Kapitola 5

Návrh predikčného systému

Navrhnutý systém z veľkej časti vychádza z dvoch prác venujúcich sa SSP s využitím CA [13] [77]. Modifikované sú tie časti systému, ktoré majú potenciál zlepšiť úspešnosť predikcie SSP. Ide hlavne o spôsob klasifikácie jednotlivých reziduí do jednej z troch motívov sekundárnej štruktúry a parametrizácie modelu CA.

Ako bolo spomenuté v sekcii 2.3, výsledná štruktúra proteínov závisí na veľkom množstve známych či neznámych parametrov. Niektoré parametre je možné získať iba experimentálne. No môžeme hodnoty týchto parametrov nahradiť hodnotami najpodobnejšej sekvencie získanej zarovnaním (napríklad pomocou algoritmu BLAST [4]), pretože konzervácia častí sekvencie spôsobuje, že proteíny v rámci jednej rodiny majú podobné vlastnosti. Aminokyselinová sekvencia však musí byť porovnávaná s celou databázou sekvencií, navyše, používané algoritmy zarovnávania majú linárnu časovú zložitosť v závislosti na dĺžke sekvencie. Jednoduchšie je využiť informácie v dátach, tzn. protriedkami štatistiky dáta nejakým spôsobom popísať, ideálne z pohľadu nastoleného problému. Takéto získanie informácii je samozrejme obmedzené pokrytím dát. Takýmto spôsobom však možno získať aj také informácie, ktoré sú neznáme, skryté, no nejakým spôsobom agregované do korelácii atp. Jedným z cieľov návrhu je, aby bola predikcia systému rýchla a aby bol systém robustný, tzn. mal by byť schopný predikovať sekundárnu štruktúru ľubovolnej aminokyselinovej sekvencie, teda mal by fungovať bez nutnosti získavania ďalších potrebných informácii o predikovanej sekvencií. Na to tu sú vhodnejšie, sofistikované prediktory využívajúce evolučné informácie získané na základe zarovnania sekvencie v rámci určitej proteínovej rodiny, alebo chemické posuny, kde je taktiež potrebné zarovnanie a hľadanie najpodobnejšej aminokyselinovej sekvencie. Uvažovaním týchto parametrov do navrhovaného systému by model CA strácal zmysel a vhodnejšie by sa javili iné predikčné modely. Samozrejme, dosahovaná úspešnosť nebude závratná, no adekvátna požadovaným vlastnostiam modelu.

Predikčným modelom je spomínaný CA, ktorého prechodová funkcia je suboptimálne parametrizovaná pomocou evolučného algoritmu, konkrétne pomocou evolučnej stratégie. Boli navrhnuté 2 prechodové funkcie. Prvá je zhodná s prechodovou funkciu vytvorenou Choprom a Benderom v [13], bola implementovaná najmä kvôli porovnávaniu s druhou, rozšírenou verziou prechodovej funkcie, ktorá využíva okrem klasickým Chou-Fasmanových koeficientov aj tzv. konformačné koeficienty (viď sekcia 5.1), ktoré štatisticky popisujú pravdepodobnosť výskytu určitej aminokyseliny v určitom konformačnom stave resp. motíve sekundárnej štruktúry.

5.1 Štatistický popis reziduí

V návrhu systému sú využité 2 štatistické vlastnosti aminokyselín, *Chou-Fasmanove koeficienty*, ktoré aminokyseliny charakterizujú z pohľadu miery výskytu v určitom motíve sekundárnej štruktúry, a *konformačné preferencie*, ktoré popisujú jednotlivé aminokyseliny na základe predispozície nachádzať sa na začiatku, resp. na konci určitého motívu sekundárnej štruktúry.

Jednou z metód predikcie SSP prvej generácie je metóda Chou-Fasman (viď 2.3 pre prehľad predikčným metód), v ich práci z roku 1974 [15] je definovaný tzv. parameter konformačnej predispozície aminokyseliny j ku konformačnému stavu i, Chou-Fasmanov koeficient P_j^i:

$$P_j^i = \frac{f_j^i}{\langle f_j^i \rangle}, \tag{5.1}$$

kde f_j^i je relatívna frekvencia aminokyseliny j v konformačnom stave i daná vzťahom 5.2 a $\langle f_j^i \rangle$ priemerná relatívna frekvencia konformačného stavu i v rámci všetkých aminokyselín vyjadrená vzťahom 5.3. Kvôli konzistencii s odkazovanými prácami sa v systéme s Chou-Fasmanovými koeficientami P_j^i pracuje v percentuálnej podobe, tzn $P_j^i \cdot 100$.

$$f_j^i = \frac{n_j^i}{n^i} \tag{5.2}$$

kde n_j^i je počet reziduí j v konformačnom stave i a n^i je celkový počet reziduí v konformačnom stave i.

$$\langle f_j^i \rangle = \frac{\sum\limits_{\forall k \, \epsilon \, AK} f_k^i}{n_j} \tag{5.3}$$

Na základe práce Guang-Zheng Zhanga a spol. [82] definujeme *konformačnú triedu* pre všetky aminokyseliny a všetky konformačné stavy (H, E, C). Nech $P = p_1, p_2, \ldots, p_n$ je primárna štruktúra proteínu (sekvencia aminokyselín) a $S = s_1, s_2, \ldots, s_n$ odpovedajúca sekundárna štruktúra proteínu dĺžky n. Ak s_i a s_{i+1} sú rôzne konformačné stavy, napríklad $s_i = H$ a $s_{i+1} = E$, hovoríme o tzv. štruktúrnom prechode (ST[1]), v tomto prípade ST_{HE}. Týmto spôsobom môžeme definovať ostatných 5 štruktúrnych prechodov: ST_{HC}, ST_{EH}, ST_{EC}, ST_{CH} a ST_{CE}.

Na základe uvedených štruktúrnych prechodov definujeme konformačnú preferenciu aminokyselín nachádzať sa na začiatku resp. konci určitého motívu sekundárnej štruktúry. Štruktúrny prechod ST_{HE} môžeme chápať ako ukončenie H a súčasne ako začiatok B. V kontexte všetkých 6 ST, počet všetkých ukončení a začiatkov H určíme nasledovne:

$$N_{\alpha\text{-ukončenie}} = N_{ST_{HB}} + N_{ST_{HC}} \tag{5.4}$$

$$N_{\alpha\text{-začiatok}} = N_{ST_{BH}} + N_{ST_{CH}} \tag{5.5}$$

[1] Z angl. Structure Transition.

kde $N(\cdot)$ reprezentuje počet rôznych štruktúrnych prechodov. Počet ukončení a začiatkov B a C sa určí analogicky. Definujeme konformačnú preferenciu CP ukončenia resp. začiatku určitého konformačného stavu aminokyseliny i, konkrétne $CP_{j,\alpha\text{-ukončenie}}$, $CP_{j,\alpha\text{-začiatok}}$, $CP_{j,\beta\text{-ukončenie}}$, $CP_{j,\beta\text{-začiatok}}$, $CP_{j,\text{Coil}-\text{ukončenie}}$ a $CP_{j,\text{Coil-začiatok}}$. Výpočet $CP_{j,\alpha\text{-ukončenie}}$ (ostatné konformačné preferencie sa získajú analogicky):

$$CP_{j,\alpha\text{-ukončenie}} = \frac{P_{j,\alpha\text{-ukončenie}}}{P_j} \tag{5.6}$$

kde $P_{j,\alpha\text{-ukončenie}}$ a P_j sa získa nasledovne:

$$P_{j,\alpha\text{-ukončenie}} = \frac{N_{j,\alpha\text{-ukončenie}}}{\sum_{i=1}^{20} N_{i,\alpha\text{-ukončenie}}} \tag{5.7}$$

kde $N_{j,\alpha\text{-ukončenie}}$ vyjadruje počet reziduí ukončujúcich H.

$$P_j = \frac{N_j}{N} \tag{5.8}$$

kde N_j vyjadruje celkový počet reziduí aminokyseliny j a N celkový počet reziduí. Uvažujúc reziduum j a jeho motív sekundárnej štruktúry, α-helix (H), definujeme konformačnú triedu (CC) konformačného stavu rezidua j (obdobne možno vyjadriť konformačné triedy pre B a C):

$$CC_{j,\alpha} = \begin{cases} b & \text{ak } CP_{j,\alpha\text{-ukončenie}} \geq 1 \ \wedge \ CP_{j,\alpha\text{-začiatok}} < 1 \\ f & \text{ak } CP_{j,\alpha\text{-začiatok}} \geq 1 \ \wedge \ CP_{j,\alpha\text{-ukončenie}} < 1 \\ n & \text{inak} \end{cases} \tag{5.9}$$

Použité znaky b, f, n značia v tomto poradí triedy *Breaker*, *Former* a *Neutral*. Konformačná klasifikácia rezidua j je vyjadrená 3-znakovým kódom – $CC_{j,\alpha}CC_{j,\beta}CC_{j,Coil}$. Pre potreby modelu nie je použitá vlastná konformačná klasifikácia $CC_{j,\alpha}$, ale konformačné preferencie, na základe ktorých sa klasifikuje, tzn. $CP_{j,\alpha\text{-ukončenie}}$ resp. $CP_{j,\alpha\text{-začiatok}}$.

5.2 1 D celulárny automat ako model aminokyselinovej sekvencie

Sekvenciu aminokyselín proteínov reprezentuje 1 D CA, ktorého bunky modelujú jednotlivé reziduá aminokyselín. Bunky môžu nadobúdať jeden z troch stavov (H, E, C). Štatistické vlastnosti aminokyselín sú modelované parametrami buniek CA. Veľkosť okolia nie je optimalizovaná pomocou evolučného algoritmu (ale je parametrizovateľná), najmä kvôli možnej paralelizácii výpočtu a relatívnej zložitosti genetických operátorov navyšujúcej čas evolúcie optimálneho pravidla. V rámci inicializácie sú každej bunke pridelené Chou-Fasmanove koeficienty, ktoré sú pri prechode do ďalšej konfigurácie CA upravované a vyjadrujú predispozície bunky nachádzať sa v určitom stave. Prechodová funkcia CA môže mať podobu základnú alebo rozšírenú. *Základná prechodová funkcia* [13] má nasledovný tvar:

$$S_{t+1,j} = \max R^i_{t+1,j} \quad i \,\epsilon\, \{H, E, C\} \tag{5.10}$$

kde $S_{t+1,j}$ je stav bunky j v čase $t+1$ a parameter $R^i_{t+1,j}$ vyjadruje mieru príslušnosti bunky resp. aminokyseliny j v kroku $t+1$ ku konformačnému stavu i (H, E, C):

$$R^i_{t+1,j} = P^i_{t+1,j} \tag{5.11}$$

$P^i_{t+1,j}$ vyjadruje Chou-Fasman koeficient bunky j v čase $t+1$ pre konformačný stav i, ktorý je váhovaným súčtom jednotlivých Chou-Fasmanových koeficientov P^i_{j-k} v okolí o:

$$P^i_{t+1,j} = \frac{\sum_{k=-o}^{o} w_k P^i_{j-k}}{\sum_{k=-o}^{o} w_k} \tag{5.12}$$

Rozšírená prechodová funkcia sa od základnej líši v definícii predispozícii bunky nachádzať sa v danom stave $R^i_{t+1,j}$:

$$R^i_{t+1,j} = \alpha \cdot P^i_{t+1,j} + \beta \cdot CP_{j,i\text{-začiatok}} + \gamma \cdot CP_{j,i\text{-ukončenie}} \tag{5.13}$$

kde α, β a γ sú váhy troch definovaných parametrov, ktoré sú optimalizované evolučným algoritmom.

5.3 Okrajové podmienky a inicializácia modelu

Pri modelovaní javov pomocou celulárneho automatu je dôležitá definícia okrajových podmienok popisujúcich situácie, kedy okolie aktuálnej bunky nie je kompletné – týka sa väčšinou buniek na okraji automatu. Podľa Chopra a Bendera, na základe expermentov, ktoré vykonali, je vhodné použiť bunky okolia mimo automatu v štruktúre Coil [13]. Vojtěch Šalanda sa takýmito bunkami zaoberal [77], experimentoval s reálnymi aj fiktívnymi, reálne neexistujúcimi aminokyselinami, a zistil, že ako najvhodnejšia sa javí fiktívna aminokyselina s označením X300, ktorej Chou-Fasmanove parametre sú 0-0-300, tzn. tendencia bunky nachádzať sa v štruktúre Coil je nenulová, má hodnotu 300.

Na inicializácii jedincov v rámci evolučného algoritmu v podstate nezáleží, no pre rýchlosť konvergencie je dôležité sa zaoberať aj touto časťou systému. Je intuitívne jasné, že vplyv reziduí v tesnom okolí predikovanej aminokyseliny bude vyšší než vplyv reziduí vzdialenejších. Platí to najmä pri motívoch α-helixu, no motívy β-sheet často vznikajú globálnou interakciou, ktorú by sa mal, vzhľadom k charaktere modelu, pokúsiť emulovať navrhnutá model celulárneho automatu.

Použitá je inicializácia hodnôt vplyvu jednotlivých okolitých reziduí (váh) na základe normalizovanej Gaussovej funkcie pre strednú hodnotu $\mu = 0$ a smerodatnú odchýlku $\sigma = 0.399$ (hodnota $f(0) \doteq 1$):

$$\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5.14}$$

5.4 Optimalizácia vektoru parametrov pomocou evolučnej stratégie

Motorom CA je jeho prechodová funkcia, ktorej expertné určenie nie je jednoduché, preto sa na jej určenie využívajú rôzne optimalizačné techniky. Keďže ide o optimalizáciu vektoru celých a reálnych čísel, je použitý algoritmus evolučnej stratégie, ktorý je podmnožinou väčšej triedy optimalizačných techník, evolučných algoritmov. Stavový priestor prechodových funkcií, ktoré sú parametrizované reálnymi číslami je teoreticky nekonečný, čo opodstatňuje použitie optimalizačných techník. Evolvovaný chromozóm základného resp. rozšíreného pravidla, C_z resp. C_r má tvar:

$$\begin{aligned} C_z &= [s, w_{-r}, w_{-r+1}, \ldots, w_{r-1}, w_r] \\ C_r &= [s, \alpha, \beta, \gamma, w_{-r}, w_{-r+1}, \ldots, w_{r-1}, w_r] \end{aligned}$$

kde s vyjadruje počet krokov EA, α vplyv predchádzajúcej predispozície na stav bunky a teda na úspešnosť predikčného modelu, β vplyv konformačného koeficientu, ktorý vyjadruje schopnosť určitej aminokyseliny začínať určitý motív sekundárnej štruktúry, γ vplyv konformačného koeficientu, ktorý vyjadruje schopnosť určitej aminokyseliny končiť určitý motív sekundárnej štruktúry, a w_i pre $i \epsilon \{-r, \ldots, r\}$ váhy jednotlivých buniek okolia.

Fitness funkciou evolučného algoritmu je jedna z dvoch funkcií porovnávajúcich podobnosť sekvencií (viď sekcia 2.4), Q_3 alebo SOV. Prehľad spôsobu implementácie evolučných operátorov ponúka tabuľka 5.1.

Evolučný operátor	**Spôsob implementácie**
Selekcia	Koleso šťastia
Kríženie	1-bodové
Mutácia	Gaussovo rozdelenie pravdepodobnosti
Náhrada populácie	Čiastočná (steady state)

Tabuľka 5.1: Navrhnuté spôsoby implementácie evolučných operátorov.

Kapitola 6

Popis implementácie systému

Implementácia prediktora, nazvaného *CASSP* (Cellular Automaton Secondary Strucure Predictor), predstavuje prepis návrhu systému do podoby núl a jedničiek, do podoby inštrukcií procesora. Tento prepis je vcelku priamočiary proces. Ide najmä o hľadanie čo najvhodnejších implementačných prostriedkov, ktoré budú čo najlepšie reprezentovať navrhnutý model.

Prediktor je implementovaný v jazyku Java JRE (Java Runstime Environment) 1.6, zaistená je bezproblémová funkčnosť aj pre JRE 1.7. Na implementáciu evolučného algoritmu bola použitá voľne dostupná knižnica JGAP [54].

6.1 Konfigurácia a API systému

Výsledný program má dve varianty – konzolovú a webovú. Konzolová aplikácia je konfigurovateľná pomocou konfiguračného súboru a argumentov príkazovej riadky s tým, že konfigurácia parametrov v príkazovej riadke má vyššiu prioritu než konfigurácia v konfiguračnom súbore, čím je zabezpečená určitá flexibilita konfigurácie programu. Veľké množstvo vlastností systému je parametrizovaných, kompletné možnosti konfigurácie možno nájsť v dokumentácii k systému. Systém má definované API, tzn. možno ho tiež použiť ako knižnicu. Príklad použitia API (použitie cross-validácie):

```
SimConfig config = new SimConfig(<conf_file_path>);
config.setDataPath(<data_path>);
config.setCrossProb(0.75);
config.setPop(100);

CASSP predictor = new CASSP(config);
predictor.crossValidate(10); // 10-stupňová cross-validácia

predictor.createEvolutionImage("evolution");
predictor.createAccClassesImage("accuracy");
predictor.createReliabImage("reliability");
```

6.2 Vlastnosti systému

Výhodou systému je možná paralelizácia výpočtu pri trénovaní prechodovej funkcie pomocou cross-validácie, ktorá je implementovaná pomocou vláken. Pre možnú vlastnú definíciu prechodovej funkcie CA je implementovaná abstraktná trieda `CARule`. Prepísaním

```
FVNQHLCGSHLVEALYLVCGERGFFYTPKA
CCCCCCCCHHHHHHHHHHHHHHHCECCCCCC
...
```

(a) Formát dát pre systém CASSP.

```
FVNQHLCGSHLVEALYLVCGERGFFYTPKA
CCCCCCCCHHHHHHHHHHHHHHHCECCCCCC
CCCCCCHHHHHHHHHHHHHHCCCCEEECCCC
95401326789999999708622662589
...
```

(b) Formát dát pre systém CASSP využívajúci nástroj PSIPRED.

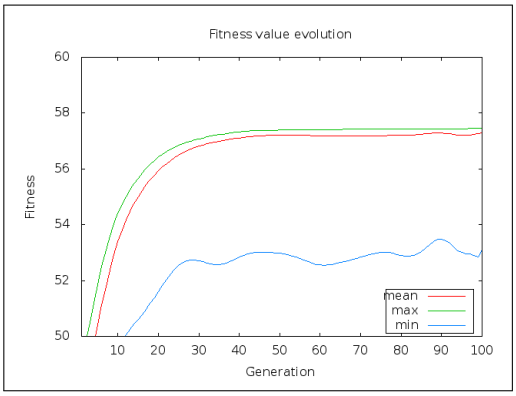
Obrázok 6.1: Formát dátových súborov pre samostatný nástroj CASSP (a) – aminokyselinová sekvencia, referenčná sekvencia sekundárnej štruktúry, a pre systém spolupracujúci s nástrojom PSIPRED (b) – aminokyselinová sekvencia, referenčná sekvencia sekundárnej štruktúry, sekvencia sekundárnej štruktúry predikovaná nástrojom PSIPRED, koeficienty spoľahlivosti predikcie nástroja PSIPRED.

metód `toChromosome`, `fromChromosome` a `nextState` možno dosiahnuť požadované správanie prechodovej funkcie. Spustením metód modulu CASSP – `createEvolutionImage`, `createReliabImage` a `createAccClassesImage` sa vytvoria obrázky popisujúce dosiahnutý výsledok (viď obrázok 6.2) vo formáte PNG [64]. Dáta potrebné pre tvorbu obrázkov sú uložené do textového súboru pre vlastné zobrazenie týchto dát. Pre neskoršie použitie získaného pravidla je implementovaná jeho serializácia, metóda `loadRule` pravidlo načíta, metóda `saveRule` pravidlo uloží do požadovaného súboru. V rámci systému je vytvorený model zapúzdrujúci nástroj PSIPRED triedou `Psipred`. Pri trénovaní/testovaní CASSPu s nástrojom PSIPRED je však nutný iný formát dát (viď obrázok 6.1).

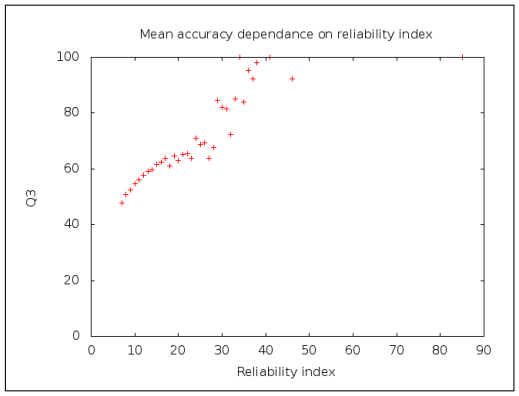
6.3 Webové rozhranie

Webové rozhranie je minimalistické, no spĺňa účel. V rámci webového rozhrania nemožno trénovať nové prechodové pravidlá, pre každý spôsob predikcie je nastavené pravidlo dosahujúce najlepšej úspešnosti predikcie. Bol použitý open-source framework Google Web Toolkit[1] (GWT). ktorý poskytuje nástroje potrebné pre jednoduchú tvorbu a správu JavaScriptových front-endových aplikácií. Ide o server-klient komunikáciu, na strane serveru je použitá technológia Java servletov, na strane klienta sú preložené funkcie aplikačného rozhrania do JavaScriptového kódu.

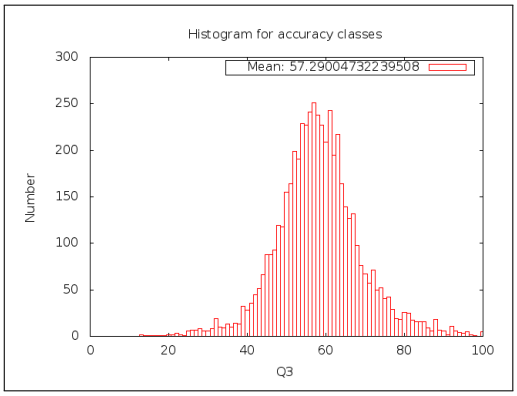
[1] Domovská stránka Google Web Toolkitu: https://developers.google.com/web-toolkit/.



(a) Evolúcia hodnoty fitness funkcie (`createEvolutionImage`).



(b) Úspešnosť predikcie v závislosti na hodnoty spoľahlivosti predikcie (`createReliabImage`).



(c) Počet sekvencií v závislosti na úspešnosti predikcie (`createAccClassesImage`).

Obrázok 6.2: Obrázky popisujúce „výkonnosť“ evolučného algoritmu a vlastnosti prediktora na základe určitej dátovej sady.

Kapitola 7

Systém je použiteľný ako sekundárny prediktor

Primárnou snahou experimentovania s navrhnutým modelom bolo zlepšiť úspešnosť predikcie sekundárnej štruktúry proteínov jedného v súčasnosti najlepších nástrojov, PSIPRED-u. Systém je však otestovaný aj ako samostatný prediktor a jeho úspešnosť porovnaná s ostatnými metódami.

Pre dosiahnutie čo najlepších výsledkov je dôležitá rozumná parametrizácia modelu. Pre jej dosiahnutie bola najskôr vykonaná optimalizácia parametra veľkosti okolia, ktoré uvažuje prechodová funkcia CA. Následne bol zistený optimálny maximálny počtet krokov CA, ktoré môžu pri evolúcii pravidla CA jednotlivé riešenia nadobúdať. Po správnom „nastavení" týchto 2 parametrov boli vykonané experimenty predikujíce sekundárnu štruktúru proteínu v spolupráci s nástrojom PSIPRED. Uvažované boli 2 varianty:

1. Primárna predikcia pomocou navrhnutého systému CASSP a následná oprava nie príliš vierohodných predikcii pomocou nástroja PSIPRED.

2. Primárna predikcia pomocou nástroja PSIPRED a následná oprava nie príliš vierohodných predikcií pomocou navrhnutého systému CASSP.

V oboch prípadoch je dôležité správne stanoviť prah opravy primárnej predikcie pomocou predikcie sekundárnej. Pre zistenie vhodného prahu bola vykonaná jeho optimalizácia pre dve varianty:

1. Použitie sekundárneho prediktora pre reziduá, ktorých vierohodnosť predikcie je nižšia než zadaný prah.

2. Použitie sekundárneho prediktora pre celú proteínovú sekvenciu, ak priemerná vierohodnosť reziduí v rámci opravovanej sekvencie je nižšia než zadaný prah.

7.1 Trénovacie a testovacie dátové sady

Pri experimentoch boli použité 3 dátové sady – RS126, CB513 a PDBselect. Dátová sada RS126 bola prvý krát použitá v článku Burkharda Rosta a Chrisa Sandera z roku 1993 [69]. Podľa ich slov ide o nehomológnu dátovú sadu. Nehomológnosť definovali tak, že žiadne 2 proteíny v dátovej sade nesmú mať viac než 25 %-nú zhodu v sekvenciách pri ich dĺžke presahujúcej 80 reziduí. Nevýhodou tohto súboru 126 sekvencií (okrem malého počtu) je,

že obsahuje páry proteínových sekvencií, ktoré sú podobné pri provnávaní inými, sofistikovanejšími metódami než obyčajnou percentouálnou zhodou. Výpočet percentuálnej zhody je totiž závislý na dĺžke zarovnania a zložení sekvencií, takže 2 sekvencie podobného, ale nezvyčajného aminokyselinového zloženia, môžu mať vysokú percentuálnu zhodu, aj keď nie sú evolučne príbuzné [19].

Dátovú sadu CB513 vytvorili páni Geoffrey Barton a James Cuff v rámci svojej štúdie z roku 1999 [19]. Zhodu dvoch aminokyselinových sekvencií, označme ich A a B, neurčovali na základe percentuálnej zhody, ale pomocou metódy, ktorá najskôr zarovná sekvencie štandardným algoritmom dynamického programovania (napríklad pomocou algoritmu Needleman-Wunsch [57]) a získa sa skóre zarovnania V. Poradie jednotlivých aminokyselín v každej proteínovej sekvencii je náhodne zmenené a následne je vykonané zarovnanie pomocou spomínaného algoritmu dynamického programovania. Tento proces sa opakuje typicky aspoň 100 krát, následne sa vypočíta priemer $\bar{x}$ a smerodatná odchýlka σ jednotlivých skóre zarovnania. Výsledná hodnota „podobnosti“ sekvencií A a B, SD, je určená nasledovne: $SD(A, B) = (V - \bar{x})/\sigma$.

Z počiatočnej dátovej sady sa odstránili multisegmentové domény, odstránené boli aj sekvencie, ktorých štruktúry získané pomocou röntgenovej kryštalografie nemali dostatočné rozlíšenie (aspoň 2,5 Å). Ďalej neuvažované boli tiež sekvencie, ktorých podobnosť s nejakou sekvenciu z dátovej sady RS126 bola $SD >= 5$, a sekvencie, ktoré nemali úplnú DSSP definíciu. Táto precízne prefiltrovaná množina sekvencií bola spojená so 126 sekvenciami dátovej sady RS126. Použitá definícia podobnosti však podľa autorov nie je schopná zachytiť všetky homológne sekvencie, na ďalšie porovnávanie a filtrovanie sekvencií bol použitý algoritmus SCOP [55]. Výsledkom bola dátová sada CP513. Pre potreby tejto práce, teda pre jednotné počítanie v sekcii 5.1 bližšie popísaných Chou-Fasmanovych a konformačných koeficientov boli nejednoznačné aminokyseliny (B,Z,X) nahradené priemernými hodnotami: pre B je to priemer hodnôt asparagínu (N) a kyseliny asparagovej (D), pre Z priemer hodnôt glutamínu (Q) a kyseliny glutámovej (E) a pre J priemer hodnôt leucínu (L) a izoleucínu (I).

Treťou použitou dátovou sadou je rozsiahly súbor približne 5 300 proteínových sekvencií z databázy PDB – PDBSelect [34]. Ide o zoznam reprezentatívnych proteínovych sekvencií s nízkou sekvenčnou podobnosťou (počítanou pomocou HSSP funkcie [1]), ktorý bol vytvorený pre potreby objektívneho štatistického vyhodnodocania okrem iného aj predikcie štruktúry proteínov. Na zarovnanie sekvencií bol využitý rýchly Huang-Miller algoritmus [39]. Jednotlivé selekcie zo zoznamu boli získané pomocou webovej služby MRS [36]. Vzhľadom k rozsiahlosti dátovej sady bola v kontexte tejto práce využívaná iba na testovanie.

Keďže vývoj v oblasti bioinformatiky je veľmi rýchly a počet záznamov v PDB rastie exponenciálne, možno považovať databázy RS126 a CB513 za zastaralé a pre reálne praktické použitie by sa siahlo po novšej, viac aktualizovanej databáze – napríklad spomínanej PDBSelect. No pre základnú charakteristiku navrhnutého modelu sú tieto 2 dátové sady dostačujúce, navyše, takmer všetky nástroje predikcie sekundárnej štruktúry spomenuté v kapitole 2 pracujú práve s týmito dátovými sadami, takže je možné priame porovnanie úspešností.

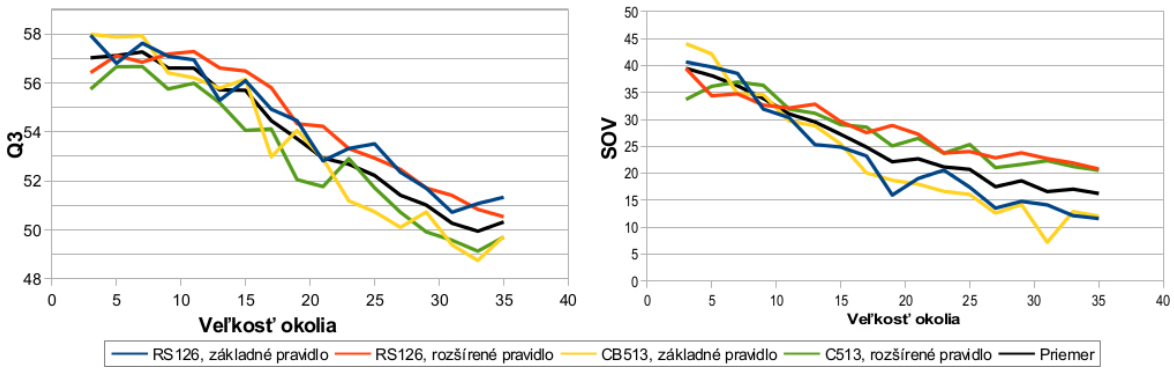
V rámci korektného popisu prediktora je veľmi dôležitý spôsob vyhodnocovania jeho úspešnosti. Azda najdôležitejšou podmienkou je, aby trénovacie a testovacie dáta nekorelovali, čo v reálnych podmienkach nie je jednoduché dosiahnúť, kedy často nie je dostatok dát a ich rozloženie je neznáme. Na mieste je teda otázka, ako ideálne rozdeliť dátovú sadu na trénovaciu a testovaciu tak, aby sme získali čo možno najdôveryhodnejšiu hodnotu úspešnosti resp. chybovosti predikcie? Simone Borra a Agostino Di Ciaccio vo svojej práci [10]

došli k záveru, že najvernejšiu hodnotu chyby prediktoru pre reálne dátové sady vykazuje 10-stupňová cross-validácia pre viac než 100 vzorkov. Leave-One-Out (LOO) cross-validácia teoreticky vykazuje objektívnejší výsledok, no pri testovaní vzhľadom k tomu, že testovacie dátové sady obsahujú iba 1 prvok, sa prejavuje veľká variabilita, čo robí problémy pri selekcii najlepšieho dielčieho modelu. Prístup 10-stupňovej cross-validácie bude teda použitý (pokiaľ nebude povedané inak) pri vyhodnocovaní úspešnosti jednotlivých experimentov. Pri získanie lepšej vierohodnosti je cross-validácia spúšťaná 3-krát a spriemerovaním je získaná výsledná úspešnosť. Na vlastné vyhodnotenie podobnosti referenčných a predikovaných sekundárnych štrukúr proteínových sekvencií boli použité 2 miery - Q_3 a SOV, bližšie popísané v sekcii 2.4.

7.2 Optimalizácia parametrov modelu

Ako bolo uvedené v úvode tejto kapitoly, pre získanie lepších výsledkov sú optimalizované dva základné parametre navrhnutého modelu – veľkosť okolia, s ktorým pracuje prechodová funkcia celulárneho automatu, a maximálny počet krokov CA (po ktorých sa získa predikovaná sekvencia), ktorý je dosiahnuteľný v rámci evolúcie evolučného algoritmu.

Ako je možné vidieť na obrázku 7.1, so zväčšujúcim sa okolím úspešnosť predikcie klesá, čo je čiastočne zrejme zapríčinené tým, že nebol dostatočný čas na trénovanie, predsalen stavový priestor pri väčších hodnotách okolia značne narastá. Ale na druhej strane to sčasti potvrdzuje slová čínskych vedcov, ktorí v 1999 v jednom článku prišli na to, že na determináciu motívu centrálneho rezidua stačí okolie 14–17 a prídavná informácia môže byť nadbytočnou a kontraproduktívnou [59]. Ako optimálne sa v tomto prípade javí *okolie o veľkosti 7*, ktoré bude použité v nasledujúcich experimentoch.



Obrázok 7.1: Klesajúca tendencia úspešnosti predikcie. Natrénované boli modely so základným aj rozšíreným pravidlom CA, s dátovou sadou RS126 aj CB513.