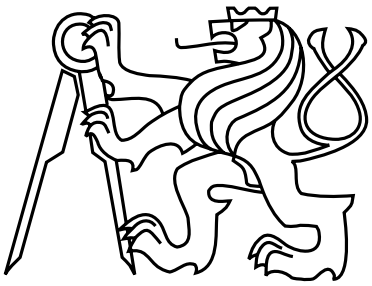
České vysoké učení technické v Praze
Fakulta elektrotechnická



DIPLOMOVÁ PRÁCE

Aplikace metrických indexovacích metod na data získaná hmotnostní spektrometrií

An Application of Metric Indexing Methods to the Mass Spectrometry Data

Jiří Novák

Vedoucí práce: RNDr. David Hoksza

Studijní program: Elektrotechnika a informatika, dobíhající magisterský

Obor: Informatika a výpočetní technika

červenec 2008

„Hmotnostní spektrometrie je jako byste rozbili židli o zeď, jednotlivé kousky zvážili a aniž byste věděli o jaký předmět se původně jednalo, se z jejich hmotností snažili poznat, že to byla židle."

IonSource.com

Poděkování

Na tomto místě bych rád poděkoval zejména RNDr. Davidu Hokszovi za pečlivé vedení, cenné připomínky při vytváření této práce a za poskytnutí obrázku 3.1.

Zvláštní poděkování bych rád vyjádřil svým rodičům Miroslavě a Jiřímu Novákovi, a také přítelkyni Lucii Dobrovodské za trpělivost při zpracování této práce a za soustavnou podporu během studia.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 22.7.2008 .....

Abstract

When we want to discover a protein function, we have to know an order of aminoacids in it's linear protein sequence. For this purpose is widely used mass spectrometry. The data are commonly interpreted using two methods - by searching in known protein sequence databases or by using graph algorithms. The research background of this work first describes currently used techniques of protein sequence identification using mass spectrometry and listed algorithms closely associate with physical and chemical principles, which are necessary for the right data interpretation.

In the second part of reserch background are described the metric indexing methods M-tree and PM-tree, which are used for reaching logarithmical complexity in similarity searching in multimedial databases. The research part of this work is based on previous background parts and focused on the interpretation by searching in databases. By using the M-tree and PM-tree structures we try to compensate the exponencially growing data abundance in protein databases and also explore the possibilities the similarity searching.

Abstrakt

Pro zjištění funkce proteinu je třeba nejprve určit posloupnost aminokyselin v jeho lineární proteinové sekvenci. Za tímto účelem se v současné době široce používá hmotnostní spektrometrie. Interpretace dat se přitom běžně provádí dvěma způsoby - vyhledáváním v databázích známých proteinových sekvencí a pomocí grafových algoritmů. Rešeršní část práce nejprve popisuje stávající techniky identifikace proteinových sekvencí z dat získaných hmotnostní spektrometrií, přičemž uvedené algoritmické postupy úzce spojuje s fyzikálními a chemickými principy, které jsou pro správnou interpretaci dat nezbytné.

Ve druhé části rešerše jsou rozebrány metrické indexovací metody M-strom a PM-strom, které se využívají pro dosažení logaritmické složitosti při podobnostním vyhledávání v multimediálních databázích. Výzkumná část práce navazuje na obě předchozí části a zaměřuje se na interpretaci dat s využitím vyhledávání v databázích. Aplikací M-stromu a PM-stromu se snažíme vyrovnat se s exponenciálně rostoucím množstvím dat v proteinových databázích a současně prozkoumat možnosti využití podobnostního vyhledávání.

Obsah

Seznam obrázků

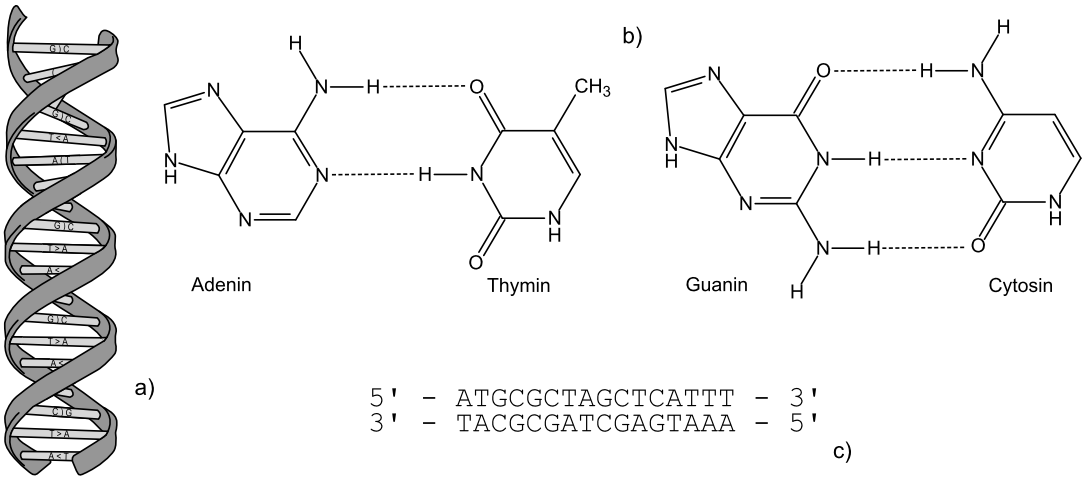
Seznam tabulek

1 Úvod

1.1 DNA

Všechny živé organismy mají v sobě zakódovanou genetickou informaci, která předurčuje jejich vlastnosti. U jednobuněčných a mnohobuněčných organismů je uložena v DNA (deoxyribonukleová kyselina). U nebuněčných organismů (např. virů) pak může být uložena i v RNA (ribonukleová kyselina). Základní jednotkou genetické informace jsou geny tj. úseky DNA, ve kterých jsou zakódovány proteiny. Proteiny (bílkoviny) jsou základními stavebními jednotkami buněk a obstarávají také většinu buněčných funkcí. Veškerou genetickou informaci uloženou v DNA resp. RNA organismu označujeme souhrnně pojmem genom.

Struktura DNA je jednoznačně určena lineární sekvencí nukleotidů adeninu (A), cytosinu (C), guaninu (G) a thyminu (T). [1] V buňce je posloupnost těchto bází uchovávána ve formě dvou komplementárních vláken vzájemně spojených vodíkovými můstky a stočených do pravotočivé dvojšroubovice. Vlákna jsou komplementární, protože na odpovídajících si pozicích v sekvencích obou vláken dochází k párování bází adeninu s thyminem (spojeny 2 vodíkovými můstky) a cytosinu s guaninem (spojeny 3 vodíkovými můstky). Kromě toho, že obě vlákna mohou být čtena ve vzájemně opačných směrech, zajišťují molekule DNA lepší stabilitu, umožňují identifikovat chyby a zjednodušují replikaci (zdvojování) molekuly. [16]



Obrázek 1.1: a) dvojšroubovice DNA, b) párování nukleotidů, c) schematický zápis komplementárních sekvencí (čísla 3' a 5' udávají vzájemnou orientaci antiparalelních řetězců; čtení DNA probíhá vždy od 5'-konce ke 3'-konci, proto se podle konvence v tomto pořadí zapisují sekvence nukleotidů)

1.2 Centrální dogma

Buňky rozdělujeme na prokaryotické (např. bakteriální) a eukaryotické (prvoci, houby, rostliny, živočichové a lidé). Zatímco DNA eukaryotických buněk tvoří dlouhé lineární molekuly organizované do chromozómů, prokaryotické buňky obsahují obvykle jednu cirkulární molekulu DNA (tzv. prokaryotický chromozóm). Počty chromozómů některých eukaryotických organismů je možné nalézt v [27].

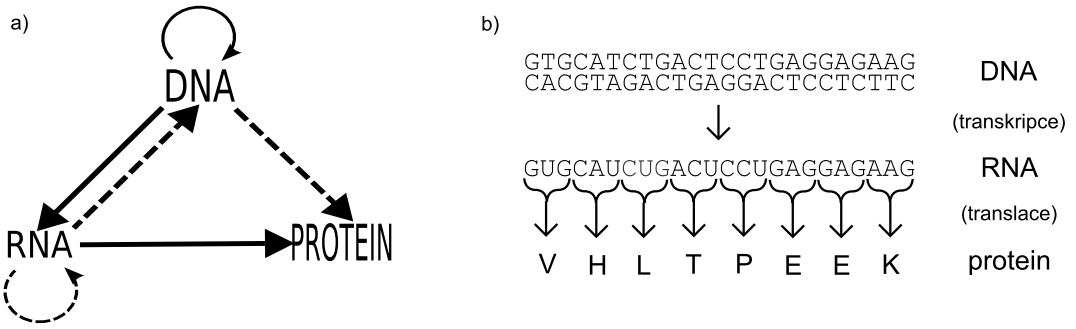
Genomy prokaryotických organismů jsou řádově kratší, např. genom bakterie *Escherichia coli* obsahuje řádově 4×10^6 bp, zatímco genom člověka má přibližně 3×10^9 bp. [2] Kromě toho

[1] V případě RNA je thymin nahrazen uracilem (U).

[2] bp = párů bází (z angl. base pairs)

u prokaryotických buněk může kódovat proteiny i více než 90% DNA, zatímco u eukaryotických je to pouze malá část celé sekvence. Odhaduje se, že pouze 5 až 10% lidské DNA kóduje proteiny.

Uspořádání prokaryotických a eukaryotických genů je naprosto odlišné. U bakterií je většina proteinů kódována nepřerušeným úsekem DNA, který je přepsán do RNA a ta se následně překládá do proteinů. U eukaryotické DNA může být úsek kódující gen rozdělen do více částí (exonů), které jsou proloženy mnohem delšími nekódujícími úseky (introny). Exony bývají dlouhé 8 bp až 17 kbp, zatímco délka intronu se pohybuje od 1 kbp do 50 kbp. [3] Před přepisem eukaryotické DNA do RNA musí tedy dojít nejprve k sestřihu, kdy jsou z úseku, kde je gen zakódován, vystříhány všechny introny a složením exonů pak vznikne mnohem kratší molekula RNA, která se překládá do proteinů. Celý proces přepisu (transkripce) DNA do RNA a následných překlad (translace) RNA do proteinů se nazývá centrální dogma molekulární biologie. [16, 4]



Obrázek 1.2: a) centrální dogma (plnými šipkami jsou označeny procesy, které běžně probíhají v buňkách, přerušovanými pak děje, které probíhají pouze za specifických podmínek např. u virů či v laboratorních podmínkách), b) schematické znázornění vzniku proteinů

1.3 Aminokyseliny a proteiny

Aminokyseliny jsou z chemického hlediska jakékoliv molekuly obsahující karboxylovou (-COOH) a aminovou (-NH_2) funkční skupinu. Zde však budeme, stejně jako v molekulární biologii, tímto pojmem rozumět 20 základních aminokyselin, které se běžně vyskytují v živých organismech jako základní stavební složky proteinů. Vzorce těchto aminokyselin prezentuje obrázek 1.3.

V DNA je každá aminokyselina zakódována trojicí bází (triplet, kodón). Protože máme 4 základní báze, snadno lze odvodit, že existuje $4^3 = 64$ kódových slov, která mohou být použita pro zakódování aminokyselin. Praxe ukázala, že kód je degenerovaný a tedy, že některé aminokyseliny jsou kódovány více triplety. Důsledkem toho je, že z posloupnosti aminokyselin nelze určit přesnou sekvenci bází, podle které byl protein vytvořen. Standardní genetický kód definující přiřazení tripletů jednotlivým aminokyselinám zobrazuje tabulka 1.1.

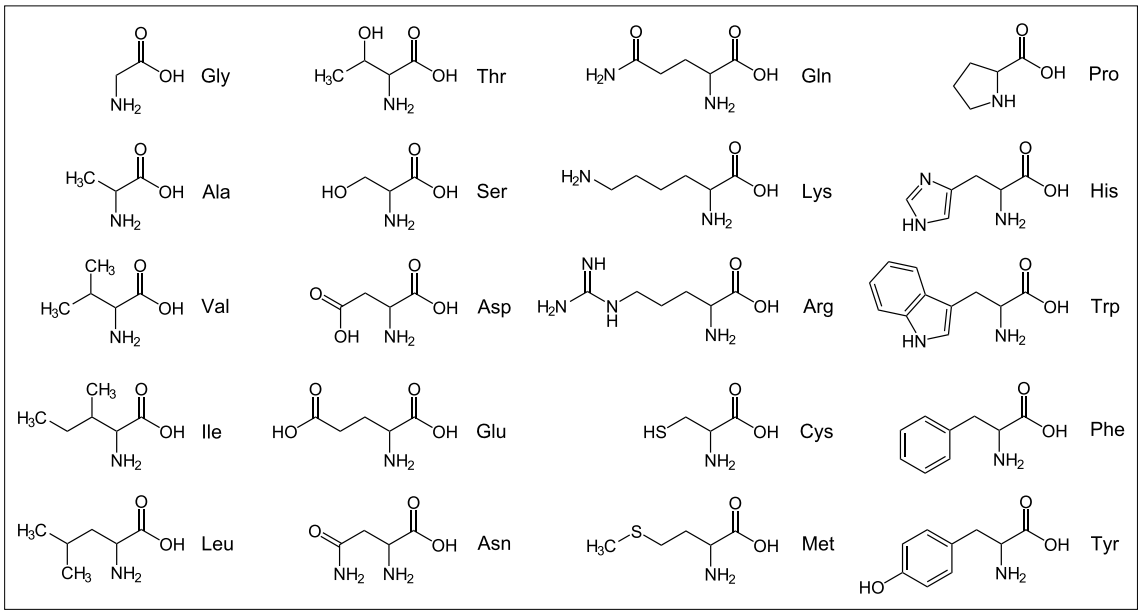
Pojmem peptidy označujeme chemické sloučeniny vznikající spojením několika aminokyselin tzv. peptidovou vazbou (-CO-NH-). Dlouhé polypeptidové řetězce tvořené lineární posloupností aminokyselin potom nazýváme proteiny. Konec lineární sekvence peptidu (proteinu), na kterém je navázána funkční skupina -NH_2, označujeme pojmem N-konec. Analogicky pak termínem C-konec rozumíme koncovou část sekvence, na které je navázána skupina -COOH (viz obrázek 2.6).

Struktura proteinů se rozlišuje na primární, sekundární, terciální a kvartérní. Primární struktura je tvořena lineární sekvencí aminokyselin. Sekundární je dána 3D motivy vyskytujícími se v uspořádání řetězce. Řetězec může vytvářet motivy šroubovice, listů skládaného

[3] kbp = kilo bp; tisíc párů bází.

1. báze		2. báze U		2. báze C		2. báze A		2. báze G		3. báze
	U	UUU	Fenylalanin	UCU	Serin	UAU	Tyrosin	UGU	Cystein	U
		UUC		UCC		UAC		UGC		C
		UUA	Leucin	UCA		UAA	stop	UGA	stop	A
		UUG		UCG		UAG	stop	UGG	Tryptofan	G
	C	CUU	Leucin	CCU	Prolin	CAU	Histidin	CGU	Arginin	U
		CUC		CCC		CAC		CGC		C
		CUA		CCA		CAA	Glutamin	CGA		A
		CUG		CCG		CAG		CGG		G
	A	AUU	Isoleucin	ACU	Threonin	AAU	Asparagin	AGU	Serin	U
		AUC		ACC		AAC		AGC		C
		AUA		ACA		AAA	Lysin	AGA	Arginin	A
		AUG	Methionin	ACG		AAG		AGG		G
	G	GUU	Valin	GCU	Alanin	GAU	Asparagová kyselina	GGU	Glycin	U
		GUC		GCC		GAC		GGC		C
		GUA		GCA		GAA	Glutamová kyselina	GGA		A
		GUG		GCG		GAG		GGG		G

Tabulka 1.1: Standardní genetický kód - kodón stop označuje konec kódující sekvence, methionin je pak zároveň start kodónem (přiřazení tripletů jednotlivým aminokyselinám resp. jejich význam se u některých typů organismů může lišit)



Obrázek 1.3: Strukturní vzorce základních aminokyselin

papíru, může obsahovat smyčky, otočky, apod. V jednom proteinu se může vyskytovat více různých motivů. Terciální strukturou pak tedy označujeme 3D strukturu celého proteinového řetězce. Kvartérní struktura je dána interakcemi (vzájemným ovlivňováním) několika řetězců, tvořících dohromady jednu funkční bílkovinu.

Různé aminokyseliny mají různé biochemické vlastnosti, a tak i jejich kombinace určují různé vlastnosti výsledných proteinů. Rovněž můžeme předpokládat, že proteiny o podobné sekvenci budou mít sklon zaujímat podobné prostorové uspořádání. Funkce proteinů je pak do značné míry dána jejich prostorovým uspořádáním. [20]

1.4 Techniky identifikace proteinů

Při výzkumu proteinů a jejich vlastností je nejprve zapotřebí určit posloupnost aminokyselin v jejich lineární proteinové sekvenci, tedy jejich primární strukturu. Pro sekvenční analýzu proteinů (resp. peptidů) se obvykle používají dvě základní metody - Edmanova degradace (přímé sekvenování) a hmotnostní spektrometrie (nepřímé sekvenování). [4]

Edmanova degradační reakce funguje na principu postupného oddělování jednotlivých aminokyselin z N-konce peptidu a jejich následné identifikaci. Metoda vyžaduje hodně zkušeností a vzorky musí být kvalitní, protože si nedokáže poradit s některými modifikacemi N-konce, což znemožňuje pokračování ve čtení sekvence. Edmanova degradace je však metoda poměrně přesná a dokáže dát velmi kvalitní výsledky. Její podstatnou nevýhodou je však doba potřebná k identifikaci celé sekvence. Proces může trvat řádově hodiny i dny, zatímco hmotnostní spektrometrie nám umožní získat data během sekund.

Hmotnostní spektrometrii označujeme jako nepřímé sekvenování, protože tato metoda sama o sobě nedokáže identifikovat proteinovou sekvenci, ale pouze nám poskytne soubor dat, na jehož základě původní sekvenci sestavujeme. Získaným souborem dat je tzv. hmotnostní spektrum nebo sada hmotnostních spekter.

1.5 Vymezení cílů práce

Hmotnostní spektrometrii a současným technikám identifikace proteinových sekvencí pomocí této metody je věnována kapitola 2. Stručně jsou popsány základní principy jednoduché (MS) a tandemové (MS/MS) hmotnostní spektrometrie, podrobněji jsou pak rozebrány algoritmické postupy interpretace získaných dat.

V případě jednoduché hmotnostní spektrometrie se pro identifikaci proteinů běžně používá metoda peptidového mapování (PMF) založená na vyhledávání v databázích známých proteinových sekvencí. Pro tandemovou hmotnostní spektrometrii je typické fragmentové mapování (PFF) a metoda identifikace s využitím tagů (Sequence Tag), obě rovněž založené na vyhledávání v databázích. Na interpretaci tandemových spekter pomocí grafových algoritmů a bez použití databáze je založeno tzv. peptidové sekvenování („De Novo" Peptide Sequencing).

Přestože techniky identifikace proteinů založené na využití databází známých proteinových sekvencí svádí k představě, že těmito metodami lze určit pouze proteiny, jejichž sekvence byly již v minulosti sestaveny, není tomu tak. Kromě anotovaných databází proteinů se totiž běžně využívají i databáze vytvořené hrubým překladem DNA sekvencí, což umožňuje identifikovat i dosud neznámé proteiny. Naopak typickým problémem technik založených na grafových algoritmech je velké množství sekvencí, které mohou odpovídat danému hmotnostnímu spektru.

Při použití databázových metod nám vyšší počet uložených sekvencí zvyšuje pravděpodobnost, že se podaří daný protein identifikovat. Čím více sekvencí však budeme analyzovat, tím déle bude celý proces trvat. Při současném trendu každoročně se exponenciálně zvyšujícího množství dat v bioinformatických databázích (viz obrázek 3.1) pak brzy zjistíme, že prohledání všech uložených záznamů omezuje praktickou použitelnost celého postupu. [5]

Pro dosažení logaritmické složitosti při vyhledávání v databázích se proto používají indexovací metody. Cílem práce je prozkoumat možnosti indexování hmotnostních spekter metrickými indexovacími metodami M-strom a PM-strom, které se používají při podobnostním vyhledávání v multimediálních databázích. Metrickým indexovacím metodám M-strom a PM-strom je věnována kapitola 3, výzkumnou částí se zabývá kapitola 4.

[4] Pehr Victor Edman (1916-1977), švédský biochemik

[5] bioinformatika = obor na pomezí biologie a informatiky

2 Hmotnostní spektrometrie

2.1 Princip metody

Hmotnostní spektrometrie (MS) je analytická metoda sloužící k převedení neutrálních molekul na nabité částice (ionty), jejich rozdělení podle poměru hmotnosti a náboje m/z (viz rovnice 2.4) a následnému záznamu relativních intenzit jednotlivých iontů. [1]

Hmotnostní spektrometr je iontově-optické zařízení separující ionty podle poměru m/z a skládá se ze tří základních částí: iontového zdroje, hmotnostního analyzátoru a detektoru. Iontový zdroj převádí neutrální molekuly zkoumané látky na ionty (tzv. ionizace). Za ním následuje hmotnostní analyzátor, který ionty za vysokého vakua rozděluje podle poměru hmotnosti a náboje m/z. Detektor pak slouží k jejich detekci a k určení relativní intenzity jednotlivých iontů. Před iontový zdroj se umisťuje ještě zařízení pro zavádění vzorků a za detektor počítač sloužící k ovládání přístroje a sběru dat. Situaci schematicky ilustruje obrázek 2.1.

Obrázek 2.1: Princip hmotnostní spektrometrie (iontový zdroj u některých metod např. ESI může pracovat za atmosférického tlaku)

Hmotnostních spektrometrů v současnosti existuje celá řada, jejich konkrétní typy často vznikají kombinacemi specifických zařízení použitých pro iontový zdroj a hmotnostní analyzátor. Popisy a fyzikální principy jednotlivých částí hmotnostních spektrometrů jsou podrobně rozebrány v [24].

2.1.1 Iontové zdroje

Způsoby ionizace, které využívá iontový zdroj, obecně rozdělujeme na tvrdé a měkké. Tvrdou ionizační technikou je např. elektronová ionizace. Metoda byla dříve označována nesprávným názvem „ionizace nárazem elektronů". To se dnes již nedoporučuje, protože zde ve skutečnosti nedochází k nárazu elektronu do molekuly, ale pouze k ovlivnění elektromagnetických polí, což způsobuje uvolnění valenčního elektronu a vznik molekulového iontu M^+.

Uvedená technika se nazývá tvrdá, protože ionizovaná molekula při ní získá nadbytek vnitřní energie a to se projevuje fragmentací molekuly na menší části. Elektronovou ionizaci lze popsat následující rovnicí.

$$M + e^- \rightarrow M^+ + 2e^- \tag{2.1}$$

Při použití měkké (šetrné) ionizační techniky na rozdíl od elektronové ionizace vznikají protonované molekuly $[M+H]^+$ (při záznamu kladných iontů) nebo deprotonované molekuly $[M-H]^-$ (při záznamu záporných iontů). Pojem protonovaná molekula označuje iont vznikající interakcí molekuly s protonem odcizeným z nějakého jiného iontu. Uvedený proces vyjadřuje

[1] Zkratka MS (z angl.) v závislosti na kontextu obvykle označuje pojmy hmotnostní spektrometrie (mass spectrometry), hmotnostní spektrometr (mass spectrometer) nebo hmotnostní spektrum (mass spectrum). Zde budeme touto zkratkou implicitně označovat metodu hmotnostní spektrometrie, nebude-li uvedeno jinak.

následující rovnice.

$$[X + H]^+ + M \rightarrow X + [M + H]^+ \tag{2.2}$$

Deprotonovanou molekulou pak označujeme iont vzniklý ztrátou protonu, což lze analogicky vyjádřit následující rovnicí.

$$[X - H]^- + M \rightarrow X + [M - H]^- \tag{2.3}$$

Použitím měkké ionizační techniky získá ionizovaná molekula mnohem menší množství energie, a proto ve výsledných spektrech pozorujeme zejména (de)protonované molekuly a minimum fragmentovaných iontů.

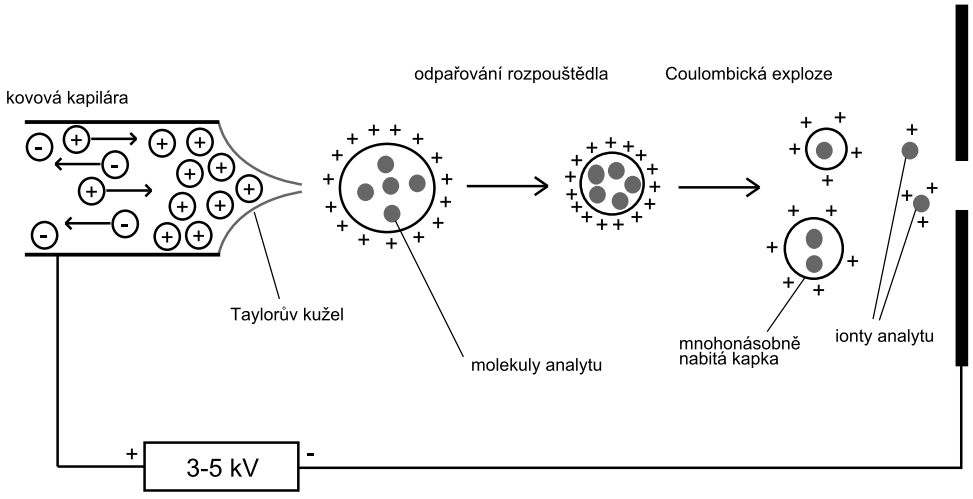
Mezi nejšetrnější měkké techniky ionizace patří ionizace elektrosprejem (ESI) a ionizace laserem za účasti matrice (MALDI), proto se často používají pro analýzu proteinů.

2.1.2 Ionizace elektrosprejem (ESI)

Ionizace elektrosprejem (Electrospray Ionization) je považována za nejšetrnější ionizační techniku vůbec a kromě toho dokáže pracovat za normálního tlaku. Fakt, že není potřeba vakuum, přispěl i k celkové oblibě této metody. Typicky se používá v zapojení spolu s hmotnostním analyzátorem tzv. iontovou pastí.

Princip metody spočívá v tom, že se analyzovaná látka (rozpuštěná ve vhodném rozpouštědle) přivádí vstupní kovovou kapilárou do iontového zdroje a v něm pak za pomoci proudu dusíku vznikají velmi malé kapičky (aerosol), které nesou díky vysokému napětí (3 - 5 kV) v kapiláře množství nábojů. Postupné odpařování rozpouštědla zmenšuje povrch i objem kapiček a povrchová hustota elektrostatického náboje postupně roste. Jakmile hustota náboje dosáhne kritické hodnoty, dojde k tzv. Coulombické explozi (rozpadu nabité kapičky na řadu mnohem menších kapiček nesoucích náboj).

Cyklus Coulombických explozí a odpařování rozpouštědla se opakuje tak dlouho, dokud není kapička dostatečně malá na to, aby mohlo dojít k uvolnění iontu z jejího povrchu. Tento proces se obvykle označuje jako „vypařování iontů".



Obrázek 2.2: Princip ESI

Při použití ESI vznikají mnohonásobně protonované ionty $[m+zH]^{z+}$, kde hodnota náboje z může nabývat řádově i několika desítek. Teoretickou hodnotu poměru hmotnosti a náboje m/z pro danou (de)protonovanou molekulu můžeme určit podle vztahu

$$\frac{m}{z} = \frac{M + zH}{|z|}, \tag{2.4}$$

kde M je relativní molekulová hmotnost neutrální molekuly a $H = 1.00794$ je relativní hmotnost atomu vodíku (resp. hmotnost protonu).

Uvažujme nyní jednoduchý motivační příklad využití ESI spolu s běžným hmotnostním analyzátorem (rozsah m/z do 3000). Dále předpokládejme, že se snažíme identifikovat neznámý protein, přičemž máme k dispozici jeho části (peptidy), které po ionizaci elektrosprejem vytvoří mnohonásobně nabité ionty, kde $z \geq 50$. Nyní dosadíme do rovnice 2.4 a pro jednoduchost uvažujeme $H = 1$.

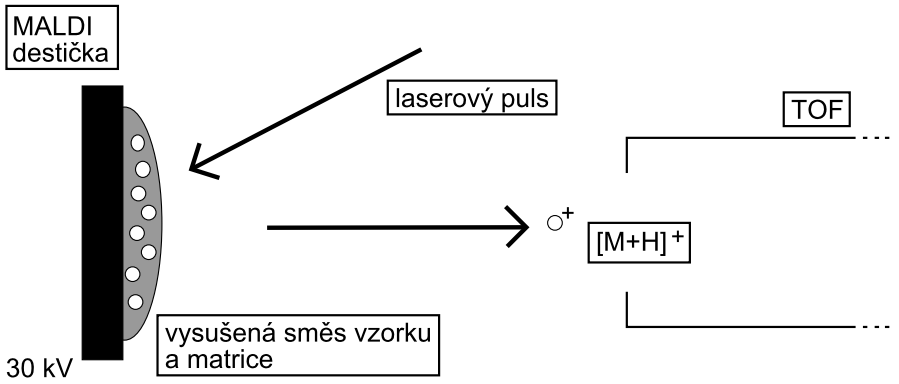
$$3000 = \frac{M + 50}{50} \Rightarrow M = 149950$$

Na hmotnostním analyzátoru s maximálním rozsahem m/z do 3000, můžeme tedy pozorovat peptidy s relativní molekulovou hmotností až do cca 150 kDa [2], pokud ovšem existují ionty, kde $z \geq 50$.

Uvedený příklad naznačuje, že ESI je velmi vhodná metoda pro analýzu biomolekul a zejména pak proteinů s velkou molekulovou hmotností. Moderní přístroje umožňují běžně analyzovat i biomolekuly s hmotností přesahující 10^6 Da.

2.1.3 Ionizace laserem za účasti matrice (MALDI)

Ionizace laserem za účasti matrice (Matrix Assisted Laser Desorption Ionization) je měkká ionizační technika (zřejmě nejšetrnější hned po ESI). Při použití této metody je vzorek nejprve rozpuštěn a smíchán s matricí, následně usušen (vykrystalizován) na MALDI destičce. Pak jsou krátkými laserovými pulsy ionizovány molekuly matrice a molekuly vzorku jsou následně ionizovány přenosem protonu z matrice (viz rovnice 2.2). Nejčastěji jsou pozorovány ionty $[M + zH]^z$, kde $z = \{-1, 1\}$. Méně často pak ionty, kde $z = 2$ a výjimečně i ionty se $z = 3$.



Obrázek 2.3: Princip MALDI

MALDI se obvykle používá ve spojení s TOF hmotnostními analyzátory. Data z MALDI-TOF hmotnostní analýzy se obvykle využívají pro metodu identifikace proteinových sekvencí nazvanou Peptide Mass Fingerprinting (PMF; viz sekce 2.3.2). Moderní MALDI-TOF vybavené reflektronovým analyzátorem však umožňují už i novější MS/MS analýzu.

[2] Da = Dalton; jednotka relativní molekulové hmotnosti

2.1.4 Hmotnostní analyzátory

Cílem hmotnostního analyzátoru je separovat ionty v plynné fázi podle poměru jejich hmotnosti a náboje (m/z). Existují různé typy analyzátorů využívající různé principy např. analyzátor doby letu, magnetický analyzátor, kvadrupólový analyzátor nebo iontová past. Oddělování iontů probíhá za vysokého vakua, aby nedocházelo ke kolizním srážkám s neutrálními atomy.

Analyzátor doby letu (Time-Of-Flight, TOF) využívá toho, že urychlené ionty o stejné kinetické energii se pohybují různou rychlostí v závislosti na hodnotě m/z, tedy toho že „menší" ionty se pohybují rychleji. Ionty jsou urychleny na vstupu do analyzátoru a potom se měří čas, za který „dolétnou" k detektoru, čímž se určí hodnota jejich poměru m/z.

Magnetický analyzátor (Magnetic sector analyser) využívá principu zakřivení dráhy iontů v magnetickém poli, které je závislé na hodnotě m/z. Při průchodu magnetickým polem dochází u iontů s nižší hodnotou m/z k většímu zakřivení jejich dráhy, u iontů s vyšší hodnotou m/z se pak dráhy zakřiví méně. Hodnota m/z je úměrná druhé mocnině poloměru dráhy iontu.

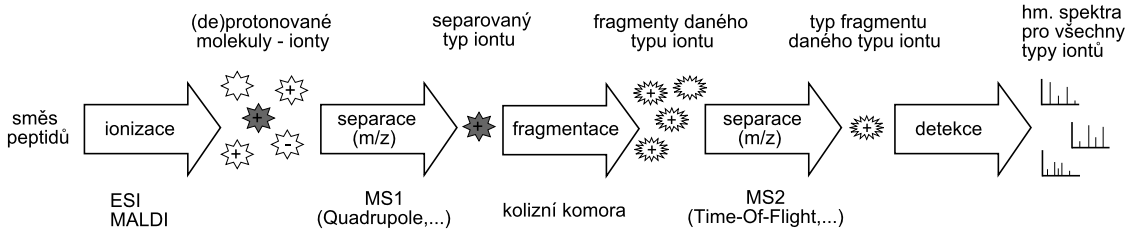
Kvadrupólový analyzátor (Quadrupole mass analyser) využívá toho, že v daný okamžik jím mohou projít pouze ionty s danou hodnotou m/z a ostatní ionty zůstanou zachyceny. Změnou napětí pak postupně projdou na detektor ionty se všemi hodnotami m/z. Podobného principu využívá i iontová past (Ion trap), kam jsou nejprve přivedeny všechny ionty a pak jsou postupně vypuzovány na detektor.

Fyzikální principy a popisy jednotlivých typů analyzátorů jsou podrobně rozebrány v [24].

2.1.5 Tandemová hmotnostní spektrometrie (MS/MS, MS^2)

Tandemový hmotnostní spektrometr má obvykle dva analyzátory oddělené kolizní komorou (collision cell), která je vyplněna inertním plynem (např. argon, xenon). Tandemová hmotnostní analýza pak probíhá ve 2 fázích.

V první fázi jsou všechny ionty podle poměru m/z postupně vpouštěny do kolizní komory, kde dochází ke kolizně indukované disociaci (Collision Induced Dissociation, CID), tj. srážkám s molekulami netečného plynu a rozpadu iontů na fragmenty. Ve druhé fázi jsou všechny fragmenty daného typu iontu (určeného poměrem m/z) opět podle poměru m/z pouštěny k detektoru. Tímto postupem získáme pro každý typ iontu hmotnostní spektrum jeho fragmentů.



Obrázek 2.4: Tandemová hmotnostní spektrometrie (MS1 značí první fázi hmotnostní analýzy, MS2 fázi druhou)

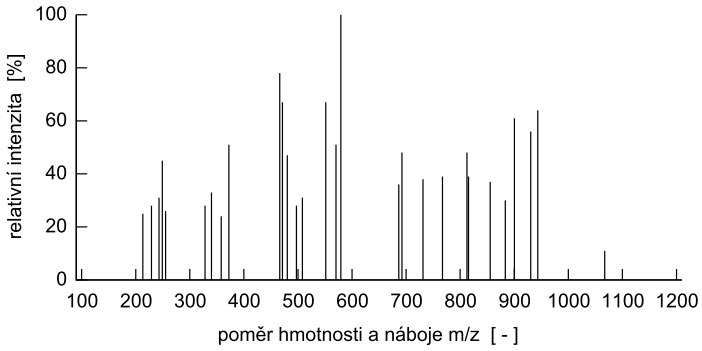
Získaná MS/MS spektra obsahují pouze fragmentové ionty vzniklé rozpadem daného typu iontu a neobsahují žádné nečistoty. Oba použité analyzátory mohou být stejného typu (Quadrupole-Quadrupole, TOF-TOF) nebo různých typů (Quadrupole-TOF tzv. QTOF), apod. [17]

Moderní spektrometrické metody umožňují nejen MS^2, ale dokonce i MS^n hmotnostní analýzu, kde $n \leq 10$. Pro účely této práce však vystačíme s MS a MS/MS hmotnostními analýzami.

2.2 Hmotnostní spektrum

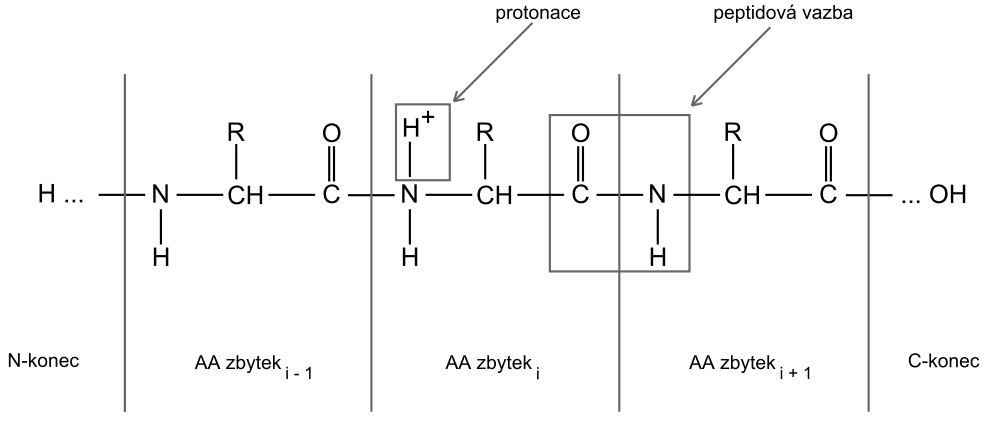
Hmotnostní spektrum zobrazuje závislost četnosti výskytu jednotlivých typů iontů na poměru jejich hmotnosti a náboje m/z. Protože hodnoty absolutní intenzity iontů jsou často i řádově závislé na konkrétním způsobu měření, převádí se četnost do normalizovaného tvaru (na relativní intenzitu vyjádřenou v %). Jednotlivé typy iontů s daným poměrem m/z jsou v grafickém zobrazení spektra viditelné jako peaky. [3] Nejvyššímu peaku ve spektru pak přísluší hodnota intenzity 100%.

Pojmem experimentální spektrum zde rozumíme spektrum získané měřením, zatímco teoretickým spektrem označujeme údaje vypočtené na základě již známé proteinové nebo peptidové sekvence. Vypočítáváme přitom pouze teoretické hodnoty poměru m/z, hodnoty intenzit určit nelze.



Obrázek 2.5: Hmotnostní spektrum

Současné metody identifikace proteinů využívají obvykle hmotnostní spektra získaná „jednoduchou" MS nebo tandemovou MS/MS hmotnostní analýzou. V případě MS hmotnostní spektrometrie jsou „nastříhané" části proteinu (peptidy) ionizovány, čímž se z jejich neutrálních molekul stávají ionty, které jsou následně zaznamenány ve výsledném spektru.

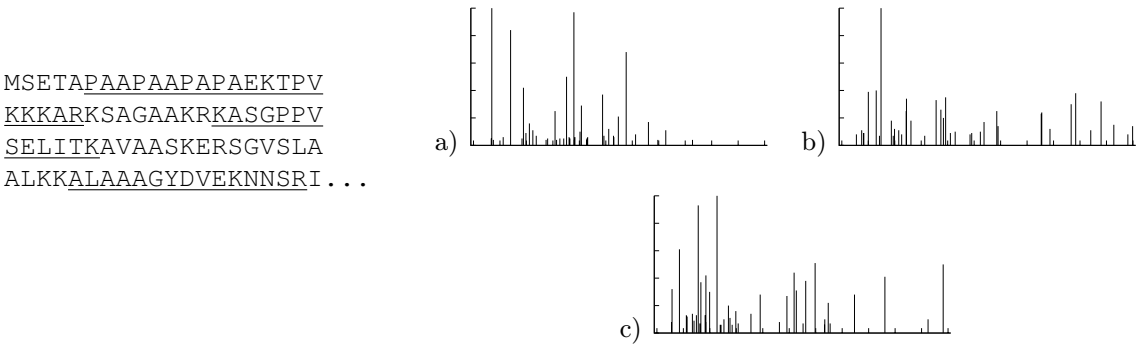


Obrázek 2.6: Struktura peptidového iontu - peptid vzniká spojením několika aminokyselinových (AA = amino acid) zbytků tzv. peptidovou vazbou (viz sekce 1.3); soustava vazeb vedoucí od N-konce k C-konci se označuje jako hlavní řetězec, části řetězce s navázanými různými radikály R jsou postranní řetězce; naznačena je protonace molekuly vznikající při ionizaci.

[3] peak [čti pík] = hrot, špička

Z principu hmotnostní spektrometrie vyplývá, že neionizované peptidy se nemohou dostat k detektoru a tudíž nejsou zaznamenány. Obvyklými ionizačními technikami pro analýzu proteinů jsou ESI a MALDI, proto budeme nadále implicitně předpokládat použití právě těchto dvou metod.

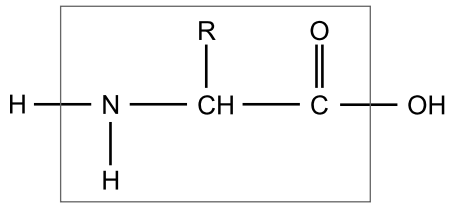
Tandemová hmotnostní spektrometrie jde oproti MS analýze ještě o krok dále a pro každý typ iontu vygeneruje hmotnostní spektrum jeho fragmentů. Při použití této metody tedy nezískáme jedno spektrum ionizovaných peptidů jako u MS analýzy, ale sadu fragmentových spekter pro jednotlivé typy iontů. Možnosti fragmentace iontů jsou podrobně rozebrány v sekci 2.4.



Obrázek 2.7: Význam sady spekter v tandemové hmotnostní spektrometrii (každé spektrum obsahuje informaci o jednom peptidu analyzovaného proteinu; odpovídající peptidy jsou pořadě zvýrazněny).

2.2.1 Využití znalosti aminokyselin

V mnoha bioinformatických aplikacích je výhodná a často pro správnou interpretaci dat i nezbytná znalost vlastností jednotlivých aminokyselin. V případě dat získaných hmotnostní spektrometrií jsou pro nás z hlediska identifikace proteinů nejzajímavější informace o jejich chemické struktuře (viz obrázky 1.3 a 2.8) a především pak o relativní molekulové hmotnosti jednotlivých aminokyselin (viz tabulka 2.2).



Obrázek 2.8: Obecná struktura aminokyseliny (výjimku tvoří prolin, kde kvůli cyklické vazbě chybí u dusíku jeden atom vodíku). „R" označuje radikál lišící se podle typu aminokyseliny.

Prvek	**Značka**	A_r
Vodík	H	1.00794
Uhlík	C	12.0107
Kyslík	O	15.9994
Dusík	N	14.00674
Síra	S	32.066

Tabulka 2.1: Prvky vyskytující se v základních aminokyselinách (A_r označuje průměrnou relativní atomovou hmotnost prvku). Monoizotopické hmotnosti uvedených prvků je možné nalézt v [21].

Obvykle se uvádí hmotnosti aminokyselin dvojího druhu monoizotopické a průměrné. [4] Při výpočtu monoizotopické hmotnosti se uvažují pouze izotopy prvků, které se v přírodě vyskytují nejčastěji. Průměrná hmotnost aminokyselin se pak stanovuje na základě četnosti výskytu jednotlivých izotopů prvků, ze kterých jsou aminokyseliny složeny.

Základní chemickou strukturu, která je u všech aminokyselin stejná, zobrazuje obrázek 2.8. V rámečku je vyznačen tzv. aminokyselinový zbytek, jehož hmotnost se v literatuře obvykle uvádí jako hmotnost aminokyseliny. Je to dáno z ryze praktického hlediska, protože hmotnosti aminokyselin se často používají pro výpočty hmotností peptidů a proteinů, kdy jsou atomy mimo vyznačenou oblast odstraňovány a nahrazovány peptidovou vazbou.

Aminokyselina	**Zkratka**	**Značka**	**Relativní molekulová hmotnost**	
			Monoizotopická	**Průměrná**
Alanin	Ala	A	71.03712	71.08
Arginin	Arg	R	156.10112	156.2
Asparagin	Asn	N	114.04293	114.1
Asparagová kyselina	Asp	D	115.02695	115.1
Cystein	Cys	C	103.00919	103.1
Fenylalanin	Phe	F	147.06842	147.2
Glutamin	Gln	Q	128.05858	128.1
Glutamová kyselina	Glu	E	129.0426	129.1
Glycin	Gly	G	57.02147	57.05
Histidin	His	H	137.05891	137.1
Isoleucin	Ile	I	113.08407	113.2
Leucin	Leu	L	113.08407	113.2
Lysin	Lys	K	128.09497	128.2
Methionin	Met	M	131.04049	131.2
Prolin	Pro	P	97.05277	97.12
Serin	Ser	S	87.03203	87.08
Threonin	Thr	T	101.04768	101.1
Tryptofan	Trp	W	186.07932	186.2
Tyrosin	Tyr	Y	163.06333	163.2
Valin	Val	V	99.06842	99.07

Tabulka 2.2: Označení a molekulové hmotnosti základních aminokyselin (uvedeny jsou hmotnosti tzv. aminokyselinových zbytků).

2.2.2 Problémy při identifikaci proteinových sekvencí

Použití hmotnostní spektrometrie pro stanovení sekvence proteinu je sice rychlejší než již v úvodu zmíněná Edmanova degradace, ale jak už to bývá, rychlost je do určité míry vykoupena kvalitou. Problémy vznikající při identifikaci sekvencí touto metodou plynou především z toho, že získaná data nám neposkytují přímo informace o obsažených aminokyselinách ani o jejich pořadí, nýbrž máme k dispozici pouze soubor hmotností a intenzit, na jehož základě se snažíme sekvenci neznámého proteinu určit. Může tedy docházet k záměnám aminokyselin se stejnou nebo podobnou hmotností.

Stejné počty atomů mají ve svých molekulách zastoupeny leucin a isoleucin, mají tedy shodné sumární vzorce a tím pádem i stejnou hmotnost, nazývají se proto izomery. Podobnou hmotnost (lišící se jen v řádech setin) mají glutamin a lysin. Problémy však nezpůsobují pouze jednotlivé aminokyseliny, ale snadno může docházet i k záměnám s jejich dvojicemi, pokud

[4] izotopy = atomy stejného prvku lišící se počtem neutronů

např. dvě aminokyseliny vyskytující se v sekvenci vedle sebe mají součet hmotností podobný hmotnosti jiné aminokyseliny nebo hmotnosti jiné dvojice.

Aminokyselina nebo dvojice	M_r		**Aminokyselina nebo dvojice**	M_r
Leu	113.08407	$\Leftrightarrow$	Ile	113.08407
Gln	128.05858	$\leftrightarrow$	Lys	128.09497
Ala + Gly	128.05858	$\Leftrightarrow$	Gln	128.05858
Ala + Gly	128.05858	$\leftrightarrow$	Lys	128.09497
Gly + Gly	114.04294	$\leftrightarrow$	Asn	114.04293
Gly + Val	156.08989	$\leftrightarrow$	Arg	156.10112
Ala + Asp	186.06407	$\Leftrightarrow$	Glu + Gly	186.06407
Ala + Asp	186.06407	$\leftrightarrow$	Trp	186.07932
Glu + Gly	186.06407	$\leftrightarrow$	Trp	186.07932
Ser + Val	186.10045	$\leftrightarrow$	Trp	186.07932
Ser + Ser	174.06406	$\leftrightarrow$	Cys (akrylamid.;+5H+3C+O+N)	174.04629
Phe	147.06842	$\leftrightarrow$	Met (oxidovaný;+O)	147.03539

Tabulka 2.3: Nejčastější záměny při identifikaci aminokyselin (uvažovány jsou pouze aminokyselinové zbytky a monoizotopické hmotnosti; M_r označuje molekulovou relativní hmotnost; $\Leftrightarrow$ označuje shodu hmotností a $\leftrightarrow$ podobnost hmotností)

Dalším problémem při identifikaci proteinových sekvencí jsou posttranslační modifikace (probíhají po translaci tj. po ukončení překladu z RNA do proteinů) a modifikace aminokyselin všeobecně. Způsobují totiž nejen strukturní změny, ale i změny hmotností, což komplikuje identifikaci sekvence. Počet známých modifikací proteinů je v současnosti již tak velký, že vznikla dokonce specializovaná databáze proteinových modifikací pro použití v hmotnostní spektrometrii [14]. V databázi UNIMOD bylo v době vytváření této práce registrováno již na 550 známých proteinových modifikací.

Typ modifikace	**Strukturní změna**	**Zasažené aminokyseliny**	**Hmotnostní rozdíl**
ztráta vody	$-H_2O$	S, T, Y; N, Q (na C-konci); D, E	-18.010565
ztráta amoniaku	$-NH_3$	K, R, N, Q (obzvláště N-koncové)	-17.026549
ztráta močoviny	$-CON_2H_4$	C-koncové R	-60.032363
hydratace	$+H_2O$	H, R; obzvláště b-ionty	+18.010565
methylace	$+CH_2$	více typů	+14.015650
hydroxylace	$+O$ (za vzniku OH)	K, P	+15.994915
oxidace	$+O$	M	+15.994915
acetylace	$+C_2H_2O$	S	+42.010565
karbamidace	$+C_2H_3NO$	C	+57.021464
karboxylace	$+C_2H_2O_2$	C	+58.005479
fosforylace	$+HPO_3$	S, T, Y	+79.966330
amidace	$+NH-O$	lib. C-konec	-0.984016
formylace	$+CO$	S, T, K; lib. N-konec	+27.994915
sulfonace	$+SO_3$	S, T, Y, C	+79.956815
kontaminace Na^+	$+Na-H$	D, E; lib. C-konec	+21.981944
kontaminace K^+	$+K-H$	D, E; lib. C-konec	+37.955588
kontaminace Cu^+	$+Cu-H$	D, E; lib. C-konec	+61.921776

Tabulka 2.4: Modifikace aminokyselin (obsažen je pouze výběr některých modifikací; uvedeny jsou monoizotopické hmotnosti). [10, 14]

Modifikace vznikají obvykle *in vivo*, *in vitro* nebo při analýze v hmotnostním spektrometru. *In vivo* nastávají přirozeným způsobem při vytváření proteinu v organismu. Některé z nich (např. glykosylace) jsou obvykle odbourávány při přípravě proteinového vzorku pro hmotnostní analýzu, zatímco jiné (sulfonace) můžeme pozorovat ve spektru v podobě „posunutých" peaků.

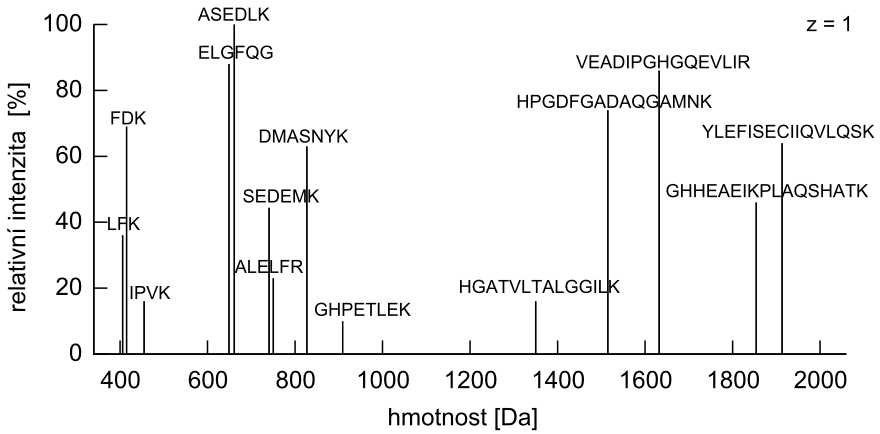
In vitro vznikají laboratorní přípravou proteinu pro analýzu. Takové modifikace mohou být vytvářeny záměrně, např. cystein bývá běžně karbamidován, aby se přerušily disulfidové můstky a ochránil se tak před další reaktivitou. Jiné nastávají spontánně, např. oxidace methioninu je důsledkem působení vzduchu. Karbamidace cysteinu se projevuje obvykle u všech cysteinových zbytků, oxidace methioninu je spíše náhodná.

Některé modifikace nastávají i ve spektrometru. Typicky se jedná o ztráty postranních řetězců aminokyselin nebo jejich částí, ztrátu vody, amoniaku, apod. Občas můžeme pozorovat kontaminaci celého spektra ionty kovů, což vzniká náhradami protonů poskytujících iontům kladný náboj za ionty kontaminujícího kovu (např. sodíkový ion Na^+). Tato forma kontaminace se pozná poměrně snadno, protože každý peak bývá „rozdělen" stejným způsobem na peaky, které jsou od sebe vzdáleny o určitý hmotnostní schodek.

2.3 Interpretace MALDI-TOF hmotnostního spektra

V případě použití MALDI-TOF hmotnostní spektrometrie dosahujeme nejlepších výsledků, pokud analyzujeme pouze jeden nebo několik málo různých proteinů. Důležité je však i to, aby zkoumaný vzorek byl kvalitní a neobsahoval zbytečné příměsi.

Nadále budeme předpokládat, že MALDI-TOF spektrometr pracuje v tzv. pozitivním módu, kdy vznikají pouze protonované molekuly. V uvedeném módu jsou typicky vytvářeny ionty s nábojem $z = 1$, připustíme-li tedy zjednodušení uvažující existenci iontů pouze s tímto nábojem, pak můžeme tvrdit, že na ose x hmotnostního spektra jsou uvedeny přímo hmotnosti jednotlivých iontů.



Obrázek 2.9: MALDI spektrum myoglobinu (svalová bílkovina) - přidělené hodnoty intenzit jsou pouze ilustrativní, ve skutečnosti nelze jejich teoretické hodnoty ze sekvence proteinu odvodit, přesto některé vyhledávače mohou intenzity využívat statisticky jako podpůrný prostředek při hodnocení proteinových sekvencí.

Každý protein musí být nejprve rozdělen na kratší peptidy, jejichž směs je následně podrobena analýze. K tomu se využívají enzymy, které dokáží štěpit proteiny na specifických místech. Nejznámějším, nejlevnějším a také nejčastěji používaným enzymem je trypsin, který „stříhá" řetězec aminokyselin vždy na C-konci lysinu nebo argininu (tj. za K nebo za R), pokud nejsou bezprostředně následovány prolinem (P).

Enzym	Dělí sekvenci	Pokud není
arg C	za R	před P
asp N	před D	
chymotrypsin	za F, (L, M,) W nebo Y	před P; za PY
bromkyan	za M	
Glu C (zásaditý)	za E	před P nebo E
Glu C (kyselý)	za D nebo E	před D nebo E
Lys C	za K	
pepsin (vysoká kyselost)	za F nebo L	
pepsin (nízká kyselost)	za A, E, F, L, Q, W nebo Y	
proteináza K	za A, C, F, G, M, S, W nebo Y	
trypsin	za K nebo R	před P

Tabulka 2.5: Enzymy využívané pro štěpení proteinů

Trypsin je obzvlášť vhodný, jestliže spektrometr pracuje v pozitivním módu. Pokud totiž v získaném spektru pozorujeme nějaký iont, znamená to, že původní neutrální molekula musela být schopna přijmout proton. Tato vlastnost je typická pro molekuly obsahující silně zásadité skupiny. Silně zásaditou skupinu má arginin, slaběji zásadité pak asparagin a lysin. Výhodou trypsinu je tedy záruka, že štěpením proteinu za argininem a lysinem vznikají peptidy, které tyto skupiny obsahují volně na koncích a tudíž jsou poměrně snadno schopné protonace (kromě peptidů z okrajových částí sekvence). Vytvářené peptidy mají rovněž vhodnou hmotnost pro analýzu (kolem 1000 Da) a obsahují v průměru 10 až 20 aminokyselin. Ideálně pak můžeme ve spektru pozorovat mezi 5 až 50 kvalitními peaky, kde každý odpovídá některému z peptidů zkoumaného proteinu nebo směsi proteinů.

V praxi je při štěpení proteinu obvykle nutné uvažovat i určité nedokonalosti, protože enzym může některá místa „střihu" vynechat. Tento jev často nastává, pokud je v sekvenci blízko u sebe několik aminokyselin určujících potencionální místa jejího dělení.

MGLSDGEWQLVLNVWGK|VEADIPGHGQEVLIR|LFK|GHPETLEK|FDK|FK|HLK|SED
EMK|ASEDLK|K|HGATVLTALGGILK|K|K|GHHEAEIKPLAQSHATK|HK|IPVK|YLE
FISECIIQVLQSK|HPGDFGADAQGAMNK|ALELFR|K|DMASNYK|ELGFQG

Peptid	M_r **[Da]**	**Peptid**	M_r **[Da]**	**Peptid**	M_r **[Da]**
MGLSDGEWQLVLNVWGK	1932.22	SEDEMK	737.78	IPVK	455.60
VEADIPGHGQEVLIR	1632.84	SEDEMKASEDLK	1381.47	YLEFISECIIQVLQSK	1913.26
LFK	406.53	ASEDLK	661.71	HPGDFGADAQGAMNK	1515.62
LFKGHPETLEK	1298.50	ASEDLKK	789.88	ALELFR	747.89
GHPETLEK	909.99	KHGATVLTALGGILK	1478.80	ALELFRK	876.07
GHPETLEKFDK	1300.43	HGATVLTALGGILK	1350.62	KDMASNYK	956.08
FDK	408.45	HGATVLTALGGILKK	1478.80	DMASNYK	827.91
FDKFK	683.81	KGHHEAEIKPLAQSHATK	1982.23	DMASNYKELGFQG	1459.59
FKHLK	671.84	GHHEAEIKPLAQSHATK	1854.06	ELGFQG	649.70
HLKSEDEMK	1116.25	HKIPVK	720.91		

Obrázek 2.10: Teoretický rozklad myoglobinové sekvence na peptidy - vyznačena jsou potencionální místa dělení trypsinem; v přehledu jsou uvedeny peptidy s relativní molekulovou hmotností v rozsahu 400 až 2000 Da s maximálně 1 vynechaným místem štěpení.

Specificita enzymu a chybovost jsou rovněž důležité parametry pro identifikaci proteinu ze získaného hmotnostního spektra. Někdy se pak používají i kombinace více enzymů a pravidla štěpení odpovídají sjednocení pravidel pro jednotlivé enzymy. V souvislosti s interpretací MALDI-TOF hmotnostních spekter rozebereme dva základní přístupy. Spíše teoretickou metodu „De Novo" a v praxi často aplikovanou metodu peptidového mapování založenou na využití databáze známých proteinových sekvencí.

2.3.1 „De Novo" pro MALDI-TOF

Technika interpretace MALDI-TOF hmotnostního spektra označovaná jako „De Novo" je založena pouze na znalosti hmotností jednotlivých aminokyselin (příp. jejich modifikací) a na znalosti struktury proteinu. [5] Princip spočívá v tom, že ke každému peaku hmotnostního spektra hledáme odpovídající peptid. Hmotnost, kterou každý peak spektra v ideálním případě (a za předpokladu, že náboj $z = 1$) zobrazuje, můžeme definovat vztahem

$$\sum_{i=1}^{n} M_r(i) + M_r(H_2O) + A_r(H), \tag{2.5}$$

kde n je počet aminokyselinových zbytků obsažených v daném peptidu, $M_r(i)$ jejich relativní molekulové hmotnosti, $M_r(H_2O)$ zahrnuje relativní molekulovou hmotnost koncových částí peptidu a $A_r(H)$ představuje relativní atomovou hmotnost protonu. Po dosazení můžeme výraz zjednodušit na tvar (2.6).

$$\sum_{i=1}^{n} M_r(i) + 19.02 \tag{2.6}$$

Pro každý peak tedy hledáme takovou multimnožinu aminokyselin, pro niž platí, že součet hmotností aminokyselin zvýšený o daný přírůstek odpovídá hmotnosti definované tímto peakem. Řešení tohoto problému můžeme převést na řešení zobecněného problému batohu bez cen [6]. Přestože je problém batohu obecně NP-těžký, časová složitost v tomto případě není hlavním kamenem úrazu. Podstatným problémem je především to, že ve výsledku získáme pro každý peak množství multimnožin a z nich víme pouze jaké aminokyseliny a v jakém množství mohou být v daném peptidu obsaženy, ale nevíme nic o jejich pořadí. Počet permutací aminokyselin získaných z jedné multimnožiny roste exponenciálně s hmotností peptidu a i pro malé peptidy je příliš velký na to, abychom mohli bezpečně určit jejich původní sekvence.

Můžeme se pokusit o vylepšení metody s využitím znalosti specificity enzymu štěpícího protein, což v případě trypsinu znamená, že většina peptidů bude mít jako poslední aminokyselinu v sekvenci K nebo R. Případně můžeme zlepšit přesnost měření spektrometru. Ani v jednom případě však řádového zlepšení už nedosáhneme. Ilustrační příklad je uveden v [10].

Při definici peaku jsme abstrahovali od možné kontaminace spektra cizími látkami, šumu, špatné kalibrace přístroje, zastoupení více izotopů jednotlivých prvků a mnoha dalších faktorů zkreslujících výsledky. Zároveň zde nebylo nutné explicitně uvažovat záměny aminokyselin a jejich modifikace (viz sekce 2.2.1 a 2.2.2).

Metoda „De Novo" má pro spektra MALDI-TOF spíše charakter teoretické úvahy a v praxi jsou pro skutečnou identifikaci proteinů z těchto spekter vhodné postupy založené výhradně na využití databází známých proteinových sekvencí. Princip přímé interpretace technikou „De Novo" je však využíván pro MS/MS hmotnostní spektra, kdy se snažíme určit posloupnost aminokyselin s využitím grafových algoritmů. Tato problematika je podrobně rozebrána v sekci 2.4.3.

[5] de novo (z lat.) = po novu

2.3.2 Metoda peptidového mapování (PMF)

Metoda peptidového mapování (Peptide Mass Fingerprinting, PMF) bývá běžně aplikována na data získaná MALDI-TOF hmotnostní spektrometrií. Dokáže však zpracovávat i spektra získaná z ESI-TOF analyzátoru, což ale není příliš obvyklé, protože ESI spektra vyžadují tzv. dekonvoluci (viz sekce 2.4.1).

Základem metody je předpoklad, že každý protein obsahuje unikátní množinu peptidů a tudíž ho lze identifikovat na základě množiny hmotností odpovídající hmotnostem jednotlivých peptidů. Vlastní identifikace peptidů pak probíhá vyhledáváním v databázích známých proteinových sekvencí, což je hlavní rozdíl oproti metodě „De Novo", která se snaží peptidové sekvence generovat přímo ze spektra.

Pro peptidové mapování je kromě přesnosti měření, důležitá i znalost specificity štěpení enzymu, který byl použit pro rozklad neznámého proteinu na směs peptidů. Při vyhledávání pak procházíme databázi proteinových sekvencí, pro každý protein simulujeme štěpení tímto enzymem a vypočítáváme hmotnosti predikovaných peptidů, které následně porovnáváme s experimentálně zjištěnými hodnotami. Přiřazení peaku a odpovídající sekvence peptidu budeme označovat jako hit. [6]

Čím je databáze proteinů větší, tím je větší i pravděpodobnost, že pro jednotlivé peaky získáme více různých a přitom odpovídajících peptidových sekvencí. Kromě sekvencí patřících peptidům z původního neznámého proteinu, tak najdeme i velký počet náhodných hitů rozptýlených po celé proteinové databázi. Snažíme se tedy vybrat takovou proteinovou sekvenci, která obsahuje více hitů. V tom, kdy přesně můžeme označit sekvenci proteinu za odpovídající zkoumanému spektru, se jednotlivé zdroje často liší. Orientačně se uvádí, že přesvědčivému výsledku odpovídá aspoň 5 různých hitů [21].

Identifikace je však v praxi složitější a pro výběr proteinu nejlépe odpovídajícího experimentálnímu spektru se používají různé typy skórování. Při vyhledávání přiřazujeme každému hitu hodnocení, které odráží jeho kvalitu a slouží jako základ pro výpočet celkového skóre daného proteinu. Výstupem pak obvykle není jedna identifikovaná sekvence, ale několik sekvencí s nejlepším skóre. Hodnocení jednotlivých hitů a potažmo celé proteinové sekvence lze stanovit na základě množství různých parametrů, nejčastější jsou tyto:

1. Do jaké míry odpovídá pozorovaný peak predikovanému peptidu ? Předpokládejme, že hmotnost definovaná peakem je např. 1234.56 ± 0.02. Pak peptid s teoretickou hmotností 1234.565 bude mít lepší hodnocení než peptid s hmotností 1234.585.

2. Jaká je hmotnost peptidu ? Peptidy s větší hmotností mají obvykle delší sekvenci a tudíž se vyskytují méně často. Pokud se nám tedy podaří hit pro peptid s velkou hmotností, je to z hlediska identifikace proteinu přesvědčivější fakt než hit peptidu o malé hmotnosti.

3. Kolik obsahuje predikovaný peptid vynechaných míst „střihu" ? Vstupním parametrem programu je obvykle i povolený počet vynechaných míst štěpení (nejčastěji 0 až 3). Hit peptidu s vynechanými místy štěpení má obvykle menší význam než hit bez vynechaných míst. V praxi však musíme uvažovat i nad tím, že pokud sekvence proteinu obsahuje za sebou např. několik lysinů, tak vynechání místa střihu je velmi běžné a případná penalizace hitu by neměla být výrazná.

4. V jakých místech proteinové sekvence klesá frekvence výskytu hitů a proč ? Podrobněji rozebráno v dalším textu.

[6] hit (z angl.) = zásah

Pro parametry popsané v bodech 2 a 3 je nejlépe stanovit konkrétní hodnoty empiricky. K tomu je nejlepší využít velký soubor dat obsahující dostatek kvalitních hitů i náhodného šumu, což nám umožní vhodným způsobem přiřadit rozdílné váhy různým typům hitů.

Při výběru nejlépe odpovídající proteinové sekvence (nebo sekvencí, pokud v analyzovaném vzorku bylo více proteinů) pro dané spektrum se řídíme i tím, že korektní hity inklinují k vytváření shluků. Skutečnost, že shluky hitů jsou pravděpodobnější než jejich rovnoměrné rozprostření po celé délce proteinu, vyplývá z prostorového uspořádání proteinů. Části proteinu, které jsou „lépe přístupné", totiž snadněji podléhají působení štěpícího enzymu. Proteinové sekvenci, která obsahuje přiléhající nebo překrývající se peptidy, pak můžeme přiřadit lepší ohodnocení. Pokud získáme jeden nebo několik proteinů obsahujících skupiny hitů a zbytek výstupu vytváří „náhodné pozadí", je interpretace výsledku zřejmá na první pohled. V některých případech však musíme k rozlišení významných skupin hitů od náhodného šumu použít statistiky.

Důležitým parametrem jsou délky proteinů uložených v databázi, protože delší sekvence generují více náhodných hitů. Pokud pak pracujeme s opravdu dlouhými proteinovými sekvencemi, většinou získáme velké počty náhodných hitů téměř pro jakékoliv spektrum. Pravděpodobnost, že daný hit je náhodný, roste lineárně v závislosti na délce proteinu.

Kromě délek proteinů pak můžeme při identifikaci využít i znalost jejich teoretických hmotností. Pokud totiž máme k dispozici aspoň hrubý odhad celkové hmotnosti zkoumaného proteinu, pak při procházení databáze vybíráme jen takové proteiny, jejichž hmotnost odpovídá námi požadovanému rozmezí hodnot.

Pokud jsme analyzovali směs proteinů, snažíme se nejprve identifikovat aspoň jeden z nich. Jakmile se nám to podaří, odstraníme ze spektra všechny peaky, s jejichž pomocí byl určen a opakujeme vyhledávání pro modifikované spektrum. Celý postup pak provádíme do té doby, dokud jsme schopni ve spektru nějaký protein bezpečně rozpoznat.

Databáze proteinů mohou být různého druhu. Některé vznikají hrubým překladem DNA sekvencí na proteinové, přičemž se využívá všech 6 možných čtecích rámců (trojice bází kóduje jeden protein, existují tedy 3 možné posuny čtecího rámce, přičemž DNA lze číst ve 2 směrech, viz sekce 1.3). Jiné databáze zohledňují i vystřihávání nekódujících intronů, další pak obsahují výhradně anotované (např. Swiss-Prot, `http://www.expasy.org/sprot/`) nebo experimentálně zjištěné proteinové sekvence (RCSB Protein Data Bank, `http://www.rcsb.org/`).

Vyhledávání je o něco složitější i tím, že proteinové databáze obsahují chyby. Nesrovnalosti v těchto sekvencích (nebo v sekvencích DNA, ze kterých byly generovány) mohou být sice početné, ale naše úvahy příliš radikálním způsobem neovlivňují. Obvyklou chybou je nalezení stejného nebo podobného proteinu vícekrát než jednou. Některé databáze totiž vznikají kombinací dat z různých zdrojů a přestože se snaží duplicity eliminovat, jejich odhalení není vždy zaručeno. Ve výsledku se pak může jednat o skutečně mírně odlišné proteinové sekvence nebo o sekvence stejné, v nichž jsou však chyby rozmístěny tak, že nebyly rozpoznány jako duplicitní. V případě, že z testovaného spektra získáme několik podobných sekvencí, jen stěží rozpoznáme, jestli jsou podobné nebo stejné a jen obsahují chyby. Nicméně některé skupiny proteinů mají velmi podobné sekvence a můžeme vycházet z předpokladu, že se jedná o varianty odvozené z jednoho konkrétního proteinu v rámci evoluce. [20]

Jestliže používáme databázi odvozenou ze 6 rámcového čtení DNA, můžeme najít hity seskupené i ve 2 nebo 3 „proteinových" sekvencích. Tento efekt vzniká, pokud původní sekvence DNA obsahují chyby v podobě vynechaných bází, čímž dochází k posuvu rámce pro daný směr čtení. Při vyhledávání v databázích si pak musíme uvědomit i fakt, že sama nemusí být úplná a pro dané spektrum vůbec nemusíme odpovídající proteinovou sekvenci najít.

Pokud analyzujeme pouze jeden protein nebo jednoduchou směs, přičemž vzniká poměrně málo peptidů a některé z nich se vyskytují ve shlucích v jedné nebo v případě směsi v něko-

lika proteinových sekvencích, pak je identifikace pomocí MALDI-TOF možná. Obvykle však samotné použití této metody nestačí, protože nemůže bezpečně prokázat o jaký protein se jedná. Poskytuje však určitý výčet pravděpodobných kandidátů, což je užitečné zejména proto, že peptidy poukazující na jednotlivé kandidáty, lze analyzovat pomocí tandemové hmotnostní spektrometrie, která nám může poskytnout přesvědčivější výsledky. [10]

2.3.3 Veřejně dostupné vyhledávače pro PMF

V současné době existuje řada aplikací umožňujících identifikaci proteinů metodou peptidového mapování, využívají přitom veřejně dostupné databáze proteinových sekvencí. Uveďme aspoň základní výčet těch, které jsou volně použitelné a přístupné prostřednictvím webového rozhraní:

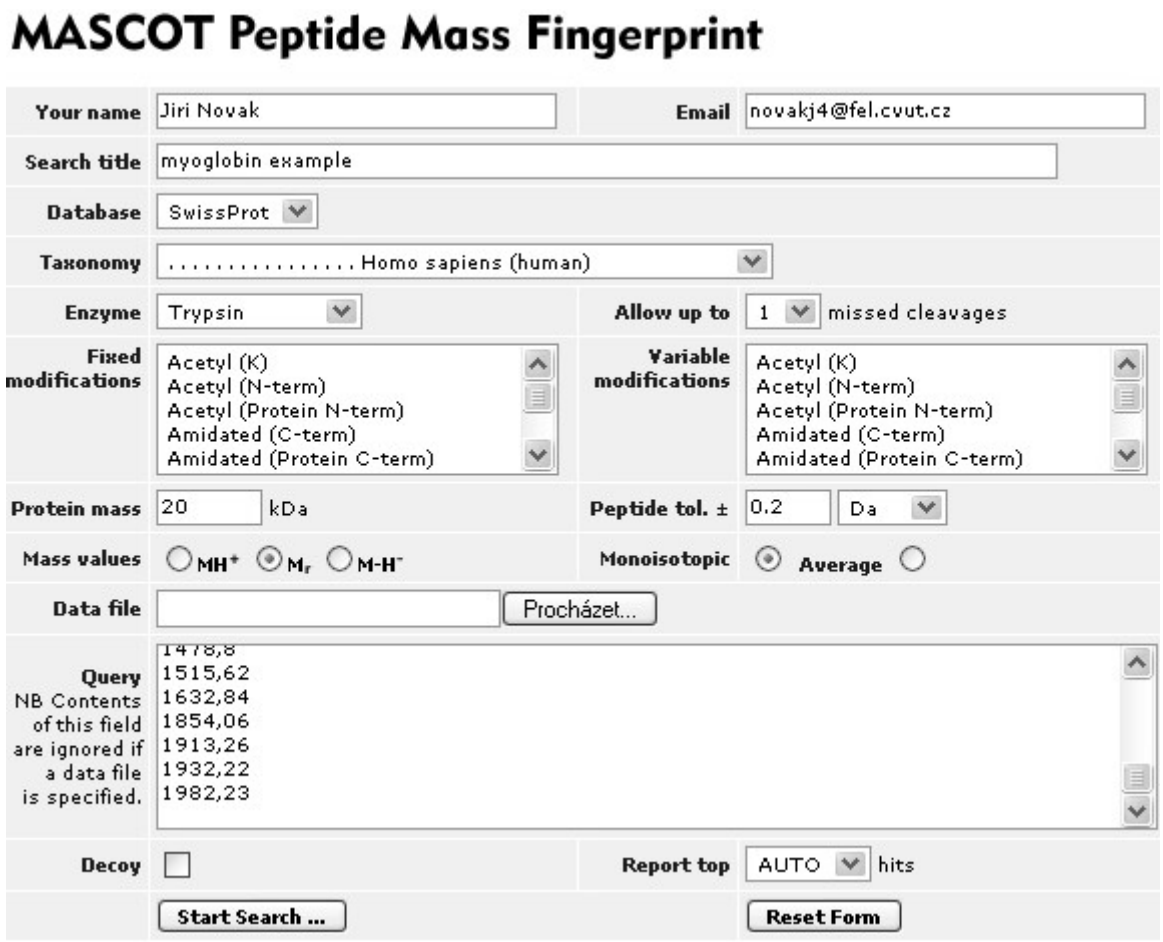
- Mascot
  `http://www.matrixscience.com/search_form_select.html`
- ProteinProspector MS-Fit
  `http://prospector.ucsf.edu/`
- Aldente
  `http://www.expasy.org/tools/aldente/`
- PeptideSearch
  `http://www.narrador.embl-heidelberg.de/GroupPages/PageLink/peptidesearchpage.html`

Různé vyhledávače mají rozdílné možnosti nastavení a každý je většinou něčím specifický. Nicméně při bližším pohledu zjistíme, že jejich společným základem je vždy seznam peaků hmotnostního spektra a několik hlavních parametrů. Seznam peaků tvoří obvykle dvojice hmotnost a intenzita. Přesnost hmotností je přitom důležitá, ale bývá také přizpůsobitelná parametrem umožňujícím nastavit odchylku, což je výhodné zejména kvůli toleranci chyby měření spektrometru, z hlediska výskytu různých izotopů jednotlivých prvků, apod. Obvykle se pak vyplatí mít k dispozici více hodnot hmotností s průměrnou přesností než jen několik málo hodnot s velmi vysokou přesností. Pro hmotnosti bývá běžně dospecifikovatelné, zda se jedná o hodnoty průměrné či monoizotopické, zda zadané údaje odpovídají (de)protonovaným nebo neutrálním molekulám a případně je možné využít i hrubý odhad celkové hmotnosti zkoumaného proteinu.

Intenzita má pro metodu peptidového mapování spíše okrajový význam, protože závisí na délce peptidu, použitém typu spektrometrické metody a několika dalších fyzikálních a chemických parametrech. Neexistuje zde tedy žádný zvláštní důvod předpokládat, že peaky s velkou intenzitou jsou zajímavější než peaky s malou intenzitou. Hlavním účelem této informace je obvykle jen odfiltrování šumu. Většina vyhledávačů ani zadání intenzit nepožaduje a specifikuje je jako volitelný parametr, který může využít statisticky k doladění výsledků.

Běžně pak bývají k dispozici možnosti výběru proteinové databáze, specifikace zkoumaného organismu, nastavení štěpícího enzymu spolu s maximálním povoleným počtem vynechaných míst a definice peptidových modifikací. [29]

Většina reálných aplikací identifikaci proteinových sekvencí zakládá na použití nějakého skórovacího algoritmu, vyhledávač Mascot používá pravděpodobnostní skórování založené na algoritmu MOWSE (MOlecular Weight SEarch). [2] Základem skórovací funkce tohoto algoritmu je matice frekvenčních koeficientů F (frequency factor matrix), ve které každý řádek reprezentuje interval 100 Da peptidové hmotnosti a sloupec interval 10 kDa proteinové hmotnosti. Hlavním důvodem pro vytvoření matice F je fakt, že peptidy s nižší hmotností vznikají častěji, přičemž tato závislost je ovlivněna ještě délkou původní proteinové sekvence. Matice tak přesně modeluje chování štěpícího enzymu, v praxi se pak používají odlišné matice pro různé enzymy a různé databáze.



Results List

1. MYG_HUMAN **Mass:** 17173 **Score:** 31 **Expect:** 15 **Queries matched:** 3

Myoglobin - Homo sapiens (Human)

Observed	Mr(expt)	Mr(calc)	Delta	Start	End	Miss	Peptide
1350.0000	1350.0000	1349.8031	0.1969	65	78	0	K.HGATVLTALGGILK.K
1478.0000	1478.0000	1477.8980	0.1020	65	79	1	K.HGATVLTALGGILKK.K
1632.0000	1632.0000	1631.8631	0.1369	18	32	0	K.VEADIPGHGQEVLIR.L

No match to: 406.0000, 408.0000, 455.0000, 649.0000, 661.0000, 671.0000, 683.
1116.0000, 1298.0000, 1300.0000, 1381.0000, 1459.0000, 1515.0000, 1854.0000,

Obrázek 2.11: Příklad s vyhledávačem Mascot - formulář s předvyplněnými údaji pro ilustrativní vyhledání sekvence myoglobinu na základě znalosti teoretických hmotností jednotlivých peptidů (viz obrázek 2.10) a úryvek odezvy vyhledávače (vidíme, že myoglobin byl identifikován na základě 3 peptidových sekvencí).

Pro výpočet frekvenčních koeficientů postupně procházíme vybranou databázi proteinových sekvencí, simulujeme štěpení daným enzymem a inkrementujeme příslušné elementy $f_{i,j}$ matice F. Všechny prvky v daném sloupci jsou poté přepočteny na pravděpodobnost jejich výskytu podle vztahu

$$f'_{i,j} = \frac{f_{i,j}}{\sum_i f_{i,j}} \tag{2.7}$$

a následně ještě normalizovány vzhledem k maximální hodnotě v daném sloupci. Normalizované hodnoty $f'_{i,j}$ označujeme jako prvky $m_{i,j}$ nové matice M (MOWSE factor matrix).

$$m_{i,j} = \frac{f'_{i,j}}{max_i\, f'_{i,j}} \tag{2.8}$$

Výsledné skóre daného proteinu je pak vypočteno podle vztahu

$$score = \frac{50000}{M_{prot} \times \prod_n m_{i,j}}, \tag{2.9}$$

kde M_{prot} je jeho relativní molekulová hmotnost, n počet hitů a normalizační hodnota 50 kDa slouží pro redukci náhodného nárůstu skóre při porovnávání s dlouhými proteinovými sekvencemi. Ve skutečnosti však na rozdíl od „klasické" varianty MOWSE používá Mascot mnohem složitější pravděpodobnostní model, jehož přesné schéma nebylo publikováno. [8]

Vyhledávání pomocí metody peptidového mapování je dnes sice široce podporováno, ale do praxe se stále častěji dostávají přístroje umožňující tandemovou hmotnostní spektrometrii, a proto v současné době již existují postupy pro mnohem přesnější identifikaci proteinů.

2.4 Interpretace tandemového hmotnostního spektra

Tandemová hmotnostní spektrometrie získává v současné době díky bouřlivému rozvoji technologií převahu nad „jednoduchou" analýzou. Pro výzkum sekvencí proteinů má pak obzvláště velký význam, protože místo jednoho spektra peptidových iontů, nám umožní získat pro každý peptid hmotnostní spektrum jeho fragmentů (viz obrázek 2.7). Tím se otevírá cesta nejen přesnější analýze proteinových sekvencí, ale i novým technikám jejich identifikace.

V souvislosti s interpretací tandemových spekter se používá metoda „De Novo" založená na grafových algoritmech, databázová metoda fragmentového mapování založená na přímém porovnávání neinterpretovaných peptidových spekter a velmi obvyklý je i hybridní způsob identifikace sekvencí kombinující výhody obou předchozích postupů, který využívá snadno interpretovatelné části spekter jako tzv. „tagy" a následně provádí dohledání výsledků v databázi.

Při samotné analýze se obvykle používá ionizace elektrosprejem. Tandemové hmotnostní spektrum lze sice získat i vylepšenou metodou MALDI Post-Source Decay (PSD), nicméně fragmentová spektra bývají poměrně často nekompletní a jejich interpretace je značně obtížná. [15]

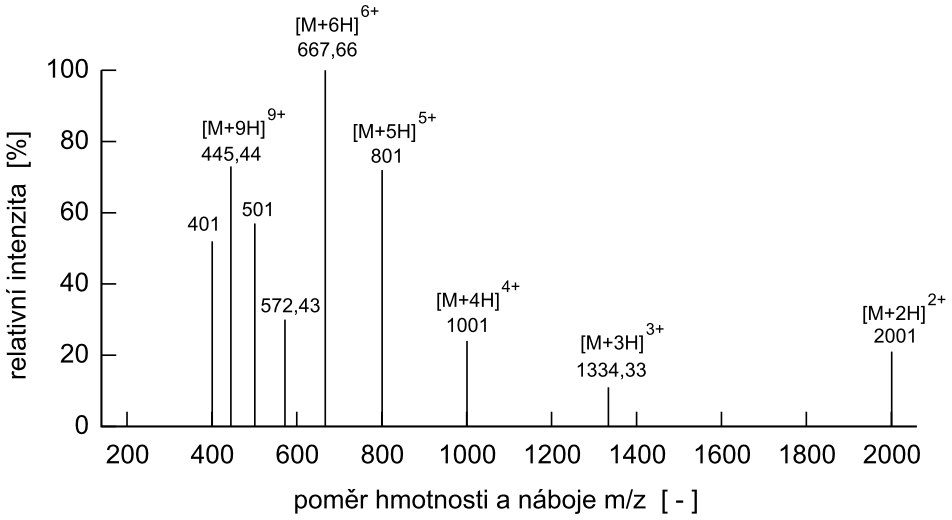
2.4.1 ESI hmotnostní spektrum

Při ionizaci elektrosprejem můžeme v získaném spektru typicky pozorovat sérii peaků odpovídající vícenásobně protonovaným iontům „stejné" molekuly. Před dalším zpracováním spektra se proto provádí tzv. dekonvoluce, při které se peaky mající různé náboje a přitom indikující tu samou neutrální molekulu zredukují na jeden peak.

Pro dekonvoluci využijeme vztah (2.4) z něhož odvodíme soustavu rovnic pro dva sousedící peaky odpovídající jedné molekule (a, b jsou konkrétní hodnoty m/z daných peaků; pro jednoduchost uvažujeme $H = 1$).

$$a = \frac{M + z_a}{z_a} \tag{2.10}$$

$$b = \frac{M + z_b}{z_b} = \frac{M + (z_a + 1)}{(z_a + 1)} \tag{2.11}$$



Obrázek 2.12: ESI hmotnostní spektrum (pro ilustraci uvádíme pouze peaky odpovídají různým nábojovým stavům jedné molekuly).

Peak jemuž odpovídá nižší hmotnost b má vyšší nábojový stav $z_b = z_a + 1$. Náboj peaku s vyšší hmotností můžeme po úpravě vyjádřit jako

$$z_a = \frac{b-1}{a-b}. \tag{2.12}$$

Pokud hodnota náboje z_a odpovídá celému číslu, lze s velkou pravděpodobností předpokládat, že se jedná o dva různě nabité ionty odpovídajících si molekul. V praxi však kvůli nepřesnostem definujeme určitou odchylku od hodnoty celého čísla, kdy ještě peaky považujeme za odpovídající jedné molekule, přičemž výsledný náboj vždy zaokrouhlujeme na celou hodnotu. Hmotnost neutrální molekuly můžeme následně dopočítat podle vztahu $M = az_a - z_a$.

Kromě odchylky náboje od celého čísla, zavádíme ještě pojem práh, kterým omezujeme hodnotu intenzity. Dekonvoluci pak provádíme pouze pro peaky, jejichž intenzita dosahuje aspoň prahové hodnoty, čímž nejen vyredukujeme šum v podobě velmi malých peaků, ale rovněž celkově urychlíme výpočet. Uveďme jednoduchý algoritmus pro dekonvoluci.

Algoritmus 2.1

```
1  ptrPeak1 = peakList.Copy(ThresholdIntensity);
2  while ptrPeak1 ≠ ∅ do
3    ptrPeak2 = ptrPeak1.GetNextPeak();
4    while ptrPeak2 ≠ ∅ do
5      m1 = ptrPeak1.Mass(); m2 = ptrPeak2.Mass();
6      if m2 > m1 then Exchange(m1,m2);
7      z = (m2-1)/(m1-m2);
8      Mr = m1*z-z;
9      if (abs(round(z)-z) ≤ tolerance) then
10       if Mr ∉ newPeakList then newPeakList.Add(Mr);
11       ptrPeak2 = ptrPeak2.RemoveAndGetNextPeak();
12     else ptrPeak2 = ptrPeak2.GetNextPeak();
13   ptrPeak1 = ptrPeak1.RemoveAndGetNextPeak();
```

Algoritmus postupně porovnává dvojice peaků a pro peak s vyšší hmotností vypočítává náboj. Jestliže hodnota náboje s danou tolerancí odpovídá celému číslu, pak pokud jsme tak již neučinili, zaznamenáme hmotnost neutrální molekuly a odstraníme všechny odpovídající

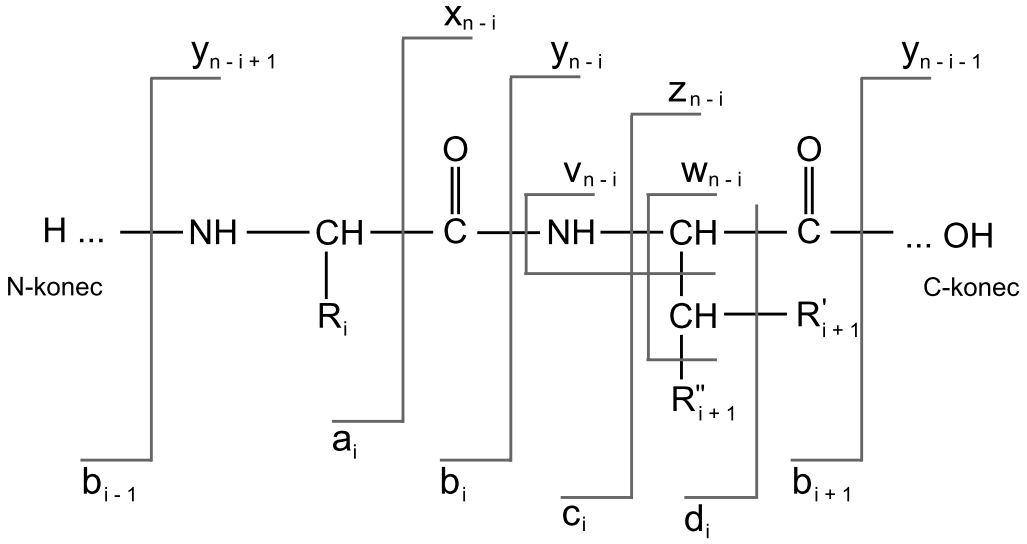
peaky. V uvedeném kódu pro jednoduchost zanedbáváme zpracování intenzit při redukci peaků. Dekonvoluce se často provádí automaticky při hmotnostní analýze. Pro ukázku jejího významu lze doporučit java applet [12].

2.4.2 MS/MS hmotnostní spektrum

Při tandemové hmotnostní spektrometrii jsou jednotlivé typy iontů postupně podrobovány fragmentaci a pro každý z nich tak získáme hmotnostní spektrum jeho fragmentů (viz obrázek 2.7), uvedený postup se označuje jako peptidové sekvenování. Kromě seznamu peaků standardně známe i poměr m/z a náboj původního ionizovaného peptidu. I v případě fragmentů platí, že jsou spektrometrem detekovány, pouze pokud mají aspoň jeden náboj. Pro tandemovou spektrometrii proteinů jsou pak typické protonované fragmenty, které budeme nadále uvažovat.

Peptidy jsou lineární molekuly a pokud předpokládáme rozdělení hlavního peptidového řetězce pouze na dva fragmenty, pak v závislosti na místě přerušení můžeme rozlišit několik základních typů fragmentových iontů. Jestliže je náboj zadržen C-koncovou částí peptidu, vznikají ionty x, y nebo z. Pokud náboj zachytí N-koncová část, mohou vzniknout ionty a, b nebo c. Nejčastěji pak vznikají typy y a b. Počet základních typů iontů pro každou koncovou část peptidu je roven třem, protože peptidová vazba může být přerušena na třech různých místech. Pro označení přesného místa se pak používá číslování iontů ve směru od konce peptidu, na jehož základě byly definovány. Hodnota pak zároveň odpovídá počtu aminokyselin v daném fragmentu (viz obrázek 2.13). Peaky odpovídající jednomu typu iontů označujeme jako iontovou sérii. Iontové série pak v praxi bývají často neúplné, tj. v experimentálním spektru nemusí nutně existovat peak pro každý teoreticky vypočtený iont daného typu.

Popsané C-koncové ionty jsou vzhledem k N-koncovým komplementární, což znamená, že na základě znalosti peptidové sekvence a y-iontu jsme schopni určit odpovídající b-iont, apod. Dále lze pozorovat, že ionty každé koncové části se od sebe liší o jistý pevně daný úsek, čili při znalosti a-iontu můžeme určit i b-iont, při znalosti z-iontu pak y-iont, atd.

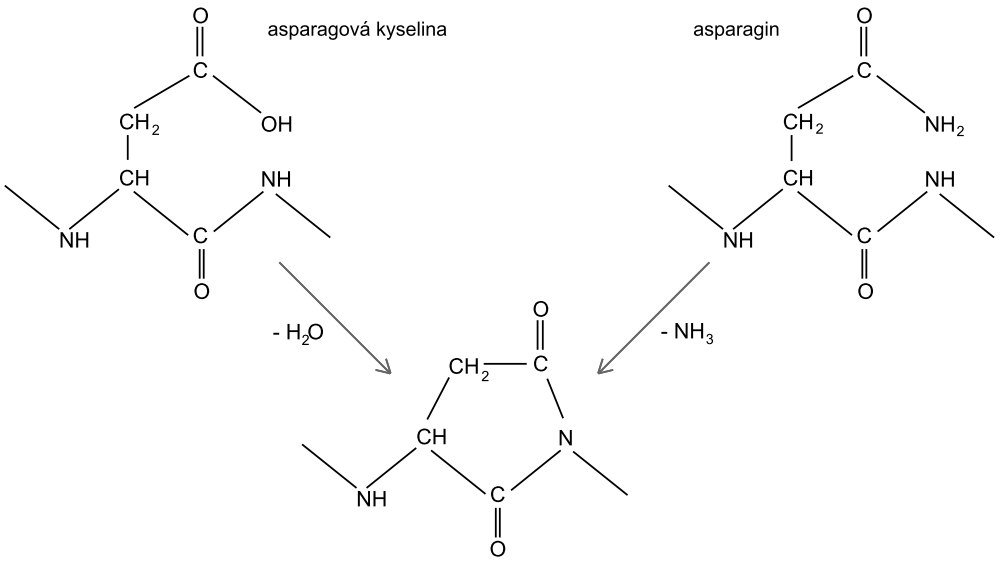


Obrázek 2.13: Typy iontů vznikající při tandemové spektrometrii

Kolizně indukovaná disociace (CID, viz sekce 2.1.5) může probíhat v závislosti na typu spektrometru buď za použití nízké nebo vysoké energie. Při nízké energii vznikají kromě uvedených iontů i jejich deriváty, které ztrácejí molekuly vody (H_2O) nebo amoniaku (HN_3). Za vysoké energie je vznik těchto derivátů spíše výjimkou, ovšem zato při ní vznikají i N-koncové ionty d,

e, f nebo C-koncové u, v, w. Jedná se o ionty odvozené z původních základních typů, u kterých došlo ke ztrátě postranního aminokyselinového řetězce nebo jeho části. Zajímavostí je, že ionty d, v nebo w mohou být využity např. k rozlišení leucinu a isoleucinu, s čímž si „jednoduchá" hmotnostní spektrometrie nedokáže poradit, protože tyto aminokyseliny mají stejnou hmotnost. Použití vysoké energie při CID je však typické zejména pro starší typy spektrometrů.

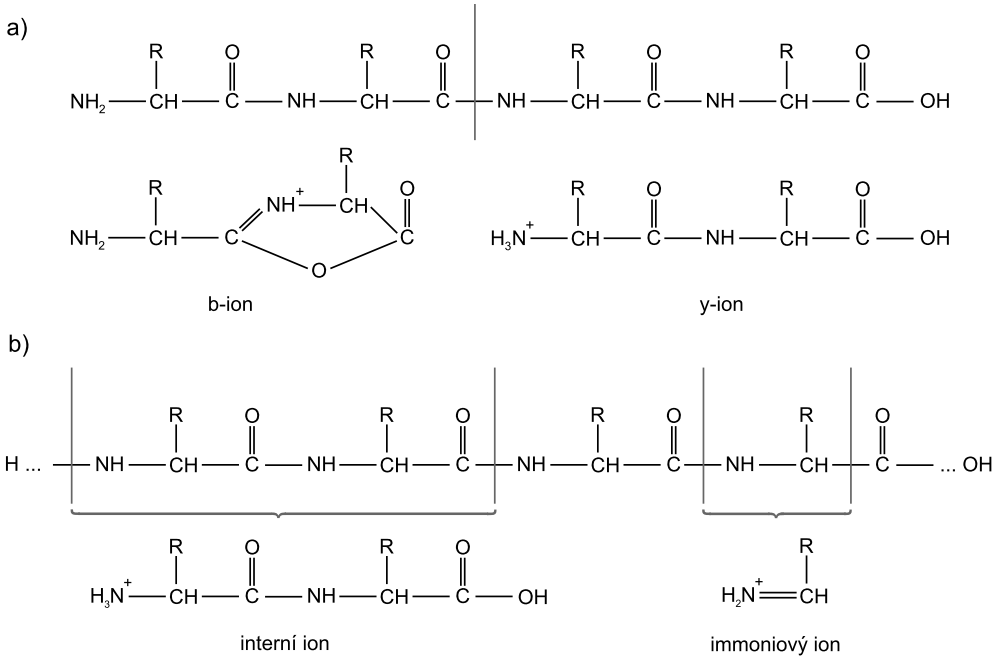
Při fragmentaci peptidu může dojít i k tomu, že je hlavní řetězec přerušen na více místech, čímž vznikají interní fragmenty. Zvláštním typem interních fragmentů jsou tzv. immoniové ionty, které jsou tvořeny pouze jedinou aminokyselinou. Výskyt těchto iontů ve spektru jednoznačně potvrzuje přítomnost dané aminokyseliny v sekvenovaném peptidu. [15]



Obrázek 2.14: Mechanismus ztráty vody a amoniaku (povšimněme si, že po dehydrataci asparagové kyseliny nebo deaminaci asparaginu nelze rozlišit o jaký aminokyselinový zbytek se původně jednalo).

Typ iontu	Kompozice iontu	Poměr m/z	Frekvence
a	$M+H-CO$	$\sum AA_i - 27.00216$	docela běžný
b	$M+H$	$\sum AA_i + 1.00794$	běžný
c	$M+H+NH+H+H$	$\sum AA_i + 18.0385$	vzácný
x	$M+OH+CO$	$\sum AA_i + 45.01744$	vzácný
y	$M+OH+H+H$	$\sum AA_i + 19.02322$	velmi běžný
z	$M+OH-NH$	$\sum AA_i + 1.99266$	velmi vzácný
dvakrát nabitý rodič	{rodičovský iont}$+H$	({m/z rodiče}+1.00794)/2	velmi běžný
třikrát nabitý rodič	{rodičovský iont}$+H+H$	({m/z rodiče}+2.01588)/3	vzácný
interní iont	$M+OH+H+H$	$\sum AA_i + 19.02322$	vzácný
immoniový iont	$M+H-CO$	$AA - 27.00216$	vzácný
a^*, b^*, y^*	{vybraný iont}$-NH_3$	{vybraný iont}−17.03056	dle typu iontu
a^o, b^o, y^o	{vybraný iont}$-H_2O$	{vybraný iont}−18.01528	dle typu iontu
d	a−{část postranního řetězce}	-	-
v	y−{celý postranní řetězec}	-	-
w	z−{část postranního řetězce}	-	-

Tabulka 2.6: Fragmentové ionty (M označuje množinu aminokyselinových zbytků (AA); frekvence je vztažena k použití nízké energie při CID; uvedeny jsou průměrné hmotnosti). [10]



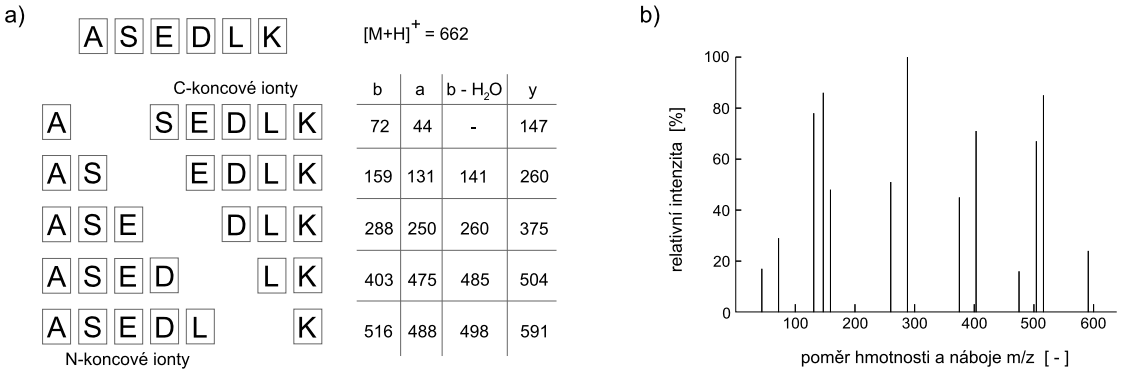
Obrázek 2.15: Vznik fragmentových iontů - a) *b* a *y* iont (při rozpadu jedenkrát protonované molekuly vzniká vždy jeden z uvedených iontů, druhý fragment je neutrální a není spektrometrem detekován), b) interní a immoniový iont.

Aminokyselina	**Značka**	**Immoniový iont [Da]**	**Odvozené ionty [Da]**
Alanin	A	44	
Arginin	R	*129*	59, 70, *73*, 87, 100, 112
Asparagin	N	*87*	*70*
Asparagová kyselina	D	*88*	*70*
Cystein	C	*76*	
Fenylalanin	F	**120**	*91*
Glutamin	Q	101	56, 84, *129*
Glutamová kyselina	E	*102*	
Glycin	G	30	
Histidin	H	**110**	*82, 121, 123, 138*, 166
Isoleucin	I	**86**	44, *72*
Leucin	L	**86**	44, *72*
Lysin	K	*101*	*70*, 84, 112, 129
Methionin	M	*104*	61
Prolin	P	**70**	
Serin	S	60	
Threonin	T	74	
Tryptofan	W	**159**	77, 117, **130**, 132, **170, 171**
Tyrosin	Y	**136**	*91, 107*
Valin	V	72	41, 55, 69

Tabulka 2.7: Hmotnosti immoniových iontů (odvozené ionty vznikají ztrátami různých částí postranních řetězců; tučně jsou zvýrazněny ionty s vysokou intenzitou, kurzívou se slabou intenzitou). [8]

2.4.3 „De Novo" peptidové sekvenování

Metoda „De Novo" (De Novo Peptide Sequencing) je založena na přímé interpretaci tandemového hmotnostního spektra a pro určení peptidové sekvence využívá grafové algoritmy. Problém identifikace peptidové sekvence převádí na vyhledávání nejdelších cest v orientovaném acyklickém grafu, který sestavujeme právě na základě experimentálního spektra. Vlastní identifikace pak probíhá pouze na základě informací o hmotnostech jednotlivých aminokyselin a s využitím znalosti struktur fragmentových iontů. Pro jednoduchost neuvažujeme intenzity peaků, hmotnosti zaokrouhlujeme na celá čísla a předpokládáme náboj $z = 1$ (resp. spektrum po dekonvoluci). [7]



Obrázek 2.16: Myoglobinový peptid „ASEDLK" a) schematické znázornění fragmentů a jejich hmotností pro nejčastější typy iontů, b) příklad hmotnostního spektra (hodnoty intenzit jsou pouze ilustrativní).

Množinu hmotností odpovídající peakům experimentálního spektra označujeme $M = \{m_1, ..., m_n\}$ a abecedou A rozumíme 20 písmen reprezentujících základní aminokyseliny (viz tabulka 2.2). Definujme zobrazení $m : A \to \mathbb{R}$, které přiřazuje každé aminokyselině její relativní molekulovou hmotnost. Rovněž předpokládejme existenci fiktivních iontů b_0 a y_0, pro které platí $m(b_0) = 1$ a $m(y_0) = 19$, a také $m(H_2O) = 18$.

Pokud peptid P reprezentujeme jako řetězec aminokyselin $a_1...a_k$, můžeme b-ionty zadefinovat zjednodušeně jako $b_j = b_0a_1...a_j$, $1 \leq j \leq k$ a y-ionty pak $y_{k-j} = a_{j+1}...a_ky_0$, $0 \leq j \leq k-1$. Pro úplnost dodejme, že fragmenty b_k a y_k jsou opět fiktivní. Ionty b_j a y_{k-j} tvoří vždy komplementární páry pro $0 \leq j \leq k$. Hmotnosti komplementárních iontů budeme označovat m^c, platí tedy $m(b_j) = m(y_{k-j})^c$ a $m(y_{k-j}) = m(b_j)^c$. Dále je důležité si uvědomit, že $m(P)$ dává pouze hmotnost aminokyselinových zbytků obsažených v peptidu P, jeho celkovou hmotnost však vypočteme jako

$$m_p = m(P) + m(H_2O) = \sum_{i=1}^{k} m(a_i) + 18, \tag{2.13}$$

a platí tedy, že

$$m(b_j) + m(y_{k-j}) = m_p + 2. \tag{2.14}$$

Definujme rozšířenou množinu hmotností experimentálního spektra $M' = \{m_{-1}, m_0, m_1, ..., m_n, m_{n+1}, m_{n+2}\}$, kde $m_{-1} = m(b_0) = 1$, $m_0 = m(y_0) = 19$, $m_{n+1} = m(b_k) = m_p - 17$ a

[7] Existují sice i různé postupy založené na metodě „De Novo", které se snaží intenzity využít statisticky, nicméně z hlediska základního principu algoritmu se jedná pouze o podružnou informaci, kterou můžeme zanedbat.

$m_{n+2} = m(y_k) = m_p + 1$. Protože experimentální spektrum nemusí být vždy úplné a může docházet k záměnám např. dvojic aminokyselin za jedinou (viz tabulka 2.3), je nutné explicitně zadefinovat množinu hmotností nejmenších a dále nedělitelných řetězců $F = \{f \in R \mid \exists_{r \in A^+}\ f = m(r) \wedge |r| \leq \gamma\}$, kde optimálně $\gamma = 2$ nebo $\gamma = 3$. Jinými slovy např. pro $\gamma = 3$ obsahuje množina F kromě hmotností jednotlivých aminokyselin také hmotnosti všech možných dvojic a trojic aminokyselin. Implicitně předpokládáme, že pro každou hmotnost v F známe původní sekvenci, ze které byla vypočtena. Množina není přímo závislá na konkrétním spektru, spíše je vhodné nastavit parametr γ v závislosti na celkové kvalitě zkoumaných spektrometrických dat.

Předpokládejme nyní, že experimentální spektrum obsahuje pouze b a y ionty. Algoritmus identifikace peptidové sekvence probíhá ve 2 fázích. V první fázi algoritmu sestavíme orientovaný acyklický graf $G = (V, E)$, kde $v \in V \Leftrightarrow v \in M' \vee v^c \in M'$ a $(v_1, v_2) \in E \Leftrightarrow v_2 > v_1 \wedge v_2 - v_1 \in F$. Ve druhé fázi hledáme řešení odpovídající nejdelším cestám. Konstrukci grafu lze vyjádřit následujícím pseudokódem, časová složitost je $O(|M'|\ log|M'|)$. [13]

Algoritmus 2.2

```
1  V = {m_{n+1}}; Q = {m_{n+1}}; E = ∅;
2  while Q ≠ ∅ do
3    v = ExtractMax(Q);
4    for each f ∈ F do
5      if ∃ m ∈ M' where v - m = f ∨ m - v^c = f then
6        v' = v - f;
7        if v' ∉ V then
8          V = V ∪ {v'};
9          Q = Q ∪ {v'};
10       E = E ∪ {(v',v)};
```

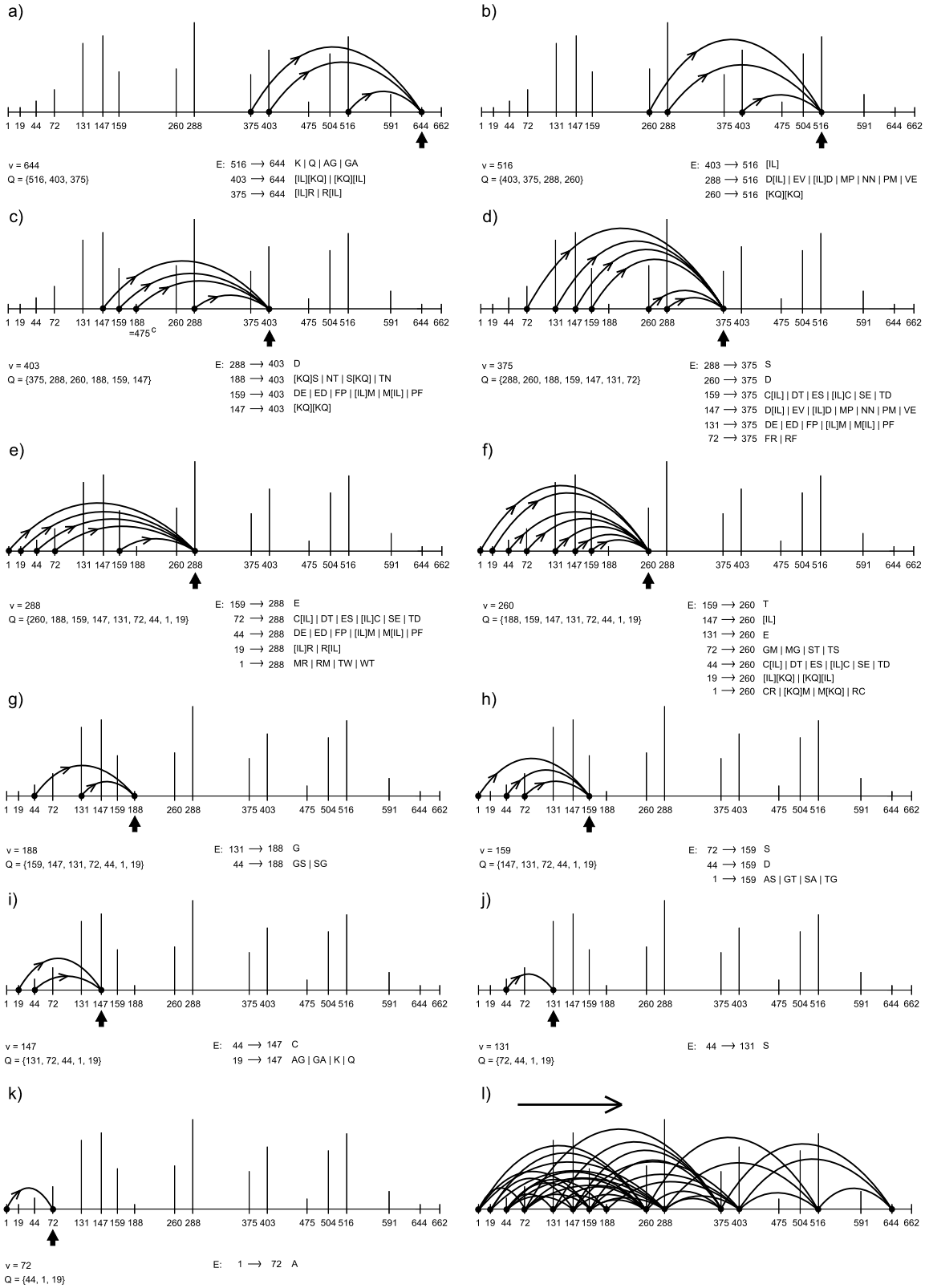
Příklad konstrukce grafu pomocí uvedeného algoritmu ilustruje obrázek 2.17, parametr $\gamma = 2$. Rozebrány jsou podrobně jednotlivé fáze algoritmu pro každou hodnotu v, která přispívá do výsledného grafu nejméně jednou hranou. Při samotné konstrukci pak můžeme postupovat buď z uzlu $m_{n+1} = m(b_k)$ do $m_{-1} = m(b_0)$ nebo z $m_{n+2} = m(y_k)$ do $m_0 = m(y_0)$. Množinu Q je nejvýhodnější reprezentovat jako prioritní frontu.

Ve druhé fázi algoritmu se snažíme z grafu vyčíst sekvenci aminokyselin odpovídající původnímu spektru. Jak je patrné už z obrázku 2.17 l), nezískáme pouze jednu konkrétní peptidovou sekvenci, ale dostaneme mnoho různých řešení. Řešením může být jakákoliv cesta začínající v uzlu m_{-1} a končící v m_{n+1} (resp. začínající v m_0 a končící v m_{n+2}). Časová složitost nalezení jednoho řešení je $O(|V|)$, nalezení všech p řešení pak $O(p|V|)$. Pro graf z obrázku 2.17 l) můžeme všechny cesty odpovídající peptidové sekvenci získat rekurzivně následujícím způsobem.

Algoritmus 2.3

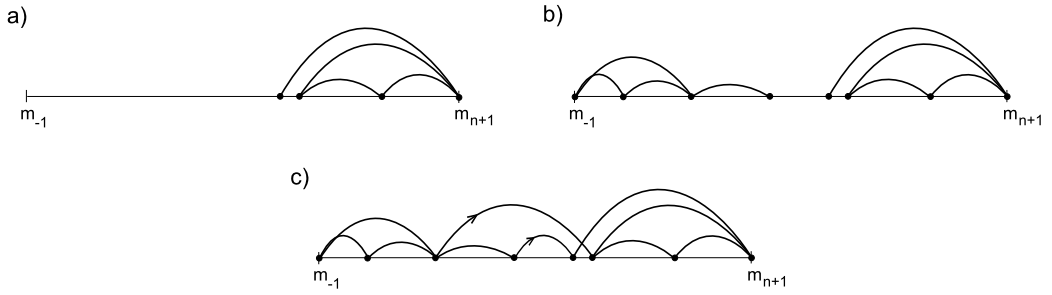
```
1  Solution(path, vertex)
2    path.Add(vertex);
3    if vertex < m_{n+1} then
4      for each (u,v) ∈ E where vertex = u do
5        Solution(path, v);
6    else path.Print();
7    path.Remove(vertex);
8  end Solution;
9
10 Solution(new Path(), m_{-1});
```

Tímto postupem dostaneme celkem 31 možných cest, přičemž každá z nich se skládá z 3 až 6 hran. Vycházíme-li z předpokladu, že experimentální spektrum bylo velmi kvalitní a vyskytovaly se v něm všechny typy b-iontů (resp. y-iontů), pak můžeme tvrdit, že nejzajímavější jsou pro nás cesty s největším počtem hran. Taková cesta je v tomto případě pouze



Obrázek 2.17: Konstrukce grafu metodou „De Novo” - a) až k) jednotlivé fáze konstrukce pro spektrum z obrázku 2.16 b) (zobrazen je aktuálně zpracovávaný uzel v, momentální stav fronty uzlů čekajících na zpracování Q a výčet hran přidávaných do výsledného grafu spolu se všemi odpovídajícími sekvencemi aminokyselin), l) kompletní graf spektra.

jedna 1 → 72 → 159 → 288 → 403 → 516 → 644 a odpovídají jí sekvence aminokyselin $ASED[I|L][AG|GA|K|Q]$. Víme, že jako štěpící enzym původního proteinu byl použit trypsin. Pokud tedy budeme předpokládat, že se nejedná o peptid z C-konce proteinu, musí posloupnost aminokyselin nutně končit K nebo R. Řešením našeho příkladu jsou tedy dvě peptidové sekvence $ASED[I|L]K$. Rozlišení $[I|L]$ je v tomto případě zcela nemožné, protože se jedná o izomery (viz sekce 2.2.2).



Obrázek 2.18: Komplikace při konstrukci grafu - a) konstrukce grafu se může „zaseknout" v případě, že ve spektru chybí některé peaky nebo se v něm vyskytují nepředvídané modifikace, b) pro řešení tohoto problému lze spustit algoritmus i „zleva-doprava", c) pokud ani obousměrné spuštění algoritmu nepomůže, doporučuje se zavést více typů iontů (množina δ; viz dále), zvětšit mohutnost množiny F nebo provést napojení nesouvislých částí grafu. [13]

Při konstrukci grafu jsme předpokládali, že spektrum obsahuje pouze b a y ionty, ve skutečnosti však obsahuje řadu různých typů iontů. Jejich existenci zohledníme tím, že budeme uvažovat množinu $\Delta = \{\delta_1, ..., \delta_k\}$, popisující relativní odchylky hmotností jednotlivých typů iontů. Odchylku obvykle vztahujeme k N-koncovým (resp. C-koncovým) fragmentům peptidu, přičemž předpokládáme jejich nulový náboj, např. $\Delta = \{b, a, b - H_2O, b - NH_3, b - H_2O - NH_3\} = \{-1, 27, 17, 16, 34\}$.

Modifikace algoritmu pak spočívá v tom, že každý uzel V_i odpovídající hmotnosti $m_i \in M$ nahradíme množinou uzlů $V_i(m) = \{m+\delta_1, ..., m+\delta_k\}$. Důležité je přitom nemíchat mezi sebou N-koncové a C-koncové typy iontů, tedy při konstrukci grafu podmínku $v-m = f \vee m-v^c = f$ nahradíme $v - m = f$ a zkonstruujeme 2 grafy pro dvě různé množiny δ, kde jedna bude odpovídat N-koncovým a druhá C-koncovým iontům. Výsledné peptidové sekvence z obou grafů pak sjednotíme, případně můžeme udělat jejich průnik a vytipovat tak pravděpodobnější řešení. Množinu M' předefinujeme na $M' = \{m_0\} \cup V_1 \cup ... \cup V_n \cup \{m_{n+1}\}$, kde $m_0 = 0$, $m_{n+1} = m_p - 18 = m(P)$ (pro N-koncové ionty) a $m_{n+1} = m_p$ (pro C-koncové ionty). Každý graf má pak nejvýše $nk + 2$ uzlů.

Identifikace peptidové sekvence je velmi často komplikována tím, že hmotnostní spektra nejsou úplná, tj. neobsahují aspoň jeden typ iontu pro všechny N-koncové nebo C-koncové fragmenty, které můžeme dělením peptidu získat. Pokud tento problém překonáme (viz obrázek 2.18) nebo pokud je původní spektrum kompletní, velmi často zjistíme, že ve výsledném grafu existuje mnoho cest, což znemožňuje spolehlivou identifikaci peptidové sekvence. Výběr nejdelší cesty v grafu totiž adekvátně neodráží význam jednotlivých uzlů, protože uzly odpovídající častějším iontům (např. b) by měly mít větší váhu než uzly pro méně obvyklé ionty (např. $b - H_2O - NH_3$), apod.

Abychom mohli jednotlivé uzly grafu ohodnotit podle jejich významu, můžeme přiřadit každému typu iontu z množiny $\Delta = \{\delta_1, ..., \delta_k\}$ pravděpodobnost jeho výskytu $\{p(\delta_1), ..., p(\delta_k)\}$. Pro jednoduchost můžeme předpokládat, že četnost výskytu daného typu iontu δ_i je nezávislá na výskytu jiného typu δ_j. Na základě vážené cesty v grafu se tedy snažíme maximalizovat pravděpodobnost, že daný peptid odpovídá zkoumanému spektru. Pokud N-koncový nebo C-

koncový fragment vytváří ionty $\delta_1, ..., \delta_l$ a nevytváří ionty $\delta_{l+1}, ..., \delta_k$, pak všech l vytvářených iontů odpovídá jednomu uzlu reprezentujícímu hmotnost tohoto fragmentu. V nejjednodušším případě takovému můžeme uzlu přiřadit ohodnocení

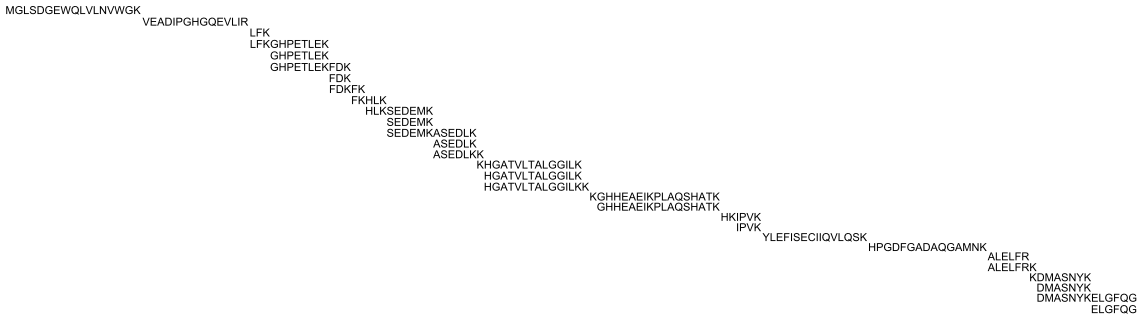
$$\left(\prod_{i=1}^{l} p(\delta_i)\right)\left(\prod_{i=l+1}^{k} (1 - p(\delta_i))\right). \tag{2.15}$$

Spektrometr však často produkuje „náhodný šum", který může na libovolné pozici ve spektru generovat peak s pravděpodobností p_R, pokud bychom tento fakt chtěli při hodnocení uzlů zohlednit, můžeme předchozí schéma upravit následujícím způsobem. [25]

$$\left(\prod_{i=1}^{l} \frac{p(\delta_i)}{p_R}\right)\left(\prod_{i=l+1}^{k} \frac{1 - p(\delta_i)}{1 - p_R}\right) \tag{2.16}$$

Komplexnější technika hodnocení založená na diskretizaci hmotností spektra je podrobně rozebrána v [3, 4].

Doposud jsme se zabývali analýzou spekter odpovídající peptidům daného proteinu. Po identifikaci všech peptidových sekvencí však chceme určit i sekvenci celého původního proteinu. Snažíme se k tomu využít redundance v podobě překrývajících se částí jednotlivých peptidů. Přitom však mohou vznikat další nejednoznačnosti a chyby vyplývající z nedostatečného překryvu. Určitou alternativou „De Novo" je proto metoda Sequence Tag (viz sekce 2.4.4), při které se z hmotnostního spektra snažíme vyčíst „zřejmé řetězce aminokyselin" a následně provádíme dohledání proteinových sekvencí v databázi.



Obrázek 2.19: Sestavení proteinové sekvence z jednotlivých peptidů.

Výhodou metody „De Novo" je fakt, že dokáže identifikovat i sekvence, které doposud nejsou uvedeny v žádné databázi. Podstatnou nevýhodou je potom počet řešení, protože danému spektru může odpovídat mnoho různých peptidových sekvencí. Technika „De Novo" je proto v praxi velmi neefektivní a její běžně dostupné implementace dokáží správně identifikovat obvykle méně než 30% sekvencí. [4] V následujícím přehledu je uvedeno několik známých implementací.

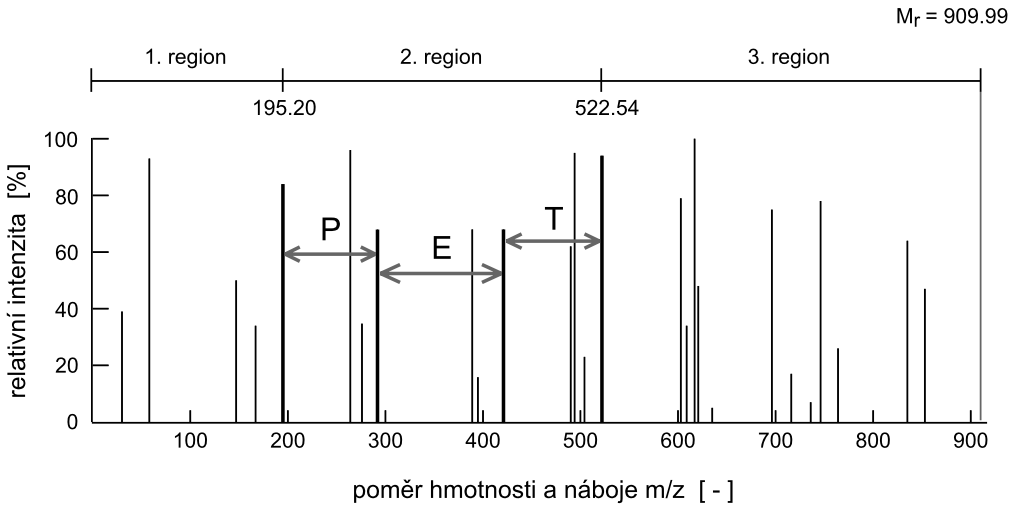
- Lutefisk
  `http://www.hairyfatguy.com/lutefisk/`
- Peaks
  `http://www.bioinformaticssolutions.com/products/peaks/index.php`
- Pepnovo
  `http://proteomics.bioprojects.org/Software/PepNovo.html`

Běžné implementace obvykle nevyužívají přesné hmotnosti aminokyselin a iontů, ale berou ohled na chybovost a pracují i s určitými odchylkami hodnot. Rovněž existují vylepšení snažící se „naučit" vyskytující se typy iontů a vytvořit si statistiky intenzit z konkrétního vzorku hmotnostních spekter. [3]

2.4.4 Identifikace s využitím tagů (Sequence Tag)

Technika Sequence Tag se snaží kombinovat výhody přímé interpretace spektra a vyhledávání v databázích. Z daného spektra peptidu se nejprve podle vzdáleností peaků snažíme vytipovat část sekvence (tag) a následně provádíme dohledání celých proteinových sekvencí v databázi. Pro definování tagu můžeme využít postupy založené na metodě „De Novo" nebo jej můžeme určit i „ručně" na základě vizuální interpretace části sousedících peaků. Právě kvůli snadným manuálním úpravám, které mohou výrazně zlepšit kvalitu výstupu, je tato metoda poměrně populární. [21]

Tag bývá obvykle zapisován jako trojice (někdy čtveřice) údajů. Nejdříve je uvedena hmotnost odpovídající peaku, který definuje začátek sekvence tagu, následuje tag samotný a za ním je uvedena hmotnost pro peak, kterým tag končí. Nezbytným údajem při identifikaci sekvence je pak i hmotnost původního neutrálního peptidu, která bývá v závislosti na konkrétní implementaci brána buď jako externí parametr nebo předchází trojici údajů a je uvedena jako první část tagu. Pro myoglobinový peptid „GHPETLEK" (viz obrázek 2.20) s průměrnou hmotností 909.99 Da, pak může zápis tagu vypadat např. „(195.20)PET(522.54)" nebo „909.99 tag(195.20,PET,522.54)". Druhou variantu zápisu používá webový vyhledávač Mascot (viz obrázek 2.22), který na rozdíl od jiných portálů, umožňuje při vyhledávání proteinu zadat i více než jen jeden peptidový tag. Vlastní sekvence tagu může být tvořena i jedinou aminokyselinou, pro jednoznačnou identifikaci proteinu je však obvykle vhodné, když obsahuje nejméně tři aminokyseliny.

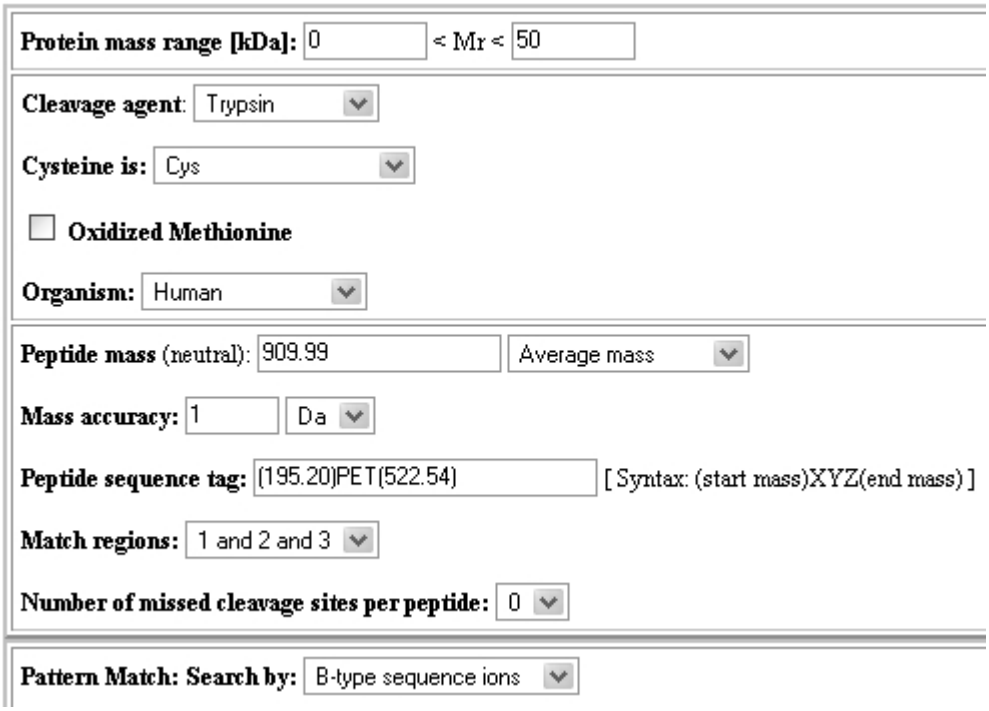


Obrázek 2.20: Sequence Tag - po vytipování posloupnosti aminokyselin je peptidové spektrum zredukováno na tři regiony, prostřední část odpovídá sekvenci tagu, krajní části určují rozdíly hmotností.

Základní princip metody je podobný jako u peptidového mapování (viz sekce 2.3.2), kdy procházíme databázi proteinů a s využitím znalosti specificity štěpícího enzymu (např. trypsinu) je dělíme na peptidy. Sequence Tag však nekončí tím, že vybere peptidy, jejichž teoretická hmotnost odpovídá experimentálním hodnotám, ale pouze si tímto způsobem předfiltruje sekvence, které se následně snaží spárovat se zadaným tagem nebo množinou tagů. U metod PMF i

Sequence Tag se nabízí možnost vytvoření databáze peptidů s ohledem na specifický štěpící enzym a zavedení indexu nad jejich hmotnostmi. Problematiku využití indexů pro tyto metody se však v dostupné literatuře nepodařilo nalézt.

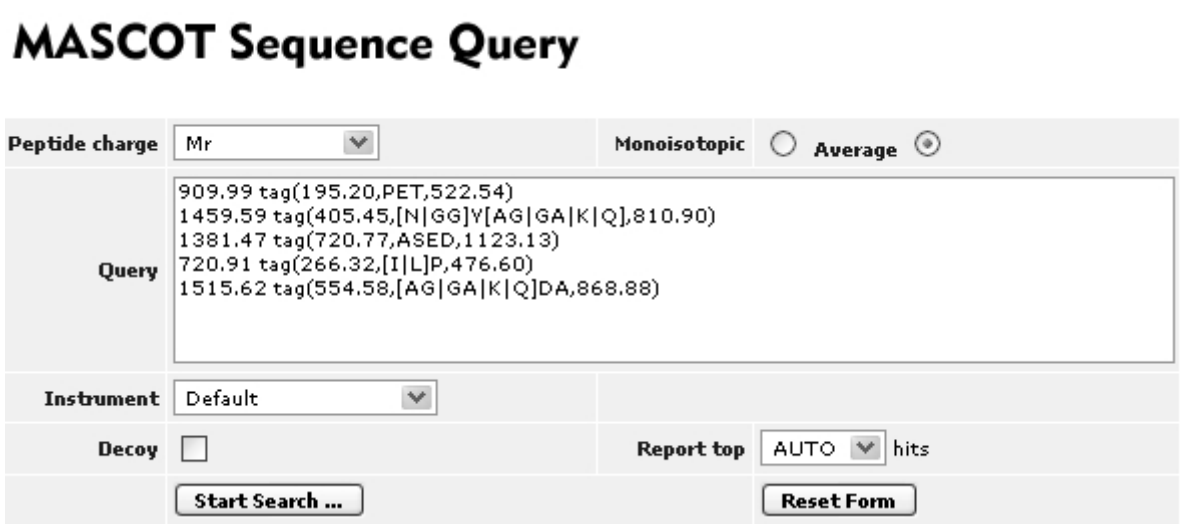
Uvažujeme-li např. tag „(195.20)PET(522.54)", musíme si uvědomit, že peaky, na jejichž základě jsme posloupnost aminokyselin určili, mohou odpovídat N-koncovým nebo C-koncovým fragmentům. Nalezená peptidová sekvence pak může obsahovat buď podřetězec „PET" nebo reverzní „TEP".



Obrázek 2.21: Vyhledávač PeptideSearch (aplikace umožňuje vyhledávání pouze s použitím jednoho tagu; zobrazena je část formuláře).

Pokud analyzovaný peptid obsahuje modifikace, může se stát, že není v databázi nalezen. Většina aplikací proto umožňuje nastavit, které části tagu budou při vyhledávání použity. Obvykle se vyřazuje region odpovídající hmotnosti, kterou tag začíná nebo končí. Uveďme několik veřejně dostupných webových aplikací podporujících metodu Sequence Tag.

- Mascot
  `http://www.matrixscience.com/search_form_select.html`
- PeptideSearch
  `http://www.narrador.embl-heidelberg.de/GroupPages/PageLink/peptidesearchpage.html`
- ProteinProspector MS-Seq
  `http://prospector.ucsf.edu/`



Obrázek 2.22: Vyhledávač Mascot (aplikace umožňuje vyhledávání proteinů s využitím více tagů; sekvence aminokyselin mohou být zadány ve formě regulárních výrazů; zobrazena je pouze část formuláře). [8]

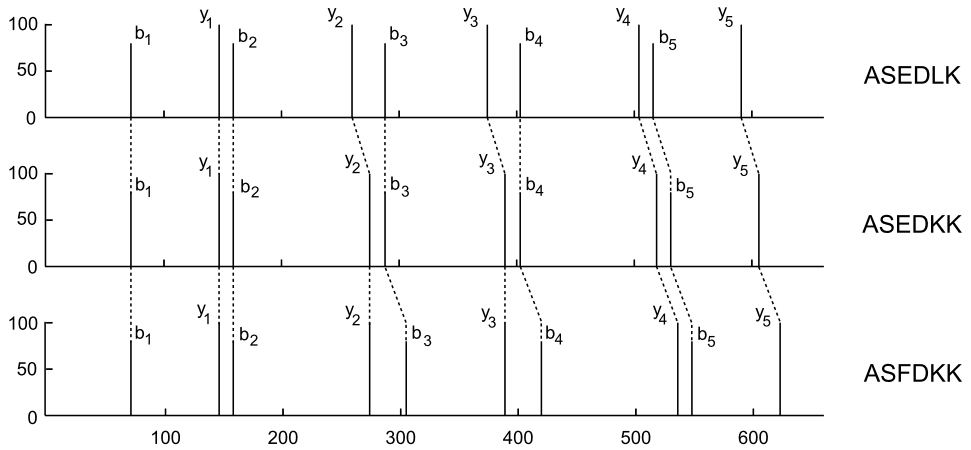
2.4.5 Metoda fragmentového mapování (PFF)

Při použití metody fragmentového mapování (Peptide Fragment Fingerprinting, PFF) porovnáváme experimentální spektra peptidů s teoretickými spektry generovanými z databáze proteinových sekvencí. Protože na rozdíl od metody peptidového mapování (PMF; viz sekce 2.3.2) máme místo jednoho spektra k dispozici sadu fragmentových spekter (viz obrázek 2.7), je identifikace tímto způsobem mnohem přesnější. Čím více spekter se nám podaří spárovat, tím je pravděpodobnější, že analyzovaný protein odpovídá dané sekvenci. Pomocí PMF se nám v extrémním případě nemusí podařit přiřadit ani jeden peak odpovídajícímu peptidu, zatímco u PFF máme pro stanovení podobnosti fragmentového spektra s daným peptidem mnohem více možností. [21]

Před vyhledáváním nejprve specifikujeme enzym, který byl použit při štěpení proteinové sekvence. Některé aplikace mají databázi pro jednotlivé enzymy předindexovanou podle hmotností peptidů, čímž je vlastní hledání výrazně rychlejší. Stinnou stránkou tohoto přístupu však může být fakt, že při každé změně v databázi potřebujeme opravit indexy pro všechny enzymy.

Vyhledávání zahájíme tím, že z databáze vybereme ty proteinové sekvence, které po rozdělení specifikovaným enzymem vytvářejí peptidy, jejichž hmotnosti s danou odchylkou odpovídají hmotnostem peptidů, pro které máme k dispozici experimentální spektra. Připomeňme, že pro každé fragmentové spektrum máme k dispozici doplňující informace o hmotnosti a náboji původního peptidového iontu, z něhož bylo vytvořeno. Získáme tedy seznam peptidů, pro které generujeme teoretická spektra jejich fragmentů. Čím menší odchylku hmotností povolíme, tím bude seznam kratší a hledání rychlejší, protože budeme srovnávat méně spekter. Tolerance však zase nesmí být moc těsná, abychom vzhledem k přesnosti spektrometru, zastoupení různých izotopů jednotlivých prvků nebo proteinovým modifikacím neodfiltrovali i potencionální řešení.

Po porovnání teoretických a experimentálních spekter jsou peptidy ohodnoceny podle toho, kolik fragmentových peaků si vzájemně odpovídalo. Pro výstup tedy vybíráme takové proteinové sekvence, které obsahují nejvíce „dobře spárovaných" peptidů. Ve výpisu je pak obvykle uvedeno několik nejpravděpodobnějších řešení, aby měl uživatel, vzhledem k možným komplikacím vznikajícím nejednoznačnostmi a nepřesnostmi při automatickém zpracování dat, možnost posoudit relevanci jednotlivých přiřazení. Techniky používané pro porovnávání spekter si rozebereme podrobněji.



Obrázek 2.23: Počet sdílených peaků ($SPC_{max} = 10$; peptidy „ASEDLK" a „ASEDKK" lišící se v jedné aminokyselině mají $SPC = 5$; „ASEDLK" a „ASFDKK" lišící se dvěma mutacemi mají $SPC = 3$).

Při analýze sekvencí můžeme vycházet z předpokladu, že podobné peptidy mají podobné sekvence. Pokud však chceme interpretovat tandemová hmotnostní spektra, brzy zjistíme, že podobné peptidové sekvence mohou mít spektra velmi odlišná. Triviální způsob porovnání dvou spekter spočívá ve zjištění počtu peaků, které se vyskytují na odpovídajících si pozicích v každém z nich. Počet sdílených peaků SPC však velmi rychle klesá s rostoucím počtem modifikací k peptidové sekvence. [8] Modifikací v tomto kontextu můžeme rozumět jak záměnu jedné aminokyseliny za jinou (mutace), tak i strukturní modifikaci stejné aminokyseliny (viz tabulka 2.4), oba dva případy se ve výsledku projeví „posunem" vybraných peaků. Schopnost rozlišit podobnost dvou peptidů s využitím hodnoty SPC je špatná pro $k = 1$ a téměř nepoužitelná pro $k > 1$.

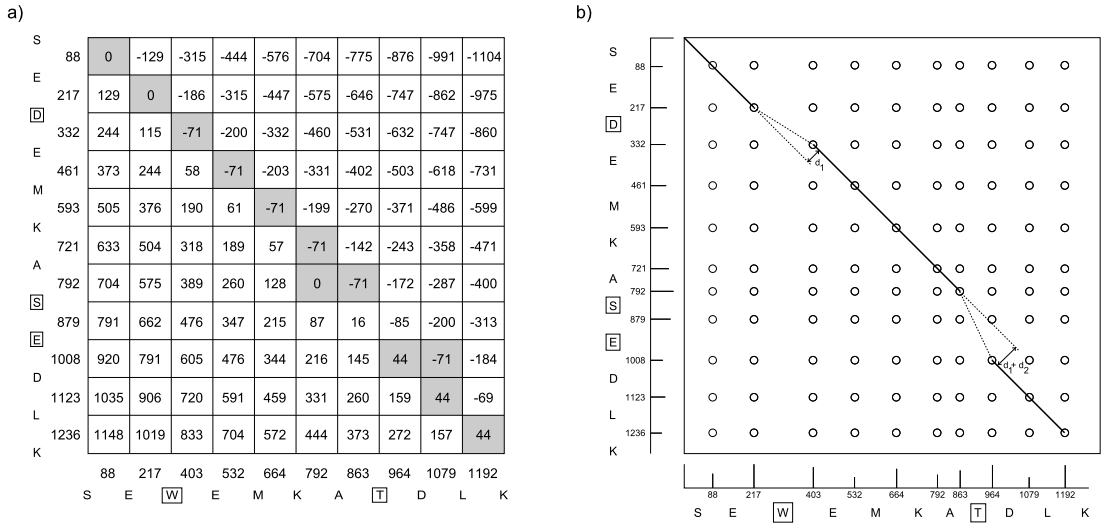
Posuny peaků se snaží zohlednit algoritmus spektrální konvoluce. Konvoluci spekter S_1 a S_2 definujeme jako multimnožinu $S_2 \ominus S_1 = \{s_2 - s_1 : s_1 \in S_1, s_2 \in S_2\}$, pro níž funkce $(S_2 \ominus S_1)(x)$ vrací násobnost prvku x čili počet dvojic (s_1, s_2), kde $s_2 - s_1 = x$. Platí, že $(S_2 \ominus S_1)(0) = SPC$. Multimnožinu $S_2 \ominus S_1$ můžeme rovněž reprezentovat jako matici o rozměrech $|S_1| \times |S_2|$, viz obrázek 2.24 a).

Jestliže se peptidy P_1 a P_2, odpovídající spektrům S_1 a S_2, liší mutací jedné aminokyseliny ($k = 1$), pak rozdíl jejich hmotností odpovídá $\delta = m(P_2) - m(P_1)$ a v multimnožině $S_2 \ominus S_1$ se kromě jiných vyskytují hodnoty $x = 0$ a $x = \delta$. Pokud se v peptidové sekvenci P délky n vyskytuje mutace na pozici t, pak sdílené peaky ($x = 0$) odpovídají N-koncovým iontům N_i pro $i < t$ a C-koncovým C_i pro $i \leq n - t$. Hodnota posunutí $x = \delta$ potom přísluší iontům N_i pro $i \geq t$ a C_i pro $i > n - t$. Přičemž platí, že $0 < i < n$ a $P = N_i C_{n-i}$ (viz obrázek 2.23).

Pro $k = 1$ jsou hodnoty $x = 0$ a $x = \delta$ v $S_2 \ominus S_1$ poměrně frekventované. Uvažujeme-li mutaci dvou různých pozic, pak se v ní často vyskytují i hodnoty $x = \delta'$ a $x = \delta - \delta'$, atd. Např. pro $k = 2$, $\delta = 80$ a $\delta' = 50$ tedy můžeme očekávat častý výskyt hodnot $x \in \{0, 80, 50, 30\}$. Podobnost tandemových spekter je proto obvykle vhodné zadefinovat na základě součtu několika nejfrekventovanějších hodnot x v $S_2 \ominus S_1$.

Spektrální konvoluce má však určitá omezení. Nechť zkoumanému peptidu P odpovídá spektrum $S = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$, přičemž pro jednoduchost uvažujeme pouze vznik b-iontů. Dále předpokládejme, že v databázi nalezneme peptidy P' a P'', kterým odpovídají spektra $S' = \{100, 200, 300, 400, 500, 550, 650, 750, 850, 950\}$ a $S'' = \{100, 150, 300, 350, 500, 550, 700, 750, 900, 950\}$. Chceme-li zjistit zda spektru S odpovídá lépe S'

[8] SPC = shared peak count (z angl.)



Obrázek 2.24: Spektrální konvoluce a alignment - příklad přiřazení b-iontových sérií dvou peptidů, a) konvoluce spekter s vyznačenými hodnotami, které se vyskytují více než 2×, b) alignment spekter - kroužky označují pozice v matici $a_n \times b_m$, kterým odpovídají nenulové hodnoty (resp. jedničky v případě spektrálního součinu).

nebo S'', stanovení počtu sdílených peaků nám nic neřekne, neboť pro oba případy je $SPC = 5$. Spektrální konvoluce nám rovněž nepomůže, protože $S \ominus S'$ i $S \ominus S''$ obsahují hodnoty $x = 0$ a $x = 50$ se stejnou frekvencí výskytu a navíc naznačuje, že oba peptidy P' i P'' můžeme získat jedinou mutací s hmotnostním rozdílem 50 Da. Důkladnější analýzou však zjistíme, že tento závěr platí pouze pro P' a nikoliv pro P'', což je dáno tím, že S' na rozdíl od S'' obsahuje shluk „posunutých" peaků. Uvedenému problému se snaží předejít algoritmus spektrálního zarovnávání (spectral alignment). [25]

Nechť $A = \{a_1, ..., a_n\}$ je uspořádaná množina celých čísel tak, že $a_i < a_{i+1}$ a nechť Δ_i transformuje všechny hodnoty této množiny kromě $i-1$ prvních prvků na $A = \{a_1, ..., a_{i-1}, a_i + \Delta_i, ..., a_n + \Delta_i\}$. Předpokládejme přitom, že pokud je posunutí Δ_i záporné, nemění pořadí prvků a platí tedy, že $a_i + \Delta_i \geq a_{i-1}$. Podobnost mezi dvěma danými množinami A a B pak definujeme jako $D(k)$, které je rovno maximálnímu počtu prvků průniku, přičemž jedna z množin může být transformována nejvýše k posunutími. Platí, že $D(0) = SPC$. Pro množiny S a S' je $D(1) = 10$, pro S a S'' pak $D(1) = 6$.

Množiny $A = \{a_1, ..., a_n\}$ a $B = \{b_1, ..., b_m\}$ můžeme reprezentovat jako pole hodnot $\{0, 1\}$, kde pole a má délku a_n a b délku b_m. Pokud pozice prvku v poli odpovídá hodnotě z příslušné množiny, nabývá prvek hodnoty 1, v opačném případě pak 0. Např. pro množinu $A = \{1, 3, 7, 10\}$ je tedy $a = [1, 0, 1, 0, 0, 0, 1, 0, 0, 1]$. V takovém modelu dat pak posunutí $\Delta_i < 0$ odpovídá jednoduše odstranění (deleci) Δ_i nul v daném poli a posunutí $\Delta_i > 0$ znamená vložení (inserci) Δ_i nul. Problém určení podobnosti dvou spekter pak můžeme převést na hledání editační vzdálenosti mezi sekvencemi a a b, kde jako elementární operace používáme deleci nebo inserci bloku nul.

Pojmem spektrální součin $A \otimes B$ rozumíme matici $a_n \times b_m$ prvků obsahující nm jedniček na pozicích $[a_i, b_j]$, kde $a_i \in A \wedge b_j \in B$, ostatní prvky jsou nulové. Počet 1 na hlavní diagonále je tedy roven hodnotě $D(0)$. Podobnost $D(k)$ pro $k > 0$ mezi dvěma spektry pak odpovídá maximálnímu počtu 1 na cestě vedoucí skrze matici $A \otimes B$ po nejvýše $k + 1$ diagonálách, viz obrázek 2.24 b).

Popíšeme si algoritmus spektrálního zarovnávání založený na dynamickém programování. Uvedená varianta se zaměřuje pouze na výpočet podobnosti spekter $D(k)$ a předpokládá exis-

tenci jednoho typu N-koncových fragmentů (b-iontů). Při implementaci ukládáme matice velikosti $n \times m$, hovoříme-li však v tomto kontextu o diagonálách matice, máme vždy na mysli hlavní diagonálu a její rovnoběžky v matici o velikosti $a_n \times b_m$ prvků, která obsahuje hodnoty na pozicích $[a_i, b_j]$ pro $a_i \in A$ a $b_j \in B$, ostatní prvky jsou nulové.

Algoritmus 2.4

```
1  A = {a1,...,an} where ai < ai+1;
2  B = {b1,...,bm} where bj < bj+1;
3  D = array[numberOfShifts+1] of Matrix(n,m);
4  for k = 0 to numberOfShifts do
5    D[k].ZeroFill();
6    for i = 1 to n do
7      for j = 1 to m do
8        if i = 1 or j = 1 then D[k].Set(i,j,1);
9        else
10         for i* = 1 to i-1 do
11           for j* = 1 to j-1 do
12             if ai - ai* = bj - bj* then x = D[k].Get(i*,j*)+1;
13             else
14               if k > 0 then x = D[k-1].Get(i*,j*)+1;
15               else x = 1;
16             if x > D[k].Get(i,j) then D[k].Set(i,j,x);
17 if numberOfShifts = 0 then return D[0].Spc();
18 return D[numberOfShifts].Max();
```

Algoritmus prochází body matice $D_{nm}(k)$ a pro každý z nich následně body podmatice $D_{ij}(k)$. Pokud pro daný bod $[i^*, j^*]$ podmatice, kde $i^* < i$ a $j^* < j$, platí rovnost $a_i - a_{i^*} = b_j - b_{j^*}$ (tj. v reprezentaci matice podle obrázku 2.24 b) body leží na stejné diagonále), vybere hodnotu $D_{i^*j^*}(k)$ (posun po stejné diagonále), v opačném případě pak hodnotu $D_{i^*j^*}(k-1)$ („skok" na rovnoběžnou diagonálu). Ze všech vybraných hodnot pro danou podmatici je následně stanoveno maximum, které je po inkrementaci uloženo na pozici $[i, j]$ v matici $D_{nm}(k)$.

Podobnost spekter $D(k)$ odpovídá pro $k > 0$ hodnotě maximálního prvku z celé matice $D_{nm}(k)$. Pro $k = 0$ je hodnota $D(0) = SPC$ rovna maximální hodnotě prvku na hlavní diagonále ovšem v myšlené reprezentaci matice s nulami. Časová složitost uvedeného algoritmu pro $m = n$ je $O(n^4 k)$. Vzhledem k tomu, že algoritmus je v praxi poměrně pomalý, zmíníme ještě alternativní řešení s časovou složitostí $O(n^2 k)$.

Algoritmus 2.5

```
1  A = {a1,...,an} where ai < ai+1;
2  B = {b1,...,bm} where bj < bj+1;
3  D = array[numberOfShifts+1] of Matrix(n,m);
4  for k = 0 to numberOfShifts do
5    D[k].ZeroFill();
6    for i = 1 to n do
7      for j = 1 to m do
8        p = Diag(i,j);
9        if p.x = 0 or p.y = 0 then x = 1;
10       else x = D[k].Get(p)+1;  // posun po stejné diagonále
11       D[k].Set(i,j,x);
12       if i > 1 and j > 1 and k > 0 then
13         x = D[k-1].RecurrenceMax(i-1,j-1)+1;  // skok na rovnoběžnou diagonálu
14         if x > D[k].Get(i,j) then D[k].Set(i,j,x);
15 if numberOfShifts = 0 then return D[0].Spc();
16 return D[numberOfShifts].Max();
```

Princip algoritmu je analogií předchozího případu, obsahuje pouze určitá implementační vylepšení. Funkce $Diag(i, j)$ vrací pozici předcházející 1 ležící při spektrálním součinu na stejné

diagonále jako bod $[a_i, b_j]$, pokud neexistuje dává $[0, 0]$. Její implementace je možná při zachování časové složitosti $O(n^2)$, podrobněji viz [9]. Metoda $RecurrenceMax(i, j)$ získává, na rozdíl od předchozí varianty algoritmu, hodnoty z matice $D_{nm}(k-1)$ pro „skok" na rovnoběžnou diagonálu pomocí předvýpočtu.

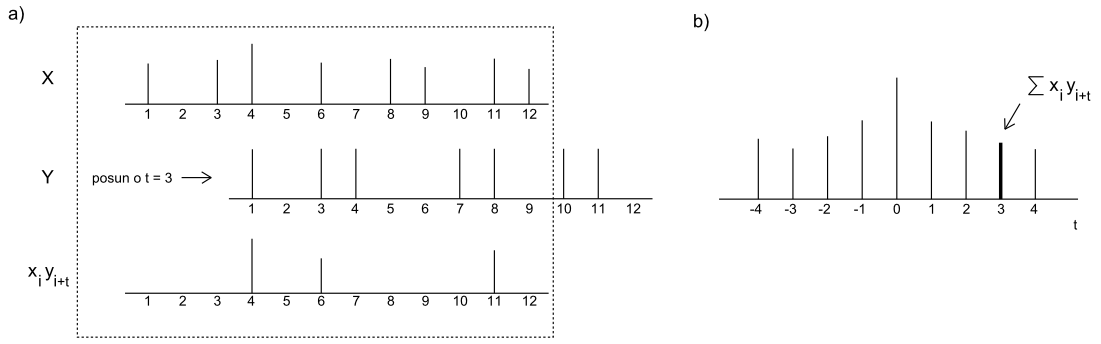
Algoritmus 2.6

```
1  Matrix.RecurrenceMax(x,y)
2    max = matrix_{x,y};
3    if x > 1 and matrix_{x-1,y} > max then max = matrix_{x-1,y};
4    if y > 1 and matrix_{x,y-1} > max then max = matrix_{x,y-1};
5    return max;
6  end RecurrenceMax;
```

Popsané algoritmy skrývají mnoho detailů, které činí ze spektrálního zarovnávání poměrně obtížný problém. Skutečné tandemové spektrum totiž obsahuje dvě prolínající se řady hodnot, kdy jedna odpovídá různým N-koncovým a druhá C-koncovým fragmentům. Kromě hlavní diagonály pro N-koncové ionty, tak vstupuje do hry i vedlejší diagonála a její rovnoběžky pro C-koncové ionty. Analýza proteinových modifikací s využitím hmotnostní spektrometrie tak stále zůstává otevřeným problémem, který doposud ještě není zcela vyřešen. [25]

2.4.6 Aplikace podporující PFF

V současné době existuje řada aplikací využívajících pro identifikaci peptidů neinterpretovaná tandemová spektra. Nejznámější z nich je zřejmě program SEQUEST. Přestože se jedná o poměrně starý produkt (1993), je pouze s malými obměnami používán dodnes. Jeho nevýhodou je však dostupnost, protože se nejedná o volně šiřitelný program.



Obrázek 2.25: Křížová korelace programu SEQUEST - a) součin vektorů X a Y po složkách pro $t = 3$, b) korelační spektrum pro $t \in \langle -4, 4 \rangle$.

SEQUEST nejprve každé experimentální fragmentové spektrum zjednoduší, aby omezil možnost nesprávné interpretace a pro jeho další zpracování předurčí maximálně 200 peaků. Z databáze pak vybere na základě specificity štěpícího enzymu a hmotnosti peptidu, pro který bylo experimentální spektrum naměřeno, až 500 odpovídajících peptidových sekvencí. Pro každou z nich následně generuje podle specifických pravidel teoretické fragmentové spektrum, které porovnává s experimentálním. Jednotlivým peptidům je přitom přiřazeno skóre

$$S_p = \left(\sum i_m\right)\left(\frac{n_i}{n_t}\right)(1+\beta)(1+\rho)\,, \tag{2.17}$$

kde $\sum i_m$ je součet intenzit spárovaných peaků, n_i jejich počet, n_t je celkový počet peaků, faktor β zohledňuje výskyt souvislých fragmentových sérií a ρ je bonus za identifikované immoniové ionty (viz tabulka 2.7). Iniciální hodnota $\beta = 0$ je zvýšena o malý přírůstek pokaždé, když jsou úspěšně spárovány peaky odpovídající b nebo y iontu. Podobně je upravována hodnota ρ pro immoniové ionty. Vyšší hodnota skóre znamená lepší výsledek, přičemž dělení hodnotou n_t zabraňuje nepřiměřenému růstu skóre pro dlouhé peptidové sekvence (resp. seznamy peaků). [5]

Kromě hodnoty S_p používá SEQUEST ještě druhý typ skórování tzv. křížovou korelaci (cross-correlation). Experimentální spektrum je nejprve normalizováno, dojde k odstranění peaků s nízkou intenzitou a hodnoty m/z jsou zaokrouhleny na nejbližší vyšší celé číslo, označme jej jako spektrum X. Pro každou peptidovou sekvenci vybranou z databáze k danému experimentálnímu spektru, je pak za pomoci jednoduchých pravidel vygenerováno teoretické spektrum Y. Skóre X_{corr} vypočteme za pomoci korelační funkce $Corr(t)$, která odpovídá součinu vektorů $X \times Y$, přičemž vektor Y je vůči X posunut o t hmotnostních jednotek.

$$Corr(t) = \sum_{i=1}^{n} x_i y_{i+t} \tag{2.18}$$

$$X_{corr} = Corr\,(0) - avg\,(\langle Corr\,(-t)\,, Corr\,(t)\rangle) \tag{2.19}$$

Funkce $avg\,()$ průměruje hodnoty intervalu diskrétních hodnot. Pro rovnici 2.19 je implicitně $t = 75$. Hodnota X_{corr} má větší vypovídací váhu o podobnosti spekter než S_p, její výpočet je však časově náročnější, a protože nenormalizuje počty spárovaných peaků je zároveň závislá na délce peptidu. Ve výstupu jsou preferovány ty peptidy, pro něž jsou hodnoty X_{corr} a S_p nejvyšší. [5, 29].

Volně přístupnou webovou aplikací pro identifikaci peptidů z tandemových spekter je vyhledávač Mascot, který využívá pravděpodobnostní skórování založené na algoritmu MOWSE (viz sekce 2.3.3). Tento algoritmus je sice typický pro metodu PMF (viz kapitola 2.3.2), nicméně Mascot jej analogicky implementuje i pro metodu fragmentového mapování. Vyhledávací formulář (viz obrázek 4.31) je pak podobný jako u peptidového mapování s tím, že informace o spektrech jsou načítány výhradně z externího souboru.

Mascot podporuje několik formátů vstupních souborů s naměřenými hodnotami z MS/MS experimentu, popíšeme si zde dva velmi běžné typy. Formát *.dta (SEQUEST formát) obsahuje na prvním řádku hmotnost jedenkrát protonovaného peptidového iontu $[M + H]^+$, pro který bylo fragmentové spektrum naměřeno, následovanou skutečně naměřenou hodnotou náboje z. Další řádky pak obsahují vždy poměr m/z a intenzitu odpovídající danému fragmentovému iontu. Jednotlivá tandemová spektra jsou pak v souboru oddělena prázdnými řádky.

a)

```
1999    2
147.2   76.03
159.16  82.75
260.36  74.3
...
```

b)

```
BEGIN IONS
PEPMASS=1000
CHARGE=2+
TITLE=nazev
147.2          76.03
159.16         82.75
260.36         74.3
...
END IONS
```

Obrázek 2.26: Formáty vstupních souborů - a) struktura *.dta, b) *.mgf; hmotnost původního neutrálního peptidu je v obou případech $M_r = 1998$, viz rovnice 2.4.

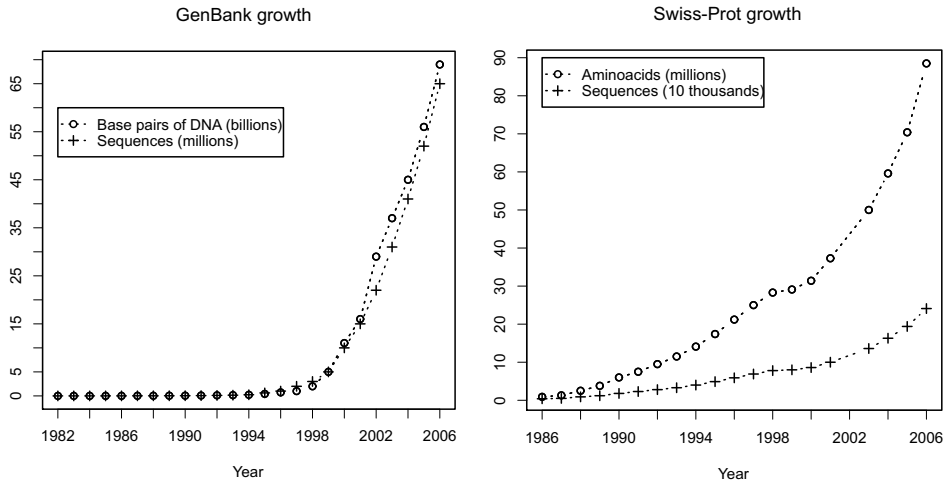
Formát *.mgf (Mascot Generic Format), jehož základní schéma je zobrazeno na obrázku 2.26 b), nabízí mnohem širší možnosti, viz [8]. Důležitým rozdílem oproti *.dta je, že pro původní peptid uvádíme skutečný poměr m/z.

Na závěr uveďme základní výčet aplikací pro metodu fragmentového mapování. Kromě programu SEQUEST se jedná o veřejně dostupné webové portály. Ukázku vyhledávače ProteinProspector prezentuje obrázek 4.32.

- SEQUEST
  `http://fields.scripps.edu/sequest/`
- Mascot
  `http://www.matrixscience.com/search_form_select.html`
- X! Tandem
  `http://www.thegpm.org/tandem/index.html`
- OMSSA
  `http://pubchem.ncbi.nlm.nih.gov/omssa/`
- ProteinProspector MS-Tag
  `http://prospector.ucsf.edu/`

3 Metrické indexovací metody

Bioinformatické databáze se vzhledem k rychlému rozvoji moderních analytických metod každoročně plní téměř exponenciálně rostoucím množstvím dat (viz obrázek 3.1). Při vyhledávání je potom z časového hlediska nevhodné procházet sekvenčním způsobem všechny uložené záznamy. Pro řešení tohoto problému byly vytvořeny indexovací metody, které nám díky logaritmické složitosti umožňují časovou náročnost vyhledávání výrazně snížit.



Obrázek 3.1: Exponenciálně rostoucí množství dat v bioinformatických databázích - příklad pro databáze GenBank (DNA) a Swiss-Prot (proteiny). [23]

Rozebereme si podrobně metrické indexovací metody M-strom a PM-strom (viz sekce 3.3 a 3.4), v nichž vlastnost metriky (viz sekce 3.1.1) umožňuje rychle prořezávat větve stromu a tím filtrovat vyhledávací prostor. Metrické metody se obvykle používají pro indexování multimediálních dat jakými jsou obrázky, videa, dokumenty, časové řady, biologická data, apod. Oproti klasickým databázím je zde hlavní rozdíl v tom, že obvykle přesně nevíme jakým způsobem kvantifikovat, které záznamy jsou při výběru (selekci) z databáze pro nás ještě relevantní a které už nikoliv.

Vlastní data je tak vhodné popsat vektorem hodnot, který co možná nejlépe vystihuje jejich vlastnosti (extrakce vlastností). Např. obrázek tak můžeme vyjádřit pomocí vektoru založeném na histogramu barev a vyhledávat podle něj. Takový vektor nám však nezaručuje, že na vybraných obrázcích budou nutně zobrazeny stejné či podobné objekty, pouze víme, že obrázky budou koncipovány z podobných barev. Provedeme-li tedy selekci podle tohoto vektoru, ve výsledku můžeme získat i obrázky, které neodpovídají našim požadavkům a naopak nemusíme nalézt ty, které představují podobné objekty v odlišném barevném složení.

Klíčovým problémem v případě hmotnostní spektrometrie je tedy najít algoritmus pro vytváření vektorů hodnot, kterými lze jednotlivá spektra efektivně popsat a spolu s vhodnou metrikou docílit nalezení jim co nejlépe odpovídajících proteinových (resp. peptidových) sekvencí.

3.1 Základní charakteristika

3.1.1 Metrika a metrický prostor

Metrický prostor $M = (D, d)$ definujeme doménou objektů D a vzdálenostní funkcí d. Metrická funkce neboli metrika $d : D \times D \mapsto \mathbb{R}$ určuje podobnost dvou objektů a musí splňovat následující axiomy:

1. reflexivitu
$$\forall x \in D, d(x, x) = 0, \tag{3.1}$$
2. pozitivitu
$$\forall x, y \in D, x \neq y \Rightarrow d(x, y) > 0, \tag{3.2}$$
3. symetrii
$$\forall x, y \in D, d(x, y) = d(y, x), \tag{3.3}$$
4. trojúhelníkovou nerovnost
$$\forall x, y, z \in D, d(x, z) \leq d(x, y) + d(y, z). \tag{3.4}$$

Konkrétní zápis axiomů a jejich počet se může v různé literatuře mírně lišit, nicméně vždy se jedná o alternativní vyjádření stejných vlastností. Pokud není splněna trojúhelníková nerovnost, označujeme funkci d jako semimetriku a M analogicky jako semimetrický prostor. Dále si rozebereme několik nejznámějších typů metrik, které můžeme využít pro porovnávání vektorů. [33]

3.1.2 Minkowského metriky

Minkowského vzdálenost tvoří celou skupinu metrických funkcí označovanou jako L_p metriky, protože její konkrétní tvar závisí na parametru $p \geq 1$. Pro obecný n-dimenzionální vektor ji definujeme jako

$$L_p\left[(x_1, ..., x_n), (y_1, ..., y_n)\right] = \sqrt[p]{\sum_{i=1}^{n} \|x_i - y_i\|^p}. \tag{3.5}$$

Pro $p = 1$ získáme tzv. Manhattanskou vzdálenost

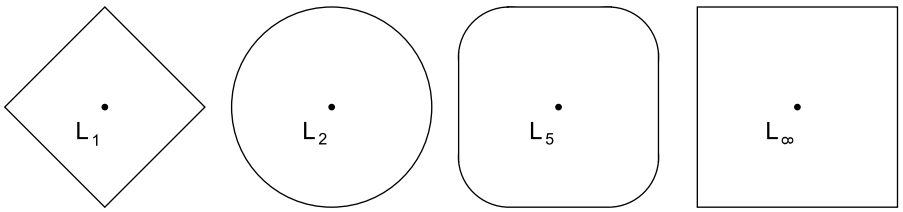
$$L_1 = \sum_{i=1}^{n} \|x_i - y_i\|, \tag{3.6}$$

je-li $p = 2$ dostaneme Eukleidovskou metriku, kterou měříme vzdálenosti objektů v reálném světě

$$L_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{3.7}$$

a v případě $p \to \infty$ získáme maximální vzdálenost

$$L_\infty = max_{i=1}^{n} \|x_i - y_i\|. \tag{3.8}$$



Obrázek 3.2: Minkowského metriky - vyznačeny jsou množiny bodů 2-dimenzionálního prostoru, které mají v dané L_p metrice stejnou vzdálenost od středu.

3.1.3 Kvadratická vzdálenost

Kvadratická metrika zakládá podobnost dvou n-dimenzionálních vektorů na matici M o velikosti $n \times n$, kde prvky $m_{i,j}$ určují, jak silný je vztah mezi složkami i a j vektorů $\vec{x}$ a $\vec{y}$. Matice M obsahuje nezáporná reálná čísla a je symetrická podle hlavní diagonály. Váhy $m_{i,j}$ jsou pak obvykle normalizovány do intervalu $0 \leq m_{i,j} \leq 1$ a pro diagonální prvky platí, že $m_{i,i} = 1$. Následující výraz reprezentuje zobecněnou kvadratickou vzdálenost d_M, kde T značí transpozici vektoru.

$$d_M(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \times M \times (\vec{x} - \vec{y})} \tag{3.9}$$

Pokud M je matice identity, tj. $m_{i,i} = 1$ a ostatní prvky jsou nulové, pak uvedená definice zahrnuje i Eukleidovskou metriku (viz rovnice 3.7). Jestliže pak předefinujeme hodnoty prvků na hlavní diagonále do intervalu $0 \leq m_{i,j} \leq 1$, získáme váženou Eukleidovskou vzdálenost.

$$d_M(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} w_i (x_i - y_i)^2} \tag{3.10}$$

Nevýhodou kvadratické vzdálenosti je, že může být v závislosti na délce n vektorů $\vec{x}$ a $\vec{y}$ výpočetně velmi náročná.

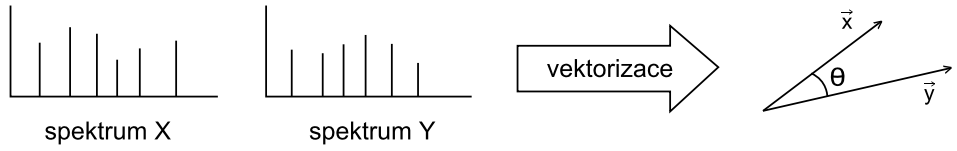
3.1.4 Kosinová podobnost

Pro porovnávání hmotnostních spekter se ukazuje jako vhodná kosinová podobnost, která měří kosinus velikosti úhlu mezi dvěma vektory v n-dimenzionálním prostoru, kde n odpovídá počtu peaků ve spektru a $\|\vec{x}\|$ je Eukleidovská norma vektoru $\vec{x}$. [1]

$$\cos\theta = \frac{\vec{x}\vec{y}}{\|\vec{x}\| \, \|\vec{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3.11}$$

Protože $-1 \leq \cos(x) \leq 1$ je zřejmé, že není splněn axiom pozitivity (viz rovnice 3.2) a tudíž kosinus úhlu není metrika. Abychom však z nemetriky udělali metriku, stačí v tomto případě použít funkci $\arccos(x)$ a pracovat přímo s velikostí úhlu. [7]

$$\theta = \arccos\left(\frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}}\right) \tag{3.12}$$



Obrázek 3.3: Kosinová podobnost (vektory popisující jednotlivá spektra můžeme sestavit z hodnot poměru m/z na jejich x-ové ose, v závislosti na počtu peaků n pak pracujeme v n-dimenzionálním prostoru).

3.1.5 Jaccardův koeficient

Pokud chceme místo podobnosti dvou objektů vyjádřit podobnost dvou množin A a B, můžeme využít Jaccardův koeficient $J(A, B)$, který je definován pomocí poměru kardinalit jejich průniku a sjednocení. Jaccardovu vzdálenost $d_J(A, B)$ pak můžeme vyjádřit jako

$$d_J(A, B) = 1 - J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|}. \tag{3.13}$$

3.1.6 Hausdorffova vzdálenost

V případě Jaccardovy metriky předpokládáme, že libovolný prvek množiny A je buď shodný s vybraným prvkem množiny B nebo se jedná o prvek zcela odlišný. Hausdorffova vzdálenost však páruje elementy množin na základě podobnostní funkce d_e, což může libovolná např. Eukleidova metrika. Předpokládejme, že platí

$$\begin{aligned} d_p(x, B) &= \inf_{y \in B} d_e(x, y), \\ d_p(A, y) &= \inf_{x \in A} d_e(x, y), \\ d_s(A, B) &= \sup_{x \in A} d_p(x, B), \\ d_s(B, A) &= \sup_{y \in B} d_p(A, y), \end{aligned}$$

pak Hausdorffovu vzdálenost nad množinami A a B definujeme jako

$$d_H(A, B) = max\{d_s(A, B), d_s(B, A)\}. \tag{3.14}$$

Hausdorffova metrika vypočte pro všechny prvky množiny A vzdálenost k nejbližšímu sousedovi v B, poté pro všechny prvky množiny B vzdálenost k nejbližšímu v A a z vypočtených vzdáleností vybere maximální hodnotu.

3.2 Vyhledávání v metrických prostorech

3.2.1 Rozsahový dotaz

V metrických prostorech nevyhledáváme nutně objekty, které se přesně shodují, ale objekty, které jsou si podobné. Běžně se používá rozsahový dotaz $R(q, r)$ (range query), kde $q \in D$ je objekt dotazu a r rádius, ve kterém jsou všechny vyskytující se objekty považovány za podobné q.

$$R(q, r) = \{o \in X, d(o, q) \leq r\} \tag{3.15}$$

Dotazovaný objekt q přitom nemusí být součástí prohledávané kolekce $X \subseteq D$ a v případě potřeby je možné, aby jednotlivé výsledky byly seřazeny podle vzdálenosti od něj. Je-li $r = 0$ pak vyhledáváme objekty shodné s q, což je typické hlavně pro algoritmy, jejichž cílem je danou položku z databáze odstranit.

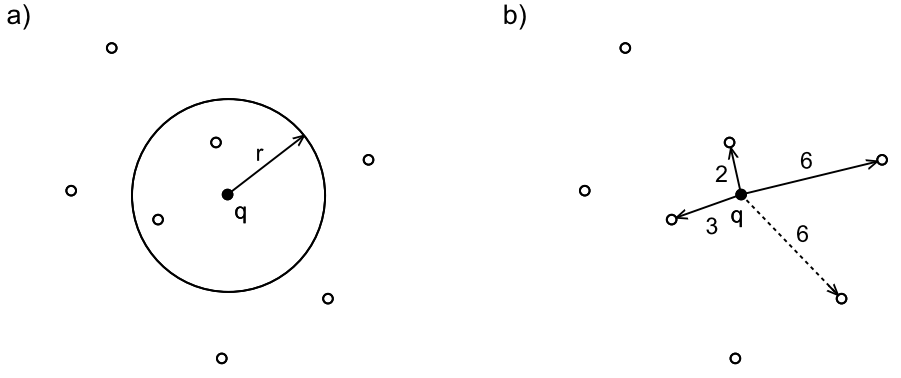
3.2.2 Dotaz na k nejbližších sousedů

Pokaždé když chceme vyhledávat pomocí rozsahového dotazu, musíme stanovit rádius r, což může být někdy poměrně obtížné. Pro dané r totiž nemusíme najít žádný objekt nebo naopak můžeme získat příliš mnoho výsledků. Alternativním způsobem pro vyhledávání podobných objektů je proto dotaz na k nejbližších sousedů (k-nearest neighbor query).

$$kNN(q) = \{R \subseteq X, |R| = k \wedge \forall x \in R, y \in X - R : d(q, x) \leq d(q, y)\} \tag{3.16}$$

Pokud máme pro posledních $p \leq k$ pozic výsledku možnost vybrat z více než p objektů ve stejné vzdálenosti od q, může být zbývajících p objektů vybráno náhodně. Někdy může být výhodné spojit rozsahový dotaz s dotazem na k nejbližších sousedů. Kombinovaný dotaz $kNN(q, r)$ nejprve vybere všechny objekty ve vzdálenosti r, a pokud je jich více než k, vrátí pouze k prvních objektů.

$$\begin{aligned} kNN(q, r) \quad &= \{R \subseteq X, |R| \leq k \ \wedge \forall x \in R, y \in X - R : \\ &\quad d(q, x) \leq d(q, y) \wedge d(q, x) \leq r\} \end{aligned} \tag{3.17}$$

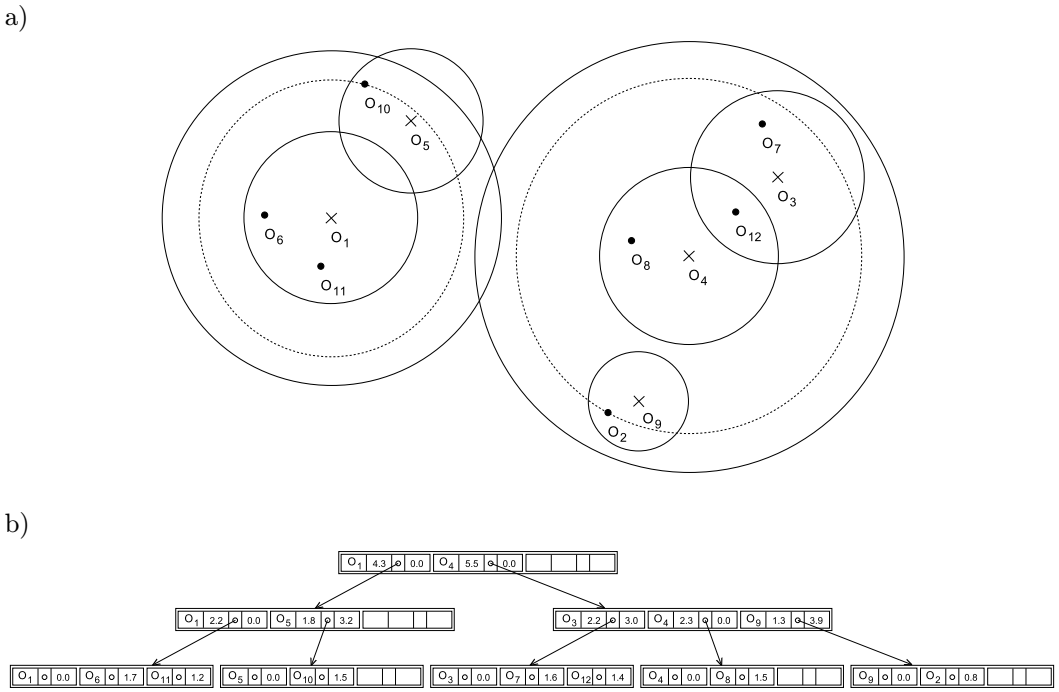


Obrázek 3.4: Typy dotazů - a) rozsahový dotaz, b) dotaz na k nejbližších sousedů (pro $k = 3$).

3.3 M-strom

3.3.1 Struktura

M-strom (Metric tree; M-tree) organizuje množinu metrických objektů S jako dynamický a vyvážený strom. Dynamika přitom umožňuje vkládat a odebírat objekty z již existujícího stromu při zachování vlastností původní struktury a vyváženost zaručuje, že rozdíl úrovní, na kterých jsou umístěny dva libovolné listy stromu, je roven nejvýše 1. Metrickým objektem $O_i \in S$ rozumíme vektor n-rozměrného prostoru, který v závislosti na typu stromového uzlu buď popisuje vlastnosti konkrétních indexovaných dat nebo slouží jako střed hypersférického regionu (O_i, r_{O_i}), kde r_{O_i} je poloměr. [1]



Obrázek 3.5: M-strom - a) struktura ve 2D Eukleidovském prostoru; pro položky kořenového uzlu je přerušovaně naznačena minimální velikost pokrývajícího poloměru r_{O_i}, b) schematické znázornění vnitřní reprezentace stromové struktury; povšimněme si, že objekt O_{12} je uložen pouze v jednom listovém uzlu, přestože spadá do dvou pokrývajících regionů objektů O_3 a O_4.

Rozlišujeme dva základní typy uzlů - vnitřní a listové. Vnitřní uzly obsahují směrovací záznamy (routing entries), které popisují n-rozměrné hypersférické regiony shlukující jednotlivé objekty O_i, listové uzly pak uchovávají odkazy na vlastní indexovaná data (ground entries). Směrovací záznam definujeme čtveřicí

$$rout(O_i) = [O_i, r_{O_i}, ptr(T(O_i)), d(O_i, Par(O_i))]\,, \tag{3.18}$$

kde $O_i \in S$ je virtuální směrovací objekt (pivot) vypočtený pro účely M-stromu, r_{O_i} pokrývající poloměr (covering radius), $ptr(T(O_i))$ ukazatel na pokrývající podstrom $T(O_i)$ (covering subtree) a $d(O_i, Par(O_i))$ je předpočítaná vzdálenost od rodičovského směrovacího objektu využívaná většinou algoritmů pro odfiltrování regionů, se kterými není nutné pracovat. Záznamy v listových uzlech definujeme trojicí

[1] hypersféra = n-rozměrná koule

$$grnd(O_i) = [O_i, oid(O_i), d(O_i, Par(O_i))], \tag{3.19}$$

kde O_i je skutečný objekt popisující konkrétní indexovanou položku a $oid(O_i)$ identifikátor externě uloženého databázového záznamu.

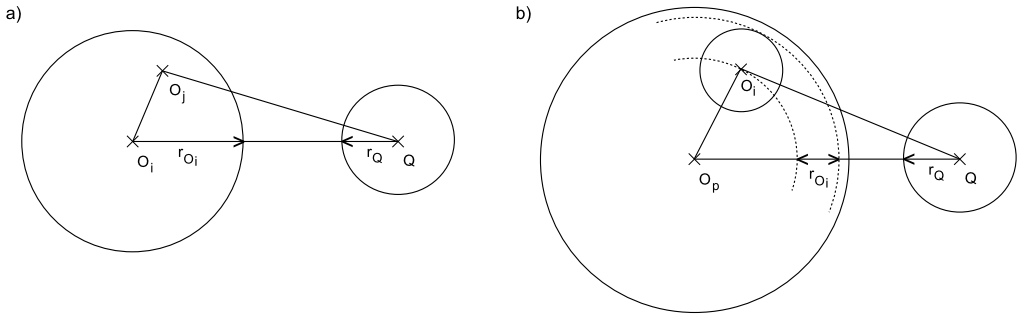
Počet položek v uzlech M-stromu je omezen jejich pevně danou kapacitou. Regiony směrovacích záznamů na stejné úrovni se mohou překrývat a mohou i „přečnívat" ven z rodičovských regionů, to však platí pouze za předpokladu, že je splněna tzv. hnízdící podmínka.

$$\forall O_j \in T(O_i) : d(O_i, O_j) \leq r_{O_i} \tag{3.20}$$

Pro daný směrovací záznam $rout(O_i)$ tedy platí, že všechny objekty uložené v pokrývajícím podstromu $T(O_i)$ jsou součástí hypersférického regionu (O_i, r_{O_i}). Při konstrukci M-stromu však tato podmínka není příliš omezující, protože pro stejná data lze vytvořit mnoho hierarchií, které ji splňují.

3.3.2 Využití trojúhelníkové nerovnosti

Algoritmy pro M-strom využívají k rychlému a bezpečnému odfiltrování regionů, které není nutné prohledávat, dvě lemmata plynoucí z trojúhelníkové nerovnosti.



Obrázek 3.6: Využití trojúhelníkové nerovnosti.

Lemma 1. Nechť (Q, r_Q) je hypersférický region dotazu na objekt Q a (O_i, r_{O_i}) region daný směrovacím záznamem $rout(O_i)$. Pokud $d(O_i, Q) > r_{O_i} + r_Q$, pak pro každý objekt O_j v podstromu $T(O_i)$ platí, že $d(O_j, Q) > r_Q$. Regiony (Q, r_Q) a (O_i, r_{O_i}) se tedy nepřekrývají a podstrom $T(O_i)$ může být z vyhledávání bezpečně vynechán, viz obrázek 3.6 a).

Důkaz plyne z trojúhelníkové nerovnosti a předpokladu $\forall O_j \in T(O_i) : d(O_j, O_i) \leq r_{O_i}$.

$$\begin{aligned} d(O_j, O_i) + d(O_j, Q) &\geq d(O_i, Q) \\ d(O_j, Q) &\geq d(O_i, Q) - d(O_j, O_i) \\ d(O_j, Q) &\geq d(O_i, Q) - r_{O_i} \qquad (\Leftarrow r_{O_i} \geq d(O_j, O_i)) \\ d(O_j, Q) &> r_Q \qquad (\Leftarrow d(O_i, Q) - r_{O_i} \geq r_Q) \end{aligned} \tag{3.21}$$

Lemma 2. Pokud platí $|d(O_p, Q) - d(O_i, O_p)| > r_{O_i} + r_Q$, kde O_p je rodičovský objekt O_i a Q objekt dotazu, pak $d(O_i, Q) > r_{O_i} + r_Q$ a podstrom $T(O_i)$ může být z prohledávání bezpečně vynechán, viz obrázek 3.6 b).

Důkaz je přímým důsledkem trojúhelníkové nerovnosti, která zaručuje správnost obou případů, na něž lze rozložit výraz v absolutní hodnotě, tedy $d(O_i, Q) \geq d(O_p, Q) - d(O_i, O_p)$ a $d(O_i, Q) \geq d(O_i, O_p) - d(O_p, Q)$.

Protože výpočet vzdálenosti dvou objektů může být časově náročný (např. kvadratická vzdálenost), jsou v uzlech M-stromu pro jednotlivé objekty O_i uloženy předpočítané vzdálenosti od rodičovských směrovacích záznamů $d(O_i, O_p)$. Test nutnosti prohledávání daného regionu pak máme v některých případech na základě uložené hodnoty „zadarmo".

3.3.3 Realizace rozsahového dotazu

Algoritmus rozsahového dotazu musí sledovat všechny větve M-stromu vedoucí k objektům O_i, které spadají do dotazovaného regionu (Q, r_Q), tj. splňují podmínku $d(Q, O_i) \leq r_Q$. Uvedeme si rekurzivní algoritmus, který má optimální počet I/O operací, protože přistupuje pouze k těm uzlům, jejichž metrické regiony překrývají oblast dotazu, přičemž menšího počtu přístupů již nelze dosáhnout. Funkce *Compute* $d(O_i, Q)$ provede výpočet vzdálenosti objektů pomocí metriky d, v ostatních případech se používají vždy předvypočtené vzdálenosti objektů ($d(O_i, O_p)$ je uloženo ve směrovacím záznamu a $d(O_p, Q)$ lze předávat jako další parametr rekurze, protože odpovídá hodnotě $d(O_i, Q)$ nadřazeného volání funkce *RangeQuery*). [19]

Algoritmus 3.1

```
1  RangeQuery(N:node, Q:query, r(Q):radius)
2    let Op be the parent object of node N;
3    if N is not a leaf then
4      ∀ Oi in N do
5        if |d(Op,Q)-d(Oi,Op)| ≤ r(Q)+r(Oi) then  // lemma 2
6          Compute d(Oi,Q);
7          if d(Oi,Q) ≤ r(Q)+r(Oi) then  // lemma 1
8            RangeQuery(*ptr(T(Oi)),Q,r(Q));
9    else
10     ∀ Oi in N do
11       if |d(Op,Q)-d(Oi,Op)| ≤ r(Q) then  // lemma 2
12         Compute d(Oi,Q);
13         if d(Oi,Q) ≤ r(Q) then
14           add oid(Oi) to the result;
15 end RangeQuery;
```

3.3.4 Realizace dotazu na k-nejbližších sousedů

Nalezení k-nejbližších sousedů je o něco málo složitější než rozsahový dotaz. Vzhledem k tomu, že vzdálenost mezi dotazovaným objektem Q a k-tým nejbližším sousedem není dopředu známá, je rádius r_Q upravován dynamicky při vyhledávání. Na počátku nastavené $r_Q = \infty$ postupně snižujeme až na konci odpovídá skutečné vzdálenosti mezi objektem Q a k-tým nejbližším sousedem. [19]

Algoritmus 3.2

```
1  kNNQuery(T:root, Q:query, k:integer)
2    PR = [T,-];
3    for i = 1 to k do NN[i] = [-,∞];  // rQ = NN[k].dmax
4    while PR ≠ ∅ do
5      nextnode = ChooseNode(PR);
6      kNNSeachNode(nextnode,Q,k);
7  end kNNQuery;
8
9  ChooseNode(PR:priority_queue)
10   let dmin(T(Oi*)) = min {∀ dmin(T(Oi)) ∈ PR};
11   remove entry [ptr(T(Oi*)),dmin(T(Oi*))] from PR;
12   return *ptr(T(Oi*));
13 end ChooseNode;
```

```
14  kNNSearchNode(N:node, Q:query, k:integer)
15    let O_p be the parent object of node N;
16    if N is not a leaf then
17      ∀ O_i in N do
18        if |d(O_p,Q)-d(O_i,O_p)| ≤ r_Q+r(O_i) then  // lemma 2
19          Compute d(O_i,Q);
20          if d_min(T(O_i)) ≤ r_Q then
21            add [ptr(T(O_i)),d_min(T(O_i))] to PR;
22            if d_max(T(O_i)) ≤ r_Q then
23              r_Q = NNUpdate([-,d_max(T(O_i))]);
24              remove from PR all entries for which d_min(T(O_i)) > r_Q;
25    else
26      ∀ O_i in N do
27        if |d(O_p,Q)-d(O_i,O_p)| ≤ r_Q then  // lemma 2
28          Compute d(O_i,Q);
29          if d(O_i,Q) ≤ r_Q then
30            r_Q = NNUpdate([oid(O_i),d(O_i,Q)]);
31            remove from PR all entries for which d_min(T(O_i)) > r_Q;
32  end kNNSearchNode;
```

Algoritmus využívá prioritní frontu PR (pending requests) pro uzly čekající na zpracování a k prvkové pole NN (nearest neighbours) pro kandidáty na k nejbližších sousedů, které na konci obsahuje výsledek. PR obsahuje požadavky $[ptr(T(O_i)), d_{min}(T(O_i))]$, kde $T(O_i)$ jsou podstromy, které v aktuální chvíli ještě není možné vyřadit z vyhledávání, protože jejich regiony (O_i, r_{O_i}) se překrývají s dynamickým regionem dotazu (Q, r_Q). Priorita prvků je dána vzdáleností $d_{min}(T(O_i))$ mezi objektem Q a metrickým regionem (O_i, r_{O_i}), požadavek s menší $d_{min}(T(O_i))$ má vyšší prioritu a je tedy zpracován dříve.

$$d_{min}(T(O_i)) = max\{0, d(O_i, Q) - r_{O_i}\}. \tag{3.22}$$

Pole NN obsahuje k položek $[oid(O_i), d(Q, O_i)]$ nebo $[-, d_{max}(T(O_i))]$ a je vzestupně setříděno podle vzdáleností objektů od Q, $oid(O_i)$ je odkaz na externí databázový záznam. Položka $[oid(O_i), d(Q, O_i)]$ na j-té pozici tedy přísluší kandidátovi na j-tého nejbližšího souseda. Hodnota $d_{max}(T(O_i))$ v záznamu $[-, d_{max}(T(O_i))]$ odpovídá maximální možné vzdálenosti mezi Q a objektem v regionu (O_i, r_{O_i}).

$$d_{max}(T(O_i)) = d(O_i, Q) + r_{O_i} \tag{3.23}$$

Metoda *NNUpdate* aktualizuje uspořádané pole NN a vrací novou hodnotu poloměru $r_Q = NN[k].d_{max}$. Protože fronta PR musí obsahovat pouze regiony, které se překrývají s oblastí dotazu (Q, r_Q), po každém snížení hodnoty r_Q z ní vymažeme všechny irelevantní záznamy, které již nesplňují podmínku $d_{min}(T(O_i)) \leq r_Q$. Uvedený algoritmus má optimální počet I/O operací, protože přistupuje pouze k těm uzlům, jejichž regiony překrývají dynamickou oblast dotazu (Q, r_Q). Počtem I/O operací je pak k-NN dotaz ekvivalentní rozsahovému dotazu. [31]

3.3.5 Dynamické vkládání objektů

M-strom lze konstruovat dynamicky vkládáním jednotlivých objektů nebo staticky, pokud máme předem k dispozici rozsáhlejší kolekci vstupních dat (viz sekce 3.3.7). Při dynamickém vkládání nejprve hledáme optimální cílový list, do kterého umístíme vkládaný objekt O_i. Snažíme se přitom vybrat takový list, jehož rodičovský objekt O_p je blízko O_i. Současně s tím požadujeme, aby rozšíření rodičovského a nadřazených regionů bylo minimální, nejlépe pak, aby nebylo potřeba vůbec. Základní myšlenku dynamického vložení objektu lze popsat následovně.

Algoritmus 3.3

```
1  Insert(T:root, Oi:object)
2    N = FindLeaf(T,Oi);
3    if N is not full then store grnd(Oi) in leaf N;
4    else Split(N,Oi);
5  end Insert;
```

Funkce *Split* implementuje tzv. politiky štěpení uzlu (viz sekce 3.3.6). Výběr cílového listu funkcí *FindLeaf* úzce souvisí s heuristickými kritérii, která se snaží, aby regiony směrovacích objektů M-stromu byly co nejmenší, nedocházelo k jejich zbytečnému překryvu a tudíž, aby samotné vyhledávání bylo rychlejší.

Algoritmus jednocestného hledání vhodného listu prochází M-strom po jediné větvi a volí vždy takový podstrom směrovacího záznamu $rout(O_j)$, pro který platí, že $d(O_j, O_i) \leq r_{O_j}$. Pokud může vybrat z více možností, zvolí ten s nejmenší vzdáleností $d(O_j, O_i)$. V případě, že není k dispozici žádný region, do něhož lze objekt O_i rovnou zařadit, je zapotřebí zvětšit pokrývající poloměr. Vybírá pak takový podstrom, pro který bude zvětšení rodičovského regionu minimální. Časová složitost algoritmu odpovídá hloubce stromu $O(\log n)$. Jednocestné hledání můžeme schematicky vyjádřit následujícím pseudokódem. [31]

Algoritmus 3.4

```
1  FindLeaf(N:node, Oi:object)
2    if N is leaf then return N;
3    N' = {∀ rout(Oj) ∈ N: d(Oj,Oi) ≤ r(Oj)};
4    if N' ≠ ∅ then
5      select rout(Oj*) ∈ N': min {d(Oj*,Oi)};
6    else
7      select rout(Oj*) ∈ N: min {d(Oj*,Oi)-r(Oj*)};
8      rOj* = d(Oj*,Oi);
9    return FindLeaf(*ptr(T(Oj*)),Oi);
10 end FindLeaf;
```

Alternativou uvedeného postupu je vícecestné vkládání, kdy se provede série bodových dotazů, která určí neplné listy, do nichž je možné objekt rovnou umístit. Z nich se pak vybere ten list, jehož rodičovský směrovací objekt O_p je vkládanému objektu O_i nejblíže, pokud žádný neexistuje provede se běžné jednocestné vložení. Výhodou vícecestného vkládání je minimalizace rozšiřování a překryvů regionů, a také méně časté štěpení uzlů. Nevýhodou je pak vyšší časová složitost $O(n)$, kde n je počet objektů ve stromu, protože místo jedné procházíme více cest. [7]

3.3.6 Politiky štěpení uzlu

Jak již bylo řečeno v předchozí části, postup při přeplnění uzlu definují politiky štěpení (split policies). Nejprve zvolíme dva nové směrovací objekty, které umístíme do rodičovského uzlu (promoting) a poté rozdělíme obsah původního uzlu mezi dva nové uzly (partitioning), přitom se snažíme brát ohled na to, aby nově vznikající regiony byly co možná nejmenší a nedocházelo tak k jejich zbytečným překryvům. Při přeplnění rodičovského uzlu postup analogicky opakujeme. Směřujeme tedy od listu ke kořenu a v nejhorším případě štěpíme všechny uzly na cestě po níž jsme se dostali do cílového listu. Pokud rozdělíme kořen, M-strom o jednu úroveň „povyroste".

Algoritmus 3.5

```
1  Split(N:node, O_i:object)
2    N_all = {∀ O_j ∈ N} ∪ {O_i};
3    if N is not the root then
4      let rout(O_p) be the parent entry of N stored in N_p;
5    allocate a new node N';
6    Promote(N_all,O_p1,O_p2);
7    Partition(N_all,O_p1,O_p2,N_p1,N_p2);
8    store N_p1's entries in N and N_p2's entries in N';
9    if N is current root then
10     allocate a new root node N_p;
11     store rout(O_p1) and rout(O_p2) in N_p;
12   else
13     replace entry rout(O_p) with entry rout(O_p1) in N_p;
14     if N_p is full then
15       Split(N_p,O_p2);
16     else
17       store rout(O_p2) in N_p;
18 end Split;
```

Funkce *Promote* vybere z uzlu N dva objekty O_{p_1} a O_{p_2}, pro které se vytvoří dva nové směrovací záznamy $rout(O_{p_1})$ a $rout(O_{p_2})$. Ty se následně uloží do rodičovského uzlu N_p. Jeden směrovací záznam přitom nahradí původní, který odkazoval na uzel N, prostor pro druhý je nově alokován. Zjednodušená potvrzující varianta (confirmed split policy) potom ponechává jeden z původních směrovacích objektů $O_{p_1} = O_p$ a vybírá pouze nový objekt O_{p_2}. Popíšeme si několik základních metod výběru dvou nových směrovacích objektů.

- *m_RAD*. Algoritmus minimálního součtu poloměrů (minimum RADii) je výpočetně nejnáročnější, protože ze všech možných kombinací dvojic O_{p_1} a O_{p_2} vybírá tu, která dává po cvičném rozdělení množiny objektů (partitioning) minimální součet pokrývajících poloměrů $r(O_{p_1}) + r(O_{p_2})$.

- *mM_RAD*. Metoda je podobná jako *m_RAD*, ale minimalizuje maximum obou poloměrů $\max\{r(O_{p_1}), r(O_{p_2})\}$.

- *M_LB_DIST*. Politika „Maximum Lower Bound on DISTance" se od předchozích případů liší tím, že využívá předpočítané vzdálenosti uložené v uzlech M-stromu. Pokud se jedná o tzv. potvrzující variantu, kde $O_{p_1} = O_p$, vybírá O_{p_2} jako nejvzdálenější objekt od O_p.

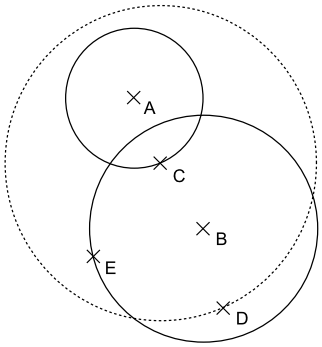
$$d(O_{p_2}, O_p) = \max_{O_j \in N}\{d(O_j, O_p)\} \tag{3.24}$$

- *RANDOM*. Tato varianta vybere pro nové směrovací záznamy dva náhodné objekty O_{p_1} a O_{p_2}. Přestože se na první pohled jedná o primitivní techniku, je rychlá a může být srovnáním pro ostatní metody.

- *SAMPLING*. Jedná se o vylepšenou metodu *RANDOM*, která náhodně zvolí vzorek objektů (sample) o velikosti s. Potom se pro každý z $s(s-1)/2$ párů vzorku vypočte rozdělení objektů a pokrývající poloměry. Ze vzorku se pak podobně jako u metody *m_RAD* vyberou dva objekty, které mají nejmenší součet poloměrů. [19]

Po výběru směrovacích objektů je potřeba rozdělit množinu původních záznamů do dvou nových uzlů, k tomu slouží funkce *Partition*. Běžně se používají dva způsoby dělení. [28]

- *Obecná nadrovina* (generalized hyperplane). Každý objekt se přiřadí k nejbližšímu směrovacímu objektu.
- *Vyvážené dělení* (balanced distribution). Střídá se přidělování k prvnímu a druhému směrovacímu objektu. V každém kroku se k danému směrovacímu objektu přidá momentálně nejbližší objekt. Počet záznamů v nových regionech se tak ve výsledku liší maximálně o 1.



Obrázek 3.7: Štěpení uzlu - naznačeno je rozdělení uzlu směrovacího objektu C s kapacitou 4 uzly (A,B,C,D) na dva nové uzly se směrovacími objekty A (obsahuje A,C) a B (obsahuje B,D,E) po přidání objektu E; upřednostnili jsme kritérium vyváženosti uzlů, nicméně bylo by možné zvolit i rozdělení A(A) a B(B,C,D,E) a minimalizovat tak překryv nových regionů.

3.3.7 Statická konstrukce M-stromu (Bulk-Loading)

Bulk-Loading můžeme využít ve chvíli, kdy máme před vlastní konstrukcí M-stromu k dispozici větší množinu objektů $O_i \in S$, které chceme zaindexovat. Algoritmus je založen na rekurzivním shlukování objektů, přičemž vytváří výslednou stromovou hierarchii.

Z množiny objektů S náhodně vybereme množinu pivotů P s mohutností m, kde m odpovídá kapacitě uzlu M-stromu. Všechny objekty $O_i \in S$ následně rozřadíme ke svým nejbližším pivotům v P. Na podmnožiny $P_1, ..., P_m$ přiřazené jednotlivým pivotům pak rekurzivně voláme Bulk-Loading. Jakmile kapacita podmnožin nepřekračuje hodnotu m, získali jsme listové uzly a po propojení jednotlivých podmnožin při návratu z rekurzivního volání dostaneme výsledný strom T.

Náhodný způsob výběru pivotů už naznačuje, že konstrukce M-stromu tímto způsobem není triviální záležitost. Pokud totiž zvolíme pivota v oblasti s řídkým výskytem objektů, bude mít výsledný podstrom menší hloubku, než pokud jej budeme vybírat v hustěji obsazeném regionu. Vytvořený strom T tak obvykle není vyvážený. Pro řešení tohoto problému se používají dvě techniky.

- Uzly, které obsahují méně objektů než je parametrem stanovená mez, jsou přerozděleny ostatním podstromům a jejich odpovídající pivoti jsou z dané množiny P odstraněni.
- Hlubší podstromy jsou rozloženy na méně hluboké a jejich nově získané kořeny jsou vloženy do P, kde nahrazují kořeny původních hlubších podstromů.

Bulk-Loading algoritmus dokáže zkonstruovat M-strom za použití méně výpočtů vzdáleností objektů než je potřeba u dynamického vkládání. Počet I/O operací je rovněž výrazně nižší, protože je hojně využívána vnitřní paměť. Co se týče efektivity vyhledávání, Bulk-Loading vytváří pouze o něco málo lepší hierarchii než dynamická konstrukce, tj. výsledný strom má menší regiony a tudíž méně často dochází k jejich překryvům. Detailní popis algoritmu je rozebrán v [30, 33].

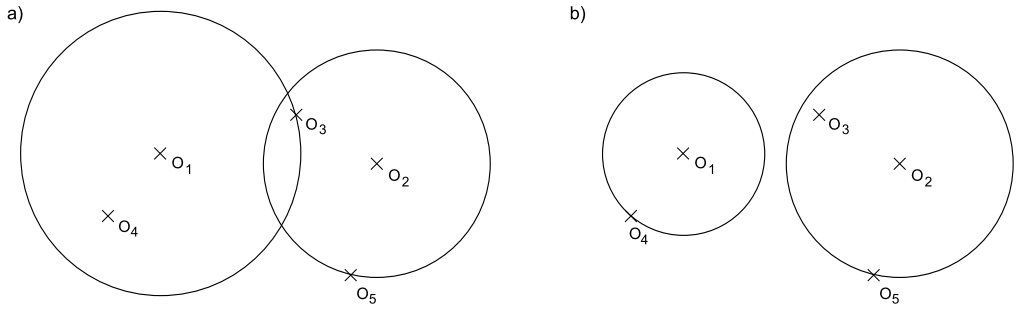
3.3.8 Slim-Tree a optimalizace algoritmem Slim-Down

Struktura označovaná jako Slim-Tree je stejná jako klasický M-strom, ale obsahuje několik vylepšení v podobě algoritmu pro vkládání objektů a algoritmu pro štěpení uzlů. U M-stromu při dynamickém vkládání objektu hledáme vždy takový region, do něhož lze objekt přímo umístit a pokud takový neexistuje, vybereme ten, u kterého je potřeba co nejméně zvětšit pokrývající poloměr. Slim-Tree však volí vždy ten region, jehož pivot je vkládanému objektu nejblíže. Pokud naopak existuje více regionů, do kterých lze objekt umístit, Slim-Tree vybírá ten s minimálním poloměrem, zatímco M-strom volí region, pro něhož je vzdálenost pivota od vkládaného objektu minimální. [33]

Algoritmus rozdělení obsahu uzlů u Slim-Tree vychází z Kruskalova algoritmu pro určení minimální kostry grafu (MST; Minimum Spanning Tree). [26] Nejprve vypočteme vzdálenosti všech možných dvojic uzlů, čímž získáme úplný graf. V něm nalezneme minimální kostru, pak odstraníme nejdelší hranu a graf se rozpadne na dvě komponenty, které tvoří obsah cílových uzlů. Nakonec v každé komponentě musíme ještě určit objekt, jehož vzdálenost je vzhledem k ostatním objektům v daném podgrafu minimální. Vybrané objekty pak nastavíme jako pivoty v rodičovských směrovacích záznamech obou regionů. Složitost tohoto postupu je $O(n^2 \log n)$, z čehož $O(n^2)$ odpovídá výpočtu vzdáleností v úplném grafu.

V případě M-stromu i Slim-Tree lze použít optimalizační algoritmus Slim-Down, který minimalizuje velikosti regionů již zkonstruovaného stromu a tím pádem i jejich překryv, což se výrazně projevuje na zlepšení efektivity vyhledávání. Základní algoritmus Slim-Down je založen na redistribuci listových záznamů, zobecněná varianta pak umožňuje i přesuny směrovacích záznamů. Redistribuce záznamů se provádí po jednotlivých úrovních stromu. Pro každý z objektů uvnitř uzlu hledáme vhodný uzel na stejné úrovni, kam lze daný objekt přemístit. Nejlepší uzel najdeme pomocí série dotazů, kde dotazem je přemisťovaný objekt a cílovými kandidáty jsou uzly na stejné úrovni stromu, do nichž se vejde region dotazovaného objektu. Vybereme přitom ten uzel, jehož rodičovský pivot je nejblíže. Algoritmus startuje vždy na listové úrovni stromu. Pokud některé uzly zůstanou téměř nebo zcela prázdné, lze zbývající položky přesunout jinam a jejich směrovací záznamy odstranit.

Nevýhodou Slim-Down jsou jak velké výpočetní náklady, tak i počet I/O operací, protože se v principu dotazujeme na každý objekt indexovaný ve stromu. Výhodou je potom kromě zmenšení velikosti regionů i fakt, že se jedná o stabilní algoritmus, který můžeme v případě potřeby kdykoliv přerušit a později znovu opět spustit. [7]



Obrázek 3.8: Slim-Down - hierarchie Slim-Tree a) před, b) po provedení optimalizace.

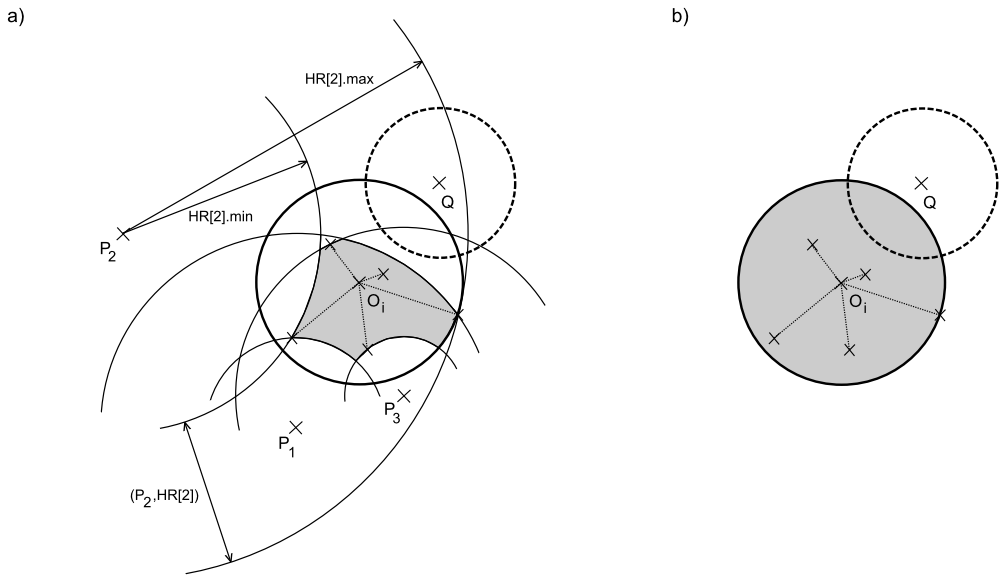
3.4 PM-strom

3.4.1 Struktura

Motivací pro konstrukci PM-stromu (PM-tree; Pivoting Metric Tree) je zmenšení metrických regionů. Rozsáhlé oblasti prázdného prostoru totiž zvyšují pravděpodobnost průniku s regionem dotazu a tím výrazně snižují efektivitu vyhledávání. Optimální tvar metrického regionu by měl obecně splňovat 4 základní požadavky:

1. Kompaktní uložení - informace o regionech uchovávané ve směrovacích záznamech by měly být co nejmenší, aby uložení vnitřních uzlů stromu zabralo mnohem méně místa než uložení listových záznamů.
2. Kompaktní tvar - regiony by měly co nejtěsněji ohraničovat skupinu objektů, aby se snížila pravděpodobnost průniku prázdného prostoru s regionem dotazu.
3. Jednoduchý test průniku - průnik regionu s oblastí dotazu by měl být snadno a rychle spočítatelný.
4. Jednoduché vytvoření nadregionu - při štěpení uzlu by měl být snadno zkonstruovatelný nadřazený region k dané množině regionů.

PM-strom se těmto požadavkům snaží přiblížit tím, že původní hypersférické oblasti M-stromu ořezává pomocí hyperprstencových (hyper-ring) regionů $(O_i, r_{O_i}^{low}, r_{O_i}^{up})$, kde $0 \leq r_{O_i}^{low} \leq r_{O_i}^{up}$, O_i je střed (pivot) dvou soustředných n-dimenzionálních koulí, $r_{O_i}^{low}$ poloměr vnitřní a $r_{O_i}^{up}$ poloměr vnější koule.



Obrázek 3.9: PM-strom - a) region PM-stromu, b) původní region M-stromu.

Pro PM-strom definujeme množinu globálních pivotů P, kde daný pivot $P_t \in S$, přičemž S je množina všech indexovaných objektů. Množina P bývá uložena v externím souboru pivotů, jehož velikost je fixní po celou dobu existence indexu ($|P| = p$). Směrovací záznam ve vnitřním uzlu PM-stromu pak definujeme jako

$$rout_{PM}(O_i) = [O_i, r_{O_i}, ptr(T(O_i)), d(O_i, Par(O_i)), HR]\,, \tag{3.25}$$

kde nový atribut HR je pole $p_{hr} \leq p$ intervalů. Daný interval $HR[t] = \langle HR[t].min, HR[t].max \rangle$ pokrývá vzdálenost mezi nejbližším a nejvzdálenějším objektem uloženým v listech podstromu $T(O_i)$ vzhledem k pivotu P_t, tedy $\forall O_j \in T(O_i)$

$$HR[t].min = \min \bigcup_j \{d(P_t, O_j)\}, \tag{3.26}$$

$$HR[t].max = \max \bigcup_j \{d(P_t, O_j)\}. \tag{3.27}$$

Množina globálních pivotů P a pole HR pak definují množinu p_{hr} hyperprstencových regionů $(P_t, HR[t].min, HR[t].max)$ neboli $(P_t, HR[t])$ takových, že všechny objekty uložené v pokrývajícím podstromu $T(O_i)$ jsou umístěny uvnitř každého hyperprstence

$$\forall O_j \in T(O_i), \forall t \leq p_{hr} \Rightarrow O_j \in (P_t, HR[t]). \tag{3.28}$$

Protože každý hyperprstencový region $(P_t, HR[t])$ obsahuje všechny objekty uložené v podstromu $T(O_i)$, musí je obsahovat i průnik hyperprstenců s hyperkoulí (O_i, r_{O_i}). Metrický region PM-stromu je pak vždy menší nebo roven velikosti původní regionu M-stromu. Hnízdící podmínka pro PM-strom navíc říká, že každý hyperprstencový region $(P_t, HR[t])$ je umístěn uvnitř rodičovského hyperprstence. Datové záznamy uložené v listech PM-stromu definujeme jako

$$grnd_{PM}(O_i) = [O_i, oid(O_i), d(O_i, Par(O_i)), PD], \tag{3.29}$$

kde nový atribut PD označuje pole $p_{pd} \leq p$ vzdáleností pivotů, tedy $PD[t] = d(O_i, P_t)$. Počty prvků $p_{hr} \geq 0$ a $p_{pd} \geq 0$ v polích HR a PD jsou konstantní po celou dobu existence indexu a jejich vhodným nastavením můžeme vyladit optimální efektivitu vyhledávání. [31, 32]

3.4.2 Zpracování dotazů

Před zpracováním dotazu Q je nejprve zapotřebí vypočítat jeho vzdálenosti od všech použitých pivotů $d(Q, P_t), \forall t \leq \max(p_{hr}, p_{pd})$. Následně procházíme PM-strom a prohledáváme pouze ty metrické regiony, jejichž směrovací záznamy určují oblast překrývající se s regionem dotazu Q, což je splněno právě tehdy, když průnik všech prstenců v poli HR a koule (O_i, r_{O_i}) protínají region (Q, r_Q).

Analogicky jako v případě M-stromu se snažíme minimalizovat použití výpočtu vzdálenosti objektů pomocí metriky $d(O_x, O_y)$, a proto kromě lemmatu 2 (viz sekce 3.3.2) na základě znalosti předvypočtených hodnot $d(Q, P_t)$ aplikujeme následující podmínku.

$$\bigwedge_{t=1}^{p_{hr}} d(Q, P_t) - r_Q \leq HR[t].max \wedge d(Q, P_t) + r_Q \geq HR[t].min \tag{3.30}$$

Tedy testujeme zda-li se všechny hyperprstence daného směrovacího záznamu překrývají s regionem dotazu. Pokud podmínka není splněna, podstrom $T(O_i)$ může být z dalšího zpracování vynechán. Obecně se přitom jedná o podmínku nezbytnou nikoliv dostačující, protože nezaručuje, že průnik všech hyperprstenců překrývá region dotazu. O postačující podmínku se však jedná v případě bodového dotazu $(Q, 0)$. Na listové úrovni můžeme vynechat ty datové záznamy, které nesplňují podmínku

$$\bigwedge_{t=1}^{p_{pd}} |d(Q, P_t) - PD[t]| \leq r_Q, \tag{3.31}$$

tj. objekt O_i není umístěn v regionu dotazu. Čím více prstenců je definováno, tím je pravděpodobnost průniku s dotazem Q nižší. Zároveň však roste velikost pole HR a režie na uložení směrovacího záznamu je naopak vyšší. Protože implementace indexových struktur běžně pracují s vnitřní pamětí (cache), znamená to pak, že do ní můžeme umístit méně uzlů a tudíž jsou častější i její výpadky.

Při předpokladu, že záznam jednoho intervalu v poli HR jsou dvě reálná čísla, která uložíme na 2×4 byty, že velikost pole $p_{hr} = 30$ a kapacita uzlu je 20 záznamů, pak uložení prstenců pro jeden uzel zabere $30 \cdot 20 \cdot 2 \cdot 4 = 4800$ bytů. Existuje však technika, která umožňuje tuto režii snížit tak, že každé reálné číslo aproximuje, aby k jeho uložení stačil 1 byte. Podrobněji viz [31, 32].

3.4.3 Realizace rozsahového dotazu

Rozsahový dotaz je velmi podobný jako u M-stromu (viz sekce 3.3.3), jedná se pouze o jeho rozšíření přidáním podmínek 3.30 a 3.31. Algoritmus zachovává optimální počet I/O operací. [31]

Algoritmus 3.6

```
1  RangeQuery(T:root, Q:query, r(Q): radius)
2    let QDist be an array of max(p_hr,p_pd) floats;
3    for t = 1 to max(p_hr,p_pd) do
4      QDist[t] = Compute d(P_t,Q);
5    RangeQueryRec(root,Q,r(Q),QDist);
6  end RangeQuery;
7
8  RangeQueryRec(N:node, Q:query, r(Q):radius, QDist:float[])
9    let O_p be the parent object of node N;
10   if N is not a leaf then
11     ∀ O_i in N do
12       if |d(O_p,Q)-d(O_i,O_p)| ≤ r(Q)+r(O_i) then  // lemma 2
13         if HROverlap(HR,QDist,r(Q)) then  // podmínka 3.30
14           Compute d(O_i,Q);
15           if d(O_i,Q) ≤ r(Q)+r(O_i) then  // lemma 1
16             RangeQueryRec(*ptr(T(O_i)),Q,r(Q),QDist);
17   else
18     ∀ O_i in N do
19       if |d(O_p,Q)-d(O_i,O_p)| ≤ r(Q) then  // lemma 2
20         if PDOverlap(PD,QDist,r(Q)) then  // podmínka 3.31
21           Compute d(O_i,Q);
22           if d(O_i,Q) ≤ r(Q) then
23             add oid(O_i) to the result;
24 end RangeQueryRec;
25
26 HROverlap(HR:float[], QDist:float[], r(Q):radius)
27   for t = 1 to p_hr do
28     if QDist[t] - r(Q) > HR[t].max or QDist[t] + r(Q) < HR[t].min then
29       return false;
30   return true;
31 end HROverlap;
32
33 PDOverlap(PD:float[], QDist:float[], r(Q):radius)
34   for t = 1 to p_pd do
35     if |QDist[t] - PD[t]| > r(Q) then
36       return false;
37   return true;
38 end HROverlap;
```

3.4.4 Realizace dotazu na k-nejbližších sousedů

Implementace dotazu na k-nejbližších sousedů rovněž vychází z algoritmu pro M-strom (viz sekce 3.3.4). Hlavní rozdíly u PM-stromu přitom spočívají v předvýpočtu vzdáleností pivotů od dotazu Q, v zabudování podmínek 3.30 a 3.31, tedy funkcí $HROverlap$ a $PDOverlap$, a dále v úpravě výpočtu hodnot $d_{min}(T(O_i))$ a $d_{max}(T(O_i))$, které zohledňují možnost překryvu prstenců s regionem dotazu.

Algoritmus 3.7

```
1  kNNQuery(T:root, Q:query, k:integer)
2    PR = [T,-];
3    for i = 1 to k do  // r_Q = NN[k].d_max
4      NN[i] = [-,∞];
5    let QDist be an array of max(p_hr,p_pd) floats;
6    for t = 1 to max(p_hr,p_pd) do
7      QDist[t] = Compute d(P_t,Q);
8    while PR ≠ ∅ do
9      nextnode = ChooseNode(PR);
10     kNNSeachNode(nextnode,Q,k,QDist);
11 end kNNQuery;
12
13 kNNSearchNode(N:node, Q:query, k:integer, QDist:float[])
14   let O_p be the parent object of node N;
15   if N is not a leaf then
16     ∀ O_i in N do
17       if |d(O_p,Q)-d(O_i,O_p)| ≤ r_Q+r(O_i) then  // lemma 2
18         if HROverlap(HR,QDist,r(Q)) then  // podmínka 3.30
19           Compute d(O_i,Q);
20           if d_min(T(O_i)) ≤ r_Q then
21             add [ptr(T(O_i)),d_min(T(O_i))] to PR;
22             if d_max(T(O_i)) ≤ r_Q then
23               r_Q = NNUpdate([-,d_max(T(O_i))]);
24               remove from PR all entries for which d_min(T(O_i)) > r_Q;
25   else
26     ∀ O_i in N do
27       if |d(O_p,Q)-d(O_i,O_p)| ≤ r_Q then  // lemma 2
28         if PDOverlap(PD,QDist,r(Q)) then  // podmínka 3.31
29           Compute d(O_i,Q);
30           if d(O_i,Q) ≤ r_Q then
31             r_Q = NNUpdate([oid(O_i),d(O_i,Q)]);
32             remove from PR all entries for which d_min(T(O_i)) > r_Q;
33 end kNNSearchNode;
```

Modifikovanou hodnotu $d_{min}(T(O_i))$ vypočteme jako

$$d_{min}(T(O_i)) = max\{0, d(O_i, Q) - r_{O_i}, d^{low}_{HRmax}, d^{low}_{HRmin}\}, \tag{3.32}$$

$$d^{low}_{HRmax} = \max \bigcup_{t=1}^{p_{hr}} \{d(P_t, Q) - HR[t].max\}, \tag{3.33}$$

$$d^{low}_{HRmin} = \max \bigcup_{t=1}^{p_{hr}} \{HR[t].min - d(P_t, Q)\}, \tag{3.34}$$

kde maximum z dvojice hodnot d^{low}_{HRmax} a d^{low}_{HRmin} určuje spodní mez vzdálenosti mezi dotazem Q a objekty umístěnými v nejvzdálenějším hyperprstenci.

Výpočet hodnoty $d_{max}(T(O_i))$ pak provedeme podle vztahu

$$d_{max}(T(O_i)) = \min\{d(O_i, Q) + r_{O_i}, d_{HR}^{up}\}, \tag{3.35}$$

$$d_{HR}^{up} = \min \bigcup_{t=1}^{p_{hr}} \{d(P_t, Q) + HR[t].max\}, \tag{3.36}$$

kde d_{HR}^{up} definuje horní mez vzdálenosti mezi objektem Q a objekty umístěnými v nejbližším hyperprstenci. Algoritmus $kNNQuery$ pro PM-strom má optimální počet I/O operací, protože přistupuje pouze k těm uzlům, jejichž metrické regiony překrývají aktuální region dotazu $(Q, d(Q, NN[k].d_{max}))$. Počet I/O operací je přitom stejný jako rozsahového dotazu PM-stromu. [31]

3.4.5 Konstrukce PM-stromu

PM-strom je konstruován stejným způsobem jako M-strom (viz sekce 3.3.5), algoritmus však musí navíc v cílovém listu vytvořit pole vzdáleností pivotů PD a aktualizovat všechna pole hyperprstenců HR ve směrovacích záznamech na „vkládací" cestě objektu O_i.

Algoritmus 3.8

```
1  Insert(T:root, Oi:object)
2    let QDist be an array of max(php,ppd) floats;
3    for t = 1 to max(php,ppd) do
4      QDist[t] = Compute d(Pt,Q);
5    N = FindLeaf(T,Oi);
6    if N is not full then
7      store grnd(Oi) in leaf N where PD = QDist;
8      update HR arrays of all the parent routing entries of TargetLeaf
9        by values stored in QDist;
10   else
11     Split(N,Oi,QDist);
12 end Insert;
```

Pole PD v listovém záznamu vytvoříme přímým přiřazením pole předvypočtených vzdáleností pivotů od vkládaného objektu $QDist$. Všechna pole HR na vkládací cestě pak modifikujeme opět s využitím předvypočtených hodnot v $QDist$ na základě vztahů 3.26 a 3.27. Metoda $FindLeaf$ je stejná jako u M-stromu. V případě rozdělení uzlu metodou $Split$, je zapotřebí provést přepočet příslušných polí HR, podrobněji viz [31]. Algoritmus vložení objektu si i po modifikaci zachovává logaritmickou složitost.

3.4.6 Výběr pivotů

Z hlediska efektivity PM-stromu je důležité ze všech objektů $O_i \in S$ vhodně vybrat množinu pivotů P, která dostatečně zredukuje velikosti a překryv regionů. Obecně lze říci, že nejlepší je zvolit P tak, aby vzdálenosti mezi jednotlivými objekty P_t byly maximální a aby pivoti nebyli umístěni ve shlucích indexovaných objektů. Pivoti, kteří jsou blízko u sebe totiž zbytečně poskytují redundantní informace.

Zadefinujme si nejprve zjednodušeně pojem výkonnostní kritérium μ_d. Nechť máme množinu pivotů $P = \{p_1, p_2, ..., p_k\}$, kde $p_i \in S$ a nechť $u \in S$ je libovolný indexovaný objekt. Pak lze sestavit vektor hodnot $[u] = (d(u, p_1), d(u, p_2), ..., d(u, p_k))$ a definovat metriku $D([x], [y]) = max_{1 \leq i \leq k} |d(x, p_i) - d(y, p_i)|$. Z množiny S tedy vybereme A náhodných dvojic objektů $\{(a_1, a'_1), (a_2, a'_2), ..., (a_A, a'_A)\}$, pro každý pár (a_i, a'_i) vypočteme hodnotu D_i a

získáme množinu vzdáleností $\{D_1, D_2, ..., D_A\}$. Výkonnostní kritérium následně určíme jako $\mu_d = \frac{1}{A} \sum_{1 \leq i \leq A} D_i$. Podrobnější rozbor je uveden v [18].

Nejjednodušší metodu, která vybírá náhodně p pivotů z množiny všech indexovaných objektů S, označujeme jako *Random*. Je výpočetně velmi levná a používá se především pro vysoce dimenzionální množiny dat, kde víceméně na volbě pivotů již nezávisí. Vylepšená technika *RandomNgroups* zvolí náhodně N skupin p pivotů, pro každou skupinu určí kritérium μ_d a vybere tu s jeho maximální hodnotou. Postup však vyžaduje $2kAN$ výpočtů vzdáleností $d(O_i, O_j)$. Její levnější varianta *RandomNmax* proto vybírá tu skupinu, pro kterou je maximální součet vzdáleností mezi všemi pivoty.

Kromě náhodného výběru pivotů můžeme použít i výběr inkrementální. Nejprve zvolíme pivota P_1 ze vzorku objektů $N \in S$, pro něhož samotného vychází hodnota μ_d maximální. Poté vybereme pivota P_2 z nového vzorku N tak, že pro množinu pivotů $\{P_1, P_2\}$, kde P_1 je konstantní, je μ_d maximální. Pokračujeme výběrem pivota P_3 opět z nové množiny N, kde μ_d pro $\{P_1, P_2, P_3\}$ je maximální, přičemž P_1 i P_2 jsou konstantní, atd. Postup opakujeme dokud nenaplníme množinu P všemi k pivoty.

Podíváme-li se na výběr pivotů z jiného úhlu, můžeme provádět jejich volbu staticky nebo dynamicky. Statickou selekci provádíme, pokud máme k dispozici větší množinu dat před vlastní konstrukcí PM-stromu, v opačném případě volíme pivoty dynamicky. Obvykle pak definujeme množinu pivotů na základě prvních k vložených objektů, přičemž pole HR a PD o velikosti $p \leq k$ jsou vypočtena až ve chvíli, kdy je vložen k-tý objekt.

V případě potřeby můžeme u PM-stromu kdykoliv provést tzv. re-pivoting, kdy zvolíme novou množinu pivotů P a přepočteme všechna pole HR a PD. Re-pivoting je výhodný např. po vložení většího množství objektů nebo nutný po použití algoritmu Slim-Down pro M-strom (viz sekce 3.3.8). Optimalizační algoritmus Slim-Down totiž v tomto případě neudržuje pole HR a PD v aktuálním stavu.

4 Výzkum a experimenty

Ve výzkumné části se budeme zabývat v současné době stále více převažující tandemovou hmotnostní spektrometrií. Zaměříme se přitom na klíčovou část identifikace, a to interpretaci peptidových sekvencí z hmotnostních spekter s využitím M-stromu a PM-stromu. Problematiku sestavování resp. skórování proteinových sekvencí na základě identifikovaných peptidů přitom pro jednoduchost ponecháme stranou, protože k tomuto účelu lze využít již dříve popsané principy (viz kapitola 2).

4.1 Kolekce testovacích dat

Pro účely testování algoritmů, které se snaží identifikovat peptidy z tandemových hmotnostních spekter, vznikají různé kolekce anotovaných MS/MS dat. Kolekce Amethyst a Opal jsou součástí projektu Quartz, který je možné nalézt na stránkách organizace GPM (The Global Proteome Machine Organization; `http://www.thegpm.org/quartz/`).

Kolekce Amethyst obsahuje směs dat získaných z QSTAR hmotnostních spektrometrů a zahrnuje spektra typu MALDI a ESI, spektrometry QSTAR jsou obdobou typu QTOF (viz sekce 2.1). Kolekce Opal pak obsahuje pouze spektra typu ESI z QTOF hmotnostních analyzátorů. V experimentech budeme konkrétně používat data ze souborů *amethyst-gv.xml* (součást archivu `ftp://ftp.thegpm.org/data/quartz/amethyst_040501.zip`) a *opal-gv.xml* (`ftp://ftp.thegpm.org/data/quartz/opal_040501.zip`), které obsahují tandemová spektra peptidů nalezených v lidském genomu.

Data jsou uložena ve formátu XML a obsahují mnoho podrobných informací, pro účely testování však budeme používat pouze několik základních údajů - seznamy peaků jednotlivých tandemových spekter, hmotnosti a náboje jejich rodičovských peptidů, peptidové sekvence příslušející daným spektrům, proteinové sekvence obsahující nalezené peptidy, modifikace peptidových sekvencí a maximální počty nalezitelných b a y-iontů.

V kolekci *amethyst-gv.xml* není implicitně rozlišeno, která spektra jsou typu MALDI a která typu ESI. Pro jednoduchost budeme proto předpokládat, že spektra s nábojem 1^+ jsou typu MALDI a spektra s vyšším nábojem typu ESI. Technikou MALDI ve výzkumné části implicitně rozumíme tandemovou hmotnostní spektrometrii MALDI-PSD (viz sekce 2.4), nikoliv jednoduchou hmotnostní MALDI-TOF analýzu.

Spektra v souboru *amethyst-gv.xml* s nábojem 1^+ obsahují od 27 do 50 peaků, s nábojem 2^+ až 4^+ od 13 do 48 peaků. V souboru *opal-gv.xml* pak libovolné spektrum obsahuje od 15 do 50 peaků. Podrobnější statistiky prezentuje tabulka 4.1.

Kolekce	**Počet spekter**						
	celkem	1^+	2^+	3^+	4^+	**s modifikacemi**	**s bodovou mutací**
amethyst-gv.xml	1825	1052	533	224	16	473	246
opal-gv.xml	622	0	477	139	6	243	127

Tabulka 4.1: Kolekce testovacích dat (modifikací rozumíme změnu hmotnosti stejné aminokyseliny, bodovou mutací záměnu jedné aminokyseliny za jinou). [11]

V případech, kdy měříme závislost na velikosti databáze, je zapotřebí rozšířit množství indexovaných proteinových sekvencí. K tomu používáme soubor proteinů nalezených v genomu člověka *human_e.fasta* (`ftp://ftp.thegpm.org/fasta/ensembl/human_e.fasta.gz`), z něhož vybereme prvních k proteinů, které přidáme ke sjednocení proteinových sekvencí získaných z kolekcí *amethyst-gv.xml* a *opal-gv.xml*. Sekvence, které obsahují symboly nepříslušející značce

žádné aminokyseliny (viz tabulka 2.2), jsou ze zpracování vynechány a redundantní proteinové sekvence jsou odstraněny. [1]

Protože na základě známých peptidových sekvencí nelze stanovit teoretické hodnoty intenzit fragmentových iontů v tandemových spektrech a protože vyšší hodnota intenzity nedává záruku, že se z hlediska identifikace peptidu jedná o významnější peak než když je intenzita nižší, provádíme v experimentech identifikaci pouze na základě hodnot poměru m/z.

Všechna měření byla prováděna na notebooku ASUS F3TC-AP008 s dvoujádrovým 64-bitovým procesorem AMD TURION TL52, 120 GB HDD, 1 GB RAM a operačním systémem Windows XP SP2 při minimálním zatížení jinými aplikacemi. Při zpracování kolekcí dat jsou pro parametry DAC_{all}, DAC_{real} (viz sekce 4.2), selektivitu a čas výpočtu implicitně uváděny průměrné hodnoty na 1 spektrum.

4.2 Framework ATOM

Pro testování metrických indexovacích metod byl použit framework ATOM (Amphora Tree Object Model), který umožňuje snadnou implementaci perzistentních stromových datových struktur a v současnosti podporuje i práci se strukturami M-strom a PM-strom (viz kapitola 3). Framework je implementován v jazyce C++ a byl vyvinut skupinou *Amphora Research Group* na Katedře informatiky FEI, VŠB - Technická univerzita Ostrava.

ATOM se snaží eliminovat dynamickou alokaci paměti, což je velmi drahá operace srovnatelná s fyzickým přístupem na disk. Na začátku alokuje paměť dané velikosti, kterou používá po celou dobu práce se stromovou strukturou a žádnou další alokaci místa pro uzly stromu již následně neprovádí. [22, 31]

Implementace indexové struktury PM-strom (M-strom v případě nastavení nulového počtu pivotů) používá vyrovnávací paměť s kapacitou $H \times N$ uzlů. Pokud se pak má při procházení indexové struktury vyzvednout z disku daný uzel, zkontroluje se nejprve, zda není uložen v cache. Počet všech přístupů k uzlům stromu bez použití cache udává parametr DAC_{all} (Disk Access Cache), skutečný počet přístupů na disk při použití cache parametr DAC_{real}.

Pro implementaci vzdálenostních funkcí je v ATOMu využívána abstraktní třída *cDistance*. Libovolnou metriku můžeme realizovat definováním metody *Compute* v jejím potomkovi.

Při vytváření PM-stromu voláme metodu *Create* třídy *cPMTree*, při otevírání již vytvořeného indexu pak analogicky metodu *Open*. Objekty můžeme jednotlivě vkládat voláním metody *Insert*. Při ukončení práce zavoláme metodu *Close*.

K dispozici jsou metody pro rozsahový dotaz *RangeQuery* a dotaz na k-nejbližších sousedů *kNNQuery*. Doimplementován byl intervalový dotaz *IntervalQuery* (viz sekce 4.5.1). Nad indexovou strukturou je rovněž možné provádět řadu dalších operací jako např. zmenšování regionů algoritmem Slim-Down *SlimDownTree*, re-pivoting *RepivotTree*, aj.

4.3 Princip konstrukce databáze

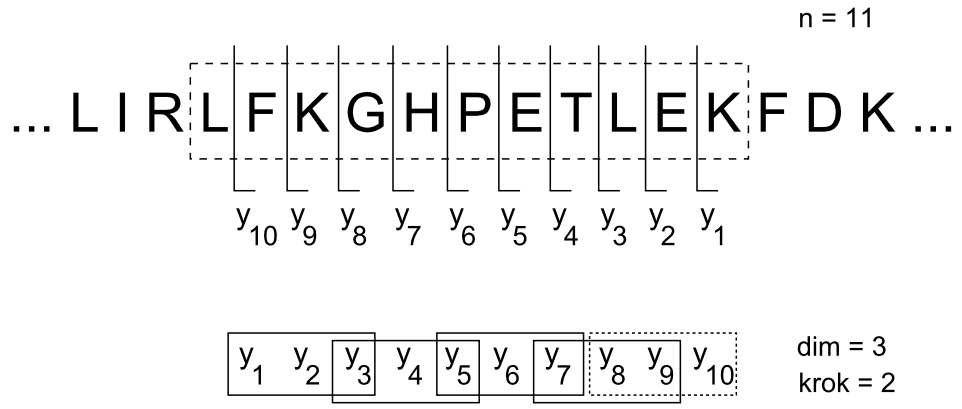
Kolekce dat *amethyst-gv.xml* a *opal-gv.xml* obsahují kromě jiného i kompletní proteinové sekvence, ve kterých byly peptidy přiřazené daným spektrům nalezeny. Po sjednocení proteinových sekvencí z těchto kolekcí a odstranění redundancí získáme 510 různých proteinů, které podrobíme pravidlům dělení trypsinem, čímž získáme teoretické sekvence peptidů pro naplnění databáze. Abychom však mohli otestovat různé metriky (viz sekce 3.1), musíme databázi naplnit vektory reálných čísel, které budou tyto peptidy vhodně popisovat.

[1] V některých případech mohou být v sekvencích obsaženy zástupné symboly * nebo X pro označení libovolné aminokyseliny, B pro označení aminokyselin D nebo N, Z pro E nebo Q, apod. [20]

Peptidy můžeme reálnými čísly popsat např. tak, že budeme vytvářet jim odpovídající teoretická hmotnostní spektra, přičemž můžeme generovat různé typy fragmentových iontů např. y, b, a, $b - H_2O$, $y - NH_3$ (viz tabulka 2.6). Čím více různých typů iontů přitom budeme vytvářet, tím bude databáze vektorů větší a současně nemáme záruku, že všechny vybrané typy iontů se budou vyskytovat i v experimentálních spektrech, naopak nám mohou výsledek hledání znepřesnit. Místo velkého počtu různých iontů budeme proto generovat a ukládat pouze y-ionty (případně y a b-ionty) a experimentální spektra na ně před vlastním vyhledáváním s využitím vhodné heuristiky redukovat (viz sekce 4.4.1 a 4.4.2).

Pokud pro každou peptidovou sekvenci délky n, kterou chceme zaindexovat, vygenerujeme teoretické hodnoty hmotností odpovídající y-iontům, získáme vektor obsahující $n - 1$ hodnot. Protože testované metriky (kromě Hausdorffovy vzdálenosti) vyžadují stejně dlouhé vektory a peptidy mají délku obecně různou, definujeme okénko, které budeme po $n - 1$ hodnotách odpovídajících hmotnostem y-iontů posouvat a jednotlivé celkově kratší a přitom stejně dlouhé vektory indexovat. Velikost okénka odpovídá dimenzi dim indexovaných vektorů. Tímto postupem lze tedy indexovat pouze peptidy, jejichž délka je nejméně $dim + 1$. Spolu s okénkem definujeme ještě krok posunu. Např. pro peptidovou sekvenci délky $n = 10$, $krok = 1$ a $dim = 3$, tak získáme $\lceil((n-1) - (dim-1))/krok\rceil = 7$ vektorů, které zaindexujeme.

Pokud nejméně jedna hodnota z konce vektoru $n - 1$ položek není součástí posledního okénka, přidáme ještě jedno okénko, které pokryje posledních dim hodnot z $n - 1$ položek, abychom nezanedbávali hodnoty, které mohou z hlediska identifikace peptidu odpovídat významným y-iontům z koncové části sekvence. Situaci ilustruje obrázek 4.1.



Obrázek 4.1: Konstrukce peptidové databáze.

Pokud definujeme $krok < dim$ vzniká určitá redundance indexovaných dat, zvyšujeme tím však přesnost identifikace. Vzhledem k faktům, že v současné době kapacita médií již není kritickým problémem a že indexovací metody mají logaritmickou složitost vyhledávání, můžeme si určitou redundanci vektorů dovolit.

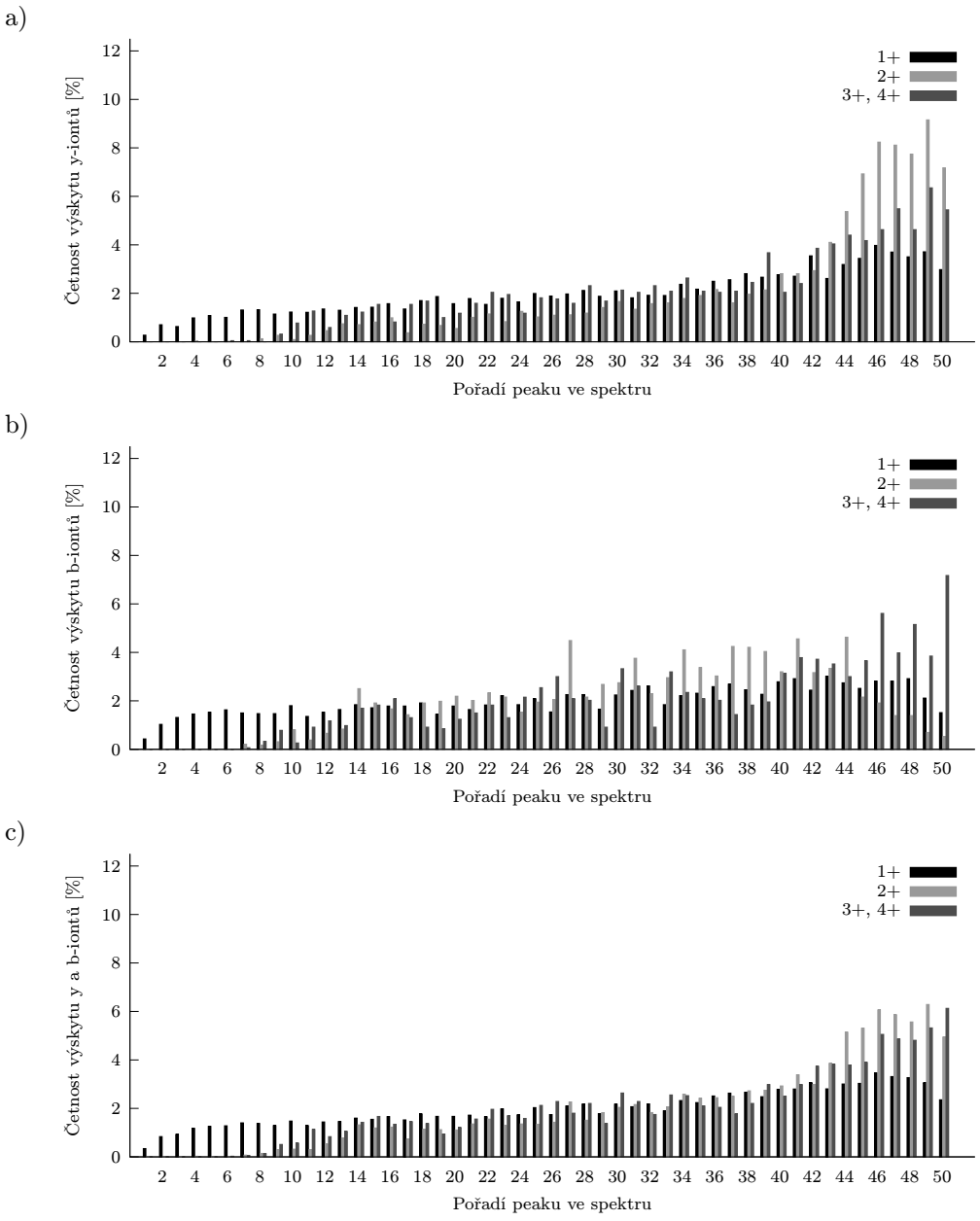
4.4 Heuristiky

Protože experimentální spektra mohou obsahovat množství různých fragmentových iontů a do databáze ukládáme pouze teoretické y-ionty (resp. y a b-ionty), je zapotřebí před provedením dotazu na databázi získat z testovaného spektra vektor hodnot, který budeme vyhledávat.

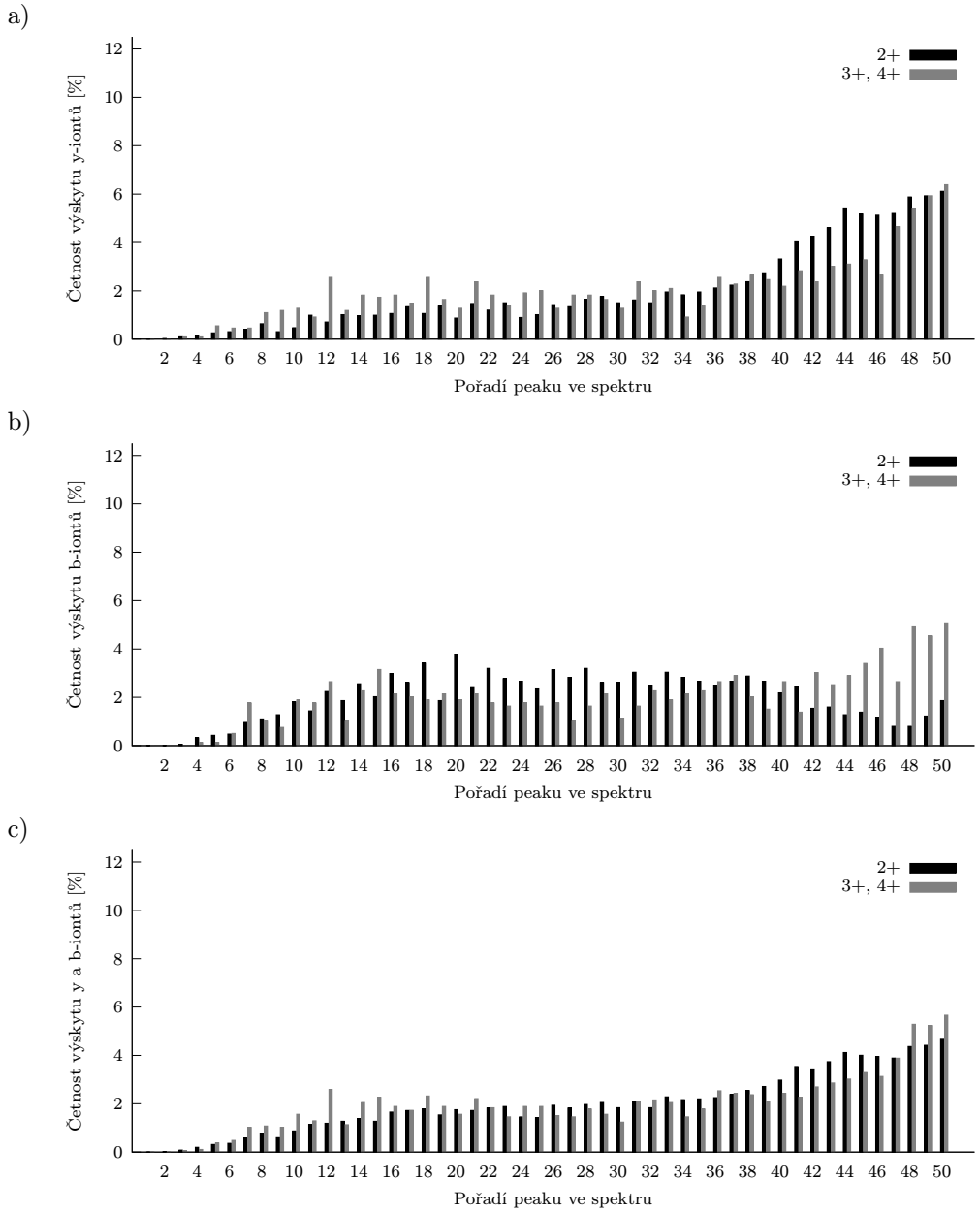
Složky tohoto vektoru by pak v ideálním případě měly odpovídat indexovaným typům iontů. Pro tento účel použijeme dvě heuristiky - první založenou na pozici iontů ve spektru, druhou založenou na hledání komplementárních iontů.

4.4.1 Heuristika založená na pozicích iontů

Peaky experimentálně získaných hmotnostních spekter jsou standardně vzestupně setříděny podle hodnot poměru m/z. Můžeme tedy udělat jednoduchou statistiku založenou na výskytu daného typu iontu v závislosti na pořadí peaku ve spektru. Na obrázcích 4.2 a 4.3 je znázorněno



Obrázek 4.2: Četnost výskytu iontů v závislosti na pozici peaku ve spektru (kolekce *amethyst-gv.xml*; spektra jsou rozdělena do skupin podle hodnoty náboje rodičovského peptidu), a) y-ionty, b) b-ionty, c) y a b-ionty.



Obrázek 4.3: Četnost výskytu iontů v závislosti na pozici peaku ve spektru (kolekce *opal-gv.xml*; spektra jsou rozdělena do skupin podle hodnoty náboje rodičovského peptidu), a) y-ionty, b) b-ionty, c) y a b-ionty.

rozložení iontů typu y a b v experimentálních spektrech kolekcí *amethyst-gv.xml* a *opal-gv.xml*. Spektra jsou vůči sobě zarovnána zprava tj. vzhledem k peaku s maximální hmotností.

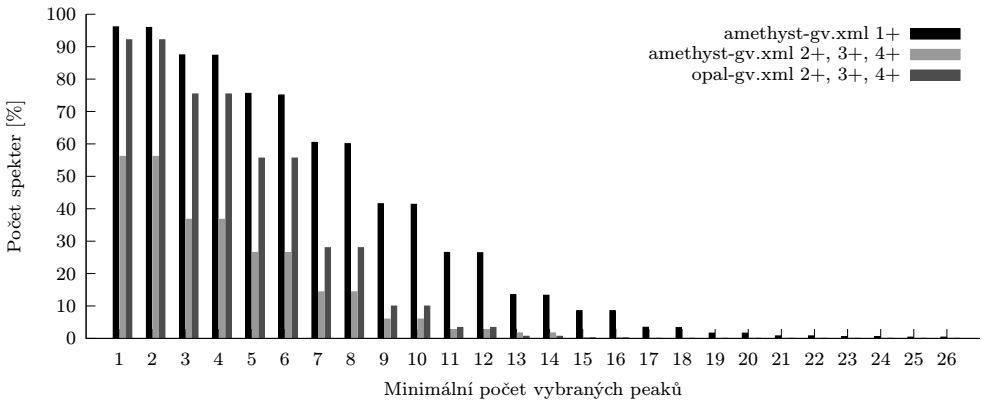
Na závislostech 4.2 c) a 4.3 c) můžeme pozorovat, že daný peak obecně odpovídá y nebo b-iontu s vyšší pravděpodobností, pokud je mezi několika posledními peaky experimentálního spektra. Na grafech 4.2 a) a 4.3 a) je potom vidět, že pro ESI spektra a několik posledních peaků je výrazná pravděpodobnost výskytu y-iontů. Heuristiku proto založíme na jednoduchém principu, a to výběru posledních k peaků experimentálního spektra, které budou odpovídat dotazu.

Abychom přitom zvýšili přesnost metody, zavádíme jednoduchý systém skórování. Pro peptidové sekvence vybrané z databáze vypočteme teoretické hmotnosti b a y-iontů a vyhledáme

je v experimentálním spektru. Sekvence, pro níž bylo ve zkoumaném spektru nalezeno nejvíce b a y-iontů, je pak považována za nejlépe odpovídající experimentálnímu spektru. Složitost skórování je $O(k\,2(N-1)\log N)$, kde k je počet vybraných peptidů, N délka sekvence peptidu a $\log N$ odpovídá vyhledání hmotnosti v setříděném experimentálním spektru (za omezujícího předpokladu, že všechny peptidy mají stejnou délku).

4.4.2 Heuristika založená na hledání párových iontů

Vzhledem ke své vysoké četnosti výskytu hrají při identifikaci peptidů klíčovou roli y a b-ionty (viz tabulka 2.6). Uvažujeme-li peptidovou sekvenci délky n, pak může v tandemovém spektru každému typu iontu teoreticky odpovídat až $n-1$ peaků (viz obrázek 2.13). Heuristika hledání párových iontů vychází z rovnice (2.14), která říká, že součet hmotností dvou komplementárních iontů b_j a y_{k-j} je roven hmotnosti neutrálního rodičovského peptidu zvýšené o 2 Da. Procházíme tedy postupně všechny peaky experimentálně získaného spektra a pro každou hmotnost hledáme předpokládaný komplementární iont. Časová složitost tohoto postupu je pro N peaků $O(N \log N)$, kde $\log N$ odpovídá hledání binárním půlením, protože peaky experimentálního spektra jsou standardně setříděny podle hmotností.



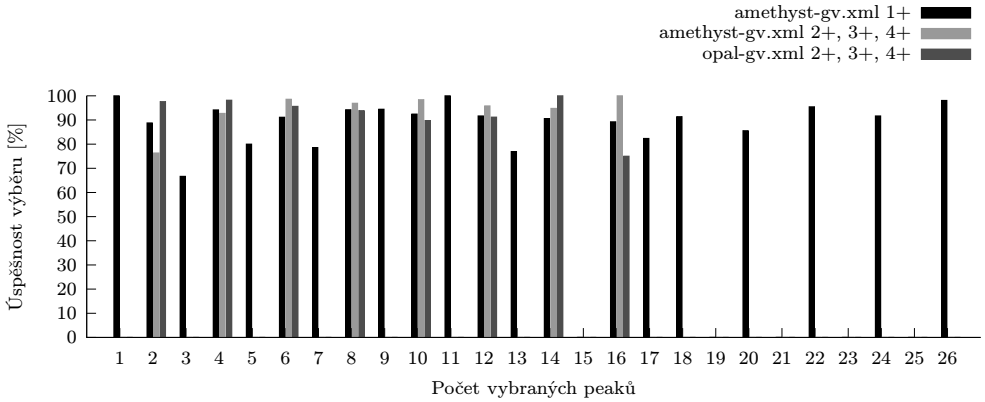
Obrázek 4.4: Hledání párových iontů - počet spekter, ve kterých byl heuristikou vybrán nejméně daný počet peaků.

Na obrázku 4.4 je uvedeno pro kolik procent spekter z testovaných kolekcí dat byl heuristikou vybrán nejméně daný počet peaků. Můžeme pozorovat, že heuristika je pro spektra MALDI (1^+) úspěšnější než pro spektra ESI ($\geq 2^+$). Rovněž lze učinit závěr, že čím větší počet vybraných peaků požadujeme, tím méně spekter bude toto omezení splňovat. Pro úspěšnější MALDI spektra obsahuje aspoň 10 vybraných peaků už jen 40 % spekter. Protože vybíráme dvojice, měl by být počet vybraných peaků roven sudému číslu, v některých případech však dochází k tomu, že počet peaků je liché číslo. Je to způsobeno tím, že oběma peakům z komplementárního páru, přísluší přibližně „stejná" hmotnost. Při hledání byla nastavena povolená odchylka hmotností na 0.2 Da.

Heuristika umožňuje z principu stanovit pouze párovost iontů, nerozlišuje přitom b a y-ionty. Protože rovnice (2.14) analogicky platí i pro c a z-ionty, může teoreticky dojít k záměně páru b a y za pár c a z, vzhledem k velmi vzácnému výskytu iontů c a z, však můžeme chybu způsobenou touto záměnou tolerovat. Častějším případem jsou chyby způsobené tím, že rovnice je splněna díky dvěma náhodným komplementárním hodnotám hmotnosti. [2] Úspěšnost

[2] Pojem „náhoda" můžeme v tomto kontextu chápat i tak, že vznikne dvojice komplementárních iontů, jejichž chemickou strukturu nejsme schopni rozpoznat.

heuristiky pro jednotlivé počty vybraných peaků uvádí obrázek 4.5. Testováno bylo kolik procent z heuristikou vybraného počtu peaků, odpovídalo teoretickým hmotnostem b nebo y-iontů vygenerovaných ze sekvence peptidu odpovídající danému spektru.



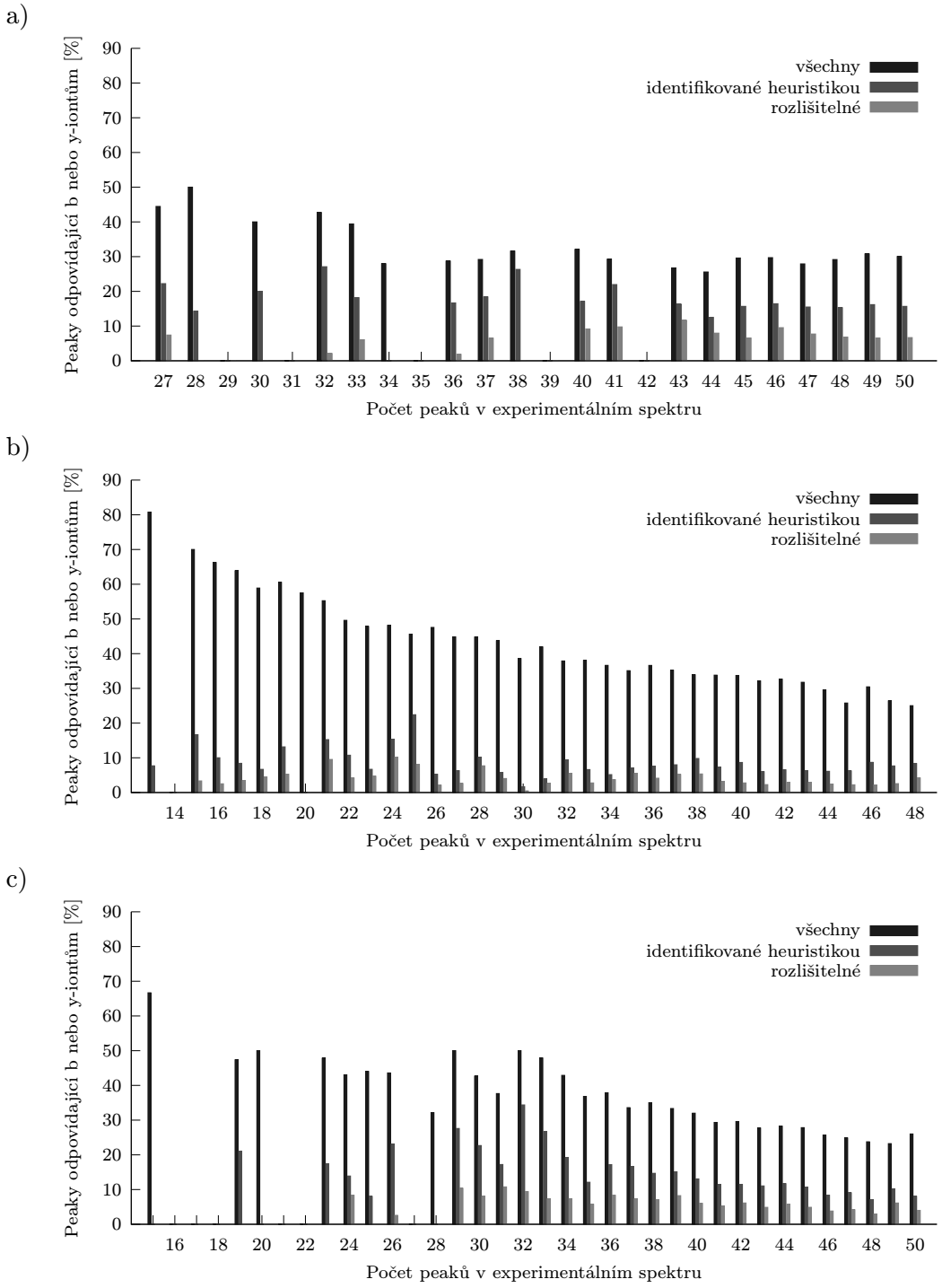
Obrázek 4.5: Hledání párových iontů - úspěšnost výběru pro jednotlivé počty vybraných peaků.

Kvalita výběru se průměrně pohybuje kolem 90%. Nevýhodou heuristiky je však potřeba, aby experimentální spektrum bylo kvalitní v tom smyslu, že obsahuje dostatek komplementárních hmotností odpovídajících b a y-iontům. Výhodou je pak to, že dělení peptidu na fragmenty probíhá ve spektrometru a tudíž výběr správných peaků není ovlivněn případnými posttranslačními modifikacemi peptidů.

Pokud se ve spektru vyskytuje málo komplementárních hmotností, je pravděpodobné, že heuristika vybere jen malý zlomek peaků, které odpovídají b nebo y iontům. Zde se otevírá prostor pro další úvahy, jak počet vybraných peaků zvýšit. Pokud víme, jaký štěpící enzym byl použit při dělení proteinu na peptidy, můžeme se pokusit určit peak b-iontu s maximální hmotností. Např. při dělení trypsinem končí většina peptidových sekvencí K nebo R, poslední b-iont pak teoreticky odpovídá hmotnosti $m(b_{k-1}) = m_p - m(OH) - m(K)$ nebo $m(b_{k-1}) = m_p - m(OH) - m(R)$.

Zároveň je možné heuristiku rozšířit tak, aby rozlišila ve vybraných dvojicích b a y-ionty. Pokud se ve spektrech vyskytuje dostatek a-iontů, můžeme definovat skórování b-iontů na základě vztahu $b_j = a_j + m(CO)$, zároveň můžeme uplatnit znalost faktu, že ionty běžně ztrácejí molekuly vody H_2O nebo amoniaku NH_3, apod. Pokud se pak v experimentálním spektru pro danou hmotnost $m(b_j)$ vyskytují s danou tolerancí i hodnoty $\{m(b_j) - m(CO), m(b_j) - m(CO) - m(H_2O), m(b_j) - m(CO) - m(NH_3)\}$ nebo jejich část, můžeme s vysokou pravděpodobností tvrdit, že se jedná o b-iont, komplementární hodnota jeho hmotnosti pak pravděpodobně odpovídá y-iontu. Můžeme rovněž testovat i násobnou ztrátu molekul H_2O, NH_3, případně jejich kombinace. Dále lze využít faktu, že hmotnosti molekul, které ztratily nějakou část již nemohou být kandidáty na b nebo y-iont a z dalšího zpracování je vyřadit, apod. Pro zvýšení účinnosti podobných přístupů, je vhodná znalost vzorku dat z konkrétního typu spektrometru, pro který budeme analyzovat větší množství spekter.

Na obrázku 4.6 jsou uvedena procentuální zastoupení peaků odpovídajících y nebo b-iontům v závislosti na celkovém počtu peaků v experimentálním spektru. Zobrazen je vždy maximální počet y nebo b-iontů, procento nalezené základní heuristikou párování iontů a část peaků, u kterých lze pomocí skórování založeném na a-iontech správně rozlišit v párech b-ionty od y-iontů. Skórování bylo definováno tak, že pokud k danému iontu existuje jeho komplementární protějšek a zároveň existuje aspoň jedna z hmotností odpovídajících $\{a, a - H_2O, a - NH_3\}$,



Obrázek 4.6: Peaky odpovídající b nebo y-iontům, a) kolekce *amethyst-gv.xml* ($z = 1^+$), b) kolekce *amethyst-gv.xml* ($z \geq 2^+$), c) kolekce *opal-gv.xml* ($z \geq 2^+$).

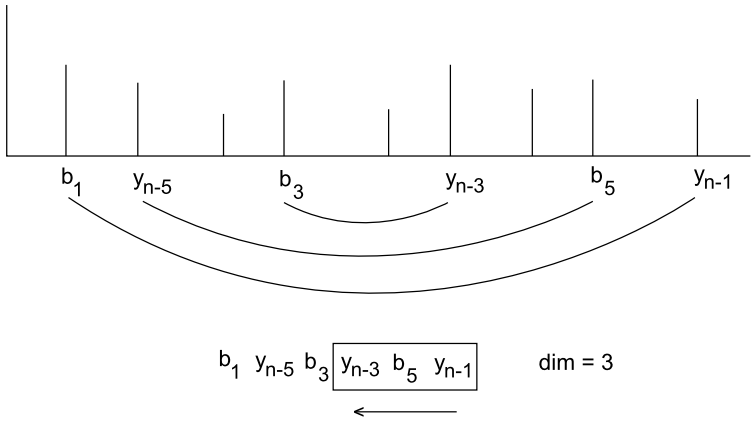
pak je považován za iont b a komplement za y. Je vidět, že na základě a-iontů se podařilo oddělit jen malou část b a y-iontů oproti základní variantě bez rozlišování, a proto budeme v dalších experimentech používat původní verzi heuristiky, která ionty neodlišuje. Popišme si nyní algoritmus identifikace peptidů založený na vyhledávání párových iontů, uvedená varianta pracuje s rozsahovým dotazem, lze však použít i k-NN dotaz.

Algoritmus 4.1

```
for i = 1 to experimental_spectrum.PeaksCount() do
  if (experimental_spectrum.FindMass(experimental_spectrum.ParentPeptideMass()
      + 2 - experimental_spectrum[i].Mass(), tolerance) then
    heuristic_masses.Add(experimental_spectrum[i].Mass());
if (heuristic_masses.Count() < query_dimension) then return false;
for i = 1 to heuristic_masses.Count() - query_dimension + 1 do
  query = heuristic_masses.GetValuesFromTo(
                              heuristic_masses.Count()-query_dimension+1-i+1,
                              heuristic_masses.Count()-i+1);
  result = RangeQuery(query,radius);
  for k = 1 to result.Count() do
    ComputeMatchedIons(experimental_spectrum,
      result[k].PeptideSequence(),tolerance);
  all_results.Append(result);
output item(s) from all_results with maximum number of matched b and y ions;
```

Algoritmus nejprve vybere z experimentálního spektra všechny hmotnosti, které s danou tolerancí splňují rovnici 2.14. Z nich pak při zachování jejich původního vzestupně setříděného pořadí použije poslední *dim*-tici hodnot a provede pro ní dotaz. Pro všechny peptidové sekvence vybrané z databáze vygeneruje teoretické hmotnosti b a y-iontů, následně je vyhledá v experimentálním spektru. V dalším kroku je vybrána *dim*-tice posunutá směrem k nižším hodnotám hmotností experimentálního spektra (opět pracujeme pouze s heuristikou zvolenými hmotnostmi) a postup se opakuje. Pokud heuristika vybere n hodnot provedeme celkem $n - dim + 1$ dotazů. Nakonec je vybrána peptidová sekvence s nejvyšším počtem b a y-iontů nalezených v experimentálním spektru.

Pro testovací účely algoritmus zjednodušujeme tím, že jakmile je nalezena správná peptidová sekvence, další dotaz již neprovádíme a vyhledávání ukončíme. V praxi pak odpovídající peptidovou sekvenci neznáme a můžeme vyhledávání ukončit ve chvíli, kdy počet nalezených b a y-iontů přesáhne stanovenou mez. Abychom mohli aplikovat algoritmus identifikace peptidu, musí být heuristikou z experimentálního spektra vybráno nejméně *dim* hodnot hmotností, v opačném případě považujeme identifikaci tímto postupem za neúspěšnou. Heuristika standardně vyhledává pouze nemodifikované peptidové sekvence, vyhledávání modifikací je zde komplikováno tím, že vzhledem k nízké úspěšnosti postupu neoddělujeme iontové série b a y-iontů.



Obrázek 4.7: Princip heuristiky párování iontů.

4.5 Vyhledávání peptidových modifikací

V této části se zaměříme na vyhledávání modifikovaných peptidových sekvencí. Změna hmotnosti aminokyseliny nebo bodová mutace jsou totiž velmi častým jevem, který v praxi nelze ponechat bez povšimnutí.

Počet výskytů	Typ modifikace	Změna hmotnosti
824	C	57,01
403	N	1
138	Q	-17,016
124	Q	1
122	M	16
69	V → L	14,0156
65	G → S	30,0106
52	A → S	15,9949
40	K → R	28,0061
27	R → Q	-28,0425

Počet výskytů	Typ modifikace	Změna hmotnosti
27	R → K	-28,0061
27	I → V	-14,0156
24	T → S	-14,0156
24	A → G	-14,0157
23	S → T	14,0156
22	N → S	-27,0109
22	K → G	-71,0735
21	G → A	14,0157
21	D	42
17	V → F	48

Tabulka 4.2: Přehled 20 nejčastějších modifikací a bodových mutací aminokyselin ve sjednocení kolekcí *amethyst-gv.xml* a *opal-gv.xml*.

4.5.1 Intervalový dotaz

Pro vyhledání vektorů, které odpovídají modifikovaným peptidům, se nabízí použití především maximové a nemodifikované Hausdorffovy metriky. Změna hmotnosti aminokyseliny se totiž obvykle projevuje posunem více složek vektoru a uvedené vzdálenosti mají výhodu v tom, že tyto přírůstky nekumulují. Můžeme tedy předpokládat, že např. vzdálenost $L_\infty = 57.01$ odpovídá karbamidaci cysteinu, $L_\infty = 16$ oxidaci methioninu, atd. Povšimněme si, že při použití metriky ztrácíme informaci o tom, zda se původně jednalo o přírůstek nebo úbytek hmotnosti. V praxi se to však projeví pouze tím, že budeme z databáze vybírat o něco vyšší počet položek odpovídající oběma alternativám.

Při použití maximové metriky a hledání modifikací však narážíme na problém selekce velkého množství objektů. Pokud bychom chtěli testovat případy, že spektrum obsahuje jednu modifikaci s výraznou odchylkou hmotnosti nebo několik modifikací, které mají v součtu opět výraznou odchylku, bude mít rozsahový dotaz velký rádius. Např. při hledání objektů ve vzdálenosti $L_\infty = 2 \times 57.01 = 114.02$ vybereme všechny objekty, které mají nejvýše tuto vzdálenost, a přitom nás zajímají pouze ty, které jsou v toleranci 114.02 ± 0.1, apod. Tato myšlenka nás přivádí k definici intervalového dotazu tvořeného pomocí hyperprstence.

$$I(q, r_{min}, r_{max}) = \{o \in X, d(o, q) \geq r_{min} \wedge d(o, q) \leq r_{max}\} \tag{4.1}$$

Analogicky jako odfiltrováváme u rozsahového dotazu oblasti, které jsou větší než ohraničující poloměr r (viz rovnice 3.15), v případě intervalového dotazu odřezáváme regiony, které nepřekrývají interval dotazu $\langle r_{min}, r_{max} \rangle$. Filtrační podmínku můžeme v případě vnitřních uzlů M-stromu vyjádřit následovně

$$d(O_i, Q) < r_{Q_{min}} - r_{O_i} \vee d(O_i, Q) > r_{Q_{max}} + r_{O_i}. \tag{4.2}$$

Pro odstranění regionů s využitím předpočítané vzdálenosti od rodičovského objektu O_p ji můžeme přepsat jako

$$d(O_p, Q) + d(O_i, O_p) < r_{Q_{min}} - r_{O_i} \vee |d(O_p, Q) - d(O_i, O_p)| > r_{Q_{max}} + r_{O_i}. \quad (4.3)$$

V případě listových uzlů je situace obdobná

$$d(O_i, Q) < r_{Q_{min}} \vee d(O_i, Q) > r_{Q_{max}}, \quad (4.4)$$

$$d(O_p, Q) + d(O_i, O_p) < r_{Q_{min}} \vee |d(O_p, Q) - d(O_i, O_p)| > r_{Q_{max}}. \quad (4.5)$$

Nyní upravíme algoritmus rozsahového dotazu pro M-strom (viz sekce 3.3.3) na intervalový dotaz aplikací vylepšených filtračních podmínek. Pro PM-strom je situace podobná, liší se pouze přidáním polí HR a PD (viz sekce 3.4.3).

Algoritmus 4.2

```
1  IntervalQuery(N:node, Q:query, r_min(Q):radius, r_max(Q):radius)
2    let O_p be the parent object of node N;
3    if N is not a leaf then
4      ∀ O_i in N do
5        if |d(O_p,Q)-d(O_i,O_p)| ≤ r_max(Q)+r(O_i)
6        and d(O_p,Q)+d(O_i,O_p) ≥ r_min(Q)-r(O_i) then
7          Compute d(O_i,Q);
8          if d(O_i,Q) ≤ r_max(Q)+r(O_i)
9          and d(O_i,Q) ≥ r_min(Q)-r(O_i) then
10           IntervalQuery(*ptr(T(O_i)),Q,r_min(Q),r_max(Q));
11   else
12     ∀ O_i in N do
13       if |d(O_p,Q)-d(O_i,O_p)| ≤ r_max(Q)
14       and d(O_p,Q)+d(O_i,O_p) ≥ r_min(Q) then
15         Compute d(O_i,Q);
16         if d(O_i,Q) ≤ r_max(Q)
17         and d(O_i,Q) ≥ r_min(Q) then
18           add oid(O_i) to the result;
19 end IntervalQuery;
```

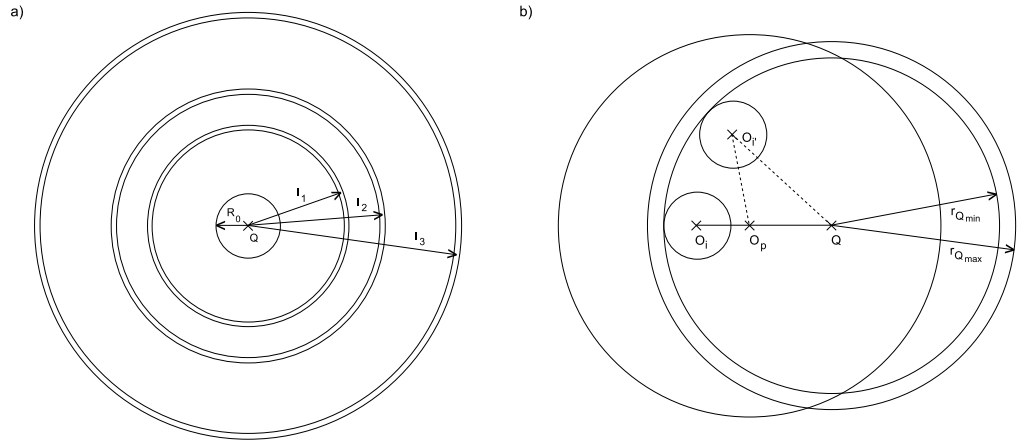
4.5.2 Algoritmus pro identifikaci peptidů s modifikacemi

Databázi peptidů konstruujeme i v případě vyhledávání s modifikacemi identicky jako v sekci 4.3. Při vyhledávání pak kromě jednoho rozsahového dotazu pro nemodifikovanou peptidovou sekvenci provedeme i sadu intervalových dotazů pro vyhledání sekvencí s předem zvolenými modifikacemi. Vyhledání jedné modifikace přitom může odpovídat použití více intervalových dotazů, pokud se vyskytuje v peptidové sekvenci na více místech. Budeme-li tedy vyhledávat např. sekvence s maximálně třemi výskyty karbamidace cysteinu, provedeme tři intervalové dotazy $I(q, 57.01 - tol, 57.01 + tol)$, $I(q, 2 \times 57.01 - tol, 2 \times 57.01 + tol)$ a $I(q, 3 \times 57.01 - tol, 3 \times 57.01 + tol)$, kde tol je povolená odchylka hmotností.

Nepříjemným faktem je, že se zvyšujícím se počtem výskytů modifikací počet potřebných intervalových dotazů exponenciálně roste. Povolíme-li maximálně m výskytů dané modifikace, pak počet intervalových dotazů stanovíme jako

$$\sum_{k=1}^{m} \left(\begin{array}{c} n+k-1 \\ k \end{array} \right), \quad (4.6)$$

kde n je počet modifikací aminokyselin (kombinační číslo odpovídá počtu kombinací s opakováním). Např. při vyhledávání $n = 3$ různých modifikací, tak provedeme pro $m = 1$ povolených výskytů 3 intervalové dotazy, pro $m = 2$ pak 9 a pro $m = 3$ dokonce 19 intervalových dotazů.



Obrázek 4.8: Intervalový dotaz - a) využití rozsahového dotazu R_0 a sady intervalových dotazů I_x pro identifikaci peptidových sekvencí s modifikacemi, b) rozšíření filtračních podmínek pro odřezání regionů uvnitř prstence intervalového dotazu.

Přestože počet známých modifikací aminokyselin lze v současné době již počítat na stovky [14], při analýze konkrétního vzorku hmotnostní spektrometrií lze ze zkušenosti vytipovat pouze několik málo modifikací, které se v získaném spektru budou pravděpodobně vyskytovat. Při vyhledávání rovněž nemusíme nutně použít všechny kombinace výskytů daných modifikací, ale můžeme zvolit jen některé z nich.

Při použití intervalového dotazu je rovněž potřeba upravit skórovací algoritmus. Pro peptidové sekvence vybrané z databáze generujeme teoretické hodnoty hmotností iontů tak, že aplikujeme posuny hmotností způsobené hledanými modifikacemi. Např. pro intervalový dotaz, který vyhledává jeden výskyt oxidovaného methioninu (+ 16 Da) za současného výskytu jednoho karbamidovaného cysteinu (+ 57.01 Da) ve vzdálenosti $L_\infty = 57.01+16 = 73.01$ použijeme při generování hmotností podle teoretické sekvence na methionin přírůstek $M_r(M)+16$ a na cystein $M_r(C)+57.01$, kde M_r je molekulová relativní hmotnost nemodifikovaných aminokyselinových zbytků. Ohodnocení výsledných kandidátů na peptidovou sekvenci nejlépe odpovídajících experimentálnímu spektru, pak provádíme opět na základě nejvyššího počtu nalezených b a y-iontů. Uveďme si nyní algoritmus pro vyhledávání peptidů s modifikacemi založený na použití intervalového dotazu.

Algoritmus 4.3

```
for i = 1 to maximum_number_of_window_shifts do
  query = experimental_spectrum.GetPeakMassesFromTo(
            experimental_spectrum.PeaksCount()-query_dimension+1-i+1,
            experimental_spectrum.PeaksCount()-i+1);
  for j = 0 to modifications.Count() do
    if j = 0 then result = RangeQuery(query,radius);
    else result = IntervalQuery(query,
                    modifications[j].Mass()-tolerance,
                    modifications[j].Mass()+tolerance);
    for k = 1 to result.Count() do
      if j = 0 then ComputeMatchedIons(experimental_spectrum,
                      result[k].PeptideSequence(),tolerance);
      else ComputeMatchedIons(experimental_spectrum,
                      result[k].PeptideSequence(),tolerance,modifications[j]);
    all_results.Append(result);
output item(s) from all_results with maximum number of matched b and y ions;
```

Podobně jako při generování databáze i algoritmus pro vyhledávání používá okénko o velikosti dim, které odpovídá dimenzi indexovaných vektorů. Nebereme tedy v úvahu pouze posledních dim hmotností experimentálního spektra, ale okénko několikrát posuneme o daný krok (v našem případě $krok = 1$) směrem k nižším hmotnostem experimentálního spektra, přičemž vyhledávání opakujeme pro každý posun okénka. Tento princip byl zvolen proto, aby při malé dimenzi např. $dim = 3$ nebyla identifikace založena pouze na posledních 3 peacích zkoumaného spektra, a nedocházelo tak k velké ztrátě informací.

V závislosti na aktuální pozici okénka tedy sestavíme dotaz $query$, který obsahuje dim hodnot hmotností experimentálního spektra. Následně provedeme na objekt $query$ jeden rozsahový dotaz a sadu intervalových dotazů pro zvolené kombinace modifikací. Pro peptidové sekvence vybrané z databáze přitom vždy generujeme teoretické hodnoty hmotností b a y-iontů, vyhledáváme je v experimentálním spektru a výsledná skóre zaznamenáváme. Na konec ze všech kandidátů na peptidovou sekvenci vybereme jednu nebo několik s nejlepším skóre tj. nejvyšším počtem nalezených b a y-iontů. Dodejme, že pro interpretaci každého hmotnostního spektra se použije nejvýše $wmax$ rozsahových dotazů a $k \times wmax$ intervalových dotazů, kde k je počet intervalů a $wmax$ maximální počet okének.

4.6 Logaritmická vzdálenost

Situace, kdy si mezi vektorem indexovaným v databázi a vektorem dotazu vzájemně neodpovídá pouze jedna nebo několik málo složek na stejných pozicích nás navádí k použití vzdálenosti, která by vystihla i podobnost takovýchto objektů. Mějme např. vektory $\vec{x} = \{200, 300, 400, 500\}$, $\vec{y} = \{200, 300, 460, 500\}$ a $\vec{z} = \{210, 305, 420, 475\}$, můžeme pozorovat, že z hlediska identifikace peptidu jsou si blíže vektory $\vec{x}$ a $\vec{y}$ než $\vec{x}$ a $\vec{z}$. Eukleidovská vzdálenost $\vec{x}$ a $\vec{z}$ je přitom naopak menší než vzdálenost vektorů $\vec{x}$ a $\vec{y}$. Zadefinujme si proto logaritmickou vzdálenost jako

$$d_{log}(\vec{x}, \vec{y}) = \sum_{i=1}^{n} d_l(x_i, y_i) \tag{4.7}$$

$$\begin{aligned} d_l(x_i, y_i) &= \log |x_i - y_i|, && |x_i - y_i| > 1 \\ &= 0, && jinak. \end{aligned}$$

Funkce logaritmus způsobí, že vektory, které se liší pouze v jedné nebo několika málo složkách byť o velký přírůstek, si budou blíže než vektory, které se liší ve větším počtu složek o malé přírůstky, a proto si $\vec{x}$ a $\vec{y}$ budou blíže než $\vec{x}$ a $\vec{z}$. Tato vlastnost je výhodná právě v případě, že se z konce experimentálního spektra snažíme vybrat souvislou dim-tici y-iontů, která je však narušena přítomností několika málo jiných hodnot hmotnosti (peaků).

Logaritmus však znevýhodňuje i vektory, které mají u všech složek malou „konstantní" odchylku, přičemž např. vektory $\vec{x'} = \{200, 300, 400, 500\}$ a $\vec{y'} = \{210, 310, 410, 510\}$ mohou popisovat stejné nebo podobné peptidové sekvence. K posunu hodnot totiž běžně dochází modifikací některé aminokyseliny, bodovou mutací aminokyselin nebo kontaminací celého spektra (např. ionty sodíku Na^+).

Funkce (4.7) obecně není metrika, protože nesplňuje trojúhelníkovou nerovnost (3.4) a narušuje axiom pozitivity (3.2). Trojúhelníková nerovnost je však splněna za předpokladu, že $\forall \vec{x}, \vec{y}, \vec{z}$ platí

$$d_{log}(\vec{x}, \vec{y}) + d_{log}(\vec{y}, \vec{z}) \quad \geq \quad d_{log}(\vec{x}, \vec{z}),$$

$$\sum_{i=1}^{n} d_l(x_i, y_i) + \sum_{i=1}^{n} d_l(y_i, z_i) \quad \geq \quad \sum_{i=1}^{n} d_l(x_i, z_i).$$

Je-li pro dané vektory $\vec{x}, \vec{y}, \vec{z}$ podmínka $|x_i - y_i| > 1$ splněna a-krát, podmínka $|y_i - z_i| > 1$ b-krát a podmínka $|x_i - z_i| > 1$ je splněna c-krát, můžeme nerovnici upravit následovně

$$\begin{aligned}
\sum_{i=1}^{a} \log|x_i - y_i| + \sum_{i=1}^{b} \log|y_i - z_i| &\geq \sum_{i=1}^{c} \log|x_i - z_i|, \\
\log \prod_{i=1}^{a} |x_i - y_i| + \log \prod_{i=1}^{b} |y_i - z_i| &\geq \log \prod_{i=1}^{c} |x_i - z_i|, \\
\prod_{i=1}^{a} |x_i - y_i| \prod_{i=1}^{b} |y_i - z_i| &\geq \prod_{i=1}^{c} |x_i - z_i|.
\end{aligned}$$

Nadále budeme předpokládat, že pro spektrometrická data součin na levé straně nerovnice nabývá vyšších hodnot než součin na pravé straně a tudíž trojúhelníková nerovnost je splněna. Ověření tohoto předpokladu a vhodnosti použití logaritmické vzdálenosti na reálných datech provedeme v experimentální části (viz sekce 4.7.2).

Pokud předpokládáme, že pro libovolnou trojici objektů je trojúhelníková nerovnost splněna, zbývá vyřešit narušení axiomu pozitivity. Z definice logaritmické vzdálenosti totiž plyne, že pokud $x \neq y \Rightarrow d_l(x, y) = 0$ pro $|x - y| \leq 1$. Funkci, která takto narušuje axiom pozitivity označujeme jako pseudometriku, můžeme jí však transformovat na metriku, pokud předpokládáme, že libovolná dvojice objektů s nulovou vzdáleností odpovídá jednomu objektu. [33]

4.7 Testování metrik heuristikou založenou na pozicích iontů

Na základě heuristiky založené na pozicích iontů (viz sekce 4.4.1) otestujeme vhodnost různých typů metrik pro identifikaci peptidů. Pro jednoduchost budeme pracovat pouze s y-ionty a ESI spektry s nábojem 2^+ z kolekce dat *amethyst-gv.xml* a zkusíme v databázi vyhledat posledních $dim \leq 8$ hodnot hmotností experimentálních spekter. Povolená odchylka teoretických a experimentálních hmotností je ± 0.1 Da.

Soubor	Dimenze vektorů	Počet vektorů	Počet peptidů	Min. délka peptidu
opal-amet-gv_y_1_1.avs	1	245574	23326	6
opal-amet-gv_y_2_1.avs	2	222248	23326	6
opal-amet-gv_y_3_1.avs	3	198922	23326	6
opal-amet-gv_y_4_1.avs	4	175596	23326	6
opal-amet-gv_y_5_1.avs	5	152270	23326	6
opal-amet-gv_y_6_1.avs	6	128944	20855	7
opal-amet-gv_y_7_1.avs	7	108089	18398	8
opal-amet-gv_y_8_1.avs	8	89691	16288	9

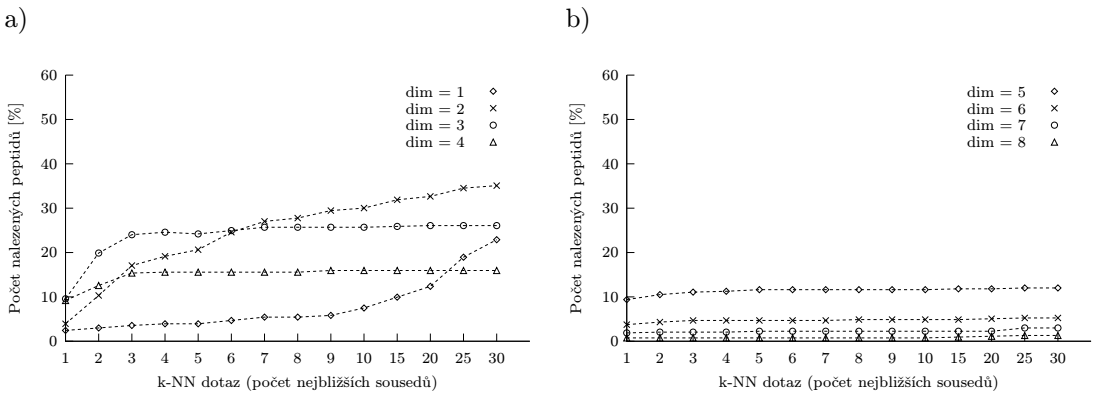
Tabulka 4.3: Soubory pro naplnění databáze (nastavení parametrů - počet proteinů: 510; štěpící enzym: trypsin; maximální počet vynechaných míst štěpení: 1; maximální délka sekvence peptidu: 20; typ generovaných iontů: y; krok: 1).

4.7.1 Eukleidovská vzdálenost

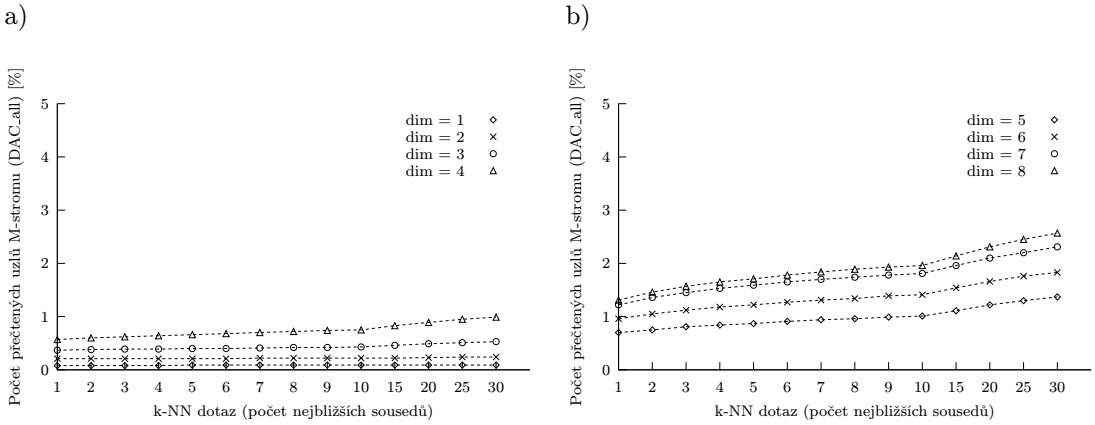
V následujícím experimentu budeme s využitím Eukleidovské vzdálenosti zkoumat vliv velikosti dimenze vyhledávaných vektorů a množství vybraných nejbližších sousedů v M-stromu na počet správně identifikovaných peptidových sekvencí a na počet přečtených uzlů DAC_{all}.

Délka vektoru	Soubor pro naplnění databáze	Kapacita listových uzlů	Počet uzlů M-stromu		
			celkem	vnitřních	listových
1	opal-amet-gv_y_1_1.avs	71	5520	170	5350
2	opal-amet-gv_y_2_1.avs	68	5118	154	4964
3	opal-amet-gv_y_3_1.avs	66	4558	140	4418
4	opal-amet-gv_y_4_1.avs	65	4060	121	3939
5	opal-amet-gv_y_5_1.avs	63	3610	107	3503
6	opal-amet-gv_y_6_1.avs	62	3110	92	3018
7	opal-amet-gv_y_7_1.avs	61	2679	83	2596
8	opal-amet-gv_y_8_1.avs	60	2245	69	2176

Tabulka 4.4: Eukleidovská vzdálenost - nastavení parametrů experimentu (struktura: M-strom; kapacita vnitřních uzlů: 50; kolekce experimentálně získaných spekter: *amethyst-gv.xml*; počet testovaných spekter: 533; náboj rodičovských peptidů: 2^+).



Obrázek 4.9: Eukleidovská vzdálenost (počet nalezených peptidů).



Obrázek 4.10: Eukleidovská vzdálenost (počet přečtených uzlů).

k-NN	$dim = 1$				$dim = 2$				$dim = 3$				$dim = 4$			
	Nalezeno		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}	
	-	%	-	%	-	%	-	%	-	%	-	%	-	%	-	%
1	13	2.44	4.60	0.08	21	3.94	10.79	0.21	51	9.57	16.71	0.37	49	9.19	23.12	0.57
2	16	3.00	4.61	0.08	55	10.32	10.81	0.21	106	19.89	17.24	0.38	67	12.57	24.16	0.60
3	19	3.56	4.64	0.08	91	17.07	10.83	0.21	128	24.02	17.64	0.39	82	15.38	25.12	0.62
4	21	3.94	4.69	0.08	102	19.14	10.87	0.21	131	24.58	17.83	0.39	83	15.57	25.83	0.64
5	21	3.94	4.69	0.09	110	20.64	10.91	0.21	129	24.20	18.20	0.40	83	15.57	26.85	0.66
6	25	4.69	4.71	0.09	131	24.58	10.94	0.21	133	24.95	18.44	0.40	83	15.57	27.71	0.68
7	29	5.44	4.74	0.09	144	27.02	11.02	0.22	137	25.70	18.68	0.41	83	15.57	28.49	0.70
8	29	5.44	4.76	0.09	148	27.77	11.07	0.22	137	25.70	19.02	0.42	83	15.57	29.26	0.72
9	31	5.82	4.79	0.09	157	29.46	11.11	0.22	137	25.70	19.31	0.42	85	15.95	29.99	0.74
10	40	7.50	4.81	0.09	160	30.02	11.18	0.22	137	25.70	19.52	0.43	85	15.95	30.60	0.75
15	53	9.94	4.89	0.09	170	31.89	11.41	0.22	138	25.89	20.89	0.46	85	15.95	33.53	0.83
20	66	12.38	4.95	0.09	174	32.65	11.71	0.23	139	26.08	22.14	0.49	85	15.95	36.27	0.89
25	101	18.95	5.05	0.09	184	34.52	12.03	0.24	139	26.08	23.22	0.51	85	15.95	38.40	0.95
30	122	22.89	5.08	0.09	**187**	**35.08**	**12.29**	**0.24**	**139**	**26.08**	**24.28**	**0.53**	85	15.95	40.10	0.99

k-NN	$dim = 5$				$dim = 6$				$dim = 7$				$dim = 8$			
	Nalezeno		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}	
	-	%	-	%	-	%	-	%	-	%	-	%	-	%	-	%
1	50	9.38	25.20	0.70	20	3.75	29.86	0.96	10	1.88	32.69	1.22	4	0.75	29.32	1.31
2	56	10.51	27.09	0.75	23	4.32	32.59	1.05	11	2.06	36.36	1.36	4	0.75	32.68	1.46
3	59	11.07	29.17	0.81	25	4.69	34.74	1.12	11	2.06	38.78	1.45	4	0.75	35.28	1.57
4	60	11.26	30.45	0.84	25	4.69	36.68	1.18	11	2.06	40.88	1.53	4	0.75	36.94	1.65
5	62	11.63	31.46	0.87	25	4.69	38.07	1.22	12	2.25	42.69	1.59	4	0.75	38.48	1.71
6	62	11.63	32.93	0.91	25	4.69	39.52	1.27	12	2.25	44.08	1.65	4	0.75	39.95	1.78
7	62	11.63	33.92	0.94	25	4.69	40.64	1.31	12	2.25	45.46	1.70	4	0.75	41.25	1.84
8	62	11.63	34.70	0.96	26	4.88	41.80	1.34	12	2.25	46.55	1.74	4	0.75	42.39	1.89
9	62	11.63	35.62	0.99	26	4.88	43.08	1.39	12	2.25	47.60	1.78	4	0.75	43.38	1.93
10	62	11.63	36.58	1.01	26	4.88	43.87	1.41	12	2.25	48.50	1.81	4	0.75	44.06	1.96
15	63	11.82	39.95	1.11	26	4.88	47.90	1.54	12	2.25	52.53	1.96	5	0.94	48.14	2.14
20	63	11.82	44.15	1.22	27	5.07	51.62	1.66	12	2.25	56.24	2.10	6	1.13	51.92	2.31
25	64	12.01	47.01	1.30	28	5.25	54.65	1.76	16	3.00	59.06	2.20	7	1.31	54.92	2.45
30	64	12.01	49.38	1.37	28	5.25	57.00	1.83	16	3.00	61.84	2.31	7	1.31	57.61	2.57

Tabulka 4.5: Eukleidovská vzdálenost (počet nalezených peptidů a počet přečtených uzlů).

Z naměřených dat můžeme vypozorovat, že počet identifikovaných peptidů je nejvyšší pro $2 \leq dim \leq 3$. Potvrdila se výhodnost zavedeného skórování (viz sekce 4.4.1), např. pro $dim = 2$ a 30-NN je počet identifikovaných peptidů $8,9\times$ vyšší, než kdybychom uvažovali pouze jednoho nejbližšího souseda, pro $dim = 3$ pak $2,7\times$ vyšší. Je-li $dim > 3$ počet identifikovaných peptidů klesá. Mezi posledními hodnotami v experimentálních spektrech je sice nejvyšší pravděpodobnost výskytu y-iontů, nicméně pravděpodobnost, že vybereme souvislou dim-tici y-iontů, která bude odpovídat některému z indexovaných objektů s rostoucí dimenzí klesá. Peaky odpovídající y-iontům jsou totiž proloženy i těmi, které y-iontům neodpovídají, případně mohou některé y-ionty ve spektru chybět. Je-li souvislost dim-tice y-iontů narušena byť jen jedinou takovou změnou, pravděpodobnost, že daný objekt bude Eukleidovskou metrikou stále zařazen mezi k vybíraných nejbližších sousedů je v praxi minimální.

Pro $dim = 1$ je počet identifikovaných peptidů menší než pro $2 \leq dim \leq 3$, protože indexování jednoho peaku (hmotnosti) způsobuje hity ve stejné vzdálenosti pro více objektů a ten, který odpovídá zkoumanému spektru nemusí být nutně mezi k nejbližšími sousedy.

Eukleidovská vzdálenost zde má určitou výhodu v tom, že počet přečtených uzlů roste se zvyšujícím se počtem vybíraných sousedů a rostoucí dimenzí jen minimálně a prochází se tak pouze malá část stromové struktury.

4.7.2 Logaritmická vzdálenost

Zopakujeme předchozí experiment s použitím logaritmické vzdálenosti (viz sekce 4.6). Budeme přitom zároveň zjišťovat, zda-li logaritmická vzdálenost pro spektrometrická data splňuje trojúhelníkovou nerovnost a je-li tak vhodné její použití spolu s metrickými indexovacími metodami. Parametr INEQ udává procento splnění trojúhelníkové nerovnosti mezi třemi libovolnými objekty na vzorku 1000 vektorů daného souboru.

Délka vektoru	Soubor pro naplnění databáze	Kapacita listových uzlů	Počet uzlů M-stromu			INEQ [%]
			celkem	vnitřních	listových	
1	opal-amet-gv_y_1_1.avs	71	5603	168	5435	99.63
2	opal-amet-gv_y_2_1.avs	68	5105	149	4956	99.93
3	opal-amet-gv_y_3_1.avs	66	4551	133	4418	99.99
4	opal-amet-gv_y_4_1.avs	65	4016	119	3897	100.00
5	opal-amet-gv_y_5_1.avs	63	3612	108	3504	100.00
6	opal-amet-gv_y_6_1.avs	62	3082	91	2991	100.00
7	opal-amet-gv_y_7_1.avs	61	2648	79	2569	100.00
8	opal-amet-gv_y_8_1.avs	60	2221	67	2154	100.00

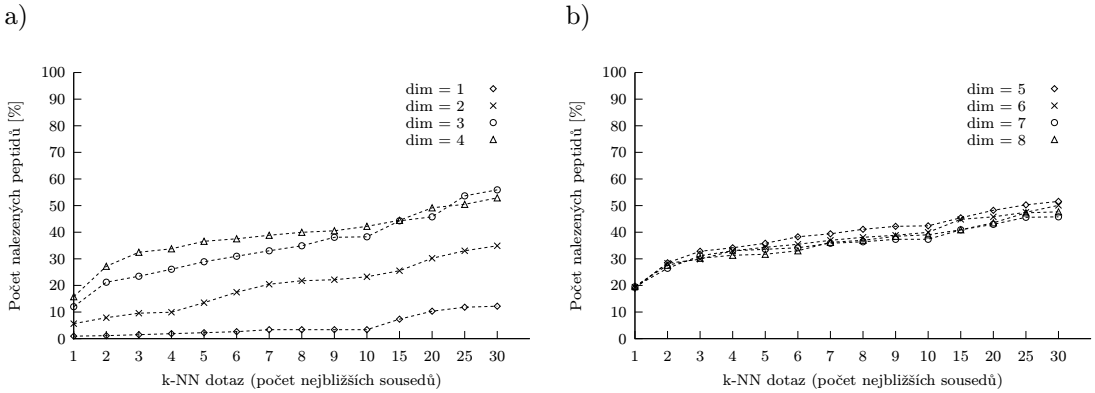
Tabulka 4.6: Logaritmická vzdálenost - nastavení parametrů experimentu (struktura: M-strom; kapacita vnitřních uzlů: 50; kolekce experimentálně získaných spekter: *amethyst-gv.xml*; počet testovaných spekter: 533; náboj rodičovských peptidů: 2^+; základ logaritmu: 100).

Z výsledků experimentu můžeme vypozorovat, že trojúhelníková nerovnost je pro testovaná data splněna s vysokou pravděpodobností. Úspěšnost logaritmické vzdálenosti je přitom pro $3 \leq dim \leq 4$ a k-NN ($k \geq 10$) přibližně $1,5\times$ až $3,3\times$ vyšší než u Eukleidovské. Zároveň můžeme pozorovat, že se zvyšující se dimenzí úspěšnost identifikace stagnuje, zatímco počet přístupů na disk skokově roste. Pro $dim = 8$ a 30-NN pak počet přečtených uzlů M-stromu dosahuje téměř 64%.

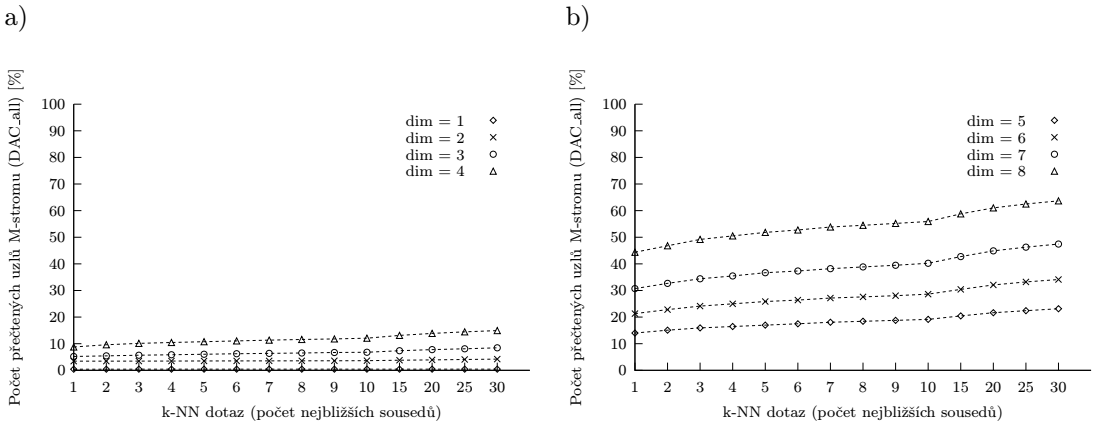
k-NN	$dim = 1$				$dim = 2$				$dim = 3$				$dim = 4$			
	Nalezeno		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}	
	-	%	-	%	-	%	-	%	-	%	-	%	-	%	-	%
1	5	0.94	23.14	0.41	30	5.63	175.46	3.44	64	12.01	236.96	5.21	84	15.76	352.16	8.77
2	6	1.13	23.55	0.42	42	7.88	176.00	3.45	113	21.20	246.66	5.42	145	27.20	385.35	9.60
3	8	1.50	23.85	0.43	51	9.57	176.50	3.46	125	23.45	258.33	5.68	173	32.46	406.69	10.13
4	10	1.88	24.10	0.43	53	9.94	177.46	3.48	139	26.08	266.14	5.85	180	33.77	419.64	10.45
5	12	2.25	24.28	0.43	72	13.51	179.27	3.51	154	28.89	274.67	6.04	195	36.59	432.05	10.76
6	14	2.63	24.39	0.44	93	17.45	180.59	3.54	165	30.96	283.40	6.23	200	37.52	443.60	11.05
7	18	3.38	24.51	0.44	109	20.45	181.21	3.55	176	33.02	289.52	6.36	207	38.84	455.32	11.34
8	18	3.38	24.55	0.44	116	21.76	182.06	3.57	186	34.90	296.20	6.51	213	39.96	465.31	11.59
9	18	3.38	24.56	0.44	118	22.14	183.16	3.59	203	38.09	304.30	6.69	216	40.53	473.81	11.80
10	18	3.38	24.56	0.44	124	23.26	184.63	3.62	204	38.27	308.72	6.78	225	42.21	484.56	12.07
15	39	7.32	24.59	0.44	136	25.52	194.19	3.80	236	44.28	335.01	7.36	237	44.47	526.72	13.12
20	55	10.32	24.60	0.44	161	30.21	200.09	3.92	244	45.78	354.23	7.78	262	49.16	556.85	13.87
25	63	11.82	24.60	0.44	176	33.02	207.99	4.07	286	53.66	368.01	8.09	269	50.47	580.23	14.45
30	65	12.20	24.60	0.44	186	34.90	214.92	4.21	**298**	**55.91**	**384.99**	**8.46**	**282**	**52.91**	**599.46**	**14.93**

k-NN	$dim = 5$				$dim = 6$				$dim = 7$				$dim = 8$			
	Nalezeno		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}	
	-	%	-	%	-	%	-	%	-	%	-	%	-	%	-	%
1	101	18.95	505.85	14.00	103	19.32	656.73	21.31	104	19.51	812.87	30.70	105	19.70	985.39	44.37
2	152	28.52	544.93	15.09	151	28.33	702.68	22.80	141	26.45	865.38	32.68	147	27.58	1039.35	46.80
3	175	32.83	575.50	15.93	159	29.83	743.57	24.13	166	31.14	910.61	34.39	161	30.21	1092.28	49.18
4	182	34.15	594.40	16.46	176	33.02	769.87	24.98	176	33.02	939.01	35.46	167	31.33	1121.89	50.51
5	191	35.83	613.15	16.98	183	34.33	796.67	25.85	179	33.58	970.96	36.67	169	31.71	1150.98	51.82
6	204	38.27	631.89	17.49	189	35.46	813.23	26.39	182	34.15	988.03	37.31	176	33.02	1171.40	52.74
7	210	39.40	652.35	18.06	197	36.96	837.16	27.16	191	35.83	1011.32	38.19	193	36.21	1196.04	53.85
8	219	41.09	665.55	18.43	203	38.09	850.19	27.59	194	36.40	1028.48	38.84	197	36.96	1209.97	54.48
9	225	42.21	678.40	18.78	207	38.84	864.36	28.05	199	37.34	1045.55	39.48	205	38.46	1225.90	55.20
10	226	42.40	691.49	19.14	213	39.96	881.64	28.61	199	37.34	1065.33	40.23	208	39.02	1241.69	55.91
15	242	45.40	738.55	20.45	239	44.84	937.31	30.41	218	40.90	1131.27	42.72	218	40.90	1305.49	58.78
20	257	48.22	781.85	21.65	244	45.78	988.77	32.08	229	42.96	1188.36	44.88	232	43.53	1355.44	61.03
25	268	50.28	809.50	22.41	253	47.47	1023.53	33.21	243	45.59	1225.24	46.27	252	47.28	1387.82	62.49
30	**275**	**51.59**	**835.52**	**23.13**	**267**	**50.09**	**1051.62**	**34.12**	244	45.78	1256.35	47.45	254	47.65	1414.23	63.68

Tabulka 4.7: Logaritmická vzdálenost (počet nalezených peptidů a počet přečtených uzlů).



Obrázek 4.11: Logaritmická vzdálenost (počet nalezených peptidů).



Obrázek 4.12: Logaritmická vzdálenost (počet přečtených uzlů).

4.7.3 Porovnání s jinými metrikami

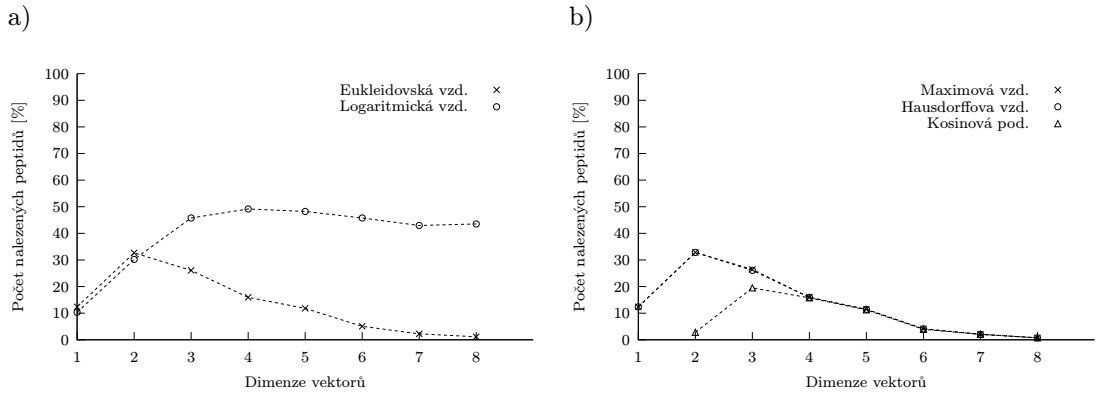
V následujícím experimentu provedeme porovnání Eukleidovské a logaritmické vzdálenosti s maximovou vzdáleností, Hausdorffovou vzdáleností a kosinovou podobností. Závislosti budeme testovat vzhledem k měnící se dimenzi vektorů $1 \leq dim \leq 8$ pro dotazy typu 20-NN. Výpočet kosinové podobnosti má z definice (viz sekce 3.1.4) smysl pro $dim > 1$.

Z definice Hausdorffovy vzdálenosti (viz sekce 3.1.6) víme, že vybírá z obou porovnávaných vektorů vždy složky s minimální vzdáleností, čímž je zaručuje určitou odolnost vůči vzájemnému posunu složek s blízkou hodnotou. Např. vektory $\vec{x} = \{300, a, 400, 500\}$ a $\vec{y} = \{300, 400, 500, b\}$, tak v závislosti na konstantách a a b mohou být stále vyhodnoceny jako podobné. Pro úplnost dodejme, že v experimentech používáme vždy stejně dlouhé vektory, Hausdorffova vzdálenost však umožňuje porovnávat i vektory různých délek.

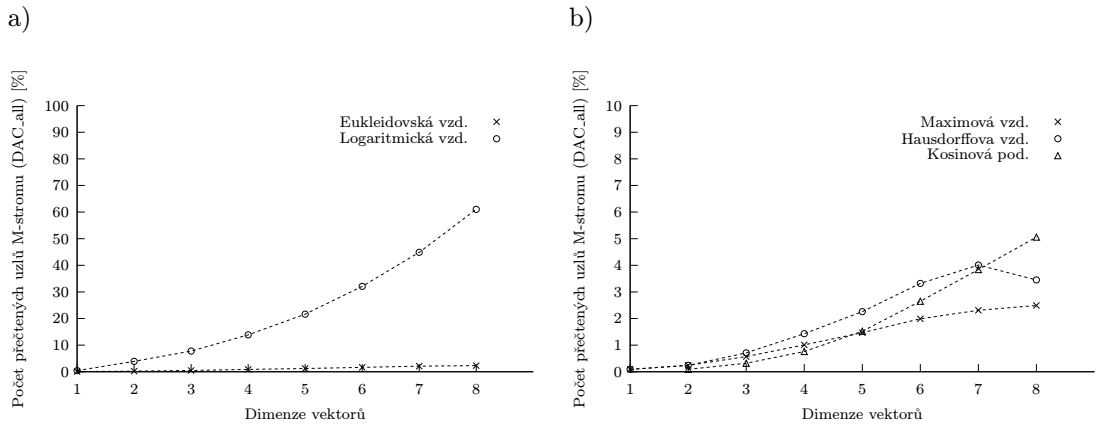
Naměřená data ukazují, že Eukleidovská, maximová a Hausdorffova metrika dávají téměř shodné výsledky. Maximová vzdálenost je přitom oproti Hausdorffově nepatrně lepší, přestože její výpočet neuvažuje posun hodnot přiřazených jednotlivým složkám vektorů a je tedy levnější. Kosinová podobnost je pro $2 \leq dim \leq 3$ slabší než uvedené metriky, v ostatních případech dává srovnatelnou úspěšnost. Nejlepších výsledků dosahuje logaritmická vzdálenost.

dim	**Maximová vzdálenost**				**Hausdorffova vzdálenost**				**Kosinová podobnost**			
	Nalezeno		DAC_{all}		**Nalezeno**		DAC_{all}		**Nalezeno**		DAC_{all}	
	-	**%**	**-**	**%**	**-**	**%**	**-**	**%**	**-**	**%**	**-**	**%**
1	66	12.38	4.95	0.09	66	12.38	4.95	0.09	-	-	-	-
2	**175**	**32.83**	**12.43**	**0.24**	**175**	**32.83**	**12.43**	**0.24**	15	2.81	4.55	0.09
3	141	26.45	25.69	0.57	139	26.08	32.42	0.71	**104**	**19.51**	**14.55**	**0.32**
4	85	15.95	40.61	1.01	85	15.95	58.07	1.43	84	15.76	30.91	0.76
5	61	11.44	53.17	1.47	61	11.44	81.83	2.26	60	11.26	54.79	1.52
6	22	4.13	61.50	1.99	22	4.13	103.13	3.32	21	3.94	81.92	2.65
7	11	2.06	61.49	2.31	11	2.06	105.88	4.01	11	2.06	102.49	3.84
8	4	0.75	55.12	2.49	4	0.75	76.98	3.45	4	0.75	113.74	5.06

Tabulka 4.8: Porovnání s jinými metrikami pro 20 nejbližších sousedů - počty správně identifikovaných peptidů a průměrný počet přečtených uzlů na 1 spektrum (DAC_{all}).



Obrázek 4.13: Porovnání s jinými metrikami (počet nalezených peptidů).



Obrázek 4.14: Porovnání s jinými metrikami (počet přečtených uzlů).

4.8 Testování algoritmu pro vyhledávání modifikací intervalovým dotazem

V této části otestujeme algoritmus pro vyhledávání peptidových modifikací založený na intervalovém dotazu (viz sekce 4.5.2). Budeme pracovat se všemi ESI spektry z testovaných kolekcí, tj. s celou kolekcí *opal-gv.xml* a s kolekcí *amethyst-gv.xml*, ze které vybereme všechna spektra mající náboj rodičovských peptidů 2^+ a více. S ohledem na výsledky experimentu v sekci 4.7.3, budeme testovat pouze maximovou vzdálenost, protože Hausdorffova metrika dosahuje podobných výsledků a protože volíme malou dimenzi vektorů $dim = 3$. Algoritmus vyhledávání pro testovací účely optimalizujeme tím, že jakmile dojde k nalezení správné peptidové sekvence, další posun okénka již neprovedeme a vyhledávání ukončíme.

Kromě jednoho rozsahového dotazu s rádiusem $r_Q = 10$, použijeme celkem 8 intervalových dotazů pro vybrané modifikace, které se v testovaných kolekcích vyskytují nejčastěji (viz tabulka 4.2). Intervalovými dotazy budeme testovat změny hmotností z množiny $\{57.01, 2 \times 57.01, 16, 17.016, 14.0156, 1, 30.0106, 57.01 + 42\}$, které odpovídají modifikacím aminokyselin {C, 2×C, M, Q, V ∨ S ∨ A ∨ T, N ∨ Q, G, C + D}, odchylka hmotností je ±0.1.

4.8.1 Maximální počet posunů okénka

Nejprve se zaměříme na vliv maximálního počtu posunů okénka $wmax$ na počet správně identifikovaných peptidů, na počet přečtených uzlů DAC_{all} a na selektivitu tj. počet objektů vybraných z databáze. Porovnáme přitom varianty algoritmu bez vyhledávání modifikací a s vyhledáváním modifikací.

Nastavení experimentu - struktura: M-strom; kapacita vnitřních uzlů: 50 položek; kapacita listových uzlů: 66 položek; indexovaný soubor: *opal-amet-gv_y_3_1.avs* (viz tabulka 4.3); počet uzlů: 4540; hloubka stromu: 3.

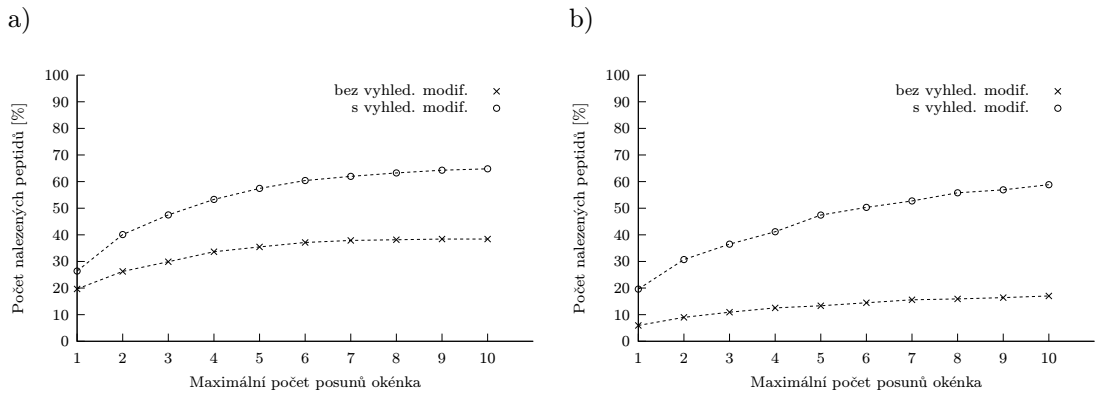
Z naměřených dat můžeme vypozorovat, že úspěšnost algoritmu v závislosti na zvyšujícím se počtu posunů okénka roste. Ve variantě bez vyhledávání modifikací však pro $wmax \geq 7$ úspěšnost pro obě testované kolekce dat začíná stagnovat a zvyšuje se již jen nepatrně. Pro variantu s vyhledáváním modifikací a $wmax = 10$ je počet identifikovaných peptidů v kolekci *amethyst-gv.xml* 1.69× vyšší než bez vyhledávání modifikací a úspěšnost identifikace dosahuje 64.81 %, v kolekci *opal-gv.xml* je pak 3.45× vyšší a úspěšnost dosahuje 58.84 %.

$wmax$	**Bez vyhledávání modifikací**						**S vyhledáváním modifikací**					
	Nalezeno peptidů		DAC_{all}		**Selektivita**		**Nalezeno peptidů**		DAC_{all}		**Selektivita**	
	-	%	-	%	-	%	-	%	-	%	-	%
1	152	19.66	20.69	0.46	39.64	0.02	204	26.39	478.48	10.54	321.00	0.16
2	203	26.26	33.53	0.74	62.94	0.03	310	40.10	870.12	19.17	612.21	0.31
3	231	29.88	44.57	0.98	84.62	0.04	367	47.48	1231.68	27.13	902.16	0.45
4	260	33.64	55.19	1.22	107.99	0.05	412	53.30	1555.42	34.26	1168.79	0.59
5	274	35.45	62.37	1.37	122.18	0.06	444	57.44	1842.49	40.58	1414.54	0.71
6	287	37.13	68.18	1.50	129.66	0.07	467	60.41	2101.85	46.30	1634.70	0.82
7	293	37.90	73.95	1.63	137.65	0.07	479	61.97	2342.15	51.59	1859.66	0.93
8	295	38.16	77.86	1.71	140.49	0.07	489	63.26	2560.60	56.40	2052.07	1.03
9	297	38.42	81.03	1.78	143.13	0.07	497	64.29	2750.99	60.59	2226.10	1.12
10	297	38.42	83.57	1.84	145.08	0.07	501	64.81	2922.06	64.36	2369.81	1.19

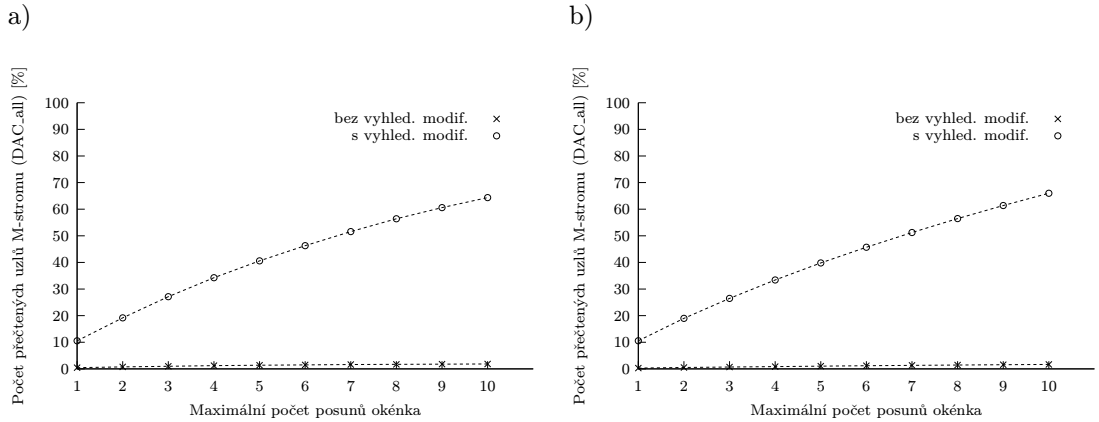
Tabulka 4.9: Úspěšnost vyhledávání, celkový počet přečtených uzlů DAC_{all} a selektivita v závislosti na maximálním počtu posunů okénka - kolekce *amethyst-gv.xml* (spektra s nábojem $z \geq 2^+$); DAC_{all} a selektivita jsou průměrné hodnoty na 1 spektrum.

wmax	**Bez vyhledávání modifikací**						**S vyhledáváním modifikací**					
	Nalezeno peptidů		DAC_{all}		**Selektivita**		**Nalezeno peptidů**		DAC_{all}		**Selektivita**	
	-	%	-	%	-	%	-	%	-	%	-	%
1	37	5.95	13.15	0.29	19.63	0.01	122	19.61	479.63	10.56	342.79	0.17
2	56	9.00	23.76	0.52	33.98	0.02	191	30.71	861.78	18.98	616.93	0.31
3	68	10.93	31.70	0.70	42.95	0.02	227	36.50	1202.03	26.48	877.10	0.44
4	78	12.54	39.14	0.86	51.59	0.03	256	41.16	1516.80	33.41	1124.61	0.57
5	83	13.34	46.45	1.02	60.75	0.03	295	47.43	1807.34	39.81	1354.70	0.68
6	90	14.47	53.61	1.18	70.08	0.04	313	50.32	2074.81	45.70	1574.78	0.79
7	97	15.59	60.08	1.32	77.36	0.04	328	52.73	2326.51	51.24	1785.75	0.90
8	99	15.92	65.95	1.45	82.84	0.04	347	55.79	2564.19	56.48	1989.92	1.00
9	102	16.40	70.98	1.56	87.48	0.04	354	56.91	2786.24	61.37	2186.32	1.10
10	106	17.04	75.68	1.67	91.11	0.05	366	58.84	2995.75	65.99	2368.83	1.19

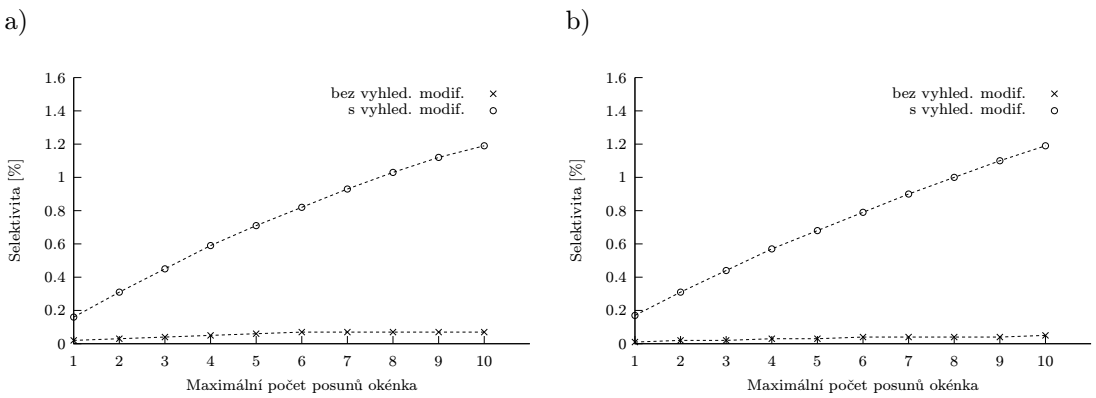
Tabulka 4.10: Úspěšnost vyhledávání, celkový počet přečtených uzlů DAC_{all} a selektivita v závislosti na maximálním počtu posunů okénka - kolekce *opal-gv.xml*; DAC_{all} a selektivita jsou průměrné hodnoty na 1 spektrum.



Obrázek 4.15: Úspěšnost vyhledávání v závislosti na maximálním počtu posunů okénka - a) kolekce *amethyst-gv.xml* (spektra s nábojem $z \geq 2^+$), b) *opal-gv.xml*.



Obrázek 4.16: Celkový počet přečtených uzlů DAC_{all} v závislosti na maximálním počtu posunů okénka - a) kolekce *amethyst-gv.xml* (spektra s nábojem $z \geq 2^+$), b) *opal-gv.xml*.



Obrázek 4.17: Selektivita v závislosti na maximálním počtu posunů okénka - a) kolekce *amethyst-gv.xml* (spektra s nábojem $z \geq 2^+$), b) *opal-gv.xml*.

Nevýhodou vyhledávání modifikací je počet přečtených uzlů DAC_{all}, který se pro obě kolekce pohybuje kolem 65 %. Efektivitu použití algoritmu nám však zachovává selektivita, která pro $wmax = 10$ dosahuje 1.19 %. S rostoucí velikostí databáze lze přitom předpokládat, že počet přečtených uzlů i selektivita budou klesat (viz sekce 4.8.4).

Provedeme-li srovnání s výsledky experimentu ze sekce 4.7.3 můžeme vypozorovat, že je výhodnější použít malou dimenzi vektorů např. $dim = 3$ a provést několik posunů okénka než použít pouze jedno okénko s velkou dimenzí např. $dim = 8$. Což je dáno tím, že pracujeme pouze s y-ionty a pravděpodobnost, že z konce ESI spektra vybereme trojici po sobě jdoucích y-iontů je vyšší než pravděpodobnost, že vybereme např. osmici. Použití menší dimenze nám rovněž umožňuje indexovat i kratší peptidy, které bychom museli při použití vyšší dimenze ze zpracování vynechat (viz sekce 4.3).

4.8.2 Kapacita vnitřních uzlů (arita stromu)

V následujícím experimentu změříme vliv změny kapacity vnitřních uzlů na celkový počet přečtených uzlů DAC_{all}, na skutečný počet uzlů přečtených z disku DAC_{real}, na čas potřebný ke zpracování dotazů a na celkový počet uzlů M-stromu.

Nastavení experimentu - indexovaný soubor: *opal-amet-gv_y_3_1.avs* (viz tabulka 4.3); maximální počet posunů okénka: 10; typ vyhledávání: s modifikacemi; velikost cache: $N = 50$, $H = 4$. Nastavujeme pouze kapacitu vnitřních uzlů (v počtu směrovacích záznamů), kapacita listových je dopočítávána automaticky.

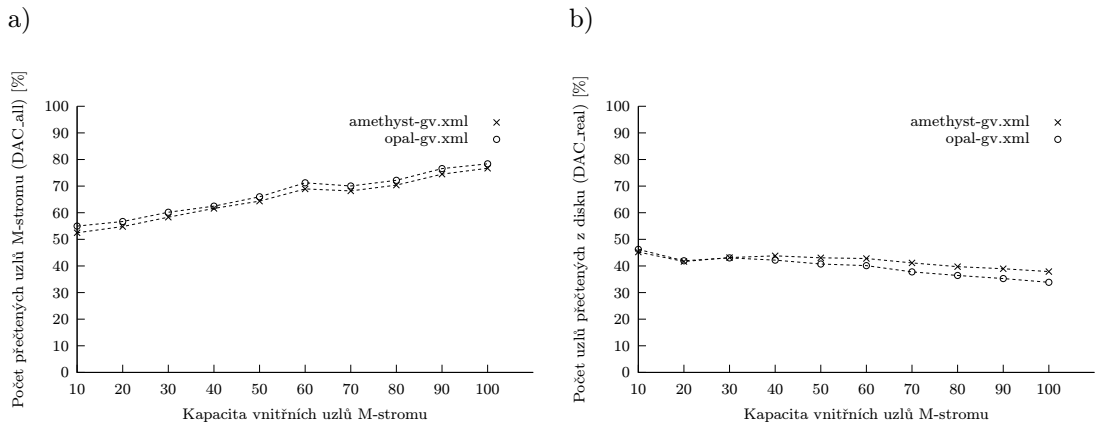
Z naměřených dat můžeme vypozorovat, že s rostoucí kapacitou uzlů hodnota DAC_{all} roste, zatímco DAC_{real} a čas potřebný pro zpracování dotazů klesají. Je to způsobeno konstantním počtem indexovaných objektů a fixní velikostí paměti cache. Počet uzlů M-stromu s jejich rostoucí kapacitou klesá logaritmicky.

Kapacita vnitřních uzlů	10	20	30	40	50	60	70	80	90	100
Kapacita listových uzlů	13	26	40	53	66	80	93	106	120	133
Celkový počet uzlů	27425	12419	7745	5723	4540	3733	3185	2779	2442	2210
Hloubka stromu	6	4	3	3	3	3	2	2	2	2

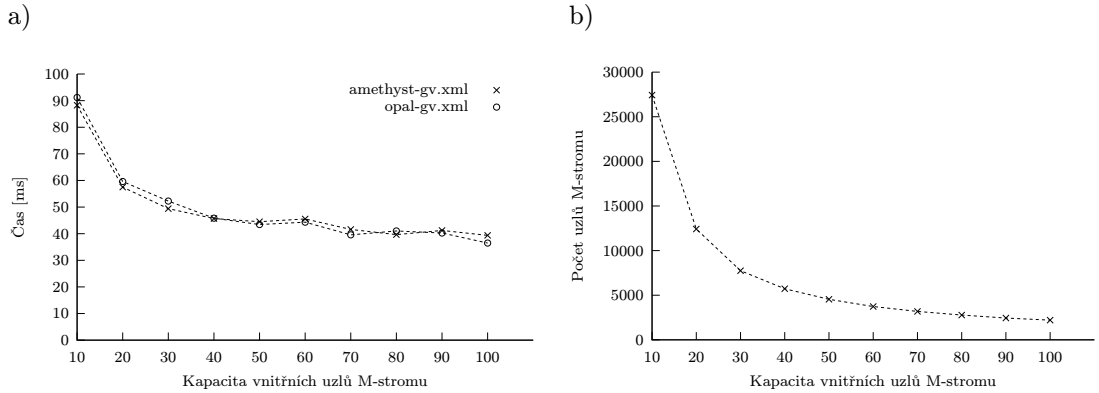
Tabulka 4.11: Parametry indexových struktur.

nsize	**amethyst-gv.xml** ($z \geq 2^+$)					**opal-gv.xml**				
	DAC_{all}		DAC_{real}		**Čas**	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]	[-]	%	[-]	%	[ms]
10	14390.89	52.47	12407.48	45.24	88.25	15077.60	54.98	12674.96	46.22	91.18
20	6806.89	54.81	5175.32	41.67	57.49	7041.27	56.70	5209.35	41.95	59.60
30	4516.65	58.32	3341.92	43.15	49.40	4660.71	60.18	3332.59	43.03	52.26
40	3526.88	61.63	2507.39	43.81	45.68	3575.57	62.48	2414.73	42.19	45.82
50	2922.06	64.36	1954.44	43.05	44.51	2995.75	65.99	1848.80	40.72	43.44
60	2573.85	68.95	1597.17	42.79	45.55	2659.65	71.25	1497.33	40.11	44.33
70	2172.01	68.20	1310.75	41.15	41.60	2230.97	70.05	1202.91	37.77	39.56
80	1955.47	70.37	1104.48	39.74	39.65	2005.90	72.18	1012.29	36.43	41.03
90	1819.17	74.50	951.25	38.95	41.21	1870.26	76.59	861.05	35.26	40.25
100	1695.17	76.70	837.58	37.90	39.37	1732.01	78.37	748.02	33.85	36.49

Tabulka 4.12: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na kapacitě vnitřních uzlů M-stromu; *nsize* udává kapacitu vnitřních uzlů M-stromu.



Obrázek 4.18: Celkový počet přečtených uzlů DAC_{all} a skutečný počet uzlů přečtených z disku DAC_{real} v závislosti na kapacitě vnitřních uzlů M-stromu.



Obrázek 4.19: Čas výpočtu a počet uzlů M-stromu v závislosti na kapacitě vnitřních uzlů.

4.8.3 Počet pivotů (PM-strom)

V tomto experimentu budeme zkoumat efektivitu použití PM-stromu (viz sekce 3.4). Otestujeme přitom nastavení počtu pivotů pro případy $|HR| = |PD|$ a $|PD| = 0$. Měřenými veličinami jsou: celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas potřebný pro zpracování dotazů.

$\|HR\|$	**amethyst-gv.xml** ($z \geq 2^+$)					**opal-gv.xml**				
	DAC_{all}		DAC_{real}		**Čas**	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]	[-]	%	[-]	%	[ms]
0	2922.06	64.36	1954.44	43.05	45.28	2995.75	65.99	1848.80	40.72	46.97
5	2727.89	63.40	1821.36	42.33	78.25	2754.03	64.00	1688.65	39.24	75.41
10	2584.40	62.87	1685.52	41.00	103.79	2636.73	64.14	1572.98	38.26	101.01
15	2577.40	64.82	1654.65	41.62	128.84	2656.05	66.80	1555.59	39.12	131.42
20	2479.98	64.57	1594.84	41.52	154.26	2557.08	66.57	1491.40	38.83	156.29
25	2377.46	63.57	1505.71	40.26	168.97	2451.43	65.55	1398.96	37.41	170.58
30	2334.87	62.73	1447.67	38.89	185.72	2425.16	65.16	1354.05	36.38	189.74
35	2326.20	64.58	1424.07	39.54	207.53	2376.69	65.98	1307.85	36.31	208.36
40	2285.83	64.08	1386.92	38.88	228.50	2337.90	65.54	1272.62	35.68	229.77
45	2255.75	63.70	1388.66	39.22	251.02	2320.90	65.54	1281.98	36.20	251.41
50	2264.87	64.49	1385.80	39.46	268.98	2330.61	66.36	1273.78	36.27	272.51

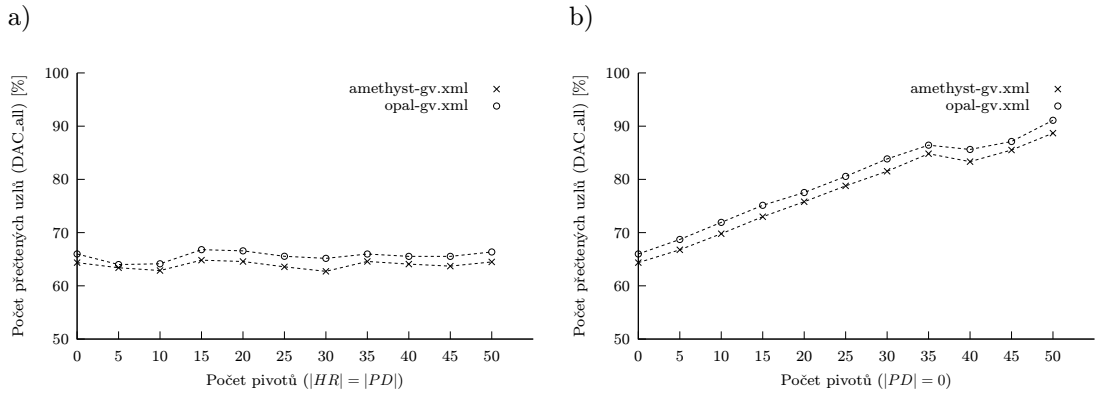
Tabulka 4.13: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na počtu pivotů ($|HR| = |PD|$).

$\|HR\|$	**amethyst-gv.xml** ($z \geq 2^+$)					**opal-gv.xml**				
	DAC_{all}		DAC_{real}		**Čas**	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]	[-]	%	[-]	%	[ms]
0	2922.06	64.36	1954.44	43.05	45.28	2995.75	65.99	1848.80	40.72	46.97
5	2497.15	66.77	1587.62	42.45	51.27	2569.68	68.71	1474.15	39.42	51.13
10	2217.86	69.79	1301.21	40.94	55.10	2285.72	71.92	1197.99	37.70	54.11
15	2028.91	72.98	1062.97	38.24	57.69	2088.60	75.13	955.33	34.36	55.09
20	1855.57	75.80	955.43	39.03	62.51	1898.00	77.53	858.28	35.06	61.94
25	1722.13	78.78	808.53	36.99	63.89	1761.00	80.56	719.84	32.93	61.13
30	1615.74	81.52	710.24	35.83	64.67	1661.92	83.85	631.16	31.84	64.62
35	1549.59	84.82	653.11	35.75	67.52	1579.00	86.43	575.58	31.50	66.11
40	1403.19	83.33	553.11	32.84	77.82	1441.77	85.62	487.09	28.92	78.96
45	1336.61	85.52	482.90	30.90	78.42	1361.70	87.12	421.28	26.95	76.02
50	1292.06	88.68	433.59	29.76	77.77	1327.34	91.10	384.38	26.38	80.64

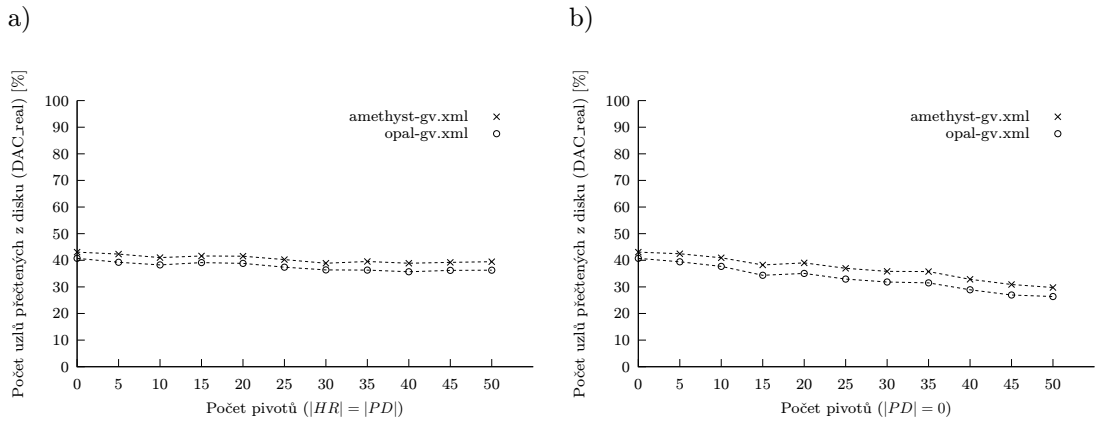
Tabulka 4.14: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na počtu pivotů ($|PD| = 0$).

Počet pivotů ($\|HR\|$)	0	5	10	15	20	25	30	35	40	45	50
Kap. list. uzlů ($\|HR\| = \|PD\|$)	66	70	73	76	78	80	81	83	84	85	86
Kap. list. uzlů ($\|PD\| = 0$)	66	80	94	108	122	136	150	163	177	191	205

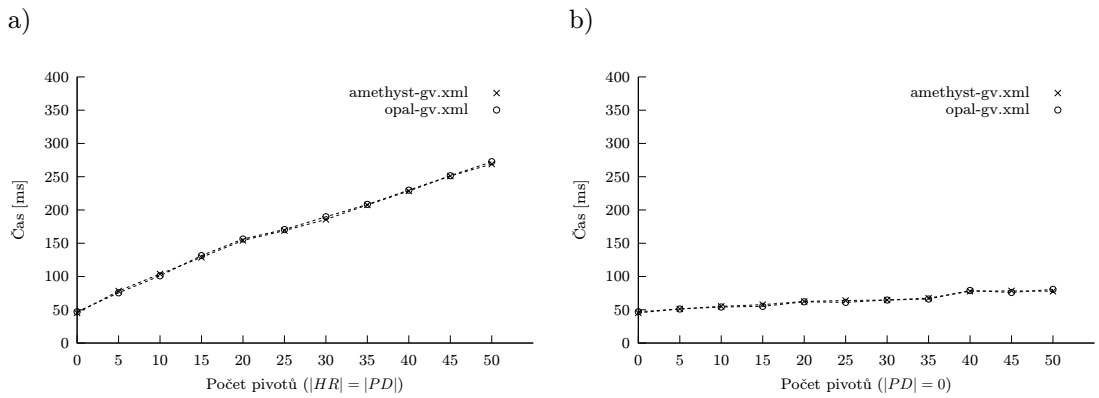
Tabulka 4.15: Kapacita listových uzlů v závislosti na nastavení počtu pivotů.



Obrázek 4.20: Celkový počet přečtených uzlů DAC_{all} v závislosti na nastavení počtu pivotů.



Obrázek 4.21: Skutečný počet uzlů přečtených z disku DAC_{real} v závislosti na nastavení počtu pivotů.



Obrázek 4.22: Čas výpočtu v závislosti na nastavení počtu pivotů.

Nastavení experimentu - indexovaný soubor: *opal-amet-gv_y_3_1.avs* (viz tabulka 4.3); maximální počet posunů okénka: 10; typ vyhledávání: s modifikacemi; velikost cache: $N = 50$, $H = 4$; kapacita vnitřních uzlů PM-stromu: 50. Poznamenejme, že pro případ $|HR| = |PD| = 0$ se jedná o M-strom.

Pro variantu $|HR| = |PD|$ procento všech přečtených uzlů DAC_{all} PM-stromu v závislosti na velikosti pole $|HR|$ stagnuje, procento uzlů přečtených z disku DAC_{real} nepatrně klesá a čas potřebný ke zpracování dotazů výrazně roste. Použití PM-stromu tedy v tomto případě není příliš výhodné, protože např. pro $|HR| = 50$ oproti $|HR| = 0$ dosáhneme snížení DAC_{real} přibližně o 4 %, přičemž čas výpočtu vzroste téměř 6×. Může to být důsledek malé velikosti testované databáze, neoptimalizované velikosti regionů PM-stromu či nevhodného rozložení indexovaných dat.

V případě $|PD| = 0$ zvýšíme kapacitu listových uzlů, protože jednotlivé záznamy vyžadují méně místa pro uložení (viz tabulka 4.15). Hodnota DAC_{all} zde sice se zvyšující se velikostí pole $|HR|$ roste, nicméně DAC_{real} klesá výrazněji než v předchozím případě. Čas výpočtu pro $|HR| = 50$ oproti $|HR| = 0$ přitom vzroste přibližně 1.7×. Nastavení $|PD| = 0$ je tedy výhodnější než $|HR| = |PD|$, přičemž lze předpokládat, že výhoda použití PM-stromu by se výrazněji projevila ve větší databázi příp. po optimalizaci stromu a provedení re-pivotingu.

4.8.4 Velikost databáze

V následujícím experimentu otestujeme vliv velikosti databáze na parametry DAC_{all}, DAC_{real} a na čas výpočtu. K původnímu testovanému souboru dat *opal-amet-gv_y_3_1.avs* budeme postupně přidávat vždy 500 proteinových sekvencí ze souboru *human_e.fasta* (viz sekce 4.1), čímž vytvoříme sadu souborů pro naplnění databáze, jak uvádí tabulka 4.16. Testovat budeme M-strom a pak i PM-strom, kde $|HR| = 40$ a $|PD| = 10$. Provedeme rovněž porovnání se sekvenčním zpracováním všech indexovaných objektů.

Nastavení experimentu - maximální počet posunů okénka: 10; typ vyhledávání: s modifikacemi; velikost cache: $N = 50$, $H = 4$; kapacita vnitřních uzlů: 50; kapacita listových uzlů M-stromu: 66; kapacita listových uzlů PM-stromu: 139.

č.	**Soubor**	**Počet proteinů**	**Počet peptidů**	**Počet vektorů**	**Sekvenční čas [ms]**	
					amet.	**opal.**
1	opal-amet-gv_y_3_1.avs	510	23326	198922	633.76	490.62
2	added_human_e_500_y_3_1.avs	1000	39782	339708	1075.32	837.80
3	added_human_e_1000_y_3_1.avs	1498	61883	527078	1666.28	1297.82
4	added_human_e_1500_y_3_1.avs	1985	82235	701574	2216.88	1728.74
5	added_human_e_2000_y_3_1.avs	2484	105860	902292	2849.68	2225.32
6	added_human_e_2500_y_3_1.avs	2981	124730	1062094	3352.20	2617.50
7	added_human_e_3000_y_3_1.avs	3481	146543	1250773	3950.32	3080.32
8	added_human_e_3500_y_3_1.avs	3978	169500	1447781	4582.82	3565.62
9	added_human_e_4000_y_3_1.avs	4476	192431	1643531	5191.56	4045.62
10	added_human_e_4500_y_3_1.avs	4972	216834	1853331	5900.94	4600.00
11	added_human_e_5000_y_3_1.avs	5468	242798	2072485	6587.18	5136.86

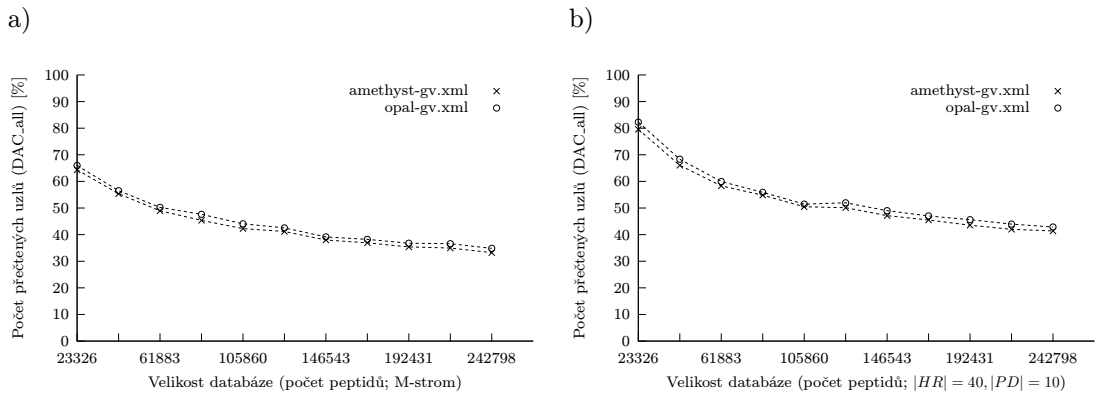
Tabulka 4.16: Soubory pro naplnění databáze (nastavení parametrů - štěpící enzym: trypsin; maximální počet vynechaných míst štěpení: 1; minimální délka sekvence peptidu: 6; maximální délka sekvence peptidu: 20; typ generovaných iontů: y; dimenze vektorů: 3; krok: 1); č. = číslo indexovaného souboru; uveden je průměrný čas potřebný pro zpracování 1 spektra; amet. = *amethyst-gv.xml* ($z \geq 2^+$); opal. = *opal-gv.xml*.

č.	**amethyst-gv.xml** ($z \geq 2^+$)					**opal-gv.xml**				
	DAC_{all}		DAC_{real}		**Čas**	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]	[-]	%	[-]	%	[ms]
1	2922.06	64.36	1954.44	43.05	48.29	2995.75	65.99	1848.80	40.72	71.00
2	4336.22	55.36	3387.33	43.24	102.85	4428.37	56.53	3359.82	42.89	123.74
3	6006.52	48.95	4966.34	40.47	164.41	6165.29	50.24	5047.58	41.13	177.94
4	7421.12	45.35	6306.92	38.54	209.52	7794.83	47.63	6588.70	40.26	231.95
5	8953.03	42.26	7767.95	36.67	280.17	9333.96	44.06	8063.86	38.07	308.40
6	10297.50	41.18	8962.29	35.84	311.50	10636.92	42.54	9225.20	36.89	339.15
7	11178.16	37.97	9884.64	33.58	352.95	11526.23	39.15	10146.08	34.47	387.37
8	12614.56	36.92	11237.85	32.89	463.52	13062.29	38.23	11590.87	33.93	504.16
9	13753.67	35.39	12367.44	31.82	504.66	14288.56	36.77	12778.74	32.88	503.87
10	15342.29	34.97	13900.90	31.68	871.62	16047.81	36.58	14449.23	32.93	594.68
11	16362.63	33.23	14968.22	30.40	1401.12	17162.98	34.85	15591.28	31.66	1033.48

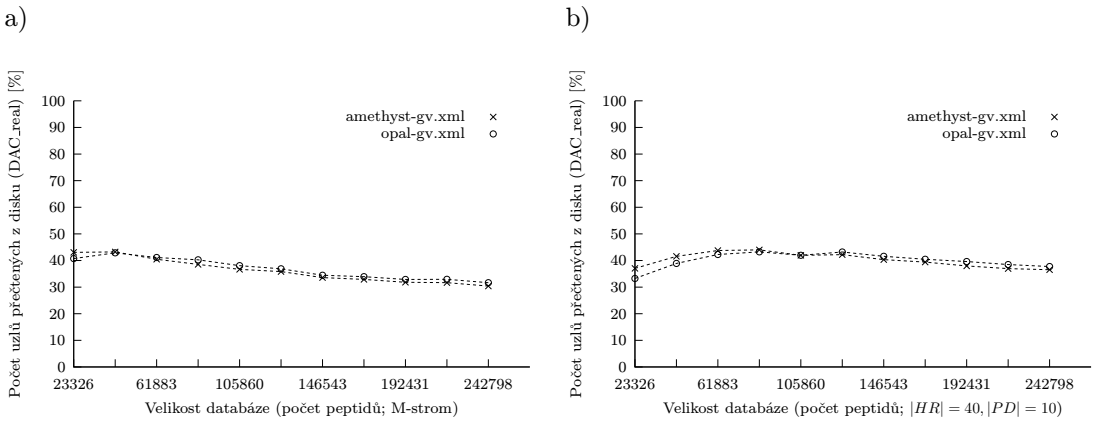
Tabulka 4.17: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na velikosti databáze (M-strom); č. = číslo indexovaného souboru.

č.	**amethyst-gv.xml** ($z \geq 2^+$)					**opal-gv.xml**				
	DAC_{all}		DAC_{real}		**Čas**	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]	[-]	%	[-]	%	[ms]
1	1706.61	79.56	794.48	37.04	142.76	1764.92	82.28	713.17	33.25	156.02
2	2446.25	66.04	1539.21	41.56	229.97	2531.07	68.33	1440.81	38.90	230.06
3	3359.70	58.31	2520.80	43.75	342.24	3454.27	59.95	2434.93	42.26	348.76
4	4220.73	54.84	3386.40	44.00	432.79	4302.24	55.90	3328.41	43.25	436.02
5	4998.42	50.47	4153.74	41.94	537.18	5096.15	51.46	4155.33	41.96	547.37
6	5856.08	50.12	4931.18	42.20	642.86	6070.51	51.96	5047.03	43.20	677.37
7	6479.44	47.13	5543.44	40.32	708.00	6737.64	49.01	5716.01	41.58	757.45
8	7315.51	45.49	6335.59	39.40	844.01	7564.02	47.03	6507.89	40.47	845.41
9	7919.06	43.55	6902.82	37.96	979.26	8296.22	45.62	7205.29	39.62	922.70
10	8641.03	41.99	7604.80	36.95	1513.14	9046.31	43.96	7920.50	38.49	1292.63
11	9530.41	41.38	8416.33	36.54	1590.62	9872.11	42.86	8690.67	37.73	1803.83

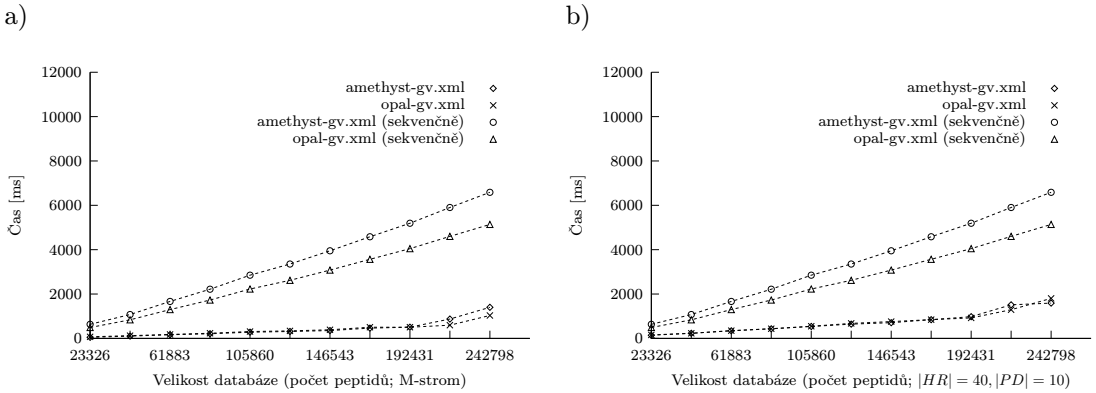
Tabulka 4.18: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na velikosti databáze ($|HR| = 40$, $|PD| = 10$); č. = číslo indexovaného souboru.



Obrázek 4.23: Celkový počet přečtených uzlů DAC_{all} v závislosti na velikosti databáze.



Obrázek 4.24: Skutečný počet uzlů přečtených z disku DAC_{real} v závislosti na velikosti databáze.



Obrázek 4.25: Čas výpočtu v závislosti na velikosti databáze.

Z výsledků měření můžeme vypozorovat, že procento přečtených uzlů DAC_{all} u M-stromu i PM-stromu s rostoucí velikostí databáze klesá. Při použití 10× větší databáze došlo u M-stromu k poklesu DAC_{all} z přibližně 65 % na 34 %, u PM-stromu pak z 81 % na 42 %. Procento uzlů přečtených z disku DAC_{real} v případě M-stromu opět kleslo z cca 42 % na 31 %, u PM-stromu hodnota DAC_{real} stagnuje v průměru na 40 %. V případě M-stromu i PM-stromu s rostoucí velikostí databáze výrazně roste i čas potřebný ke zpracování dotazů. Průměrný čas potřebný na zpracování jednoho spektra při použití databáze s cca 250 tis. peptidovými sekvencemi u M-stromu činí 1 až 1.4 s a u PM-stromu pak 1.6 až 1.8 s, v případě sekvenčního zpracování je však potřeba 5.1 až 6.6 s.

4.9 Testování heuristiky založené na hledání párových iontů

V následující části otestujeme použití logaritmické vzdálenosti spolu s heuristikou založenou na hledání párových iontů (viz sekce 4.4.2). Budeme používat soubory uvedené v tabulce 4.21, které byly vytvořeny postupem popsaným v sekci 4.8.4 s tím rozdílem, že v tomto případě používáme $dim = 5$ a generujeme ionty typu b a y. Vzhledem k výsledkům statistiky prezentované v sekci 4.4.2 budeme interpretovat spektra s nábojem 1^+ z kolekce *amethyst-gv.xml* (MALDI). Minimální počet $dim = 5$ peaků, nalezitelných popsanou heuristikou bez rozlišování b a y-iontů, přitom obsahuje pouze 795 spekter (z 1052) a pouze s těmito spektry budeme pracovat. Povolená odchylka hmotností je 0.2 Da.

4.9.1 Rádius rozsahového dotazu

Nejprve budeme zjišťovat úspěšnost, počet všech přečtených uzlů DAC_{all}, skutečný počet přečtených uzlů DAC_{real}, selektivitu a čas výpočtu v závislosti na rádiusu rozsahového dotazu. Provedeme rovněž porovnání se sekvenčním zpracováním všech indexovaných objektů.

Nastavení experimentu - indexovaný soubor: *added_human_e_1000_yb_5_1.avs* (viz tabulka 4.21); struktura: M-strom; velikost cache: $N = 50$, $H = 4$; kapacita vnitřních uzlů: 50; kapacita listových uzlů: 63; celkový počet uzlů: 24973; hloubka stromu: 3; základ logaritmu: 10.

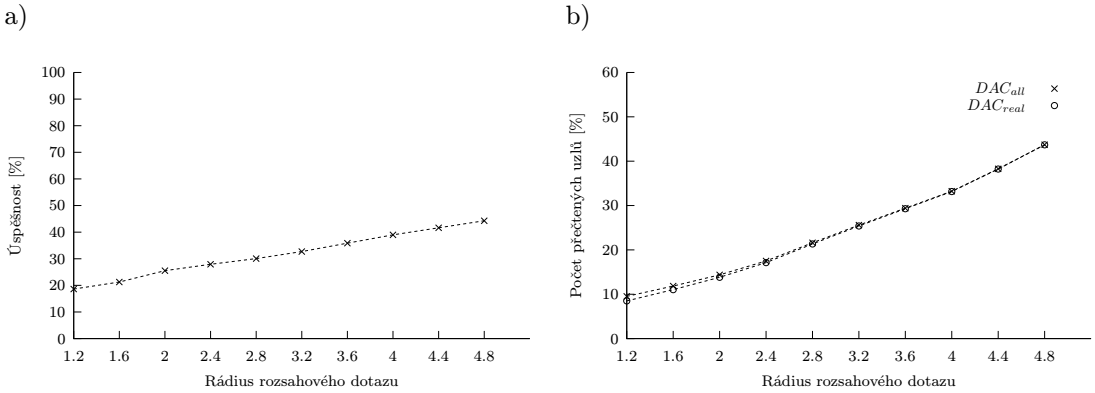
Rádius	DAC_{all}		DAC_{real}		**Selektivita**	
	[-]	%	[-]	%	[-]	%
1.2	2377.80	9.52	2122.47	8.50	1.94	0.00018
1.6	2958.33	11.85	2754.58	11.03	5.25	0.00050
2.0	3588.98	14.37	3452.10	13.82	13.14	0.00125
2.4	4371.83	17.51	4277.20	17.13	29.97	0.00284
2.8	5397.63	21.61	5328.62	21.34	74.35	0.00705
3.2	6389.31	25.58	6341.86	25.39	164.57	0.01561
3.6	7342.57	29.40	7311.39	29.28	342.05	0.03245
4.0	8297.79	33.23	8278.26	33.15	654.81	0.06212
4.4	9558.64	38.28	9546.74	38.23	1341.09	0.12722
4.8	10915.77	43.71	10908.12	43.68	2463.10	0.23366

Tabulka 4.19: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a selektivita v závislosti na rádiusu rozsahového dotazu (logaritmická vzdálenost).

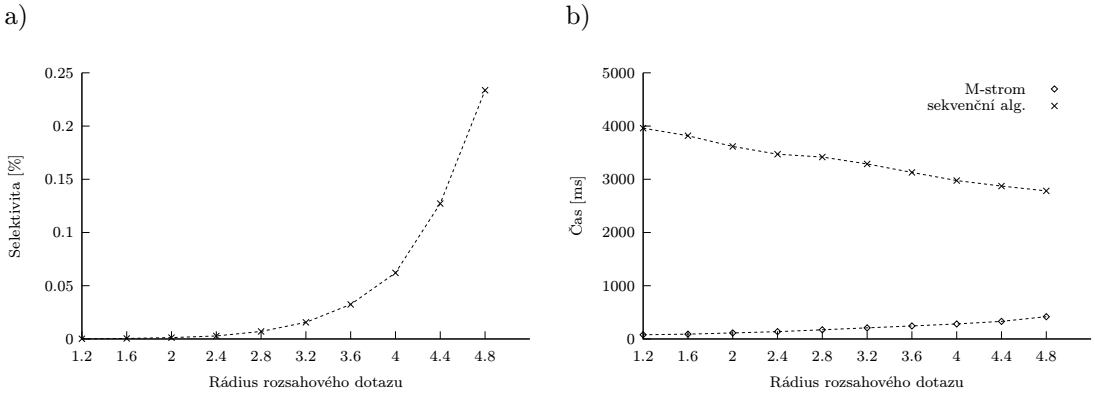
Rádius	**Sekvenční zpracování**			**M-strom**			
	Úspěšnost		**Čas**	**Úspěšnost**		**Čas**	**Zrychlení**
	[-]	%	[ms]	[-]	%	[ms]	[-]
1.2	148	18.62	3960.79	148	18.62	78.85	50.23
1.6	169	21.26	3817.73	169	21.26	90.20	42.33
2.0	203	25.53	3616.67	203	25.53	112.88	32.04
2.4	222	27.92	3470.70	222	27.92	137.25	25.29
2.8	239	30.06	3417.67	239	30.06	172.57	19.80
3.2	260	32.70	3288.19	260	32.70	208.60	15.76
3.6	285	35.85	3128.79	285	35.85	244.81	12.78
4.0	310	38.99	2976.07	310	38.99	281.79	10.56
4.4	331	41.64	2872.36	331	41.64	330.51	8.69
4.8	352	44.28	2781.66	352	44.28	418.76	6.64
Průměr	251.90	31.69	3333.06	251.90	31.69	207.62	16.05

Tabulka 4.20: Porovnání se sekvenčním zpracováním (logaritmická vzdálenost).

Z výsledků experimentu můžeme vypozorovat, že DAC_{all}, DAC_{real} i selektivita v závislosti na rádiusu rozsahového dotazu rostou. I přestože při nastavení rádiusu na 4.8 přečteme téměř 44 % všech uzlů, selektivita dosahuje pouze 0.23 %, což naznačuje výhodnost použití M-stromu. Úspěšnost identifikace při použití logaritmické vzdálenosti spolu s M-stromem je ve všech testovaných případech stejná jako při sekvenčním zpracování a tedy logaritmická vzdálenost, i přestože obecně není metrikou, se pro spektrometrická data ukazuje jako vhodná. Čas výpočtu je při použití M-stromu v průměru 16× nižší než při sekvenčním zpracování. Můžeme však předpokládat, že při zvětšení testované databáze bude zrychlení výrazně vyšší.



Obrázek 4.26: Úspěšnost, DAC_{all} a DAC_{real} v závislosti na rádiusu rozsahového dotazu.



Obrázek 4.27: Selektivita a čas výpočtu v závislosti na rádiusu rozsahového dotazu.

4.9.2 Velikost databáze

Nyní otestujeme závislost na velikosti databáze, zaměříme se přitom na parametry DAC_{all}, DAC_{real}, selektivitu, čas potřebný pro zpracování dotazů a čas indexování databáze. Provedeme rovněž porovnání se sekvenčním průchodem všech indexovaných objektů.

Č. souboru	**Soubor**	**Počet proteinů**	**Počet peptidů**	**Počet vektorů**
1	opal-amet-gv_yb_5_1.avs	510	23326	397844
2	added_human_e_500_yb_5_1.avs	1000	39782	679416
3	added_human_e_1000_yb_5_1.avs	1498	61883	1054156
4	added_human_e_1500_yb_5_1.avs	1985	82235	1403148
5	added_human_e_2000_yb_5_1.avs	2484	105860	1804584
6	added_human_e_2500_yb_5_1.avs	2981	124730	2124188
7	added_human_e_3000_yb_5_1.avs	3481	146543	2501546
8	added_human_e_3500_yb_5_1.avs	3978	169500	2895562
9	added_human_e_4000_yb_5_1.avs	4476	192431	3287062

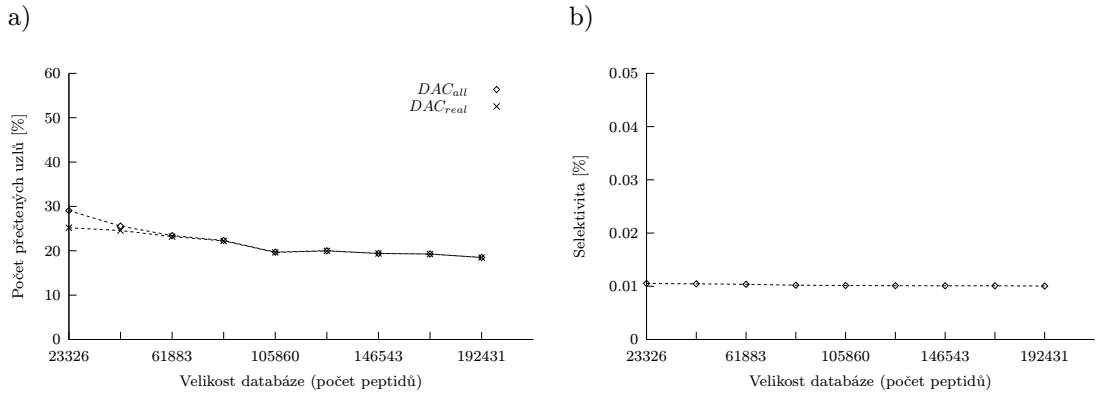
Tabulka 4.21: Soubory pro naplnění databáze (nastavení parametrů - štěpící enzym: trypsin; maximální počet vynechaných míst štěpení: 1; minimální délka sekvence peptidu: 6; maximální délka sekvence peptidu: 20; typ generovaných iontů: y, b; dimenze vektorů: 5; krok: 1).

Č. souboru	DAC_{all}		DAC_{real}		**Selektivita**	
	[-]	%	[-]	%	[-]	%
1	2712.71	29.06	2351.35	25.19	41.89	0.01053
2	4103.43	25.57	3944.01	24.57	71.10	0.01046
3	5850.87	23.43	5793.45	23.20	109.35	0.01037
4	7434.38	22.32	7402.86	22.22	143.08	0.01020
5	8455.81	19.70	8433.14	19.65	182.95	0.01014
6	10090.83	20.01	10076.14	19.98	214.72	0.01011
7	11552.69	19.41	11541.25	19.39	252.45	0.01009
8	13291.29	19.28	13279.42	19.26	292.29	0.01009
9	14446.93	18.50	14438.23	18.49	330.20	0.01005

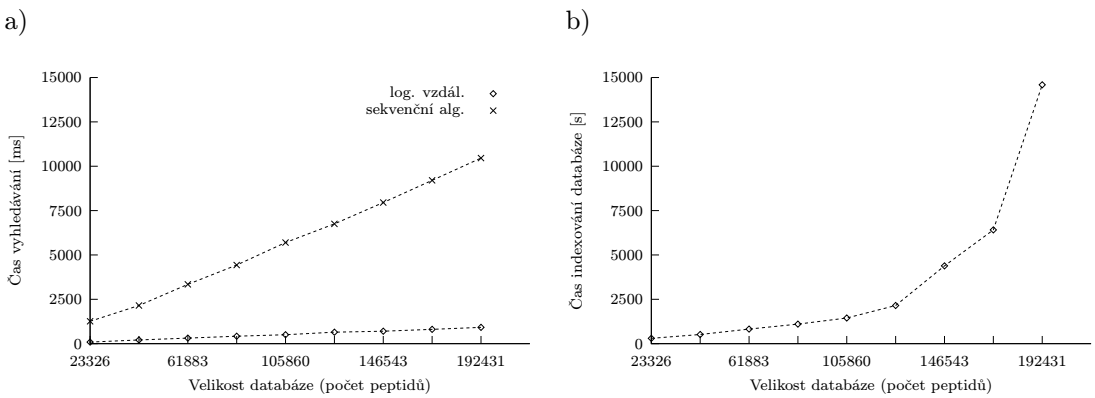
Tabulka 4.22: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a selektivita v závislosti na velikosti databáze (logaritmická vzdálenost).

Č. souboru	**Čas**			**Zrychlení vyhl.**
	indexování	**sekvenční vyhl.**	**vyhledávání**	
	[s]	**[ms]**	**[ms]**	
1	300.27	1261.69	89.89	14.04
2	516.45	2151.10	209.85	10.25
3	822.30	3342.59	313.49	10.66
4	1100.13	4430.31	419.28	10.57
5	1445.05	5699.42	502.51	11.34
6	2151.33	6751.42	646.78	10.44
7	4384.80	7953.83	701.80	11.33
8	6411.48	9204.99	805.48	11.43
9	14591.10	10460.53	917.59	11.40

Tabulka 4.23: Čas indexování, vyhledávání a sekvenčního zpracování (logaritmická vzdálenost); čas vyhledávání a sekvenčního vyhledávání jsou průměrné hodnoty na 1 spektrum.



Obrázek 4.28: Počet přečtených uzlů a selektivita v závislosti na velikosti databáze.



Obrázek 4.29: Čas vyhledávání a čas indexování v závislosti na velikosti databáze.

Nastavení experimentu - struktura: M-strom; velikost cache: $N = 50$, $H = 4$; typ dotazu: rozsahový; rádius dotazu: 3.0; základ logaritmu: 10. Indexování databáze bylo provedeno jednocestným vkládání (viz sekce 3.3.5).

Z naměřených dat můžeme usoudit, že procento přečtených uzlů DAC_{all} i DAC_{real} s rostoucí velikostí databáze klesá. Rozdíl mezi parametry DAC_{all} a DAC_{real} je přitom minimální a zřejmě by pomohlo zvětšení kapacity vyrovnávací paměti. Selektivita ve všech případech činí přibližně 0.01 %. Při použití sekvenčního průchodu i M-stromu bylo správně identifikováno 251 peptidů (31.57 %). Výpočet byl při použití M-stromu průměrně 11.27× rychlejší než při sekvenčním zpracování.

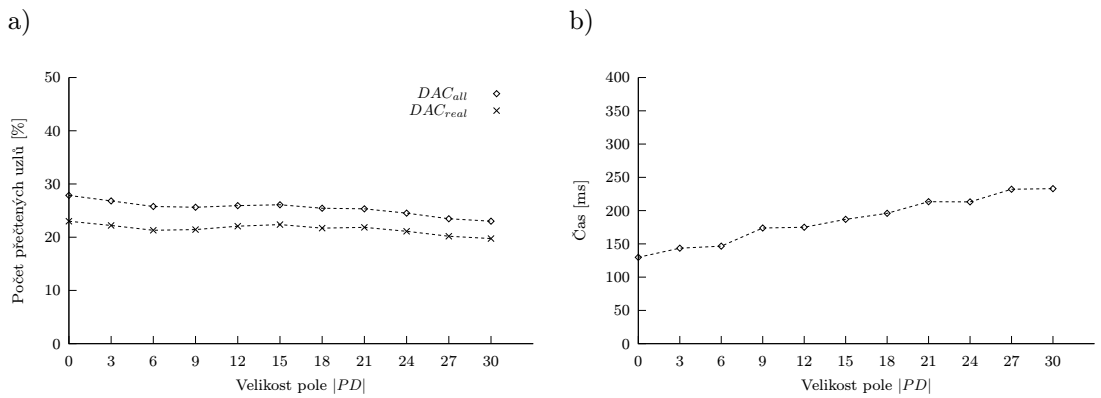
4.9.3 Poměr velikosti polí *HR* a *PD*

V následujícím experimentu otestujeme vliv poměru velikosti polí *HR* a *PD* na parametry DAC_{all}, DAC_{real} a na čas potřebný pro zpracování dotazů.

Nastavení experimentu - indexovaný soubor: *opal-amet-gv_yb_5_1.avs*; struktura: PM-strom; velikost cache: $N = 50$, $H = 4$; kapacita vnitřních uzlů: 50; velikost pole *HR*: 30 pivotů; typ dotazu: rozsahový; rádius dotazu: 3.0; základ logaritmu: 10.

$\|PD\|$	DAC_{all}		DAC_{real}		**Čas**
	[-]	%	[-]	%	[ms]
0	1254.53	27.85	1035.91	23.00	129.79
3	1278.56	26.82	1059.19	22.21	143.50
6	1309.43	25.77	1082.22	21.30	146.57
9	1382.03	25.63	1155.91	21.43	173.81
12	1483.76	25.93	1262.91	22.07	174.94
15	1568.17	26.10	1344.02	22.37	186.84
18	1608.28	25.44	1371.36	21.70	195.91
21	1671.19	25.33	1441.56	21.85	213.38
24	1699.14	24.52	1462.98	21.11	213.06
27	1705.68	23.47	1466.83	20.18	232.11
30	1734.55	23.01	1489.13	19.75	232.99

Tabulka 4.24: Celkový počet přečtených uzlů DAC_{all}, skutečný počet uzlů přečtených z disku DAC_{real} a čas výpočtu v závislosti na poměru velikosti polí *HR* a *PD* ($|HR| = 30$).



Obrázek 4.30: Počet přečtených uzlů a čas vyhledávání v závislosti na poměru velikosti polí $|HR|$ a $|PD|$.

Z výsledků experimentu můžeme vypozorovat, že s rostoucí velikostí pole *PD* (tj. klesajícím poměrem velikosti polí *HR* a *PD*) počty přečtených uzlů DAC_{all} i DAC_{real} klesají, čas potřebný pro zpracování dotazů přitom roste. Úspěšnost vyhledávání je stejná jako v předchozím experimentu (viz sekce 4.9.2).

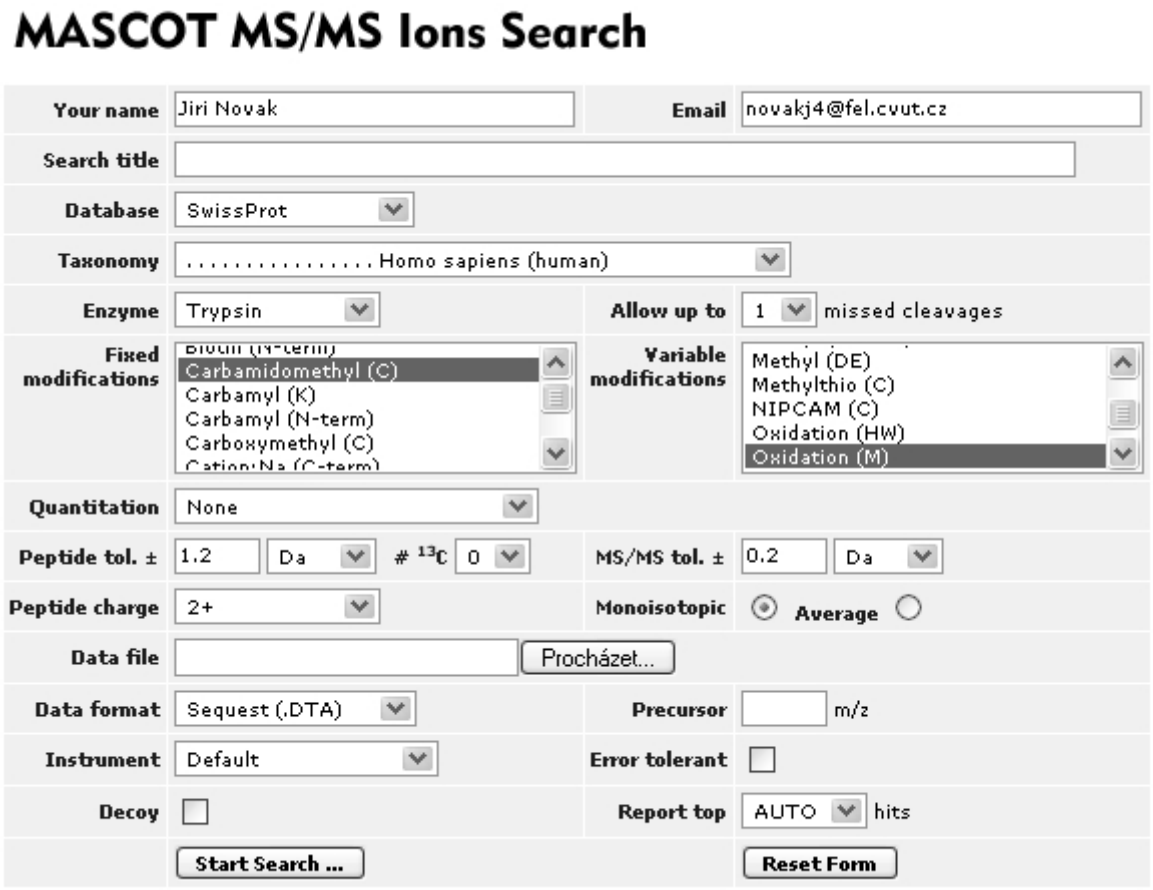
4.10 Porovnání se stávajícími metodami vyhledávání v databázích

V této části provedeme porovnání metod založených na metrických indexech s veřejně dostupnými vyhledávači Mascot a ProteinProspector (verze 4.27.2 Basic). Protože interpretujeme tandemová hmotnostní spektra, budeme v případě Mascotu testovat aplikaci MS/MS Ions Search a v případě ProteinProspectoru aplikaci MS-Tag. Jednomu analyzovanému proteinu v praxi odpovídá sada tandemových hmotnostních spekter, které přísluší jeho částem (peptidům; viz obrázek 2.7), protože jsme se však ve výzkumné části zaměřili na klíčovou identifikaci peptidové sekvence z tandemového spektra, přizpůsobíme tomuto předpokladu i metodiku porovnávání se stávajícími metodami.

Budeme analyzovat vždy jedno spektrum a testovat, zda-li peptidová sekvence, kterou vyhledávač k danému spektru přiřadí jako nejlepší hit odpovídá referenční sekvenci. Pro testování použijeme prvních 50 tandemových hmotnostních spekter z kolekce dat *opal-gv.xml*, které mají náboj rodičovského peptidu 2^+. Referenční sekvencí přitom rozumíme peptidovou sekvenci uvedenou právě v souboru *opal-gv.xml*.

Nastavení vyhledávače Mascot uvádí obrázek 4.31. Vyhledáváme tedy v databázi SwissProt a v genomu člověka. Při štěpení proteinů na peptidy používáme trypsin, přičemž jsme povolili maximálně 1 vynechané místo štěpení. Vyhledávač umožňuje nastavení fixních modifikací (pro všechny výskyty dané aminokyseliny v peptidu) a variabilních modifikací (pro některé výskyty dané aminokyseliny). Nastaveno je vyhledávání fixních modifikací Carbamidomethyl C (+57.01 Da) a variabilních modifikací Oxidation M (+16 Da), Gln→pyro-Glu pro N-koncové Q (−17.016 Da) a Deamidated N, Q (+1 Da). Používáme monoizotopickou hmotnost, tolerance odchylky teoretické hmotnosti peptidu od experimentálně zjištěné hodnoty je 1.2 Da a tolerance hmotností jednotlivých fragmentů je 0.2 Da. Jednotlivá spektra zadáváme ze samostatných souborů *.dta v SEQUEST formátu (viz obrázek 2.26). Nastavení ProteinProspectoru je podobné, viz obrázek 4.32.

Z metod založených na metrických indexech použijeme pro srovnání algoritmus využívající intervalový dotaz spolu s maximovou metrikou (viz sekce 4.5.2) a vyzkoušíme i variantu tohoto algoritmu s Hausdorffovou vzdáleností. Dále otestujeme logaritmickou vzdálenost,



Obrázek 4.31: Mascot (MS/MS Ions Search) - nastavení vyhledávače.

pro kterou nám z uvedeného algoritmu vypadne vyhledávání modifikací tj. použití intervalového dotazu (zdůvodnění viz sekce 4.5.1). Při vyhledávání používáme heuristiku založenou pozicích y-iontů (viz sekce 4.4.1). Pracujeme s celou kolekcí proteinů nalezených v genomu člověka (*human_e.fasta*), čímž jsou simulovány podobné výchozí podmínky jako u veřejně dostupných vyhledávačů.

Nastavení experimentu - indexovaný soubor: *added_human_e_y_3_1.avs*; počet proteinů: 47781; počet peptidů: 2427652; počet vektorů: 20700407; dimenze vektorů: 3; struktura: M-strom; kapacita vnitřních uzlů: 50; kapacita listových uzlů: 66; hloubka stromu: 4; velikost cache při indexování: $N = 1000$, $H = 25$; velikost cache při vyhledávání: $N = 1000$, $H = 5$; maximální počet posunů okénka: 10.

Logaritmická vzdálenost - rádius rozsahového dotazu: 2.0; základ logaritmu: 10; počet vnitřních uzlů: 16255; počet listových uzlů: 499863; celkový počet uzlů: 516118; čas indexování: 57329.2 s.

Hausdorffova metrika - rádius rozsahového dotazu: 10.0; počet vnitřních uzlů: 15719; počet listových uzlů: 489716; celkový počet uzlů: 505435; čas indexování: 83370.0 s.

Maximová metrika - rádius rozsahového dotazu: 10.0; počet vnitřních uzlů: 15710; počet listových uzlů: 490997; celkový počet uzlů: 506707; čas indexování: 75496.1 s.

Při použití sady intervalových dotazů v případě maximové a Hausdorffovy metriky vyhledáváme modifikace aminokyselin {C, M, Q, N $\vee$ Q} odpovídající po řadě změnám hmotností $\{57.01, 16, 17.016, 1\}$, kde vnitřní a vnější rádius intervalového dotazu určíme na základě odchylek ± 0.1 pro danou hodnotu.

U porovnání se stávajícími metodami zjišťujeme kromě toho, zda nalezená peptidová sekvence odpovídá referenční sekvenci, také kolik b a y iontů bylo identifikováno. Výsledky srovnání prezentuje tabulka 4.25, kde b je počet identifikovaných b-iontů, y je počet y-iontů, $b + y$ odpovídá součtu identifikovaných b a y-iontů, m určuje zda byla nalezena správná sekvence (1) nebo zda byla uvedena jiná sekvence (0). V případě vyhledávače Mascot ještě měření doplňujeme o hodnotu p, která udává, zda byla správnost sekvence jednoznačně potvrzena (1) nebo zda výsledek hledání nelze 100 % zaručit (0).

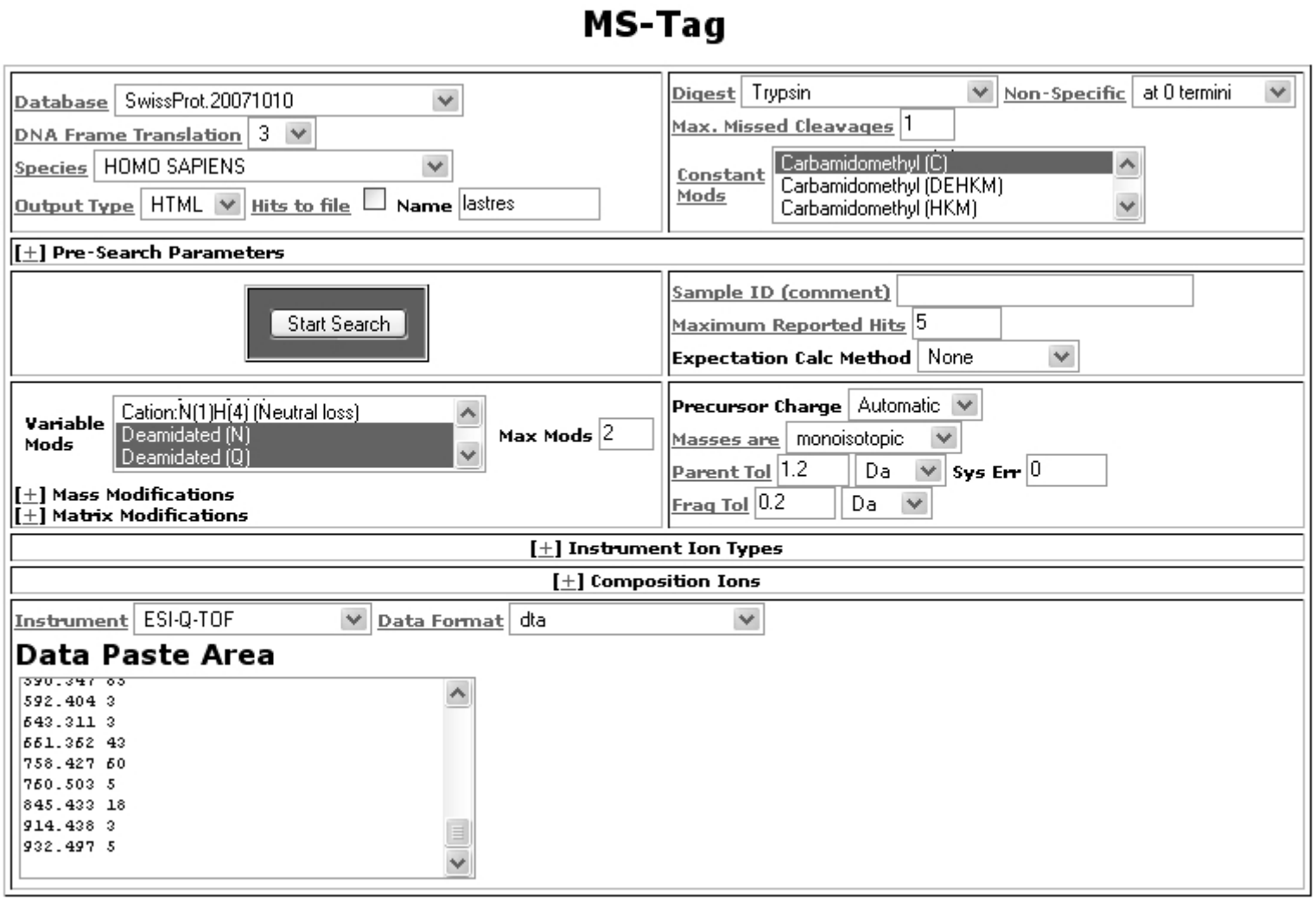
Z výsledků experimentu se jako nejlepší ukázal vyhledávač ProteinProspector, který správně identifikoval 36 z 50 testovaných sekvencí. Druhým v pořadí je vyhledávač Mascot, který korektně určil 34 peptidů, přičemž s jistou potvrdil 31 sekvencí. Zajímavá je úspěšnost algoritmu používajícího sadu intervalových dotazů, který spolu s maximovou metrikou správně identifikoval 26 sekvencí a spolu s Hausdorffovou metrikou 28 sekvencí. S využitím logaritmické vzdálenosti jsme správně určili 24 peptidových sekvencí. I přestože při vyhledávání používáme poměrně jednoduchou heuristiku (viz sekce 4.4.1), můžeme vypozorovat, že úspěšnost metrických indexovacích je srovnatelná i s tak kvalitními vyhledávači jako jsou Mascot a ProteinProspector.

Vyhledávač ProteinProspector potřeboval pro zpracování jednoho spektra 4 s, současná verze vyhledávače Mascot čas výpočtu neuvádí. Zpracování algoritmem používajícím sadu intervalových dotazů trvalo při použití maximové metriky průměrně 149.07 ms, při použití Hausdorffovy metriky pak 165.33 ms. S využitím logaritmické vzdálenosti vyhledávání byla průměrná doba identifikace 263.56 ms. Tabulka 4.26 prezentuje podrobnější výsledky experimentu pro metrické indexovací metody. Lze vypozorovat, že oproti sekvenčnímu průchodu dosahujeme při použití metrických indexovacích metod řádově 1000 násobného zrychlení.

V případě algoritmu založeného na použití sady intervalových dotazů počet všech přečtených uzlů DAC_{all} pro maximovou metriku dosahuje průměrně 3.32 % a selektivita 0.27 %, pro Hausdorffovu metriku pak DAC_{all} činí 3.80 % a selektivita 0.28 %. V případě použití logaritmické vzdálenosti je sice DAC_{all} o něco vyšší 11.15 %, ale selektivita vychází v průměru pouze 0.1 %.

spektrum	Mascot					ProteinProspector				interv. dotaz + max. vzd.				interv. dotaz + Haus. vzd.				log. vzdál.				opal-gv.xml		
	m	p	b	y	$b+y$	m	b	y	$b+y$	m	b	y	$b+y$	m	b	y	$b+y$	m	b	y	$b+y$	b	y	$b+y$
34.dta	0	0	3	0	3	0	0	6	6	0	1	5	6	0	1	5	6	1	2	9	11	2	9	11
65.dta	1	0	5	6	11	1	5	6	11	0	2	3	5	0	2	3	5	0	2	2	4	5	6	11
72.dta	1	1	3	7	10	1	4	7	11	1	4	7	11	1	4	7	11	0	2	4	6	4	7	11
138.dta	1	1	4	7	11	1	4	8	12	1	4	8	12	1	4	8	12	0	2	4	6	4	8	12
179.dta	1	1	2	8	10	1	3	8	11	1	3	8	11	1	3	8	11	1	3	8	11	3	8	11
191.dta	1	1	0	10	10	1	2	11	13	1	2	11	13	1	2	11	13	1	2	11	13	3	11	14
199.dta	1	1	1	12	13	1	2	12	14	1	2	12	14	1	2	12	14	1	2	12	14	2	12	14
236.dta	1	1	5	4	9	1	5	6	11	0	2	4	6	0	4	2	6	1	1	6	7	5	6	11
302.dta	0	0	0	6	6	0	0	6	6	1	1	9	10	1	1	9	10	1	1	9	10	1	9	10
304.dta	1	1	7	8	15	1	7	8	15	1	7	8	15	1	7	8	15	1	7	8	15	7	8	15
305.dta	0	0	2	1	3	0	5	2	7	0	4	2	6	0	4	2	6	0	3	2	5	5	7	12
351.dta	0	0	1	4	5	0	2	4	6	0	2	4	6	0	2	4	6	0	2	3	5	3	9	12
367.dta	1	0	2	8	10	1	2	8	10	1	2	8	10	1	2	8	10	1	2	8	10	2	9	11
376.dta	1	1	2	8	10	1	3	8	11	1	3	8	11	1	3	8	11	1	3	8	11	3	8	11
382.dta	1	1	7	10	17	1	7	10	17	1	7	10	17	1	7	10	17	1	7	10	17	7	10	17
403.dta	1	1	2	9	11	1	2	9	11	1	2	9	11	1	2	9	11	1	2	9	11	2	9	11
405.dta	1	1	1	11	12	1	1	11	12	1	1	11	12	1	1	11	12	0	4	2	6	1	11	12
407.dta	1	1	4	8	12	1	5	9	14	1	5	9	14	1	5	9	14	0	3	3	6	5	9	14
418.dta	1	1	3	7	10	1	4	7	11	1	4	7	11	1	4	7	11	0	2	4	6	4	7	11
476.dta	0	0	3	0	3	0	0	6	6	0	4	1	5	0	4	1	5	1	1	9	10	1	9	10
553.dta	1	1	5	12	17	1	5	8	13	1	5	10	15	1	5	10	15	1	5	10	15	5	10	15
559.dta	1	1	8	9	17	1	9	9	18	0	1	2	3	0	0	3	3	0	0	6	6	9	9	18
566.dta	1	1	4	12	16	1	6	12	18	0	2	3	5	0	2	3	5	0	3	4	7	6	12	18
573.dta	1	1	2	11	13	1	3	11	14	1	3	11	14	1	3	11	14	1	3	11	14	3	11	14
576.dta	1	1	6	12	18	1	6	12	18	0	3	4	7	0	2	6	8	0	4	3	7	6	12	18
578.dta	1	1	1	9	10	1	3	10	13	1	3	10	13	1	3	10	13	0	1	4	5	3	10	13
585.dta	0	0	4	1	5	0	1	4	5	0	2	1	3	0	1	2	3	1	4	9	13	4	9	13
592.dta	1	1	2	13	15	1	3	14	17	0	0	4	4	0	0	4	4	0	4	4	8	4	14	18
593.dta	1	1	3	8	11	1	5	10	15	0	3	3	6	0	3	3	6	0	2	6	8	5	10	15
595.dta	1	1	1	7	8	1	4	9	13	0	3	3	6	0	2	5	7	0	1	3	4	4	9	13
596.dta	0	0	4	0	4	0	5	3	8	0	4	2	6	0	4	2	6	0	1	1	2	5	10	15
602.dta	1	1	2	10	12	1	4	10	14	1	4	10	14	1	4	10	14	1	4	10	14	4	10	14
610.dta	1	1	4	12	16	1	5	12	17	1	5	12	17	1	5	12	17	0	1	3	4	5	12	17
633.dta	1	1	1	8	9	1	2	10	12	1	2	10	12	1	2	10	12	1	2	10	12	2	10	12
648.dta	1	0	5	7	12	1	6	7	13	0	2	2	4	0	1	3	4	0	2	3	5	6	7	13
653.dta	1	1	3	7	10	1	4	7	11	0	1	3	4	1	3	7	10	1	3	7	10	3	7	10
654.dta	0	0	5	0	5	0	4	5	9	0	5	2	7	0	5	2	7	0	2	5	7	5	9	14
656.dta	0	0	0	5	5	1	3	8	11	0	3	5	8	1	4	8	12	1	4	8	12	3	9	12
667.dta	0	0	0	4	4	0	4	3	7	1	2	12	14	1	2	12	14	1	2	12	14	4	12	16
694.dta	0	0	2	0	2	0	0	4	4	0	3	3	6	0	3	3	6	0	1	1	2	3	7	10
699.dta	0	0	2	8	10	1	2	8	10	0	1	4	5	0	1	4	5	0	5	3	8	2	11	13
768.dta	1	1	3	10	13	1	6	10	16	1	6	10	16	1	6	10	16	0	1	5	6	6	10	16
778.dta	1	1	4	8	12	1	4	8	12	1	4	8	12	1	4	8	12	1	4	8	12	4	8	12
793.dta	1	1	5	14	19	1	5	14	19	1	2	14	16	1	2	14	16	1	2	14	16	5	14	19
794.dta	1	1	8	9	17	1	8	9	17	0	1	4	5	0	1	4	5	0	0	0	0	8	9	17
804.dta	0	0	0	0	0	0	6	2	8	1	0	8	8	1	0	8	8	1	0	8	8	3	8	11
806.dta	1	1	6	12	18	1	6	13	19	0	2	5	7	0	2	5	7	0	3	5	8	6	13	19
810.dta	0	0	4	0	4	0	1	3	4	1	2	7	9	1	2	7	9	1	2	7	9	2	7	9
822.dta	0	0	4	0	4	0	5	2	7	0	1	5	6	0	1	5	6	0	3	4	7	3	11	14
823.dta	0	0	1	3	4	0	3	4	7	0	5	5	10	0	5	5	10	0	3	1	4	3	10	13
celkem	**34**	**31**	156	345	501	**36**	191	393	584	**26**	142	326	468	**28**	142	338	480	**24**	125	306	431	200	472	672
průměr	0.68	0.62	3.12	6.9	10.02	0.72	3.82	7.86	11.68	0.52	2.84	6.52	9.36	0.56	2.84	6.76	9.6	0.48	2.5	6.12	8.62	4	9.44	13.44

Tabulka 4.25: Porovnání se stávajícími metodami vyhledávání v databázích.



Obrázek 4.32: ProteinProspector (MS-Tag).

spektrum	interv. dotaz + max. vzd.					interv. dotaz + Haus. vzd.					log. vzdál.				
	Čas [ms]	**Sekv. čas [s]**	DAC_{all} [%]	DAC_{real} [%]	**Selektivita [%]**	**Čas [ms]**	**Sekv. čas [s]**	DAC_{all} [%]	DAC_{real} [%]	**Selektivita [%]**	**Čas [ms]**	**Sekv. čas [s]**	DAC_{all} [%]	DAC_{real} [%]	**Selektivita [%]**
34.dta	316.17	697.42	6.1	4.88	0.74	393.95	779.27	7.1	5.3	0.79	233.64	96.72	4.06	3.62	0.07
65.dta	202.33	173.53	2.75	2.3	0.13	161.7	290.33	3.05	2.49	0.14	280.81	97.72	14.42	9.23	0.00
72.dta	276.91	283.31	3.28	2.51	0.27	193.39	249.97	3.48	2.71	0.28	650.67	461.67	20.1	15.63	0.56
138.dta	110.19	242.52	1.84	1.5	0.14	139.28	118.03	1.98	1.63	0.15	926.19	466.41	19.63	15.53	0.43
179.dta	8.83	34.55	0.15	0.12	0.01	12.61	21.88	0.18	0.16	0.01	73.14	24.94	2.81	2.01	0.01
191.dta	259.41	226.77	3.84	3.25	0.16	136.44	120.27	2.67	2.31	0.08	246.72	94.31	11.72	7.57	0.01
199.dta	0.78	8.70	0.03	0.02	0.01	0.95	25.39	0.04	0.02	0.01	42.86	28.92	1.71	1.2	0.02
236.dta	323.14	663.08	6.53	5.12	0.73	383.44	716.39	7.77	6	0.79	238.7	158.00	8.02	5.97	0.17
302.dta	7.31	40.75	0.15	0.12	0.03	9.08	33.00	0.17	0.13	0.03	111.86	117.09	2.79	2.33	0.08
304.dta	134.94	153.06	2.26	2.03	0.16	146.66	156.50	2.57	2.31	0.17	109.55	147.61	2.68	2.32	0.13
305.dta	271.37	447.45	5.63	4.06	0.36	249.58	514.67	6.9	4.47	0.38	304.09	175.86	16.1	10.23	0.06
351.dta	88.69	128.52	2.35	1.64	0.07	109.78	242.14	2.76	1.99	0.07	227.11	95.27	14.18	8.73	0.00
367.dta	117.66	162.17	2.6	2.18	0.11	162.45	249.41	4.23	3.44	0.17	224.44	192.55	11.9	7.73	0.05
376.dta	1.26	31.58	0.04	0.03	0.01	1.3	13.39	0.03	0.02	0.01	25.58	101.97	1.76	1.21	0.03
382.dta	4.83	78.99	0.32	0.18	0.10	9.19	59.66	0.33	0.2	0.10	92.28	186.58	2.66	2.26	0.13
403.dta	16.06	34.05	0.43	0.33	0.01	14.33	28.80	0.45	0.32	0.01	128.89	48.81	5.26	3.66	0.01
405.dta	15.03	31.78	0.44	0.29	0.03	23.94	38.89	0.48	0.31	0.03	422.51	314.42	18.74	13.85	0.30
407.dta	237.39	730.74	5.31	4.19	0.43	255.59	510.58	6.33	4.8	0.48	396.03	178.30	16.56	11.97	0.13
418.dta	114.78	260.16	3.27	2.36	0.24	133.2	286.49	3.47	2.58	0.25	534.25	515.59	20.22	15.61	0.55
476.dta	231.61	497.66	5.17	4.06	0.59	268.52	903.52	6.07	4.62	0.68	66.41	30.91	1.8	1.56	0.01
553.dta	16.7	26.76	0.26	0.23	0.01	20.58	51.31	0.32	0.27	0.01	84.81	49.06	3.39	2.09	0.02
559.dta	120.27	208.89	2.76	2.03	0.14	150.19	315.86	3.53	2.28	0.15	175.72	139.83	12.7	6.41	0.03
566.dta	204.74	333.56	5.02	3.96	0.31	267.02	549.64	7.62	5.24	0.45	320.94	138.31	15.63	9.95	0.04
573.dta	17.36	40.41	0.66	0.43	0.03	23.74	61.11	0.75	0.46	0.03	130.25	59.13	6.98	4.3	0.01
576.dta	171.56	322.50	5.2	3.87	0.31	203.58	609.30	7.15	4.78	0.40	377.33	118.09	15.56	9.91	0.02
578.dta	164.94	383.39	4.52	3.42	0.32	249.03	590.45	5.88	4.57	0.40	546.31	299.97	19.38	14.67	0.28
585.dta	168.59	342.69	4.48	3.39	0.24	191.17	445.61	4.82	3.54	0.25	99.67	35.00	3.25	2.63	0.01
592.dta	425.41	1155.55	10.94	8	1.03	505.03	1304.74	12.21	8.89	1.08	464.11	401.78	18.46	13.21	0.41
593.dta	145.84	267.77	3.42	2.59	0.22	183.27	407.86	5.19	3.25	0.29	319.77	178.05	15.06	9.76	0.02
595.dta	275.25	431.47	5.8	4.68	0.49	405.5	742.50	9.03	6.83	0.72	319.45	123.16	16.74	11.76	0.03
596.dta	122.11	254.20	2.97	2.25	0.21	286.25	409.75	4.66	2.87	0.30	306.78	109.88	14.51	9.35	0.01
602.dta	4.92	25.77	0.1	0.08	0.02	6.42	26.58	0.11	0.09	0.02	78.97	38.59	1.95	1.77	0.03
610.dta	11.45	23.69	0.22	0.2	0.01	14.8	29.06	0.26	0.23	0.02	390.66	190.33	16.96	11.36	0.11
633.dta	8.27	35.95	0.19	0.16	0.01	10	17.95	0.2	0.17	0.01	10.8	14.48	1.38	0.89	0.00
648.dta	91.66	158.52	2.62	1.86	0.09	119.16	266.86	3.42	2.29	0.10	192.19	107.99	12.59	6.91	0.01
653.dta	516.84	1077.88	11.23	9.41	1.28	183.73	300.75	3.74	3.15	0.27	401.75	198.16	16.87	12.7	0.18
654.dta	353.36	740.08	7.71	6.1	0.86	393.16	1168.36	8.52	6.41	0.88	404.83	279.48	17.51	12.87	0.17
656.dta	328.39	925.47	7.47	5.67	1.03	344.16	914.23	7.08	5.43	0.90	367.98	326.42	16.28	12.09	0.27
667.dta	36.14	123.26	0.5	0.48	0.08	45.19	120.14	0.67	0.6	0.09	128.45	81.94	3.72	3.09	0.09
694.dta	123.87	204.23	2.77	1.97	0.15	172.2	417.38	3.79	2.67	0.19	204.87	121.09	14.33	8.87	0.00
699.dta	202.41	447.61	4.98	3.85	0.31	258.03	502.00	5.81	4.23	0.32	296.63	143.59	14.58	9.15	0.05
768.dta	54.08	36.38	0.98	0.89	0.02	71.64	69.25	1.34	1.2	0.03	343.05	133.02	14.77	9.39	0.04
778.dta	11.28	48.80	0.22	0.19	0.06	12.53	46.69	0.22	0.19	0.06	102.38	102.72	2.71	2.25	0.09
793.dta	8.77	61.67	0.19	0.16	0.07	7.92	84.11	0.23	0.17	0.07	94.09	123.45	2.38	2.06	0.09
794.dta	92.05	206.52	2.89	1.88	0.14	111.92	361.24	3.68	2.13	0.15	213.39	137.50	12.84	6.73	0.02
804.dta	284.12	490.09	6.29	5.08	0.52	339	621.30	7.16	5.78	0.57	256.12	99.61	9.21	6.48	0.04
806.dta	198.14	370.73	4.73	3.85	0.29	238.14	569.47	7.21	5.2	0.41	295.08	139.59	15.53	10.16	0.04
810.dta	22.48	84.45	0.76	0.47	0.06	23.08	135.20	0.7	0.41	0.07	129.08	104.55	11.36	7.18	0.03
822.dta	389.17	703.50	9.68	7.51	0.75	440.95	921.47	10.65	7.9	0.77	473.2	221.39	18.81	14.36	0.19
823.dta	144.81	315.66	3.86	2.79	0.27	183.95	478.20	4.13	2.85	0.28	313.38	131.44	14.81	9.68	0.04
průměr	149.07	295.48	3.32	2.57	0.27	165.33	357.94	3.80	2.80	0.28	263.56	157.62	11.15	7.69	0.10

Tabulka 4.26: Metrické indexovací metody a sekvenční zpracování.

5 Závěr

První část rešerše (viz kapitola 2) popisuje problematiku analýzy proteinových sekvencí pomocí hmotnostní spektrometrie a mapuje stávající algoritmické postupy interpretace hmotnostních spekter. Rozebrány jsou metody pro jednoduchou hmotnostní analýzu (MS) a pro tandemovou hmotnostní spektrometrii (MS/MS). Interpretace dat je přitom založena buď na vyhledávání v databázích známých proteinových sekvencí (PMF, PFF, Sequence Tag) nebo na použití grafových algoritmů („De Novo" peptidové sekvenování).

Úspěšnost současných technik založených na využití grafů je však v praxi nižší než 30 % [4]. Je to důsledkem toho, že experimentálně získaná hmotnostní spektra obvykle obsahují nekompletní iontové série a že interpretace peaků často není jednoznačná. Nepřítomnost některých peaků a záměny aminokyselin s podobnou hmotností totiž způsobují, že ve výsledku dostaneme pro dané spektrum mnoho odpovídajících sekvencí, z nichž nejsme schopni rozpoznat tu správnou.

Množství nalezených sekvencí minimalizují metody založené na vyhledávání v databázích. Umožňují přitom interpretovat nejen spektra již známých sekvencí, ale s využitím databází získaných překladem DNA jsou vhodné i pro identifikaci dosud neznámých proteinů. S rostoucí velikostí databáze však kromě větší pravděpodobnosti nalezení odpovídající sekvence roste i čas potřebný k jejímu prohledání. Komplikace rovněž způsobují posttranslační modifikace sekvencí, které nelze při konstrukci databáze algoritmickým překladem DNA podchytit, protože v živých organismech probíhají až dodatečně po překladu DNA do proteinů. Hledání posttranslačních modifikací stále zůstává otevřeným problémem.

Cílem práce bylo prozkoumat možnosti indexování hmotnostních spekter metrickými indexovacími metodami M-strom a PM-strom. Metrické indexovací metody jsou rozebrány ve druhé části rešerše (viz kapitola 3). Implementováno a otestováno bylo s využitím frameworku ATOM několik běžně známých typů metrik - Eukleidovská vzdálenost, maximová vzdálenost, kosinová podobnost a Hausdorffova vzdálenost. Definována byla logaritmická vzdálenost a intervalový dotaz pro metrické struktury M-strom a PM-strom. Na základě použití maximové metriky (resp. Hausdorffovy metriky) a sady intervalových dotazů byl navržen algoritmus pro vyhledávání modifikovaných peptidových sekvencí.

Pro sestavování dotazovaných objektů z experimentálních spekter byly navrženy dvě heuristiky - první založená na pozicích výskytu y-iontů, druhá na hledání párových y a b-iontů. Úspěšnost použití M-stromu a PM-stromu je přitom do značné míry ovlivněna právě těmito heuristikami a jejich případné vylepšení otevírá prostor pro další možnosti výzkumu. Heuristika založená na pozicích y-iontů se ukazuje jako výhodnější pro ESI spektra, heuristika založená na hledání párových y a b-iontů pak pro tandemová MALDI spektra.

Na základě provedených experimentů se z testovaných metrik ukázalo jako nejvhodnější použití logaritmické vzdálenosti. Při vyhledávání peptidových modifikací se osvědčil algoritmus založený na použití maximové (resp. Hausdorffovy) metriky a sady intervalových dotazů. Porovnání se současnými metodami vyhledávání v databázích ukázalo, že metrické indexovací metody jsou vhodné pro interpretaci tandemových hmotnostních spekter. I při použití jednoduchých heuristik pro sestavování dotazů z experimentálních spekter totiž dokáží dát výsledky kvalitativně srovnatelné se současnými metodami vyhledávání v databázích. Detailní porovnání časů výpočtu nebylo provedeno, protože lokální instalace vyhledávačů ProteinProspector a Mascot nejsou volně dostupné. Oproti sekvenčnímu zpracování celé databáze při použití metrických indexovacích metod dosahujeme řádově 1000 násobného zrychlení (viz tabulka 4.26).

6 Literatura

[1] Achieving I/O Improvements in a Mass Spectral Database.
http://www.acm.org/crossroads/xrds13-2/dataio.html.

[2] Acquisition of Proteomic Data Using Mass Spectrometry and Their Bioinformatics Interpretation.
http://chemicke-listy.cz/common/content-issue_4-volume_101-year_2007.html.

[3] De Novo Peptide Sequencing via Tandem Mass Spectrometry.
http://www2.in.tu-clausthal.de/~hammer/biosem/paper/dancik.pdf.

[4] Doplňující materiály ke knize „An Introduction to Bioinformatics Algorithms".
http://www.bioalgorithms.info/slides.php.

[5] Introduction to Computational Proteomics.
http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1933459.

[6] Materiály pro výuku předmětu „Problémy a algoritmy".
http://service.felk.cvut.cz/courses/36PAA/.

[7] Materiály pro výuku předmětu „Vyhledávání v multimediálních databázích".
http://siret.ms.mff.cuni.cz/skopal/DBI030.htm.

[8] Matrix Science - Help Topic Index.
http://www.matrixscience.com/help/.

[9] Mutation-tolerant protein identification by mass spectrometry.
http://www.cse.ucsd.edu/~ppevzner/papers2000.html.

[10] Peptide Mass Spectrum Interpretation.
http://www.weddslist.com/ms/.

[11] The Global Proteome Machine Organization.
http://www.thegpm.org/.

[12] The SpectraPhile ESI MS Deconvolution applet.
http://home.iprimus.com.au/pakholt/lcms/spectraPhile.html.

[13] Un algoritmo efficiente per il sequenziamento de novo di un peptide.
http://www.dsi.unive.it/~simeoni/ElisaMori.pdf.

[14] UNIMOD - databáze proteinových modifikací pro použití v hmotnostní spektrometrii.
http://www.unimod.org/.

[15] Základy MALDI-TOF MS pro studium bílkovin.
http://biomikro.vscht.cz/maldiman/cz/theory/.

[16] Alberts, B.; Bray, D.; Johnson, A.; aj.: *Základy buněčné biologie - Úvod do molekulární biologie buňky*. Espero Publishing, Ústí nad Labem, české vydání, 1998.

[17] Ashcroft, A.: An Introduction to Mass Spectrometry.
http://www.astbury.leeds.ac.uk/facil/MStut/mstutorial.htm.

[18] Bustos, B.; Navarro, G.; Chávez, E.: Pivot Selection Techniques for Proximity Searching in Metric Spaces. 2003.

[19] Ciaccia, P.; Patella, M.; Zezula, P.: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of 23rd International Conference on VLDB*, editace M. Kaufmann, 1997, s. 426–435.

[20] Cvrčková, F.: *Úvod do praktické bioinformatiky.* Academia, Praha, první vydání, 2006.

[21] Guzzetta, A.: IonSource - Mass Spectrometry and Biotechnology Resource. `http://www.ionsource.com`.

[22] Hoksza, D.: *Vícerozměrné indexování pro relační SŘBD.* Diplomová práce, Matematicko-fyzikální fakulta UK v Praze, Katedra softwarového inženýrství, 2006.

[23] Hoksza, D.; Skopal, T.: Index-based approach to similarity search in protein and nucleotide databases. 2007.

[24] Holčapek, M.: Materiály pro výuku předmětu „Hmotnostní spektrometrie v organické analýze". `http://holcapek.upce.cz/teaching_CZ.htm`.

[25] Jones, N. C.; Pevzner, P. A.: *An Introduction to Bioinformatics Algorithms.* MIT Press, Cambridge, Massachusetts, 2004.

[26] Kolář, J.: *Teoretická informatika.* Česká informatická společnost, Praha, 2004.

[27] Kočárek, E.: *Genetika.* Scientia, Praha, první vydání, 2004.

[28] Krátký, M.; Skopal, T.; Snášel, V.: Porovnání některých metod pro vyhledávání a indexování multimediálních dat. 2002.

[29] Matthiesen, R.: *Mass Spectrometry Data Analysis in Proteomics.* Humana Press, Totowa, New Jersey, 2007.

[30] Patella, M.: *Similarity Search in Multimedia Databases.* Dipartimento di Elettronica Informatica e Sistemistica, Bologna, 1999, `http://www-db.deis.unibo.it/Mtree/index.html`.

[31] Skopal, T.: *Metric Indexing in Information Retrieval.* Dizertační práce, VŠB – Technická univerzita Ostrava, 2004.

[32] Skopal, T.; Pokorný, J.; Snášel, V.: PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. ADBIS, Budapest, Hungary, 2004.

[33] Zezula, P.; Amato, G.; Dohnal, V.; aj.: *Similarity Search - The Metric Space Approach.* Springer, USA, 2006.

A Obsah přiloženého CD

```
/
|_ /source                              - soubory implementační části
|    |_ /atom                           - framework ATOM
|    |_ /data                           - datové soubory pro experimentální část
|    |    |_ /converted                 - konvertované testovací kolekce dat
|    |    |                               (viz index.htm)
|    |    |_ /dta                       - soubory .dta použité v sekci 4.10
|    |    |_ /original                  - nekonvertované soubory testovaných kolekcí
|    |    |_ /packed                    - původní komprimované balíky kolekcí dat
|    |    |_ added_human_e.zip          - soubory dat ve FASTA formátu
|    |    |_ added_human_e_y.zip        - soubory dat, viz tabulka 4.16 a sekce 4.10
|    |    |_ added_human_e_yb.zip       - soubory dat, viz tabulka 4.21
|    |    |_ opal-amet-gv_y.zip         - soubory dat, viz tabulka 4.3
|    |
|    |_ /gen_avs                        - utilita generující soubory pro naplňení
|    |                                    indexových struktur
|    |_ /parse_xml                      - utilita pro konverzi kolekcí testovacích
|    |                                    dat z formátu XML
|    |_ /spectrometry                   - utilita pro indexování a vyhledávání dat
|    |_ index.htm                       - popis utilit a generování datových souborů
|
|_ /tex                                 - zdrojové soubory textu práce
     |_ /gnuplot                        - soubory pro vykreslení grafů GNUPLOTem
     |_ /graphs                         - grafy ve formátu PDF
     |_ /images                         - obrázky ve formátech PDF a JPG
     |_ /svg                            - obrázky ve vektorovém formátu SVG
     |_ diplomka.tex                    - hlavní soubor projektu pro LaTeX
     |_ diplomka.pdf                    - vlastní text práce
```

Seznam použitých zkratek

AA	Amino Acid
A_r	Atomová relativní hmotnost
ATOM	Amphora Tree Object Model
CID	Collision Induced Dissociation
Da	Dalton
DAC	Disk Access Cache
DNA	Deoxyribonucleic Acid
ESI	Electrospray Ionization
FEI VŠB	Fakulta elektrotechniky a informatiky, Vysoká škola báňská
GPM	The Global Proteome Machine Organization
HR	Hyper-ring Regions
I/O	Input/Output
kNN	k-Nearest Neighbor
MALDI	Matrix Assisted Laser Desorption Ionization
MOWSE	MOlecular Weight SEarch
M_r	Molekulová relativní hmotnost
MS	Mass Spectrometry, Mass Spectrometer, Mass Spectrum
MS/MS, MS^2	Tandem Mass Spectrometry
MST	Minimum Spanning Tree
M-tree	Metric Tree
NN	Nearest Neighbours
OMSSA	Open Mass Spectrometry Search Algorithm
PD	Pivot Distances
PFF	Peptide Fragment Fingerprinting
PMF	Peptide Mass Fingerprinting
PM-tree	Pivoting Metric Tree
PR	Pending Requests
PSD	Post Source Decay
QTOF	Quadrupole TOF
QSTAR	obdoba QTOF
RNA	Ribonucleic Acid
SPC	Shared Peak Count
TOF	Time Of Flight
XML	eXtensible Markup Language

Rejstřík