UNIVERSITATEA BABEȘ-BOLYAI
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

Modele bazate pe Instruirea Automată pentru Rezolvarea unor Probleme din Bioinformatică

Rezumatul Tezei de Doctorat

Doctorand: Maria Iuliana Bocicor
Îndrumător științific: Prof. Dr. Gabriela Czibula

Septembrie 2013

Lista publicațiilor

Publicații în ISI Web of Knowledge

Publicații în ISI Science Citation Index Expanded

1. [CBC13] Gabriela Czibula, **Iuliana M. Bocicor** and Istvan Gergely Czibula. Temporal Ordering of Cancer Microarray Data through a Reinforcement Learning Based Approach. *PLoS ONE*, Vol. 8, No. 4, e60883, doi:10.1371/journal.pone.0060883, 2013. **(IF: 4.092)**
2. [CBC12] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. Promoter Sequences Prediction Using Relational Association Rule Mining. *Evolutionary Bioinformatics*, Vol. 8, pag. 181–196, 2012. **(IF: 1.229)**
3. [BCC11a] **Maria Iuliana Bocicor**, Gabriela Czibula and Istvan Gergely Czibula. A Distributed Q-Learning Approach to Fragment Assembly. *SIC Journal, Studies in Informatics and Control* Vol. 20, Issue 3, pag. 221–232, 2011. **(IF: 0.671)**

Publicații în ISI Conference Proceedings Citation Index

4. [BCC11b] **Maria Iuliana Bocicor**, Gabriela Czibula and Istvan Gergely Czibula. A Reinforcement Learning Approach for Solving the Fragment Assembly Problem. In *Proceedings of the 3th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '11)*, pag. 191-198, IEEE Computer Society, 2011.

Articole publicate în jurnale internaționale și în volume ale unor conferințe internaționale

1. [CCB13] Gabriela Czibula, Istvan Gergely Czibula and **Iuliana M. Bocicor**. A Comparison of Reinforcement Learning Based Models for the DNA Fragment Assembly Problem. *Studia Universitatis "Babes-Bolyai", Informatica*, LVIII(2), pag. 90-102, 2013. **(indexed Mathematical Reviews)**
2. [BCG$^+$13] **Iuliana Bocicor**, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri and Marco Antoniotti. Reconstructing Colorectal Cancer Progression through Copy Number Alteration Data. *Complex Systems Models in Biology and Medicine: Generic Properties and Applications* (Va fi trimisă pentru publicare).
3. [Boc12a] **Iuliana M. Bocicor.** A Study on using Reinforcement Learning for Temporal Ordering of Biological Samples. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(4), pag. 63-74, 2012. **(indexed Mathematical Reviews)**
4. [BCG$^+$12] **Iuliana M. Bocicor**, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri and Marco Antoniotti. Ordering copy number alteration data to analyze colorectal cancer progression. Proceedings of NETTAB 2012: Workshop on Integrated Bio-Search, *EMBnet.journal*, Vol. 18, Suppl. B (NETTAB 2012), pag. 84-86, 2012. **(indexed Google Scholar)**
5. [Boc12c] **Maria Iuliana Bocicor.** A Study on Using Association Rules for Predicting Promoter Sequences. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(2), pag. 32-42, 2012. **(indexed Mathematical Reviews)**

6. [CBC11a] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. A Distributed Reinforcement Learning Approach for Solving Optimization Problems. In *Proceedings of the 5th International Conference on Communications and Information Technology (CIT '11)* Greece, pag. 25–30, 2011. **(indexed INSPEC)**
7. [CBC11c] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. A Reinforcement Learning Model for Solving the Folding Problem. *IJCTA - International Journal of Computer Technology and Applications*, Vol. 2, Issue 1, pag. 171–182, 2011. **(indexed Index Copernicus)**
8. [CBC11b] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. An Experiment on Protein Structure Prediction using Reinforcement Learning. *Studia Universitatis "Babes-Bolyai", Informatica*, LVI(1), pag. 25–34, 2011. **(indexed Mathematical Reviews)**
9. [CBC11d] Gabriela Czibula, **Maria Iuliana Bocicor** and Istvan Gergely Czibula. Solving the Protein Folding Problem Using a Distributed Q-Learning Approach. *International Journal of Computers*, Volume 5, Issue 3, pag. 404–413, 2011. **(indexed INSPEC)**
10. [CCB11a] Istvan Gergely Czibula, Gabriela Czibula and **Maria Iuliana Bocicor**. A Software Framework for Solving Combinatorial Optimization Tasks. *Studia Universitatis "Babes-Bolyai", Informatica, Special Issue*, LVI(3), pag. 3–8, 2011. **(indexed Mathematical Reviews)**

Articole publicate în volume ale unor conferințe naționale

1. [Boc12b] **Maria Iuliana Bocicor**. Experiments on Promoter Sequences Prediction using Association Rules. In *Proceedings "Zilele Academice Clujene 2012, Departamentul de Informatica"*, Presa Universitara Clujeana, pag. 32–35, Cluj Napoca, 2012.
2. [CCB11b] Istvan Gergely Czibula, Gabriela Czibula and **Maria Iuliana Bocicor**. A Reinforcement Learning Based Framework for Solving Optimization Problems. In *Post proceedings of Knowledge Egineering Principles and Techniques*, pag. 235-246, Presa Universitara Clujeana, 2011.

Alte publicaţii

1. [Boc11a] **Maria Iuliana Bocicor**. Invatarea automata pentru identificarea regiunilor promotor in ADN. *National Symposum "Interferente", 3rd Edition*, Universitatea de Nord Baia Mare, ISBN 978-606-536-226-0, pag. 68-70, 2012. **(CNCSIS)**
2. [Boc11b] **Maria Iuliana Bocicor**. Modele pentru problema plierii proteinei. *National Symposum "Interferente", 2nd Edition*, Universitatea de Nord Baia Mare, ISBN 978-606-536-146-1, pag. 191–193, 2011. **(CNCSIS)**
3. [Boc10a] **Maria Iuliana Bocicor**. Algoritmi evolutivi aplicati in chemoterapie. *National Symposum "Interferente", 1st Edition*, Ceconi, Baia-Mare, ISBN 978-606-8086-29-3, pag. 145-148, 2010.
4. [Boc10b] **Maria Iuliana Bocicor**. Bioinformatica si aplicatiile ei. *Scoala Maramureseana*, Nr. 41-45, ISSN 1583-2171, pag. 222-223, 2010.

În cazul publicațiilor având mai mulți autori, toți autorii au contribuit în egală măsură la: conceperea, proiectarea și efectuarea experimentelor, analiza datelor, dezvoltarea structurii și argumentelor lucrărilor, redactarea manuscriselor, revizuirea critică și aprobarea versiunilor finale ale articolelor.

Cuprins

Introducere

Această teză de doctorat reprezintă rezultatul cercetărilor noastre în domeniul instruirii automate, concentrându-se în mod special pe *modele bazate pe instruirea automată pentru rezolvarea unor probleme din domeniul bioinformaticii*. Cercetările au fost inițiate în 2010 sub îndrumarea doamnei Prof. Dr. Gabriela Czibula.

Direcţia principală de cercetare pe care ne axăm este aplicarea unor modele din domeniul instruirii automate pentru rezolvarea unor probelme complexe din bioinformatică.

Instruirea automată (machine learning - ML) [Mit97], o ramură a intreligenței artificiale, se ocupă cu dezvoltarea de algoritmi care învață automat, prin experiență. ML este un domeniu interdisciplinar care folosește cunoștințe și rezultate obținute într-o varietate mare de domenii: inteligență artificială, matematică, statistică şi probabilităţi, teoria informaţiei, psihologie, neurobiologie. Chiar dacă în prezent calculatoarele nu sunt capabile să învețe la fel de bine ca și oamenii, există o multitudine de modele teoretice și algoritmi care învață prin experiență, care sunt foarte eficienți pentru o gamă largă de probleme complexe.

Bioinformatica este un domeniu interdisciplinar de cercetare, care își propune rezolvarea unor probleme din biologie, biochimie sau medicină prin aplicarea unor tehnici computaționale pentru colectarea, gestionarea, organizarea, dar mai ales pentru analiza informațiilor biologice. Multe dintre problemele importante din bioinformatică sunt foarte complexe și pentru astfel de sarcini dificile metodele ML s-au dovedit a fi foarte adecvate.

Problemele pe care am decis să le abordăm se numără printre marile provocări din bioinformatică. În scopul de a găsi soluții optime ale acestor probleme au fost dezvoltați o serie algoritmi în domeniul inteligenței computaționale. Scopul nostru principal este de a dezvolta noi modele și algoritmi de instruire automată care ar putea oferi soluții comparabile și chiar mai bune decât cele existente, atât în ceea ce privește calitatea soluțiilor, cât și timpul computațional al algoritmilor.

Prin urmare, cercetările noastre se axează în principal pe două direcții. Prima se referă la aplicarea *regulilor de asociere relaționale* [SCC06] pentru rezolvarea unor probleme de clasificare din bioinformatică. A doua direcție se referă la aplicarea unor tehnici bazate pe *învățarea prin întărire* pentru a rezolva probleme de optimizare combinatorială NP-complete din bioinformatică.

În plus față de direcțiile majore de cercetare menționate mai sus, în această teză mai prezentăm o nouă metodologie care vizează o anumită problemă din bioinformatică (problema ordonării temporale) și un tip specific de date biologice. Această abordare a fost dezvoltată în colaborare cu grupul de cercetare BIMIB de la Universitatea Milano-Bicocca, Italia. De asemenea, mai prezentăm o nouă interfață de programare pentru rezolvarea problemelor de optimizare folosind tehnici de învățare prin întărire.

Teza este structurată în patru capitole, după cum urmează.

Primul capitol, **Probleme din Bioinformatică. Context General**, descrie pe scurt domeniul bioinformaticii și prezintă câteva dintre cele mai dificile probleme din acest domeniu. Capitolul începe printr-o scurtă introducere în biologia moleculară. În continuare, se prezintă câteva tehnici fundamentale ale instruirii automate și ale inteligenței computaționale care au fost aplicate cu succes pentru rezolvarea unor probleme din bioinformatică. Apoi descriem mai în detaliu cele patru probleme abordate în această teză: *predicția regiunilor promotor*, *asamblarea fragmentelor ADN (acid dezoxiribonucleic)*, *predicția structurii terțiare a proteinelor* și *ordonarea temporală a unor date biologice*.

Capitolele 2, 3 și 4 conțin contribuțiile noastre originale în direcția propunerii de noi modele ML pentru rezolvarea unor probleme complexe din bioinformatică.

Capitolul 2, **O Nouă Abordare bazată pe Reguli de Asociere Relaționale pentru Predicția Regiunilor Promotor**, prezintă un nou model de clasificare bazat pe reguli de asociere relaționale, pe care îl propunem pentru identificarea regiunilor promotor în ADN. Începem prin a descrie regulile de asociere relaționale și prezentăm un algoritm pentru identificarea regulilor de asociere ordinale

[CSTM06]. Apoi descriem tehnica originală pe care o propunem, bazată pe extragerea unor reguli de asociere relaționale și pe învățarea supervizată, pentru recunoașterea regiunilor promotor. Prezentăm algoritmul de clasificare și două extensii ale acestuia. Cei trei algoritmi sunt evaluați experimental, iar rezultatele obținute sunt analizate. Clasificatorul principal este comparat cu alți clasificatori din literatură, precum și cu cei doi algoritmi reprezentând extensiile sale. Concluziile capitolului și posibilele cercetări ulterioare sunt conturate în ultima parte a acestuia.

Capitolul 3, **Noi Abordări bazate pe Învățarea prin Întărire pentru Probleme din Bioinformatică**, prezintă trei modele noi, dintre care unul distribuit, bazate pe învățarea prin întărire, folosite pentru a oferi soluții la două probleme importante din bioinformatică. Începem capitolul prin a prezenta câteva noțiuni teoretice de bază despre învățarea prin întărire și continuăm prin a descrie trei noi modele generale bazate pe acest tip de învățare, pentru o anumită clasă de probleme de optimizare combinatorială. Prezentăm, de asemenea, un nou mecanism inteligent de selecție a acțiunilor utilizat în procesul de învățare prin întărire. Aceste modele sunt particularizate, aplicate și evaluate experimental pe două probleme din bioinformatică: asamblarea fragmentelor ADN și predicția structurii terțiare a proteinelor. Abordările propuse sunt analizate și comparate între ele, precum și cu alte abordări existente în literatura de specialitate. În cele din urmă, capitolul cuprinde câteva concluzii și direcții de cercetare care vor fi investigate în continuare.

Capitolul 4, **Noi Abordări pentru Ordonarea Temporală a Datelor Biologice**, prezintă două abordări diferite pe care le-am propus pentru problema ordonării temporale a unor date biologice. Acest capitol prezintă, de asemenea, o nouă interfață de programare pentru rezolvarea problemelor de optimizare folosind tehnici de învățare prin întărire, care este aplicată pentru a găsi soluții la problema menționată mai sus. Prima abordare, dezvoltată în colaborare cu o echipă de cercetare în timpul stagiului meu de cercetare de la Universitatea Milano-Bicocca, introduce o nouă metodologie elaborată pentru abordarea problemei ordonării temporale. Prezentăm o evaluare experimentală a algoritmului care implementează această metodologie pe un studiu de caz care se referă la date asociate cancerului colorectal, precum și o analiză a rezultatelor. Apoi, arătăm cum unul dintre modelele bazate pe învățarea prin întărire introduse în capitolul 3 este adaptat și modificat pentru a aborda problema ordonării temporale. Abordarea este evaluată experimental pe mai multe seturi de date de expresie genică reale. Rezultatele obținute sunt analizate și comparate cu alte abordări existente în literatura de specialitate. Mai multe variante ale acestei abordări originale sunt propuse și rezultatele sunt comparate între ele. Ultimul subcapitol introduce un nou framework de programare (software framework) care folosește învățarea prin întărire și arată modul în care acesta poate fi utilizat pentru a dezvolta o aplicație pentru problema ordonării temporale. În final, prezentăm concluziile capitolului, precum și alte direcții de cercetare care vor fi investigate în continuare.

Contribuțiile originale introduse în această teză sunt conținute în Capitolele 2, 3 și 4, fiind detaliate mai jos:

- Un model bazat pe învățarea supervizată și pe reguli de asociere relaționale pentru predicția regiunilor promotor în ADN (Subcapitolul 2.2) [CBC12].
- Algoritmi de învățare supervizată bazați pe reguli de asociere relaționale pentru predicția regiunilor promotor în ADN (Secțiunea 2.2.4 și Subsecțiunile 2.2.4.2, 2.2.4.3) [CBC12, Boc12c, Boc12b].
- Evaluări experimantale ale acestor algoritmi folosind un studiu de caz, o comparație a abordării propuse cu alte abordări similare din literatură (Secțiunea 2.3.2) [CBC12] și comparații între algoritmii propuși (Subsecțiunea 2.3.2.2) [Boc12c, Boc12b].
- Trei noi modele generale bazate pe învățarea prin întărire, *modelul determinării unui drum*, *modelul permutărilor* și un *model distribuit* pentru un tip specific de probleme de optimizare combinatorială (Secțiunile 3.2.1, 3.2.2) [CBC13, CCB13, Boc12a, BCC11a, BCC11b, CBC11c, CBC11b].
- Un nou mecanism inteligent de selecție a acțiunilor utilizat în procesul de învățare prin întărire, definit pentru a ghida un agent care învață înspre soluții performante (Secțiunea 3.2.4) [CBC13].
- Particularizarea celor trei modele pentru problema asamblării fragmentelor ADN, precum și evaluări experimentale pe mai multe secvențe ADN (dintre care și una aparținând genomului uman), analiza rezultatelor și comparații cu alte abordări din literatură (Secțiunile 3.3.1, 3.3.2, 3.3.4) [BCC11b, BCC11a].

- O comparație între modelul determinării unui drum și cel al permutărilor, aplicate pe o mică secvență ADN și analiza rezultatelor (Secțiunile 3.3.3, 3.3.4) [CCB13].
- O particularizare a modelului determinării unui drum și a abordării distribuite pentru problema predicției structurii terțiare a proteinelor, precum și evaluările experimentale pe mai multe proteine mici, analiza rezultatelor și comparații cu alte abordări din literatură (Subcapitolul 3.4) [CBC11c, CBC11b, CBC11d, CBC11a].
- O nouă metodologie care adaptează o soluție care a fost propusă anterior pentru date de expresie genică [GBJ08] la un tip diferit de date biologice: alterații cromozomiale ale numărului de copii (Secțiunea 4.1.2) [BCG+12, BCG+13].
- Evaluări experimentale ale metodologiei introduse pe un set de date conținând alterații ale numărului de copii prelevate de la pacienți în diferite stadii ale cancerului de colon și compararea rezultatelor (Secțiunea 4.1.3) [BCG+12].
- O particularizare a modelului determinării unui drum pentru problema ordonării temporale a unor date biologice (Secțiunea 4.2.1) [CBC13] și evaluări experimentale de mai multe seturi de date reale de expresie genică, o nouă măsură de evaluare pentru a cuantifica performanța algoritmului nostru (Secțiunea 4.2.2), precum și analiza rezultatelor și comparații cu alte abordări din literatura de specialitate (Secțiunea 4.2.4) [CBC13].
- Algoritmi bazați pe Q-Learning care implementează modelul determinării unui drum pentru rezolvarea problemei ordonării temporale (Secțiunea 4.2.3) [CBC13, Boc12a] și evaluări experimentale ale acestor algoritmi pe un set de date de expresie genică aparținând unui anumit organism (drojdia), analize și comparații ale rezultatelor obținute (Secțiunile 4.2.3, 4.2.4) [Boc12a].
- Un framework de programare generic și o aplicație dezvoltată folosind acest framework pentru abordarea problemei ordonării temporale (Subcapitolul 4.3) [CCB11a, CCB11b].

Capitolul 1

Probleme din Bioinformatică. Context General

Bioinformatica este un domeniu de cercetare interdisciplinară care încearcă să rezolve probleme din biologie, biochimie sau medicină utilizând metode din matematică, statistică și informatică [HMTA08]. se referă la aplicarea de tehnici computaționale pentru gestionarea, organizarea și mai ales pentru analiza informațiilor biologice. Bioinformatica este o interfață între științe computaționale și biologie. Scopul său principal este de a procesa imensele cantități de informație biologică, în scopul de a elucida modul de funcționare a organismelor vii.

1.1 Concepte Fundamentale ale Biologiei Moleculare

Pentru a înțelege subiectul de studiu al bioinformaticii prezentăm elementele de bază care sunt studiate în biologia moleculară și care sunt utilizate în problemele din bioinformatică. Conceptele descrise în cele ce urmează sunt preluate din lucrarea lui Brazma *et al.* [BPSS01].

Genomul tuturor organismelor vii este codificat în ADN (acid dezoxiribonucleic) și reprezintă totalitatea informațiilor ereditare. ADN-ul poate fi compus dintr-unul sau două lanțuri de molecule. Un lanț ADN este compus din patru tipuri de molecule organice complexe, numite *nucleotide*: adenină (A), guanină (G), citozină (C) și timină (T). Un exemplu de astfel de ADN este: $AGTCCAAGCTT$. Când două lanțuri ADN se leagă între ele acestea formează o structură stabilă, cunoscută sub numele de dublu helix al ADN-ului. ARN-ul, sau acidul ribonucleic, este compus dintr-un singur lanț format din 4 tipuri de nucleotide: adenină, guanină, citozină și uracil (care înlocuiește timina din ADN).

Genele sunt segmente ale materialului ADN, fiind considerate moleculele de bază care conțin informația ereditară. Ele codifică proteine specifice. O genă poate sau nu poate fi exprimată într-o anumită celulă, ceea ce duce sau nu la sinteza unui produs genic (proteină sau ARN). De vreme ce toate celulele din organism conțin aceeași informație genetică (ADN-ul este identic), diferențele în expresia genelor sunt responsabile pentru diferențierea celulelor.

Aminoacizii sunt molecule mici. Ei pot fi integrați în molecule mari (macromolecule), sau pot avea roluri independente. Aminoacizii se leagă într-o anumită ordine pentru a forma o proteină, prin urmare, ei pot fi considerați componentele de bază ale proteinelor. Există 22 de aminoacizi care compun proteinele din corpul uman, aceștia fiind notați cu litere ale alfabetului.

Proteinele sunt molecule foarte importante, compuse din secvențe de aminoacizi, care se pot înlănțui în orice ordine. Secvența de aminoacizi formează structura primară a proteinei, care poate fi reprezentată ca un șir de simboluri reprezentând cei 22 de aminoacizi (o proteină poate fi văzută ca un cuvânt peste alfabetul format din cele 22 de litere reprezentând aminoacizii). De îndată ce sunt sintetizate ca secvențe liniare de aminoacizi, proteinele se pliază în câteva secunde, formând o structură tridimensională stabilă numită starea nativă a proteinelor. Se presupune că informațiile pentru procesul de pliere sunt conținute exclusiv în secvența de aminoacizi. Structura secundară a unei proteine este forma tridimensională generală a segmentelor locale și este apare datorită interacțiunilor locale dintre aminoacizi. Ca rezultat al acestor interacțiuni, anumite părți ale lanțului proteic ajung în contact unele cu altele și datorită forțelor de atracție și repulsie molecula adoptă o structură tridimensională fixă și relativ stabilă - structura terțiară. Această structură este foarte importantă, deoarece definește funcția proteinei.

1.2 Inteligență Computațională și Instruire Automată în Bioinformatică. Provocări și Perspective

Unele dintre cele mai importante probleme în bioinformatică sunt prea complexe pentru a putea fi rezolvate folosind tehnici convenționale. Cantitățile imense de date biologice, faptul că părți semnificative ale acestora nu sunt adnotate, sau faptul că ele conțin mult zgomot sunt obstacole care fac imposibilă rezolvarea acestor probleme cu metode și algoritmi tradiționali. Din aceste motive, metodele oferite de inteligența computațională (computational intelligence - CI) și instruirea automată (machine learning - ML) sunt foarte potrivite pentru astfel de sarcini [PPN09]. Aceste metode oferă și un anume grad de flexibilitate în ceea ce privește datele de intrare, precum și posibilitatea extinderii progresive, în scopul de a ține pasul cu cerințele rezultate din creșterea continuă a cantităților de date biologice.

Există un număr mare de exemple care ilustrează cum sunt aplicate tehnicile CI și ML pentru a rezolva probleme din bioinformatică.

Expresia genică se referă la procesul prin care informațiile genice sunt transformate în produse genice funcționale (ARN sau proteine). Tehnologia modernă microarray este folosită experimental pentru a detecta nivelurile de expresie a mii de gene, supuse unor diferite condiții, în timp. După colectarea datelor, unul dintre primii pași în analiza acestora este clusterizarea (clustering). Diferiți algoritmi din domeniile CI și ML au fost propuși pentru clusterizarea datelor de expresia genică: o rețea neuronală asociativă de clusterizare [YLGW02], o abordare hibridă folosind optimizarea bazată pe comportamentul de grup (particle swarm optimization - PSO) și hărți cu auto-organizare (self organizing maps - SOM) [XDE+03], o metodă k-means fuzzy [AHO03] sau metoda k-vecinului cel mai apropiat (k-nearest neighbor) [NMB+03, OP09]. Un alt tip de analiză a datelor de expresie genică se referă la selectarea genelor exprimate diferențiat. Pentru *selecția genelor* literatura de specialitate oferă diferite metode: rețele neuronale probabilistice cu o metodă de selecție a atributelor [HL03], calcul evolutiv împreună cu rețelele neuronale artificiale (artificial neural networks - ANNs) [Fog05] sau rețele bayesiene, rețele neuronale cu funcții de bază radiale și arbori neuronali (neural trees) [HCP+00].

Structura secundară a unei proteine sau a ARN-ului este forma tridimensională generală a segmentelor locale și apare datorită interacțiunilor locale dintre moleculele componente. Problema determinării acestei structuri poate fi modelată ca o problemă standard de clasificare. În literatura de specialitate există mai multe abordări de clasificare pentru predicția structurilor secundare: metode care folosesc ANNs [NT88, Ste93], k-cel mai apropiat vecin și varianta fuzzy [GP08, BS06], mașini cu suport vectorial (support vector machines - SVMs) [LHJYFHB04] și modele Markov ascunse (HMMs) [MGR06, YV04].

Clasificarea proteinelor se referă la caracterizarea proteinelor noi utilizând secvențele lor și detectarea unor relații între secvențe apărute ca urmare a evoluției. Literatura de specialitate oferă două tipuri de metode pentru rezolvarea acestei probleme de clasificare: generative și discriminative. Printre metodele propuse se numără un HMM generativ pentru o familie de proteine [JDH99], o tehnică SVM [LEN02] sau o metodologie pentru construirea unui clasificator bazat pe rețele neuronale [WLD03].

Alinierea secvențelor joacă un rol important în analiza moleculară a secvențelor biologice. Aceasta se referă la procesul de aranjare a secvențelor primare de ADN, ARN sau proteine pentru a identifica regiunile similare care ar putea fi o consecință a relațiilor funcționale, structurale sau evolutive între secvențe. Pentru această problemă, mai multe metode au fost propuse: algoritmi genetici [HYYY02, CL05], o metodă de optimizare bazată pe furnici (ant colony optimization - ACO) [CPCC06], un HMM [RK03] sau o metodă bazată pe logica fuzzy [NVNM07].

Alte probleme majore din bioinformatică, care vor fi abordate în această teză folosind diferite modele ML pe care le introducem, vor fi prezentate în detaliu în următoarele subcapitole. Aceste probleme sunt: *predicția regiunilor promotor în ADN*, *asamblarea fragmentelor ADN*, *predicția structurii terțiare a proteinelor* și *ordonarea temporală a datelor biologice*.

1.3 Predicția Regiunilor Promotor în ADN

Promotorii sunt regiuni ale secvenței ADN care semnalează începutul unei gene în timpul procesului de transcriere, un prim proces implicat în sinteza proteinelor. *Problema identificării regiunilor*

promotor este de o importanță majoră în bioinformatică, din două motive. În primul rând, identificarea promotorilor este un pas semnificativ în procesul detecției genelor. În al doilea rând, promotorii sunt esențiali în reglarea expresiei genelor. Deoarece nu sunt cunoscute condițiile în care o regiune din ADN se comportă ca un promotor, metodele ML sunt potrivite pentru a aborda această problemă. Aceste metode pot învăța descrieri utile ale unor concepte având ca intrări doar niște exemple - secvențe ADN, care conțin anumite tipare de bază, dar totuși încă necunoscute [TYA08].

În contextul învățării supervizate, identificarea promotorilor poate fi enunțată astfel [CS94]: fiind date două seturi de secvențe de ADN de lungime fixă, una conținând secvențe cu regiuni promotor, iar cealaltă conținând secvențe fără prezența acestui semnal, să se genereze un clasificator capabil să prezică dacă o secvență ADN de lungime fixă conține sau nu o regiune promotor.

Mai multe metode ale instruirii automate au fost aplicate în scopul de a recunoaște regiuni promotor în ADN: un sistem de învățare hibrid numit KBANN (Knowledge-Based Artificial Neural Networks), care combină învățarea bazată pe explicații cu cea empirică [TSN90], alte abordări folosind rețele neuronale [PE95], o metodă de tip *grey relational analysis* [TPAG11], abordări SVM [KP04]. După cunoștințele noastre, învățarea bazată pe reguli de asociere nu a fost utilizată în literatura de specialitate pentru a identifica promotori.

1.4 Problema Asamblării Fragmentelor ADN

Stabilirea ordinii nucleotidelor, sau procesul de secvențiere a ADN-ului, este de mare importanță în cercetarea din biologie, medicină, biotehnologie și biologie criminalistică. Pentru determinarea secvenței nucleotidelor în lanțuri ADN mari, acestea sunt mai întâi întretăiate în porțiuni mai mici. *Problema asamblării fragmentelor ADN* constă în reconstrucția secvenței moleculei originale, pornind de la secvențele fragmentate. Această problemă este NP-completă [Pev00], prin urmare soluțiile exacte sunt foarte dificil de obținut. Mai mult, diferite erori de secvențiere care pot afecta fragmentele obținute ridică noi obstacole în rezolvarea problemei.

Ilustrăm în continuare procesul de asamblare, folosind un exemplu simplu, preluat din [Kos07]. Presupunem că pentru secvența ADN $TTACCGTGC$, avem următorul set de fragmente: $F_1 = ACCGT$, $F_2 = CGTGC$, $F_3 = TTAC$ și $F_4 = TACCGT$. În primul rând trebuie să determinăm suprapunerea dintre fiecare fragment și celelalte trei. Aceasta se face de obicei cu ajutorul unui algoritm de aliniere și o măsură de similaritate. Apoi trebuie să găsim ordinea corectă a acestor fragmente, bazându-ne pe similaritățile calculate. Ordinea este: $F_3F_4F_1F_2$.

F_3	$\rightarrow$	T	T	A	C	$-$	$-$	$-$	$-$	$-$
F_4	$\rightarrow$	$-$	T	A	C	C	G	T	$-$	$-$
F_1	$\rightarrow$	$-$	$-$	A	C	C	G	T	$-$	$-$
F_2	$\rightarrow$	$-$	$-$	$-$	$-$	C	G	T	G	C
		T	T	A	C	C	G	T	G	C

În literatura de specialitate au fost dezvoltate diverse metode euristice pentru a aborda problema asamblării fragmentelor ADN: algoritmi genetici (îmbunătățiți) [KC06, PFB95, LATK06], algoritmi euristici de clusterizare [LK04], algoritmi ACO [MC03, WCT06], precum și abordări de învățare supervizată, cum ar fi rețelele neuronale recurente [AAdFG99].

1.5 Predicția Structurii Terțiare a Proteinelor

Predicția structurii proteinelor este una dintre marile provocări ale bioinformaticii, fiind o direcție importantă de cercetare, datorită numeroaselor sale aplicații în medicină (proiectarea de noi medicamente, predicția bolilor) și în ingineria genetică (modelare celulară, modificarea și îmbunătățirea funcțiilor anumitor proteine). Imediat după ce sunt sintetizate ca secvențe liniare de aminoacizi, proteinele se pliază adoptând o structură tridimensională fixă, numită stare nativă, care determină și funcțiile proteinelor.

O clasă importantă de modele conceptuale care abstractizează această problemă include modelele bazate pe matrici - compuse dintr-o matrice care descrie pozițiile posibile ale aminoacizilor în spațiu și o funcție de energie a proteinelor, care depinde de aceste poziții. Scopul este de a găsi minimul

global al acestei funcții, deoarece se presupune că o proteină în starea sa nativă are o energie minimă și că procesul de pliere implică minimizarea acestei energii [Anf73]. Unul dintre modelele cele mai des folosite, bazat pe matrici bidimensionale, este modelul hidrofob-polar (Hydrophobic-Polar - HP) definit de către Dill [Dil85]. Acest model se bazează pe observația că forțele hidrofobe sunt factori foarte importanți în procesul de pliere a proteinelor, direcționându-le înspre structura lor tridimensională nativă. În modelul HP, fiecare aminoacid este considerat fie hidrofob (respinge apa) fie polar (hidrofil - absoarbe apa) și funcția de energie este definită astfel încât să surprindă interacțiunile dintre aminoacizii hidrofobi vecini în matrice.

Predicția structurii terțiare a proteinelor modelată folosind modelul HP este o problemă NP-completă [BL98], prin urmare diverse metode euristice și de aproximare au fost propuse pentru a o aborda. Dintre acestea, se remarcă algoritmi ACO [SH05, TMM08], algoritmi genetici [UM93], algoritmi hibrizi - care combină algoritmi genetici și căutarea tabu [ZWL+09], metode evolutive folosind operatori genetici de căutare locală [Chi10], precum și tehnici de învățare supervizată, cum ar fi SMVs și ANNs [DD01].

1.6 Ordonarea Temporală a Datelor Biologice

Analiza și ordonarea temporală sunt probleme importante în domeniul bioinformaticii și al biologiei computaționale, deoarece analiza temporală a evenimentelor care descriu un anumit proces biologic ar putea oferi o perspectivă semnificativă asupra dezvoltării și progresului acestuia. În multe situații, ordonarea în timp a unor instanțe din seturi de date oferă informații mai semnificative decât separarea instanțelor în diferite clase. Prin urmare, problema generală a ordonării temporalăe este comparabilă, ca importanță, cu problema clasificării [CSS99].

Problema ordonării temporale poate fi exprimată în diverse forme. Una dintre definiții se referă la determinarea și descrierea succesiunii de evenimente ce caracterizează un proces biologic. În cazul în care acest proces este cancerul, scopul este identificarea ordinii temporale a modificărilor genetice și a alterațiilor căilor biologice care apar în timpul genezei și evoluției acestei boli. Se cunoaște faptul că cele mai multe tumori se dezvoltă din cauza mutațiilor care apar în anumite gene-cheie (oncogene sau gene supresoare de tumori) [Fra07]. De aceea, pentru a înțelege mai bine evoluția cancerului, este necesar studiul ordinii apariției acestor mutații. În literatura de specialitate există mai multe lucrări care abordează problema ordonării temporale, așa cum a fost ea definită mai sus. Unele dintre aceste metode sunt probabilistice [HHL06], bazate pe rețele bayesiene [GBHB09, GEL+11], iar altele se concentrează asupra construirii de modele reprezentând arbori care codifică evenimente genetice posibile [DJK+99, DJK+00, BJK+05a, BJK+05b, PSBM09], sau de modele matematice [ACB+10, CBL+12] pentru identificarea ordinii mutațiilor genetice care apar în procesul de dezvoltare a cancerului.

O a doua direcţie de cercetare se axează pe o definiție diferită a problemei ordonării temporale. Aceasta se referă la construirea unei colecții de date biologice multidimensionale care reflectă o evoluție temporală corectă a unui proces biologic. În literatură există în principal două lucrări care tratează problema după această definiție, ambele folosind date de expresie genică. Primul studiu [GBJ08] folosește problema comis-voiajorului pentru a obține o ordonare temporală corectă a unor probe biologice reprezentate prin date de expresie genică, iar soluția prezentată în cel de-al doilea [MLK03] se bazează pe arbori de acoperire minimi (minimum spanning trees) și pe arbori PQ.

Capitolul 2

O Nouă Abordare bazată pe Reguli de Asociere Relaționale pentru Predicția Regiunilor Promotor

În acest capitol abordăm problema predicției regiunilor promotor și propunem un model de clasificare bazat pe extragerea inteligentă a regulilor de asociere relaționale (RAR) pentru identificarea promotorilor în ADN.

Abordările prezentate în acest capitol sunt lucrări originale publicate în [CBC12, Boc12c, Boc12b].

Regulile de asociere relaționale [SCC06] au fost introduse pentru a putea surprinde diferite tipuri de relații între atributele unor instanțe din seturi de date. Pornind de la ideea de a descoperi RAR în cadrul unui set de date, propunem un model de clasificare pentru problema predicției promotorilor. Intuitiv, am presupus că în problema de a decide dacă o secvență ADN conține sau nu regiuni promotor, relațiile dintre nucleotidele care formează secvența ADN pot fi relevante. Aceste relații ar putea exprima diferite informații conținute în secvența ADN care ar putea influența în mod semnificativ clasificarea. Astfel, ne focalizăm pe dezvoltarea unui model computațional de instruire automată, care va fi suficient de puternic pentru a surprinde aspecte relevante pentru a face distincție între secvențe ADN care conțin sau nu o regiune promotor. În construirea clasificatorului încercăm să caracterizăm secvențele ADN prin proprietățile lor biologice și chimice și să exploatăm tehnici de extragere de cunoștințe pentru a descoperi modele ascunse în date.

2.1 Reguli de Asociere Relaționale. Context General

Învățarea bazată pe reguli de asociere este o metodă des folosită de a descoperi corelațiile relevante care există între datele din seturi mari de date. Având un set de atribute, numite elemente, și un set de tranzacții, care compun setul de date de intrare, unde fiecare tranzacție conține un subset de elemente, o regulă de asociere este o implicație a două seturi de elemente. Există mai multe tipuri de reguli de asociere care pot fi descoperite în date (binare, cantitative, fuzzy), unul dintre acestea fiind reguli de asociere ordinale, care au fost introduse în [MML01]. Cu toate acestea, există situații reale în care regulile de asociere ordinale nu sunt suficient de puternice pentru a descrie diferite regularități între date. Prin urmare, regulile de asociere ordinale au fost extinse în [SCC06] înspre reguli de asociere relaționale, în scopul de a putea surprinde diferite tipuri de relații între atributele unor instanțe.

Similar cu [SCC06], notăm cu $R = \{r_1, r_2, \ldots, r_n\}$ un set de *instanțe* (entități sau înregistrări), unde fiecare instanță este caracterizată de o listă de m atribute, $(a_1, \ldots, a_m)$. Notăm valoarea atributului a_i pentru instanța r_j cu $\Phi(r_j, a_i)$, $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$. Fiecare atribut poate avea valori aparținând unui domeniu D_i, care conține și o valoare vidă. Cu M notăm mulțimea tuturor relațiilor posibile care pot fi definite pe mulțimea $D_i \times D_j$. În [SCC06] o *regulă de asociere relațională* este definită ca $(a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}) \Rightarrow (a_{i_1}\ \mu_1\ a_{i_2}\ \mu_2\ a_{i_3} \ldots \mu_{\ell-1}\ a_{i_\ell})$, unde $\{a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}\} \subseteq \mathcal{A} = \{a_1, \ldots, a_m\}$, $a_{i_j} \neq a_{i_k}$, $j, k = 1..\ell$, $j \neq k$ și $\mu_i \in M$ este o relație din $D_{i_j} \times D_{i_{j+1}}$, D_{i_j} fiind domeniul atributului a_{i_j}. *Suportul* unei reguli este procentul de instanțe (din cele n) în care $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ apar împreună. Similar cu [SCC06], notăm mulțimea instanțelor în care $a_{i_1}, a_{i_2}, a_{i_3}, \ldots, a_{i_\ell}$ apar împreună cu $R' \subseteq R$ și $\Phi(r_j, a_{i_1})\ \mu_1\ \Phi(r_j, a_{i_2})\ \mu_2\ \Phi(r_j, a_{i_3})\ \ldots\ \mu_{\ell-1}\ \Phi(r_j, a_{i_\ell})$ este adevărat

pentru fiecare instanță r_j din R'. Atunci $c = |R'|/|R|$ se numește *confidența* regulii.

Lungimea unei reguli de asociere relaționale este numărul de atribute din regulă. Această lungime poate fi egală cel mult cu numărul m de atribute care descriu instanțele.

În cele mai multe cazuri se caută RAR *interesante* dintr-un set de date, adică reguli relaționale care sunt adevărate pentru un număr minim de instanțe, având suportul cel puțin s_{min}, și confidența cel puțin c_{min} (s_{min} și c_{min} sunt valori prag furnizate de utilizator).

2.1.1 Algoritmul *DOAR*

În [CSTM06] a fost introdus un algoritm de tip A-Priori [AS94], numit *DOAR - Discovery of Ordinal Association Rules*, pentru identificarea regulilor de asociere ordinale relevante într-un set de date. Acesta găsește într-un mod eficient toate regulile de asociere ordinale (RAR în care relațiile sunt relații de ordine) de orice lungime, care sunt adevărate într-un set de date.

Algoritmul *DOAR* identifică reguli de asociere ordinale folosind un proces iterativ care constă în generarea unor reguli candidat și verificarea dacă acestea îndeplinesc condițiile minime de suport și confidență. *DOAR* efectuează mai multe tranziții peste setul de date. În prima tranziție se calculează suportul și confidența regulilor de lungime 2 și se determină care dintre ele sunt interesante. Fiecare tranziție ulterioară începe cu un set de reguli de pornire interesante de lungime $(k-1)$ $(k \geq 3)$, identificate în tranziția anterioară, folosit pentru a genera noi posibile reguli interesante de lungime k, denumite reguli candidat. Generarea candidaților este un element cheie al algoritmului *DOAR*. Datele sunt din nou scanate pentru a calcula suportul și confidența candidaților. La finalul acestei etape, algoritmul păstrează doar acele reguli candidat care sunt interesante și acestea vor fi folosite în următoarea iterație. Procesul se oprește atunci nu se mai găsesc noi reguli interesante în ultima iterație. Mai multe despre algoritmul *DOAR* și validarea teoretică a acestuia se pot vedea în [CSTM06].

În abordarea noastră, algoritmul *DOAR* este extins înspre un algoritm similar, *DRAR* algoritm (*Discovery of Relational Association Rules*) pentru a găsi RAR interesante, adică reguli de asociere, care pot surprinde diferite tipuri de relații între atributele unor instanțe. Implementarea actuală găsește toate RAR interesante de orice lungime, precum și toate RAR interesante maximale de orice lungime: în cazul în care o regulă interesantă de o anumită lungime poate fi extinsă cu un atribut și ea rămâne interesantă, doar regula de extinsă este păstrată.

2.2 O Nouă Metodă pentru Predicția Promotorilor

În acest subcapitol propunem o tehnică de învățare supervizată bazată pe RAR pentru predicția promotorilor în ADN, numită *PCRAR* (*Promoter sequences Classifier using Relational Association Rules*). Acest clasificator a fost construit în scopul de face distincție între secvențele ADN care conțin și cele care nu conțin promotori. Clasificatorul *PCRAR* nu se bazează pe mecanisme biologice specifice, avantajului lui constând în capacitatea sa de a învăța în mod automat diferențele dintre secvențe ADN care includ sau nu regiunile promotor, având ca date de intrare doar aceste secvențe, fără alte informații biologice suplimentare.

Avem de a face cu o problemă de clasificare binară. Există două clase posibile: una conține *instanțe pozitive* - secvențe ADN cu regiuni promotor, iar cealaltă conține *instanțe negative* - secvențe ADN fără promotori (sau non-promotor).

Ideea principală a abordării noastre este descrisă în cele ce urmează. Într-un scenariu de învățare supervizată pentru identificarea promotorilor se dau două seturi care conțin exemple pozitive și negative. Acestea sunt folosite pentru a construi clasificatorul. În faza de antrenare utilizăm algoritmul *DRAR*. Chiar dacă *DRAR* poate fi folosit pentru a descoperi RAR de orice lungime într-un set de date, în primă instanță l-am folosit pentru a descoperi doar RAR binare, adică reguli de lungime 2. Am detectat în datele de intrare toate RAR binare interesante (reguli între două atribute), ținând cont de valorile prag furnizate pentru suport și confidență. După ce faza de antrenare este încheiată, atunci când o nouă instanță (secvența ADN) trebuie clasificată (ca fiind pozitivă sau negativă), raționăm după cum urmează. Considerând regulile binare descoperite în timpul antrenării în seturile de instanțe *pozitive* și *negative*, vom calcula un scor pentru noua instanță, scor definit astfel încât să varieze între 0 și 1, care indică gradul în care instanța poate fi considerată pozitivă. Dacă acest scor este mai mare sau egal cu 0.5, atunci instanța va fi clasificată ca fiind *pozitivă*, altfel va fi considerată *negativă*.

Considerăm o secvență ADN (instanță) ca fiind un lanț n-dimensional $S = (s_1, s_2, \ldots, s_n)$ conținând cele patru litere A, T, G și C, care reprezintă nucleotidele ce compun ADN-ul (Subcapitolul 1.1). Prin urmare, setul de atribute care descriu instanțele este reprezentat de o listă n-dimensională, în care al

i-lea atribut corespunde nucleotidei i din secvența ADN, $i \in \{1, \ldots, n\}$. Prin urmare, fiecare atribut are 4 valori posibile: caracterele A, T, G și C.

Pentru a clasifica o secvență ADN în una din cele două clase (cu sau fără regiuni promotor), vor fi efectuați următorii pași:

1. Definirea relațiilor.
2. Preprocesarea datelor.
3. Antrenarea/construirea clasificatorului.
4. Testarea/clasificarea.

2.2.1 Definirea Relațiilor

În această etapă definim relațiile între valorile atributelor, relații care vor fi utilizate în procesul de extragere a RAR. Mai exact, ne concentrăm pe identificarea relațiilor dintre două nucleotide dintr-o secvență ADN (A, T, G sau C), care pot fi relevante pentru a decide dacă secvența conține sau nu o regiune promotor.

Primul pas se referă la căutarea unor proprietăți chimice și fizice care pot caracteriza fiecare nucleotidă. Am extras din PubChem [BWTB08] 5 proprietăți măsurabile: masa molară, densitatea, aria suprafeței topologice polare, numărul de atomi grei și complexitatea. La acestea se adaugă compoziția de bază, una dintre caracteristicile fundamentale ale unei secvențe ADN care se referă la procentul din fiecare dintre cele patru nucleotide pe un singur lanț ADN. Prin urmare, asociem fiecărei nucleotide (valoare a unui atribut) o listă de șase coduri numerice, reprezentând valorile celor șase proprietăți enumerate mai sus.

Al doilea pas constă în identificarea codurilor care sunt relevante pentru sarcina de clasificare (din cele 6 coduri). În această direcție efectuăm o analiză statistică a datelor de intrare pentru a identifica acele coduri care implică o corelație puternică între atribute și valorile țintă. Pentru a determina dependențele dintre atribute și valorile țintă, utilizăm coeficientul de corelație Spearman [Spe04]. Pentru a defini relațiile dintre atribute și pentru a extrage RAR interesante vor fi luate în considerare numai codurile care oferă cele mai mari corelații cu valorile țintă.

2.2.2 Preprocesarea Datelor

După identificarea unui set cu tipurile de coduri care sunt relevante în definirea modelului bazat pe RAR (Secțiunea 2.2.1), o altă analiză statistică este aplicată pe datele de intrare în scopul de a reduce dimensionalitatea acestora, prin eliminarea atributelor care nu influențează valoarea țintă. Pentru a determina dependențele dintre atribute și valorile țintă folosim tot coeficientul de corelație Spearman. Scopul acestui pas este de a elimina acele atribute (nucleotide la o anumită poziție, pentru exemplul nostru), care nu au o influență semnificativă asupra valorii țintă, adică sunt foarte slab corelate cu aceasta (valoarea absolută a corelației este sub un prag foarte mic, pozitiv).

2.2.3 Construirea Clasificatorului

În această etapă RAR interesante sunt descoperite în seturile de date de antrenare. Modelul de clasificare constând din RAR interesante descoperite în seturile de date de antrenare va fi utilizat în continuare pentru a clasifica toate instanțele de test.

Antrenarea constă în aplicarea algoritmului *DRAR* pentru a determina două seturi de RAR cu *suport* și *încredere* minime: un set de RAR pozitive, notat cu RAR_+, folosind datele de intrare conținând instanțe pozitive și un set de RAR negative, notat cu RAR_-, obținut folosind instanțele care nu conțin regiuni promotor.

2.2.4 Clasificarea

După încheierea fazei de antrenare și construirea clasificatorului $PCRAR$, atunci când o nouă secvență ADN S trebuie să fie clasificată, se va calcula un scor $P_+(S)$ (notat mai simplu P_+), care va specifica dacă S conține o regiune promotor și $P_-(S)$ (notat mai simplu P_-), un alt scor, care indică faptul că S nu conține o regiune promotor. Propunem două tehnici de calcul pentru aceste scoruri. Prima depinde exclusiv de numărul total de reguli de asociere generate (pozitive și negative) și de numărul de reguli pe care noua secvență le verifică sau nu, fără a lua în considerare confidențele

regulilor. A doua se bazează pe confidențele RAR generate. Pornim de la ideea că RAR relevante vor induce scoruri precise. De aceea obiectivul nostru principal este identificarea relațiilor corecte și semnificative în datele de intrare.

2.2.4.1 Calculul scorului utilizând numărul de reguli

Scorul P_+ calculat pentru a determina dacă o instanță este *pozitivă* este:

$$P_+ = \frac{n_+ + m_-}{|RAR_+| + |RAR_-|} \tag{2.1}$$

unde n_+ este numărul de RAR pozitive pe care le verifică instanța și m_- indică numărul de reguli negative pe care *nu* le verifică instanța dată.

Dacă $P_+ \geq 0.5$ atunci instanța S va fi clasificată ca fiind *pozitivă*, altfel ea va fi considerată *negativă*.

În mod similar poate fi calculat și scorul P_- pentru a determina dacă o instanță ar trebui clasificată ca fiind *negativă*. Însă acest pas poate fi omis, deoarece se poate ușor demonstra că rezultatele furnizate de clasificatorul *PCRAR* sunt consistente logic, adică pentru o anumită instanță S, $P_+ + P_- = 1$.

Algoritmul care utilizează modul de calcul al scorului descris mai sus este *PCRAR*.

2.2.4.2 Calculul scorului utilizând confidențele regulilor

O a doua metodă de calcul al scorurilor se bazează pe confidențele regulilor verificate/neverificate. Pentru o instanță ADN S, scorul P_+ va fi calculat astfel:

$$P_+ = \frac{1}{2}(\frac{s_+(S)}{s_+} + \frac{sn_-(S)}{s_-}) \tag{2.2}$$

unde $s_+(S)$ este suma confidențelor regulilor din setul RAR_+ pe care le verifică S, $sn_-(S)$ este suma confidențelor regulilor din setul RAR_- pe care *nu* le verifică S, iar s_+ and s_- sunt sumele totale ale confidențelor tuturor regulilor din RAR_+ și RAR_-, respectiv.

Ca și în cazul precedent, dacă $P_+ \geq 0.5$ atunci instanța S va fi clasificată ca fiind *pozitivă*, altfel ea va fi considerată *negativă*. Scorul P_- poate fi calculat în mod analog, dar acest pas poate fi omis deoarece se poate demonstra că pentru o instanță $P_+ + P_- = 1$.

Algoritmul care utilizează calculul scorului descris mai sus se numește *BRSC* (Binary Rules, with Score computation based on the Confidence of the rules).

Remarca 1 *În cazul lui PCRAR este suficientă doar generarea regulilor relaționale binare interesante, deoarece o regulă cu o lungime mai mare decât doi este verificată dacă sub-regulile sale binare sunt verificate. Acest lucru reduce semnificativ timpul de antrenare a clasificatorului. Dacă în calculul scorului este inclusă și confidența, putem considera fie doar reguli binare, fie reguli de orice lungime. BRSC utilizează reguli binare. Un alt algoritm, numit* KRSC *(*K*-length Rules Generation), care utilizează regulile de orice lungime, este prezentat în cele ce urmează.*

2.2.4.3 Generarea Regulilor de lungime k

Ca o a doua extensie a lui *PCRAR*, propunem generarea regulilor de orice lungime k, lungimea maximă fiind numărul de atribute ale unei instanțe (pentru o instanță S, notăm cu $|S|$ numărul de atribute). O regulă de lungime k este verificată de către o instanță dacă toate cele $k-1$ sub-reguli binare ale sale sunt verificate. După generarea tuturor regulilor de lungime k ($k \in \{2, 3, \ldots, |S|\}$), atunci când o nouă secvență ADN trebuie să fie clasificată, vom calcula scorurile pentru această secvență folosind modul de calcul descris în Subsecțiunea 2.2.4.2.

2.3 Evaluare Experimentală

În acest subcapitol dorim să evaluăm experimental abordarea noastră propusă pentru a detecta regiunile promotor în ADN, folosind RAR. Oferim, de asemenea, o comparație cu alte abordări similare existente.

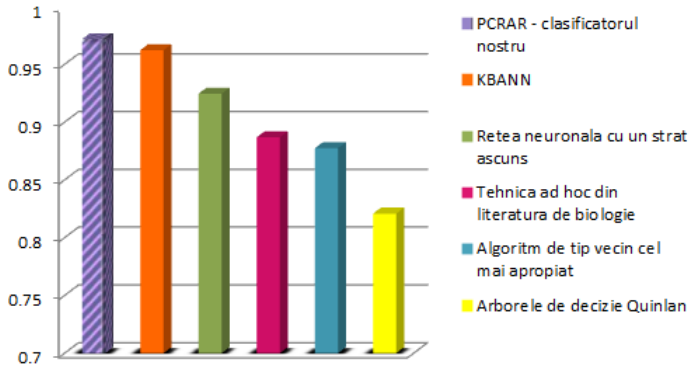


Figura 2.1: Rezultate comparative. Clasificatorul nostru *PCRAR* surclasează alți clasificatori din literatură [TSN90] care au mai fost aplicați pentru a identifica regiuni promotor.

2.3.1 Setul de Date

Setul de date pe care l-am folosit pentru a testa eficiența clasificatorilor se numește "E. coli promoter gene sequences (DNA) with associated imperfect domain theory". Acesta a fost preluat din UCI repository [FA10] și conține 106 secvențe ADN, cu și fără regiuni promotor, fiecare având o lungime de 57 de nucleotide. Jumătate dintre secvențe reprezintă cazuri pozitive și jumătate sunt negative. Am ales acest set de date pentru evaluarea experimentală din două motive: în primul rând, pentru că acesta este public și în al doilea rând, pentru în literatură există și alți clasificatori care au fost validați folosind acest set de date, ceea ce ne permite să efectuăm comparații între modelul nostru și cele existente.

2.3.2 Rezultate și Discuție

Metodologia și algoritmii descriși în Subcapitolul 2.2 sunt aplicați pe setul de date considerat și rezultatele sunt prezentate în cele ce urmează.

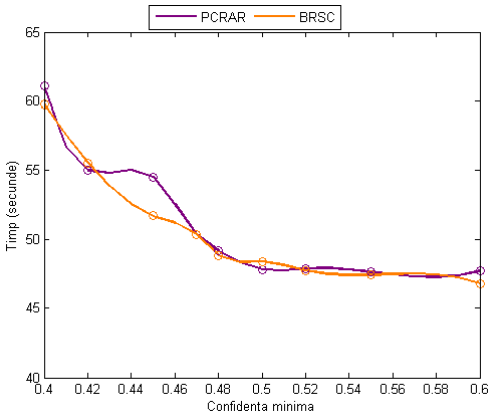
2.3.2.1 *PCRAR* - Rezultate și analiză

Am executat algoritmul de clasificare *PCRAR* prezentat în subcapitolul 2.2 folosind pragul minim de suport $s_{min} = 0.9$ și diferite valori pentru pragul minim de confidență c_{min} : $\{0.6, 0.55, 0.52, 0.5, 0.48, 0.47, 0.45, 0.42, 0.4, 0.38, 0.36\}$. Pentru evaluarea performanței abordării noastre, am folosit setul de date descris în Secțiunea 2.3.1 și am aplicat o metodă de validare de tip cross-validation. Cel mai bun rezultat a fost obținut pentru valoarea pragului de confidență **0.4**, caz în care s-a raportat o eroare de clasificare de 0.018867 (**2/106**) în urma validării, care a fost finalizată în **64.430** secunde. Relativ la alți clasificatori din literatura de specialitate propuși pentru recunoașterea regiunilor promotor [TSN90], *PCRAR* depășește cel mai bun clasificator existent. Această comparație este ilustrată în Figura 2.1 [CBC12]. În această figură, fâșia hașurată indică performanța clasificatorului nostru *PCRAR*. Un alt avantaj al abordării noastre în comparație cu abordările existente este faptul că etapa de antrenare a lui *PCRAR* se execută foarte rapid, deoarece este suficientă numai generarea regulilor de asociere relaționale binare.

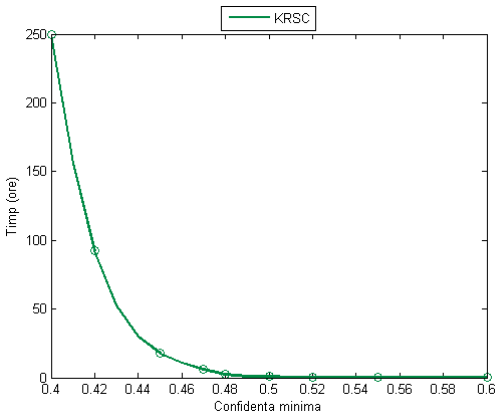
2.3.2.2 *PCRAR* și extensiile sale - Rezultate comparative

Cei doi algoritmi prezentați în Subsecțiunile 2.2.4.2 și 2.2.4.3 au fost testați pe același set de date, iar rezultatele au fost comparate cu cele raportate de către *PCRAR*.

În ceea ce privește numărul de RAR generate (pentru întregul set de date pozitive și negative), ambii algoritmi *PCRAR* și *BRSC* identifică reguli binare, care depind doar de setul de date de intrare, prin urmare, acești algoritmi vor genera întotdeauna același număr de reguli, pentru o anumită valoare a pragului de confidență. În schimb, numărul de reguli generate de *KRSC* va fi semnificativ mai mare, pentru că acest algoritm determină regulile de orice lungime, pornind de la un set de reguli binare



(a) Timpi de execuție comparativi: *PCRAR* și *BRSC* (secunde). Se poate observa că timpii de execuție ai celor doi algoritmi sunt apropiați, diferența maximă fiind pentru pragul de confidență 0.45.



(b) Timpul de execuție al lui *KRSC*. Se poate observa că funcția confidență-timp este exponențială.

Figura 2.2: Timpi de execuție comparativi pentru cei trei algoritmi bazați pe RAR.

generate. Lungimea maximă posibilă pentru o regulă de lungime k este de $k = 57$ (numărul de atribute dintr-o instanță), dar lungimea maximă efectivă a regulilor care au fost generate a fost de $k = 8$.

Cel mai bun rezultat obținut de către *BRSC* este la fel cu cel obținut de *PCRAR*: **104** cazuri clasificate corect, din 106, pentru confidența minimă $c_{min} = 0.4$. *KRSC*, pe de altă parte, s-a dovedit mai puțin performant, cel mai bun rezultat obținut de către acesta fiind **97** de instanțe clasificate corect, din 106 (o eroare de 0.084905), pentru o valoare de 0.6 a confidenței.

Ne vom referi acum la timpii de execuție ai algoritmilor. Aceștia sunt timpii de validare, adică timpii totali în care fiecare clasificator efectuează validarea (incluzând aici și timpul de antrenare). Ambii algoritmi care iau în considerare numai reguli binare au timpi de execuție foarte mici ($\sim$ 2 minute), după cum se observă în Figura 2.2a [Boc12c]. Cel care generează reguli de orice lungime rulează mult mai lent (de la 5 de minute la 250 ore, în funcție de valorile confidenței), după cum se poate observa în Figura 2.2b [Boc12c].

Rezultatele obținute demonstrează că algoritmii care generează și folosesc doar RAR binare obțin rezultate mai bune decât cel care generează reguli de orice lungime, atât în ceea ce privește acuratețea clasificării cât și timpul de execuție. Concluzionăm astfel că, pentru problema considerată, regulile binare sunt suficient de relevante pentru a obține o clasificare precisă a unei secvențe ADN în una din cele două clase, indentificând astfel dacă ea conține sau nu regiuni promotor.

2.4 Concluzii și Cercetări Ulterioare

În acest capitol am introdus un model de clasificare bazat pe descoperirea regulilor de asociere relaționale pentru predicția regiunilor promotor, prezentat în lucrarea originală [CBC12]. Evaluarea experimentală a modelului propus a demonstrat puterea clasificatorului nostru, care îi surclasează pe cei existenți în literatură, pentru setul de date considerat, indicând astfel potențialul propunerii noastre. Am introdus, de asemenea, două extensii pentru modelul de clasificare bazat pe RAR, prezentate în articolele [Boc12c], [Boc12b]. Am evaluat experimental și am comparat cei trei algoritmi.

Performanța modelului de clasificare introdus în acest capitol conduce înspre concluzia că modelele de instruire automată și tehnicile de extragere inteligentă a datelor sunt instrumente importante capabile să recunoască tipare în date biologice, care sunt greu de identificat folosind tehnici convenționale.

Cercetări ulterioare vor fi făcute pentru a identifica și lua în considerare diferite tipuri de relații între nucleotidele unei secvențe ADN. De asemenea, ne propunem să îmbunătățim precizia clasificatorilor bazați pe RAR prin utilizarea unor tehnici de învățare supervizată pentru a identifica valorile cele mai potrivite pentru parametrii utilizați. Vom investiga și posibilitatea hibridizării modelului nostru de clasificare, prin combinarea sa cu alte modele de instruire automată [Mit97].

Capitolul 3

Noi Abordări bazate pe Învățarea prin Întărire pentru Probleme din Bioinformatică

Acest capitol începe prin a introduce principalele aspecte ale învățării prin întărire (reinforcement learning) și apoi continuă prin a prezenta contribuțiile noastre originale, mai specific, două noi modele de învățare și o abordare distribuită bazate pe învățarea prin întărire și utilizate pentru a oferi soluții la două probleme importante în bioinformatică: asamblarea fragmentelor ADN (Subcapitolul 1.4) și predicția structurii terțiare a proteinelor (Subcapitolul 1.5).

Cele două noi modele și abordarea distribuită bazate pe învățarea prin întărire prezentate în acest capitol, precum și aplicarea acestora în domeniul bioinformaticii au fost prezentate în publicațiile originale [CBC13, CCB13, BCC11a, CBC11a, BCC11b, CBC11c, CBC11b, CBC11d].

3.1 Învățarea prin Întărire. Context General

Învățarea prin întărire (reinforcement learning - RL) [SB98] este o abordare a instruirii automate care combină două discipline pentru a rezolva problemele pe care nici una dintre discipline nu le pot aborda în mod individual: *programarea dinamică* și *învățarea supervizată*. În literatura instruirii automate, RL este considerat a fi printre cele mai fiabile tipuri de învățare, deoarece este cea mai apropiată de modul de învățare uman.

RL se preocupă de modul în care un agent autonom, care percepe și acționează în mediul său poate învăța să aleagă acțiunile optime pentru atingerea obiectivelor sale [Mit97]. Într-un scenariu RL, sistemul de învățare selectează acțiuni pe care le efectuează în mediul său și primește *recompense* (sau *întăriri*), sub forma unor valori numerice care reprezintă o evaluare a acțiunilor selectate [PU98]. Agentului îi este dat un obiectiv pe care să îl realizeze și el învață cum să își atingă acest obiectiv prin interacțiuni continue cu mediul său, folosind metoda învățării din greșeli. Agentului nu i se spune ce acțiuni să aleagă, ci el trebuie să descopere care acțiuni conduc către cea mai mare recompensă. Într-o sarcină RL scopul agentului este de a maximiza suma recompenselor primite, pornind de la o stare inițială și până când ajunge la o stare finală.

O problemă RL este caracterizată de patru componente: un *spațiu al stărilor* $\mathcal{S}$, care specifică toate configurațiile posibile ale sistemului; un *spațiu al acțiunilor* $\mathcal{A}$, care conține toate acțiunile disponibile agentului care învață; o *funcție de tranziție* δ, care precizează stocastic rezultatele obținute atunci când se alege orice acțiune, din orice stare; *o funcție a recompenselor*, care definește recompensele ce pot fi primite în urma selecției acțiunilor.

Cele două concepte care stau la baza învățării prin întărire sunt metoda învățării din greșeli (trial and error) și recompensa întârziată (delayed reward) [CK91]. Sarcina agentului este de a învăța o politică de control $\pi : \mathcal{S} \to \mathcal{A}$, care maximizează suma preconizată a recompenselor primite E, în care recompensele ulterioare sunt reduse exponențial, în funcție de cât de departe sunt: E este definit ca fiind $r_0 + \gamma \cdot r_1 + \gamma^2 \cdot r_2 + ...$ ($0 \leq \gamma < 1$ este factorul de actualizare).

Există două modele de bază RL. Agentul poate învăța fie o *funcție de utilitate* (U) pe stări (sau istoricul stărilor) și o utilizează pentru a selecta acțiuni care maximizează utilitatea așteptată a

rezultatelor lor, sau poate învăța o *funcție acțiune-valoare* (Q), care oferă utilitatea pentru perechi de tip (stare, acțiune). Cea din urmă conduce înspre abordări de tip *Q-learning*.

3.1.1 Q-learning - Un Algoritm RL folosind Funcții de Tip Acțiune-Valoare

Unul dintre cei mai utilizați algoritmi RL este *Q-learning*, în care agentul învață o funcție de tip acțiune-valoare. În loc de a avea o funcție între stări și valorile stărilor, Q-learning învață o funcție între perechi de tip (stare, acțiune) și valorile acestor perechi, aceasta fiind funcția Q. În fiecare stare există o Q-valoare asociată fiecărei acțiuni posibile din acea stare. Definiția Q-valorii este utilitatea pe termen lung, sau suma recompenselor (care poti fi și reduse) care se primesc atunci când se alege acțiunea asociată și se urmează o politică dată. O valoare Q optimă este suma recompenselor primite atunci când se alege o acțiune asociată Q-valorii și apoi se continuă utilizând o politică optimă.

Dacă notăm cu $Q(s, a)$ valoarea alegerii acțiunii a din starea s, cu $r(s, a)$ recompensa primită atunci când se alege acțiunea a din starea s și cu s' starea mediului în care se trece în urma selecției acțiunii a din starea s, atunci ecuația cel mai des utilizată pentru actualizarea Q-valorilor la fiecare iterație a algoritmului este următoarea [WD92]:

$$Q(s, a) = (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r(s, a) + \gamma \cdot \max_{a'} Q(s', a')). \tag{3.1}$$

unde γ este factorul de actualizare și $\alpha \in [0, 1]$ este rata de învățare.

A fost demonstrat faptul că algoritmul Q-learning converge înspre o funcție de tip acțiune-valoare optimă și către o politică optimă, dacă toate perechile (stare, acțiune) sunt vizitate de un număr infinit de ori și politica converge către politica *greedy* [PS94].

3.1.2 Politici de Selecție a Acțiunilor

Un aspect important în RL este menținerea unui echilibru între *exploatare* și *explorare* [Thr92]. Agentul trebuie să fie capabil să acumuleze multe recompense, prin alegerea celor mai bune acțiuni, folosindu-și experiența, dar trebuie să și exploreze mediul său, în încercarea de a găsi acțiuni noi (poate nu cele optime), care pot duce ulterior la recompense mai mari. În literatură există mai multe reguli (politici) de selecție a acțiunilor, care definesc modul în care se fac tranzițiile între stări în procesul de învățare.

Politica *greedy* se referă la faptul că agentul alege acțiunea cu cea mai mare valoare în fiecare stare [SB98]. O altă metodă, care menține un echilibru între explorarea stărilor noi și exploatarea cunoștințelor actuale, este ϵ-*greedy* [SB98]. Un agent care urmează această politică va alege, în cele mai multe cazuri (cu probabilitate $1 - \epsilon$), acțiunea având cea mai mare recompensă, dar cu o probabilitate mică (ϵ) agentul va explora noi acțiuni dintr-o stare, selectând o acțiune în mod aleator, după o distribuție uniformă și independent de valoarea acesteia. Această politică are și un dezavantaj, anume faptul că atunci când o acțiune se alege aleator, cea mai nesatisfăcătoare acțiune poate fi selectată cu aceeași probabilitate ca și a doua cea mai bună. O modalitate de a contracara acest dezavantaj este folosirea unei politici care alege des acțiuni mai bune - *politica softmax*. Acest mecanism de selecție are avantajul că acțiunile sunt clasificate în funcție de estimările valorilor lor și fiecare acțiune este aleasă cu o probabilitate calculată folosind valoarea sa [SB98].

3.1.3 Urme de Eligibilitate (Eligibility Traces)

Urmele de eligibilitate au fost introduse în [Klo72] și ele reprezintă un mecanism de bază în RL pentru gestiunea *delay*-ului [SS96]. Ideea este că de fiecare dată când o stare este vizitată, aceasta este marcată printr-o urmă, care apoi scade treptat în timp, exponențial, în funcție de un parametru de degradare λ ($0 \leq \lambda \leq 1$) și în funcție de factorul de actualizare γ. Urma indică faptul că starea este *eligibilă* pentru învățare [SS96].

Există două implementări posibile pentru aceste urme de eligibilitate. Prima este *urme de eligibilitate cu acumulare (accumulating eligibility traces)* - urma crește de fiecare dată când starea este vizitată. Li se atribuie mai mult credit stărilor vizitate mai recent și mai des. Al doilea tip este *urme de eligibilitate cu înlocuire (replacing eligibility traces)* - de fiecare dată când o stare este vizitată urma sa este resetată la 1, indiferent de informațiile anterioare legate de această urmă.

3.2 Noi Modele Bazate pe Învățarea prin Întărire. Considerații Teoretice

În acest subcapitol propunem două noi modele bazate pe RL și o abordare RL distribuită, care vor fi ulterior utilizate pentru a aborda trei probleme importante în bioinformatică.

Dintr-o perspectivă computațională, fiecare dintre cele trei probleme pe care dorim să le abordăm utilizând tehnici RL (asamblarea fragmentelor ADN - Subcapitolul 1.4, predicția structurii terțiare a proteinelor - Subcapitolul 1.5 și problema ordonării temporale - Subcapitolul 1.6) este o problemă NP-completă de optimizare combinatorială. O altă caracteristică comună tuturor celor trei probleme este că fiecare poate fi privită ca o *problemă de identificare a unei permutări generalizate*, ceea ce înseamnă că soluția poate fi codificată într-o permutare generalizată a unui anumit set de obiecte, care îndeplinește anumite condiții și optimizează o funcție obiectiv. Prin termenul de *permutare generalizată*, ne referim la *permutările ordinare* (în care toate obiectele sunt distincte) și la *permutările cu repetiții*, în care pot exista mai multe copii ale aceluiași obiect.

Introducem o definiție generală a unei probleme de optimizare, în care se cere determinarea unei permutări generalizate. Avem ca date de intrare un set de n obiecte și o funcție obiectiv, generică, definită pe mulțimea tuturor permutărilor generalizate ale obiectelor, care asociază fiecărei permutări o valoare reală, indicând calitatea și relevanța acesteia. Scopul este de a determina o permutare generalizată σ a mulțimii $\{1, 2, \ldots, n\}$, care optimizează funcția obiectiv a secvenței de obiecte considerate în ordinea dată de σ. Notăm cu m lungimea permutării σ, unde $m \geq n$ ($m = n$ pentru cazurile în care obiectele nu pot fi repetate).

3.2.1 Modelul Determinării unui Drum

În cele ce urmează, vom introduce un prim model RL, numit *modelul determinării unui drum*. Ideea sa principală se bazează pe construirea unei permutări generalizate a obiectelor date, care optimizează funcția obiectiv, pornind de la o secvență vidă și adăugând iterativ obiectele cele mai adecvate până când permutarea este complet formată.

Mediul poate fi vizualizat ca un arbore, care conține $\frac{n^{m+1}-1}{n-1}$ noduri, fiecare nod corespunzând unei stări. *Starea inițială* este reprezentată de către rădăcină. Spațiul acțiunilor conține n acțiuni corespunzătoare celor n valori posibile $1, 2, \ldots, n$, utilizate pentru reprezentarea unei soluții. La un moment dat, se poate trece de la o stare la una dintre cele n stări succesor, prin executarea uneia dintre cele n acțiuni posibile. Tranzițiile între stări sunt echiprobabile, probabilitatea de tranziție fiind egală cu $1/n$. Pentru fiecare tranziție, agentul primește o recompensă, diferită de la o problemă la alta, în cele mai multe cazuri acesta fiind definită astfel încât să optimizeze funcția obiectiv.

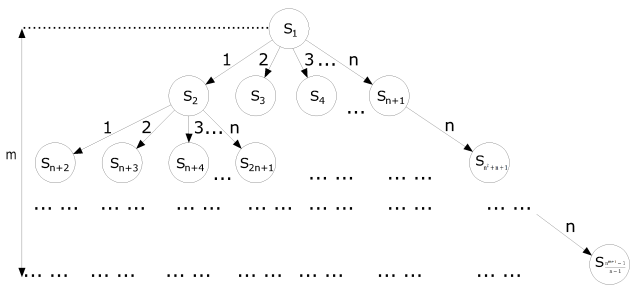
O reprezentare grafică a spațiului de stări asociat modelului determinării unui drum este dată în Figura 3.1a. De la starea inițială s_1, agentul va alege una dintre cele n acțiuni pentru a efectua o tranziție. Acest proces se va repeta pentru fiecare stare nouă, rezultând într-o "coborâre" a agentului cu câte un nivel (în arbore), până când se ajunge la o *stare finală* - una dintre stările poziționate pe ultimul nivel. În acest mediu, un drum până la o stare finală este compus din noduri distincte (stările), iar două noduri adiacente sunt conectate printr-un arc (acțiune). Secvența de acțiuni obținută în urma tranzițiilor succesive între stările unui drum va fi denumită *configurație de acțiuni* asociată drumului și aceasta determină o secvență (o permutare generalizată) a obiectelor de intrare.

Pentru fiecare problemă în care se caută o permutare generalizată optimă, agentul va învăța (folosind algoritmul Q-learning), pe parcursul mai multor episoade de antrenare, un drum cu valoarea optimă a funcției obiectiv. După încheierea procesului RL de instruire, agentul învață să execute acele tranziții care maximizează suma recompenselor primite pe un drum de la o stare inițială la una finală.

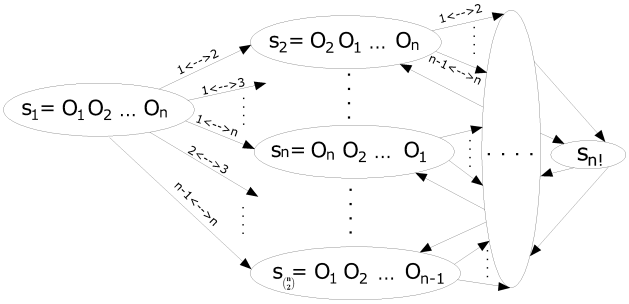
3.2.2 Modelul Permutărilor

Al doilea model RL, numit *modelul permutărilor* este prezentat în continuare. Acesta se referă numai la permutări ordinare, în care toate elementele trebuie să fie distincte (pentru care $m = n$). Ideea este de a ajunge la o permutare de obiecte având valoarea optimă a funcției obiectiv. Se pornește de la permutarea identică și se folosește o rafinare iterativă (prin derivarea de noi permutări) până când se ajunge la o permutare finală având valoarea asociată a funcției obiectiv suficient de aproape de o valoare țintă.

În cazul acestui model, fiecare stare reprezintă o soluție posibilă, adică o permutare a obiectelor date. Mediul este reprezentat grafic în Figura 3.1b. El este compus din $n!$ stări posibile. *Starea inițială*



(a) Mediul asociat modelului determinării unui drum. Cercurile reprezintă stările, iar tranzițiile între stări sunt indicate de săgețile etichetate cu acțiunea care cauzează tranziția de la o stare la alta.



(b) Mediul asociat modelului permutărilor. O_i indică al i-lea object din setul de date. Stările sunt reprezentate în interiorul elipselor iar tranzițiile între stări sunt indicate de săgețile etichetate cu acțiunea care cauzează tranziția de la o stare la alta.

Figura 3.1: Reprezentare grafică a mediilor pentru cele două modele RL.

este permutarea identică a obiectelor și considerăm că o stare este *finală* dacă valoarea asociată a funcției obiectiv este suficient de aproape de o valoare țintă. Pentru a putea defini o stare finală sunt necesare cunoștințe a priori despre problemă. Pentru a trece de la o stare la alta, agentul poate alege una din cele $\binom{n}{2} = \frac{n(n-1)}{2}$ acțiuni. Fiecare acțiune este o pereche de indici specificând că obiectele aflate pe pozițiile definite de acești indici în starea actuală vor fi interschimbate pentru a ajunge la o stare succesor. Tranzițiile între stări sunt echiprobabile. Funcția recompensă este definită luând în considerare funcția obiectiv pentru o secvență de obiecte și este specifică fiecărei probleme.

Spre deosebire de modelul determinării unui drum, în acest caz nu ne interesează secvența de acțiuni determinată de către agent până la o stare finală, ci mai degrabă dorim să aflăm starea finală în care se află mediul după ce agentul își execută politica învățată. Această stare finală, care reprezintă o permutare a obiectelor inițiale, ar trebui să aibă o valoare asociată foarte aproape de valoarea optimă a funcției obiectiv.

3.2.3 Modelul RL Distribuit

În scopul accelerării procesului de antrenare, extindem modelele RL propuse înspre o abordare RL distribuită. Propunem o abordare Q-learning concurentă, în care mai mulți agenți cooperativi învață să se coordoneze, în scopul de a determina o politică optimă în mediul lor.

În arhitectura distribuită avem două tipuri de agenți: *agenți locali*, care rulează în procese sau fire de execuție separate și sunt instruiți folosind algoritmul Q-learning; un *agent supervizor*, care supraveghează procesul de învățare și sincronizează datele învățate de către agenții locali individuali. Se păstrează un tabel [GBF+07] care stochează estimările globale ale Q-valorilor. Agentul supervizor actualizează Q-valorile globale numai în cazul în care noile valori primite de la agenții locali sunt mai bune decât cele existente în tabel.

Pe parcursul unor episoade de antrenare, agenții locali individuali experimentează drumuri de la o stare inițială către una finală, actualizând estimările Q-valorilor prin folosirea unei metode Q-learning (Secțiunea 3.1.1) și cu ajutorul uneia dintre cele două modele prezentate anterior în această secțiune (modelul determinării unui drum sau modelul permutărilor). După finalizarea antrenării sistemului multi-agent, soluția învățată de către agentul supervizor este construită pornind de la starea inițială și folosind o politică *greedy* până la determinarea stării finale.

3.2.4 O Nouă Politică Inteligentă de Selecție a Acțiunilor

Introducem o nouă politică inteligentă de selecție a acțiunilor, cu scopul de a ghida mai eficient agentul RL înspre soluții corecte. Acest mecanism de selecție este derivat din mecanismul ϵ-Greedy (Secțiunea 3.1.2) și folosește un procedeu de tip *look-ahead*, în scopul de a explora mai bine spațiul de căutare. Pentru selectarea unei acțiuni dintr-o stare, se utilizează următorul mecanism de selecție: cu probabilitatea $1 - \epsilon$ se alege acțiunea care maximizează Q-valoarea stării succesor (selecție ϵ-*Greedy*) și cu probabilitate ϵ se alege acțiunea pe care optimizează valoarea funcției obiectiv corespunzătoare configurației curente (pasul *look-ahead*).

3.3 Modele bazate pe Învățarea prin Întărire pentru Problema Asamblării Fragmentelor ADN

În acest subcapitol, cele trei modele RL introduse în Secțiunile 3.2.1, 3.2.2 și 3.2.3 sunt adaptate pentru a aborda problema asamblării fragmentelor ADN (fragment assembly - FA). Această problemă a fost prezentată în Subcapitolul 1.4 și se referă la reconstrucția unei secvențe ADN originale din fragmente ADN mai mici.

3.3.1 Metodologie

În cele ce urmează considerăm că $\mathcal{S}eq$ este o secvență ADN și $\mathcal{FS} = \{F_1, F_2, \ldots, F_n\}$ este un set de fragmente. Fiecare fragment este o subsecvență mai mică a ADN-ului original și fragmentele conțin regiuni care se suprapun. Problema FA constă în determinarea ordinii în care aceste fragmente trebuie să fie asamblate pentru a construi molecula ADN inițială, pe baza subșirurilor comune ale fragmentelor. Prin urmare, problema FA poate fi privită ca problema de a genera o permutare σ a mulțimii $\{1, 2, \ldots, n\}$ care optimizează performanța secvenței $\sigma_{\mathcal{FS}} = (F_{\sigma_1}, F_{\sigma_2}, \ldots, F_{\sigma_n})$ $(n > 1)$. Funcția obiectiv care trebuie maximizată este o măsură de performanță PM, obținută prin însumarea unor scoruri care cuantifică suprapunerile tuturor fragmentelor adiacente [PFB95].

În primul rând am adaptat și aplicat modelul determinării unui drum pentru a rezolva această problemă. Spațiul stărilor și cel al acțiunilor, precum și funcția de tranziție rămân neschimbate. Oferim două definiții posibile pentru funcția de recompensă, ambele depinzând de măsura de performanță a unei permutări. Prima funcție recompensă, definită în Formula (3.2), ilustrează situații în care feedback-ul este dat la sfârșitul fiecărui episod al procesului RL. Această funcție poate fi modificată astfel încât să ofere feedback agentului după fiecare tranziție efectuată, chiar dacă starea nu este finală. Prin urmare, a doua definiție a funcției recompensă este dată în Formula (3.3).

$$r(\pi_k) = \begin{cases} PM(F_{a_\pi}) & dacă \quad k = n \\ \tau & altfel \end{cases} \qquad (3.2)$$

$$r(\pi_k) = \begin{cases} 0 & dacă\ k = 0\ sau\ k = 1 \\ w(F_{a_{\pi_{k-1}}}, F_{a_{\pi_{k-2}}}) & altfel \end{cases} \qquad (3.3)$$

unde $r(\pi_k)$ este recompensa primită de către agent în starea π_k, după istoricul mediului $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{k-1}$, $a_\pi = (a_{\pi_0} a_{\pi_1} \ldots a_{\pi_{n-1}})$ este o posibilă configurație a acțiunilor, iar $w(a, b)$ reprezintă scorul de similaritate a fragmentelor a și b.

3.3.2 Evaluare Experimentală și Rezultate

Experimentele computaționale sunt efectuate pe un exemplu mic artificial,i pe o secvență ADN aparținând bacteriei *Escherihia coli (E. coli)*, precum și pe o secvență ADN din genomul uman.

3.3.2.1 Experimentul 1 - Exemplu artificial

Am testat abordările propuse pe un mic exemplu descris în Subcapitolul 1.4. Secvența ADN este $TTACCGTGC$, iar setul de fragmente este: $F_1 = ACCGT$, $F_2 = CGTGC$, $F_3 = TTAC$, $F_4 = TACCGT$. Ordinea corectă a fragmentelor care determină secvența ADN originală este: $F_3F_4F_1F_2$.

Agentul a fost antrenat folosind funcția recompensă definită în Formula (3.2). Am utilizat mecanismul ϵ-Greedy de selecție a acțiunilor și 240 de episoade de antrenare. După etapa de antrenare au fost raportate două soluții optime: $F_3F_4F_1F_2$ și $F_2F_1F_4F_3$, ambele având valoarea maximă a măsurii de similaritate.

Pe acest exemplu am aplicat și modelul RL distribuit prezentat în Secțiunea 3.2.3. Am folosit doi agenți locali, fiecare utilizând modelul determinării unui drum și Q-learning. Aceleași două soluții optime au fost raportate de către agentul supervizor, după încheierea antrenării agenților locali.

3.3.2.2 Experimentul 2 - Secvență ADN aparținând bacteriei E. coli

În acest al doilea exemplu, am ales o mică porțiune din ADN-ul bacteriei *Escherihia coli (E. coli)*. Secvența conține 25 de nucleotide: $TACTAGCAATACGCTTGCGTTCGGT$. Folosind scripturile Perl din [ZCY+11] am obținut 10 fragmente, fiecare având o lungime de 8 nucleotide. Pentru a obține secvența originală, ele trebuie ordonate astfel: $F_6F_3F_{10}F_5F_7F_9F_1F_8F_2F_4$.

În experimentele noastre, am folosit ambele funcții recompensă definite în Secțiunea 3.3.1 și două politici de selecție a acțiunilor: politica ϵ-Greedy și politica inteligentă prezentată în Secțiunea 3.2.4.

Soluția raportată în urma antrenării agentului este cea corectă, având valoarea maximă a măsurii de performanță, pentru toate testele.

În prima serie de teste am folosit funcția recompensă definită în Formula (3.2) și politica ϵ-Greedy, cu $\epsilon = 0.8$. Soluția, adică permutarea având măsura optimă de performanță, se obține după $486 \cdot 10^3$ episoade, în medie. În al doilea rând, am testat modelul RL folosind funcția recompensă definită în Formula (3.3) și politica ϵ-Greedy, cu aceeași valoare a parametrului ϵ. Deoarece agentul este recompensat după fiecare tranziție, învățarea este mai rapidă. Ca urmare, permutarea corectă a fragmentelor se obține după o medie de $334 \cdot 10^3$ episoade. În cele din urmă, modelul RL a fost testat folosind funcția recompensă definită în Formula (3.3) și politica inteligentă de selecție a acțiunilor introdusă în Secțiunea 3.2.4. Prin utilizarea acestui mecanism de selecție, agentul ajunge la soluția optimă mult mai rapid, în medie, după mai puțin de 10^3 episoade.

3.3.2.3 Experimentul 3 - Secvență ADN aparținând genomului uman

Pentru a evalua algoritmul nostru pe secvențe mari, am ales o secvență de ADN uman. Aceasta a fost preluată de la NCBI (National Center for Biotechnology Information) [LAR+10]. Secvența se referă la o secțiune numită "Human MHC class III region DNA with fibronectin type-III repeats" și are o lungime de 3835 de nucleotide. Am folosit un program numit GenFrag [EB96] pentru a genera 20 fragmente pentru secvența menționată. Lungimea medie a unui fragment este de 766 de nucleotide.

Pentru a evita gruparea unor fragmente care conțin regiuni lungi care se suprapun, dar care nu trebuie să fie adiacente în permutarea finală, pentru acest exemplu vom folosi o funcție de performanță diferită. Această funcție, introdusă de către Parsons *et. al* [PFB95] penalizează soluții în care apar suprapuneri puternice între fragmente îndepărtate din aliniament și ea trebuie minimizată.

Considerând măsura de performanță menționată mai sus, funcția recompensă se modifică de asemenea, fiind definit în Formula 3.4:

$$r(\pi_k) = \begin{cases} 0 & dacă \quad k = 0 \; sau \; k = 1 \\ -2 \cdot \sum_{i=0}^{k-2} (k-1-i) \cdot w(F_{a_{\pi_i}}, F_{a_{\pi_{k-1}}}) & altfel \end{cases} \tag{3.4}$$

unde $r(\pi_k)$ este recompensa primită de către agent în starea π_k, după istoricul mediului $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{k-1}$, iar $w(a, b)$ reprezintă scorul de similaritate a fragmentelor a și b.

Se poate demonstra că suma totală a recompenselor este de fapt egală cu opusul măsurii de performaţă. Recompensa va fi maximizată, deci funcția performanță va fi minimizată.

Am evaluat permutările obținute din punct de vedere al numărului de secvențe continue. Permutarea optimă obținută formează o singură secvență continuă, care va fi apoi folosită pentru a obține molecula ADN originală. Algoritmul nostru bazat pe RL a fost rulat de 5 ori, iar convergența se atinge relativ rapid, după $32 \cdot 10^3$ episoade, în medie.

3.3.3 O Comparație între *Modelul Găsirii unui Drum* și *Modelul Permutărilor*

Pentru a compara performanțele celor două modele, considerăm aceeași secțiune de ADN de la bacteria *Escherihia coli (E. coli)*, dar în acest caz am generat doar 8 fragmente, care alcătuiesc secvența ADN-ului, după cum urmează: $F_4F_7F_5F_6F_8F_3F_2F_1$.

Ambele modele RL sunt aplicate pentru a determina o permutare optimă a fragmentelor. Modelul determinării unui drum este aplicat în mod similar celor două experimente anterioare, folosind ca recompensă funcția definită în Formula (3.3). În ceea ce privește modelul permutărilor, spațiile stărilor și acțiunilor, precum și funcția de tranziție sunt cele prezentate în Secțiunea 3.2.2. Funcția recompensă asociată unei tranziții dintr-o stare $s_j = \sigma^j_{\mathcal{FS}}$ într-o stare $s_l = \sigma^l_{\mathcal{FS}}$ ($j,\ l \in \{1, \cdots, n!\},\ l \neq j$), prin executarea acțiunii $a_k, 1 \leq k \leq N_a$, este:

$$r(s_l = F_{\sigma^l} | s_j, a_k) = \begin{cases} PM(\sigma^l_{\mathcal{FS}}) - PM(\sigma^j_{\mathcal{FS}}) & dacă \; s_l \; nu \; este \; stare \; finală \\ PM(\sigma^l_{\mathcal{FS}}) - PM(\sigma^j_{\mathcal{FS}}) + PM(s_0) & altfel \end{cases} \tag{3.5}$$

unde s_0 este permutarea identică.

Comparăm cele două modele prin rularea implementărilor corespunzătoare, folosind algoritmul Q-learning [SB98] și trei mecanisme de selecție a acțiunilor: ϵ-Greedy, softmax (Secțiunea 3.1.2)

și mecanismul inteligent de selecție (Secțiunea 3.2.4). Modelul permutărilor a depășit abordarea determinării unui drum, pentru studiul de caz considerat, în ceea ca privește numărul de epoci, precum și timpul de execuție. Totuși, în cazul modelului permutărilor, menționăm că este dificilă determinarea unei stări finale și pentru a putea face acest lucru avem nevoie de informații a-priori în legătură cu valorile posibile ale măsurii de performanță.

Pentru modelul permutărilor numărul de stări ale mediului este mai mic decât în cazul abordarea care determină un drum. Cu toate acestea, un dezavantaj important al acestui model este faptul că necesită cunoștințe suplimentare despre problemă pentru a putea defini o stare finală și aceste informații nu sunt tot timpul disponibile. Astfel, modelul determinării unui drum este mai general și pentru că rezultatele obținute sunt bune, atât în ceea ce privește precizia și în ceea ce privește timpul de execuție, putem concluziona că acest model este mai eficient.

3.3.4 Discuție

Am evaluat experimental fiecare model RL nou pe mai multe secvențe ADN, folosind algoritmul Q-learning, cu două funcții recompensă diferite și trei politici de selecție a acțiunilor cu diverse valori ale parametrilor.

În ceea ce privește modelul RL al determinării unui drum aplicat pentru asamblarea fragmentelor ADN, antrenarea pe parcursul unui episod are o complexitate de $\theta(n)$ (n fiind numărul de fragmente), atunci când algorimul Q-learning este folosit cu politica ϵ-Greedy. În cazul în care agentul efectuează selecția acțiunilor folosind mecanismul inteligent (procedura *look-ahead*), complexitatea procesului în timpul unui episod este de $\theta(n^2)$. În consecință, presupunând că numărul episoadelor de antrenare este k, complexitatea totală a algoritmului de antrenare a agentului este $\theta(k \cdot n)$ sau $\theta(k \cdot n^2)$, în funcție de politica aleasă.

Am introdus două funcții recompensă pentru un agent care învață folosind modelul determinării unui drum. Funcția care răsplătește agentul după fiecare tranziție conduce la o învățare mai rapidă decât cea care îi oferă feedback abia la sfârșitul unui episod. Procesul de învățare este și mai mult accelarat o dată cu utilizarea politicii inteligente de selecție a acțiunilor (Secțiunea 3.2.4).

Pentru a evita considerarea unui număr foarte mare de episoade pentru a obține o soluție, am introdus și un model RL distribuit. Principalul avantaj al abordării distribuite este faptul că, prin utilizarea mai multor agenți în timpul fazei de antrenare, timpul total de execuție este redus considerabil. Totuși, în continuare trebuie să investigăm cum se poate păstra acuratețea rezultatelor în abordarea distribuită.

3.4 Modele bazate pe Învățarea prin Întărire pentru Problema Predicției Structurii Terțiare a Proteinelor

Acest subcapitol descrie modul în care două dintre modelele RL introduse în subcapitolul 3.2, mai precis modelul determinării unui drum (Secțiunea 3.2.1) și modelul distribuit (Secțiunea 3.2.3), sunt aplicate pentru a rezolva problema predicției structurii terțiare a proteinelor (Protein Tertiary Structure Prediction - PTSP), care se referă la estimarea structurii tridimensionale a unei proteine, pornind de la secvența liniară de aminoacizi. Această problemă a fost prezentată în Subcapitolul 1.5.

3.4.1 Metodologie

Pentru definirea sarcinii de învățare RL, vom folosi modelul hidrofob-Polar (Hydrophobic-Polar HP) [Dil85] (Subcapitolul 1.5). Pentru o anumită proteină, compusă din aminoacizi (fiecare aminoacid fiind fie hidrofob fie polar), problema este de a găsi poziția fiecărui aminoacid într-o matrice bidimensională, astfel încât să fie minimizată o funcție de energie.

O soluție bidimensională a acestei probleme, corespunzând unei proteine $\mathcal{P}$ de lungime q ar putea fi reprezentată printr-o secvență de lungime $q-1$, în care fiecare poziție codifică direcția aminoacidului curent față de cel precedent (L (left) - stânga, R (right) - dreapta, U (up) - sus, D (down) - jos).

Prezentăm în cele ce urmează modul în care modelul RL care determină un drum introdus în Secțiunea 3.2.1 este adaptat și aplicat pentru a rezolva problema PTSP. Având în vedere că o soluție poate fi privită ca o secvență compusă din cele patru simboluri (L, R, U, D) care exprimă direcțiile aminoacizilor, problema PTSP poate fi modelată ca o problemă de generare a unei permutări generalizate π care minimizează funcția de energie în modelul HP. În acest caz, permutarea nu este

una ordinară, ci o permutare cu repetiții, deoarece fiecare dintre cele patru simboluri pot apărea de oricâte ori este necesar. Prin urmare, remarcăm că obiectele menționate în Subcapitolul 3.2 sunt aici reprezentate de cele patru direcții, adică $n = 4$ și lungimea m a unei permutări cu o unitate mai mică decât numărul de aminoacizi care formează secvența primară a proteinei.

Spațiile stărilor și acțiunilor, precum și funcția de tranziție sunt similare cu cele prezentate în Secțiunea 3.2.1. Funcția recompensă este definită astfel încât să se asigure minimizarea energiei proteinei. Prin urmare, după o tranziție într-o stare intermediară (care nu este finală), agentul este recompensat cu o constantă mică pozitivă; pentru o tranziție la o stare finală agentul primește ca recompensă valoarea opusă a funcției de energie asociată configurației obținute. În acest fel, încercând maximizarea sumei recompenselor, agentul minimizează, de fapt, funcția de energie.

3.4.2 Evaluare Experimentală și Rezultate

Modelul determinării unui drum și modelul RL distribuit (Secțiunea 3.2) sunt evaluate experimental folosind câteva proteine HP generate artificial, precum și o proteină de referință utilizată în general pentru predicția structurii bidimensionale a proteinelor.

3.4.2.1 Experimentul 1

Considerăm un exemplu simplu, o secvență proteică HP $\mathcal{P} = HHPH$, compusă din patru aminoacizi, adică $q = 4$. Agentul este antrenat folosind algoritmul Q-learning. Am folosit polica de selecție ϵ-Greedy, cu $\epsilon = 1$ și 100 de episoade de antrenare.

Soluția raportată după ce antrenarea agentului a fost încheiată are valoarea minimă a funcției de energie asociate egală cu -1. Această soluție este optimă și are *configurația* (ULD). Soluția echivalentă cu următoarele configurații: (RUL), (DRU) și (LDR), care sunt toate rotații ale sale în jurul punctului fix din matrice unde este poziționat primul aminoacid. De asemenea, această soluție este echivalentă cu configurația (URD), care este versiunea sa oglindită, precum și rotațiile oglindite echivalente (LUR), (RDL) și (DLU). Agentul poate ajunge la oricare dintre aceste soluții, care sunt toate corecte și au energia minimă asociată $E^* = -1$.

Pe acest exemplu simplu, am aplicat și modelul RL distribuit introdus în Secțiunea 3.2.3. Doi agenți locali sunt instruiți pe parcursul a 40 de episoade, folosind abordarea determinării unui drum. După încheierea etapei de antrenare, soluția raportată de agentul supervizor este aceeași configurație optimă: (ULD).

3.4.2.2 Experiment 2

În al doilea experiment secvența proteică este compusă din $q = 11$ aminoacizi: $\mathcal{P} = HHHPHPPP$ PPH. Exemplul a fost preluat din [Chi10] și valoarea minimă a funcției de energie este $E^* = -2$.

Agentul a fost antrenat de-a lungul a 10^5 episoade și folosind mecanisul ϵ-Greeedy de selecție a acțiunilor, cu $\epsilon = 0.8$, pentru a permite explorarea spațiului de căutare, dar și exploatarea Q-valorilor învățate. Agentul are nevoie, în medie, de $43 \cdot 10^3$ episoade pentru a obține o configurație având energia minimă. Rezultatele raportate reprezintă media a 5 rulări ale algoritmului. În urma antrenării, agentul găsește mai multe soluții optime, toate având energia -2: $(DRDLDLULUR)$, $(DDRUUULLDD)$, $(DLDRDRRUUL)$, $(RURDRRDLLL)$, $(ULURRRDDLU)$, $(URULLLDDRU)$, $(ULURURRDLD)$, împreună cu rotațiile aferente și cu versiunile oglindite.

3.4.2.3 Experiment 3

Considerăm în acest exemplu o proteină HP $\mathcal{P} = HPHPPHHPHPPHPHHPPHPH$ compusă din 20 de aminoacizi, adică $q = 20$. Aceasta este o secvență proteică de referință utilizată în general pentru predicția structurii bidimensionale a proteinelor. Poate fi găsită și în [UM93] și valoarea optimă a energiei este $E^* = -9$.

Am antrenat agentul de-a lungul a 10^5 episoade și folosind mecanisul ϵ-Greeedy de selecție a acțiunilor, începând cu $\epsilon = 1$, pentru a favoriza explorarea, apoi am scăzut treptat valoarea lui ϵ pe durata procesului de antrenare, în final ajungându-se la o valoare mică, aceasta însemnând că la finalul antrenării este favorizată exploatarea.

Folosind parametrii definiți mai sus și presupunând că perechile (stare, acțiune) sunt vizitate în mod egal în timpul antrenamentului, soluția raportată în urma antrenării agentului este configurația $(RUULDLULLDRDRDLDRRU)$. Figura 3.2 [CBC11b] ilustrează această configurație.

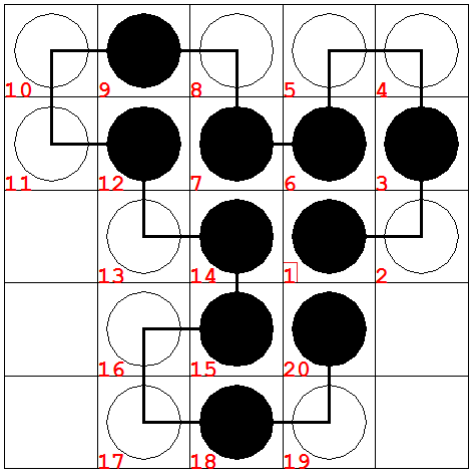


Figura 3.2: O configurație pentru secvența proteică $\mathcal{P} = HPHPPHHPHPPHPHHPPHPH$, de lungime 20. Configurația învățată este $(RUULDLULLDRDRDLDRRU)$. Cercurile negre reprezintă aminoacizii hidrofobi, iar cele albe îi reprezintă pe cei hidrofili. Valoarea funcției de energie pentru această configurație este -9.

3.4.3 Discuție

În acest subcapitol am abordat cazul bidimensional al problemei predicției structurii terțiare a proteinelor folosind două modele bazate pe învățarea prin întărire care au fost prezentate anterior. Aceste modele sunt adaptate pentru a adresa problema PTSP, care poate fi privită, din punct de vedere computațional, ca problema generării unei permutări optime cu repetiții ce conține direcțiile aminoacizilor proteinei.

În ceea ce privește modelul determinării unui drum pentru rezolvarea problemei bidimensionale PTSP, remarcăm următoarele. Pentru abordarea Q-learning folosită în combinație cu politica ϵ-Greedy, procesul de instruire a unui episod are o complexitate de timp de $\theta(n)$, unde n este lungimea secvenței proteinei HP. Prin urmare, presupunând că numărul de episoade de antrenare este k, complexitatea generală a algoritmului de antrenare a agentului este $\theta(k \cdot n)$. Dacă dimensiunea n a secvenței proteice HP este mare și prin urmare, spațiul stărilor devine foarte mare, ar trebui utilizată o metodă de aproximare a funcțiilor pentru a stoca estimările Q-valorilor (de exemplu o rețea neuronală sau mașini cu suport vectorial).

Principalul dezavantaj al abordării noastre este că trebuie luat în considerare un număr foarte mare de episoade de antrenare pentru a obține rezultate precise, ceea ce duce la o convergență lentă. În scopul de a accelera procesul de convergență vom considera îmbunătățiri suplimentare.

3.5 Concluzii și Cercetări Ulterioare

În acest capitol am prezentat noi modele bazate pe învățarea prin întărire pentru a rezolva o clasă generală de probleme de optimizare combinatorială și am adaptat și aplicat aceste modele pentru a rezolva două probleme majore în bioinformatică: problema asamblării fragmentelor ADN (Subcapitolul 1.4) și problema predicției structurii terțiare a proteinelor (Subcapitolul 1.5). Abordările prezentate în acest capitol au fost publicate în [CBC13, CCB13, BCC11a, CBC11a, BCC11b, CBC11c, CBC11b, CBC11d].

Tehnicile nou introduse au fost evaluate experimental pe mai multe seturi de date din domeniul problemelor abordate, iar rezultatele obținute au fost prezentate și analizate. Buna performanță a modelelor bazate pe învățarea prin întărire demonstrează potențialul propunerilor noastre.

Intenționăm să extindem evaluarea algoritmilor de învățare Q-learning pe secvențe ADN și secvențe proteice mai mari, pentru a dezvolta în continuare analizele noastre. De asemenea, vom investiga posibile îmbunătățiri ale acestor modele prin adăugarea unor mecanisme de căutare locală, prin combinarea politicilor softmax cu procedura inteligentă de selecție a acțiunilor (Secțiunea 3.2.4), prin modificarea parametrilor politicilor de selecție în timpul procesului de antrenare, sau prin considerarea unui model RL distribuit bazat pe modelul permutărilor pentru asamblarea fragmentelor ADN.

Capitolul 4

Noi Abordări pentru Ordonarea Temporală a Datelor Biologice

Acest capitol prezintă două abordări diferite pe care le-am propus pentru problema ordonării temporale (temporal ordering - TO) a datelor biologice (Subcapitolul 1.6). Prima abordare, dezvoltată în timpul stagiului meu de cercetare împreună cu un grup de cercetare de la Universitatea Milano-Bicocca, adresează problema TO dintr-un punct de vedere computațional, dar concentrându-se și pe aspecte biologice ale acesteia [BCG+12, BCG+13]. A doua adresează problema TO dintr-o perspectivă computațională și se referă la aplicarea unuia dintre modelele de învățare prin întărire (reinforcement learning - RL) introduse în capitolul precedent pentru rezolvarea acestei probleme [CBC13, Boc12a]. Mai mult, acest capitol mai prezintă o nouă interfață de programare pentru rezolvarea problemelor de optimizare folosind tehnici RL, care este aplicată pentru determinarea soluțiilor problemei TO [CCB11a, CCB11b].

4.1 Ordonarea Temporală a unor Probe asociate Cancerului Colorectal folosind Date conținând Alterații Cromozomiale

Acest subcapitol prezintă o nouă abordare a problemei TO. Metoda introdusă a fost dezvoltată în timpul stagiului meu de cercetare de la Universitatea din Milano-Bicocca, în colaborare cu grupul de cercetare BIMIB.

4.1.1 Concepte Biologice

Această secțiune prezintă câteva concepte biologice de bază necesare pentru o mai bună înțelegere a metodologiei pe care o propunem pentru ordonarea temporală a unui set de date care conține informații despre alterații cromozomiale.

Cancerul colorectal (Colorectal cancer - CRC) este al treilea cel mai frecvent tip de cancer la nivel mondial și a doua cea mai frecventă cauză de deces în ceea ce privește cancerul [JSXW10]. Cele mai multe CRC se dezvoltă printr-o serie de etape morfologice distincte, puternic corelate cu o funcționare defectuoasă a rețelelor complexe care guvernează dinamica criptei intestinale și a homeostaziei, modificări induse de instabilitatea genetică și acumularea de alterații în funcțiile unor gene-cheie [VFH+88, FV90, Fra07, Net12].

Alterațiile cromozomiale ale numărului de copii (copy number alterations - CNA) se referă la regiuni ale ADN-ului care au fost fie șterse (pierderi sau ștergeri) fie duplicate de un anumit număr de ori (amplificări) pe cromozomi. Aceste modificări pot afecta funcția unor gene prin modificarea expresiei lor și au fost asociate cu susceptibilitatea sau rezistența la anumite boli. În cancer, CNA pot duce, de asemenea, la activarea oncogenelor si inactivarea genelor supresoare de tumori. În timpul progresiei de la adenom la carcinom invaziv, apar frecvent anumite aberații cromozomiale specifice, cum ar fi amplificări și ștergeri pe anumiți cromozomi [ASD+10]. CNA sunt direct corelate cu cancerul și analiza CNA din celulele tumorale ar putea oferi informații suplimentare referitoare la procesul de progresie al CRC.

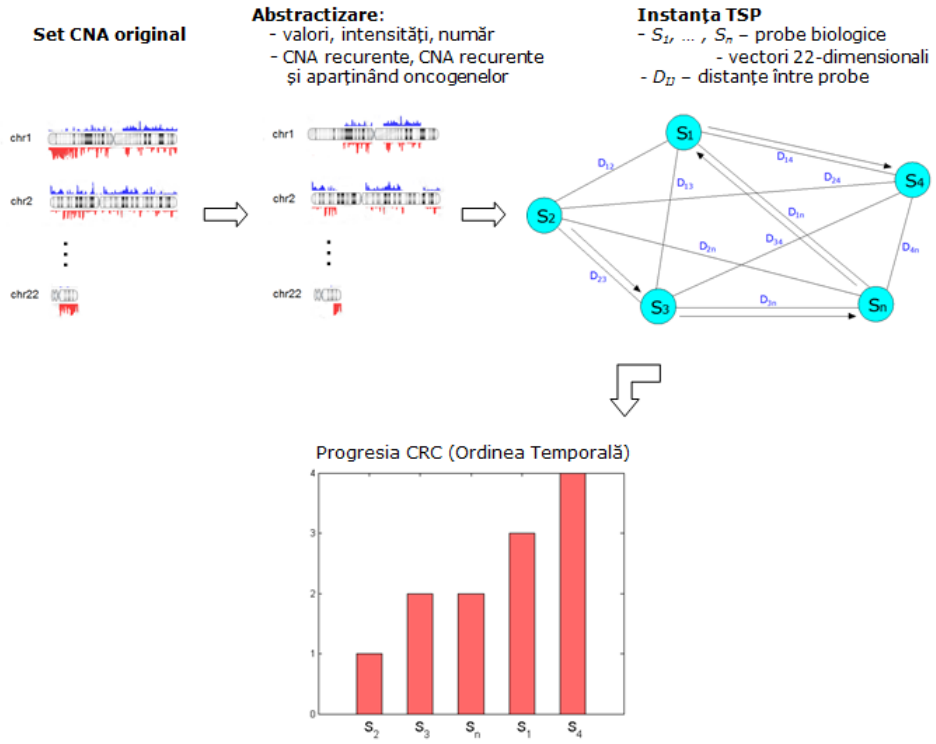


Figura 4.1: Reprezentarea metodologiei propuse. Pornind de la setul de date de intrare, folosim mecanisme de abstractizare pentru a defini diferite submulțimi ale acestuia. Un exemplu TSP este construit pentru fiecare nou set de date și în final soluția problemei TSP reprezintă ordonarea temporală a probelor biologice.

4.1.2 Metodologie

Această secțiune prezintă metodologia pe care o propunem pentru a obține o ordonare temporală a unui set de date CNA statice prelevate de la pacienți în diferite stadii ale cancerului colorectal. Propunerea noastră este axată pe o abordare specifică a problemei TO, propusă anterior de Gupta și Bar-Joseph [GBJ08], care funcționează pentru date de expresie genică. Tehnica prezentată în [GBJ08] se bazează pe reducerea problemei de ordonare la problema comis-voiajorului (travelling salesman problem - TSP), respectând două ipoteze biologice stabilite pentru date de expresie genică. Presupunând că datele CNA verifică, de asemenea, aceste două ipoteze, vom dezvolta o metodologie care ne permite să aplicăm tehnica propusă în [GBJ08] pe un set de date de tip CNA.

Figura 4.1 [BCG+12] ilustrează pe scurt metodologia noastră, subliniind cele mai importante etape care au fost folosite pentru a determina o ordonare temporală a unui set de probe biologice. Aceste etape vor fi detaliate în următoarele secțiuni.

În scopul de a surprinde aspecte distincte ale fenomenului complex CNA, definim mai multe măsuri cromozomiale și anumite filtre care vizează porțiuni semnificative din cromozomi. De asemenea, ne propunem să identificăm care dintre aceste măsuri cromozomiale dă cele mai multe informații în ceea ce privește progresia tumorii sau dacă amplificările sau ștergerile cromozomiale, considerate separat, ar putea influența rezultatul. Ca măsuri cromozomiale, introducem următoarele noțiuni: *valoare, intensitate, număr* și, analog, mediile acestora: *media valorilor și media intensităților*, toate referindu-se la modificări, ștergeri și amplificări. Mai mult, propunem două metode de filtrare care sunt aplicate pe setul inițial de date, ce ar putea conduce spre obținerea unor ordini mai exacte: (i) CNA recurente - avem în vedere acele CNA care aparțin unor regiuni ale cromozomilor care au suferit modificări în mai multe probe biologice; (ii) CNA recurente, precum și CNA aparținând regiunilor care includ cel puțin una dintre gene cunoscute a fi implicate în progresia tumorii (cancer driver genes).

Pentru a construi instanța TSP, considerăm nodurile ca fiind probele 22-dimensionale (fiecare dimensiune corespunde unui cromozom; nu luăm în considerare cromozomul sexual) și utilizăm o matrice a distanțelor pentru a defini distanța între oricare două probe. Pentru calculul distanțelor folosim două metrici: distanța L_1 și distanța euclidiană.

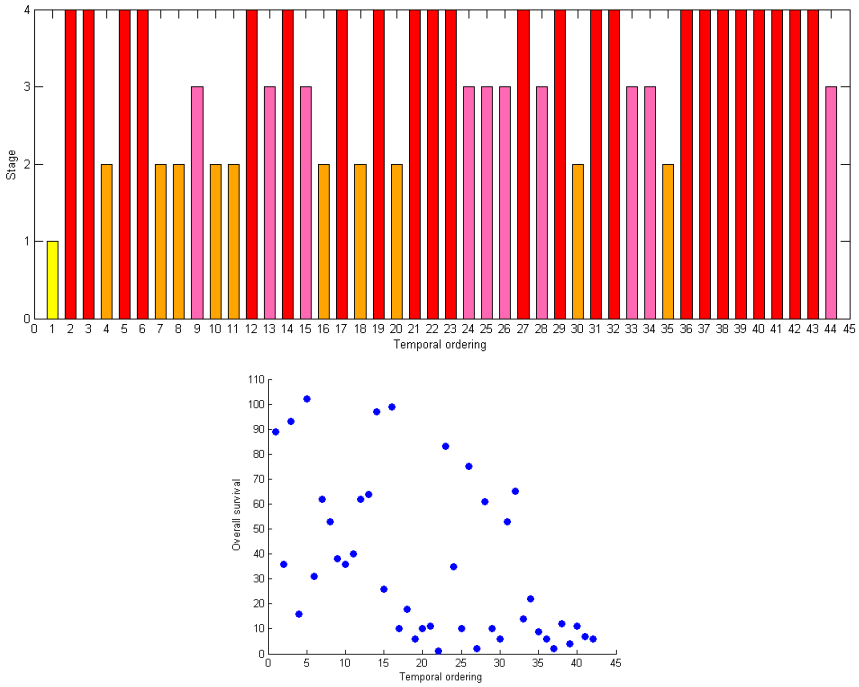


Figura 4.2: Cea mai bună ordine obținută pentru setul de date CRC. Prima figură reprezintă graficul ordinii obținute în raport cu stadiile histologice: I (galben), II (portocaliu), III (roz) și IV (roșu). A doua reprezintă graficul ordinii în raport cu timpul de supraviețuire. Probele din partea stângă au timpi mai mari de supraviețuire, după cum era de așteptat.

4.1.3 Experimente

Această secțiune prezintă experimentele dezvoltate în scopul de a testa eficiența metodologiei noastre. Evaluările experimentale sunt realizate pe un set CNA conținând date prelevate de la pacienți care suferă de cancer colorectal.

Setul de date a fost preluat din studiul lui Reid *et al* [RGS+09]. Acesta conține 44 de probe biologice prelevate de la pacienți în diferite stadii de CRC.

Au fost realizate trei tipuri de teste, unul pentru setul inițial de date de intrare și două pentru submulțimile obținute prin aplicarea filtrelor descrise mai sus. Prin urmare s-au obținut mai multe ordini temporale. După calculul valorilor cromozomiale ale tuturor probelor, folosind măsurile propuse, instanța TSP rezultată este rezolvată prin utilizarea programului Concorde TSP Solver [Coo11]. Ca și criteriu de validare am folosit timpul de supraviețuire a fiecărui pacient, după ce a fost diagnosticat. Definim *o ordine ideală* în care prima probă are asociat timpul maxim de supraviețuire, iar ultima - timpul minim. Pentru a determina cât de bună este o soluție calculăm distanța de la ea până la soluția ideală, folosind o distanță bazată pe devierea pătratică (*squared deviation distance - SDD*) [SS05]. Prin urmare, ordinile temporale având SDD mai mici (față de ordinea ideală) sunt considerate a fi mai precise.

Cel mai bun rezultat a fost obținut pentru testul care folosește CNA recurente, precum și CNA aparținând regiunilor care includ oncogene, cu măsura cromozomială care ia în calcul *media valorilor alterațiilor* și pentru *distanța euclidiană*. Figura 4.2 [BCG+13] (jos) reprezintă acest rezultat și ilustrează corelația între ordinea obținută și timpii de supraviețuire. Probele din jumătatea din stânga a graficului aparțin pacienților a căror timpi de supraviețuire (relativ la momentul diagnosticului) sunt mai mari, iar cele din jumătatea din dreapta au timpi mai mici de supraviețuire. Deși în datele noastre etapele histologice ale CRC nu sunt întotdeauna corelate direct cu timpul de supraviețuire (pe măsură ce stadiul bolii avansează timpul de supraviețuire ar trebui să scadă), în Figura 4.2 [BCG+13] (sus), se poate observa că pentru cea mai bună ordine există mai multe probe în stadiile I și II în jumătatea din stânga (9 probe) decât în cea din dreapta (2 probe) și mai multe probe în stadiile III și IV în jumătatea din dreapta (20 probe) decât în cea din stânga (13 probe). Observăm astfel că ordinea este, într-o anumită măsură corelată și cu etapele histologice. Putem remarca, de asemenea, că proba având stadiul I este pe prima poziție.

4.2 Model bazat pe Învățarea prin Întărire pentru Problema Ordonării Temporale a Datelor Biologice

Acest subcapitol prezintă modul în care modelul de învățare prin întărire (reinforcement learning - RL) care determină un drum (Secțiunea 3.2.1) poate fi adaptat și modificat pentru a aborda problema ordonării temporale a datelor biologice. Această problemă a fost descrisă în subcapitolul 1.6. Ea se referă la construirea unei colecții sortate de date biologice multi-dimensionale, care să reflecte o evoluție temporală a unui anumit proces biologic. Ne limităm aici la examinarea unor seturi conținând date de expresie genică obținute din experimente cu tehnologii de tip microarray.

4.2.1 Metodologie

Notăm setul de date de intrare cu $\mathcal{DS}$, acesta conținând n ($n > 1$) probe multi-dimensionale: $\mathcal{DS} = \{S_1, S_2, \ldots, S_n\}$, iar fiecare probă este identificată printr-un set de atribute. Pentru tipul de date considerat, fiecare atribut este reprezentat de o singură genă și are ca valoare un număr real ce măsoară nivelul de expresie a genei în cauză. Un prim pas al abordării noastre este preprocesarea datelor. Deoacere dimensionalitatea datelor de intrare poate fi extrem de mare (mii de niveluri de expresie a genelor pentru fiecare probă), scopul acestui pas este eliminarea acelor gene nu care oferă nici o informație semnificativă în procesul de ordonare temporală. În acest scop, realizăm o analiză statistică pe setul de date și cu ajutorul coeficientului de corelație Pearson [Tuf11], selectăm doar acele gene care sunt puternic corelate cu informațiile biologice suplimentare alese (în cazul setului de date asociate cancerului, aceste informații vor fi de timpii de supraviețuire).

Din punct de vedere computațional problema TO poate fi modelată ca problema generării unei permutări σ a mulțimii $\{1, 2, \ldots, n\}$, care maximizează similaritatea totală Sim a secvenței de probe considerate în ordinea dată de σ: $S_\sigma = (S_{\sigma_1}, S_{\sigma_2}, \ldots, S_{\sigma_n})$ ($n > 1$). Menționăm că în cazul problemei TO, permutarea căutată este ordinară, ceea ce înseamnă că toate elementele sale componente trebuie să fie distincte, prin urmare, lungimea permutării este egală cu numărul de probe. Similaritatea totală Sim însumează similaritățile tuturor probelor adiacente și trebuie să fie maximizată. Similaritatea totală a secvenței $S_\sigma = (S_{\sigma_1}, S_{\sigma_2}, \ldots, S_{\sigma_n})$ este definită ca fiind $Sim(S_\sigma) = \sum_{i=1}^{n-1} sim(S_{\sigma_i}, S_{\sigma_{i+1}})$, unde $sim(x_i, x_j)$ este similaritatea între vectorii multidimensionali x_i și x_j și este definită ca fiind $sim(x_i, x_j) = Max - d_E(x_i, x_j)$. Prin d_E notăm distanța euclidiană, iar Max este o constantă foarte mare.

Aplicăm modelul determinării unui drum introdus în Secțiunea 3.2.1 pentru a obține un drum de la o stare inițială la una finală, care codifică o permutare a probelor ce maximizează măsura de similaritate. Un drum $\pi = (\pi_0 = s_1, \pi_1, \pi_2, \ldots, \pi_n)$ este *valid* dacă toate acțiunile din *configurația* lui sunt distincte și fiecare probă din secvența $\mathcal{S}eq_\pi$ este mai similară cu proba care o urmează imediat decât cu orice altă probă. Spațiul stărilor și cel al acțiunilor, precum și funcția de tranziție rămân cele definite în Secțiunea 3.2.1. Pentru a obține o configurație optimă funcția recompensă este definită după cum urmează (Formula (4.1)):

$$r(\pi_k) = \begin{cases} 0 & dacă \quad k = 1 \\ -\infty & dacă\ \pi\ nu\ este\ valid \\ sim(S_{a_{\pi_{k-1}}}, S_{a_{\pi_{k-2}}}) & altfel \end{cases} \tag{4.1}$$

unde $r(\pi_k)$ este recompensa primită de către agent în starea π_k, după istoricul mediului $\pi_0 = s_1, \pi_1, \pi_2, ...\pi_{k-1}$, $k \in \{1, \ldots, n\}$, iar $a_\pi = (a_{\pi_0} a_{\pi_1} \ldots a_{\pi_{n-1}})$ este o posibilă configurație a acțiunilor.

Agentul primește o recompensă negativă pe drumurile invalide, prin urmare va învăța să exploreze doar drumuri valide. Având în vedere recompensa definită în Formula (4.1), se poate demonstra că agentul învață să găsească un drum valid care maximizează similaritatea totală a ordinii temporale asociate.

4.2.2 Evaluare Experimentală și Rezultate

În această secțiune ne propunem evaluarea experimentală a abordării RL pentru rezolvarea problemei TO, folosind mai multe seturi de date reale. Unele dintre acestea sunt serii temporale (ordinea temporală este cunoscută), permițându-ne astfel să validăm metoda noastră. Alte două seturi de date conțin probe extrase de la pacienți care suferă de cancer.

Set de date	**Ordinea RL obținută (S)**	*SMD* **(S)**	**Timp de execuție (sec.)**	**Ordinea din literatură (S')**	**SMD (S')**	**Îmb.**
Celule de drojdie afectate de schimbări ale mediului [GSK+00]						
"Heat shock"	1, 2, 3, 4, 5, 6, 7, 8	0	< 2	1, 8, 7, 6, 5, 4, 3, 2 [GBJ08]	2	Da
"DTT exposure"	1, 2, 3, 4, 5, 6, 7, 8	0	< 2	1, 2, 3, 4, 5, 6, 7, 8 [GBJ08]	0	La fel
"Amino acid starvation"	1, 2, 3, 4, 5	0	< 2	1, 2, 3, 4, 5 [GBJ08]	0	La fel
"Nitrogen depletion"	4, 3, 2, 1, 5 6, 7, 8, 9, 10	2	< 2	4, 3, 2, 1, 5, 6, 7, 8, 9, 10 [GBJ08]	2	Same
"Diauxic shift"	1, 2, 3, 4, 5, 6, 7	0	< 2	1, 2, 3, 4, 5, 6, 7 [GBJ08]	0	La fel
"α factor-based synchronization of the *Saccharomyces cerevisiae* yeast cells" [SSZ+98]						
	1, 2, 3, 4, 5, 6, 7, 8, 9, 17, 14, 15, 16, 18, 10, 11, 12, 13	5	~ 5	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 18, 17, 16, 15 14, 13, 12, 11 [SSZ+98]	2	Nu
Răspunsul celulelor umane la infecția cu *Listeria monocytogenes* [BVBT02]						
"Wild type 1"	1, 2, 3, 4, 5, 6	0	< 2	1, 2, 3, 4, 5, 6 [GBJ08]	0	La fel
"Wild type 2"	1, 2, 3, 4, 5, 6	0	< 2	3, 2, 1, 5, 4, 6[GBJ08]	4	Da
"Mutant 1"	1, 4, 2, 3, 5, 6	2	< 2	1, 3, 2, 6, 5, 4 [GBJ08]	4	Da
"Mutant 2"	1, 2, 3, 4, 5, 6	0	< 2	1, 2, 3, 4, 6, 5 [GBJ08]	2	Da

Tabela 4.1: Rezultate obținute de algoritmul RL pentru seriile temporale. Pentru fiecare set de date, prezentăm soluția obținută de algoritmul nostru RL, valoarea măsurii de evaluare SMD a soluției, timpul de execuție (în secunde), precum și alte ordini temporale obținute în literatura de specialitate pentru aceleași seturi de date și valorile corespunzătoare ale măsurii de evaluare. Ultima coloană a tabelului (Îmbunătățiri) specifică dacă metoda noastră conduce la soluții mai bune (fie ordini corecte, fie ordini cu valori mai mici ale măsurii de evaluare) în comparație cu cele care au fost deja raportate în literatura de specialitate.

4.2.2.1 Serii Temporale conținând Date de Expresie Genică

Pentru a ne testa metoda pe date cu ordini temporale cunoscute, am folosit mai multe serii temporale. O serie temporală este o colecție de date rezultate dintr-un anumit tip de experiment biologic: sunt extrase de la același individ probe conținând țesuturi la momente cunoscute în timpul progresiei unui proces biologic. Seturile de date utilizate în aceste experimente sunt serii temporale conținând date biologice de la un organism numit drojdie, precum și din țesuturi umane.

Pentru a putea compara rezultatele noastre cu alte rezultatele prezentate în literatura de specialitate pentru aceleași seturi de date, precum și pentru a cuantifica performanța algoritmului nostru RL, introducem o măsură de evaluare care estimează calitatea unei soluții obținute pentru un set de date, în funcție de ordinea corectă, care se cunoaște. Definim o măsură numită SMD (*Samples Misplacement Degree*), care, în opinia noastră, exprimă cât de corect au fost plasate probele într-o anumită ordine. Prin definiția sa, măsura SMD sancționează soluțiile în care probele nu sunt în poziția corectă în ordine, cu privire la probele învecinate.

Agentul este antrenat folosind modelul determinării unui drum, cu algorimul Q-learning și cu mecanismul inteligent de selecție a acțiunilor utilizat cu $\epsilon = 0.8$. Numărul de episoade de antrenare este de 13000. Soluțiile raportate în fiecare caz, după finalizarea antrenării agentului sunt ordinile temporale valide și optime.

Pentru fiecare serie temporală considerată, Tabelul 4.1 [CBC13] prezintă rezultatele obținute. Menționăm că pentru fiecare din cele zece seturi de date, ordinile corecte sunt cele începând cu prima probă (extrasă în primul moment) și în care cresc consecutiv până la proba care a fost prelevată ultima. Se poate observa că algoritmul nostru a obținut ordinile corecte pentru *șapte* din cele zece seturi de date. În ceea ce privește timpul de calcul, pentru seturile de date mai mici (cele care conțin mai puțin, sau exact 10 probe), algoritmul RL obține foarte rapid soluțiile, în mai puțin de 2 secunde, pe un PC cu 3 GHz și 4 GB RAM, în toate cele nouă cazuri (Table 4.1 [CBC13]). Timpul de execuție al algorimului nostru pentru seria compusă din 18 probe [SSZ+98] a fost și el destul de mic (~ 5 secunde).

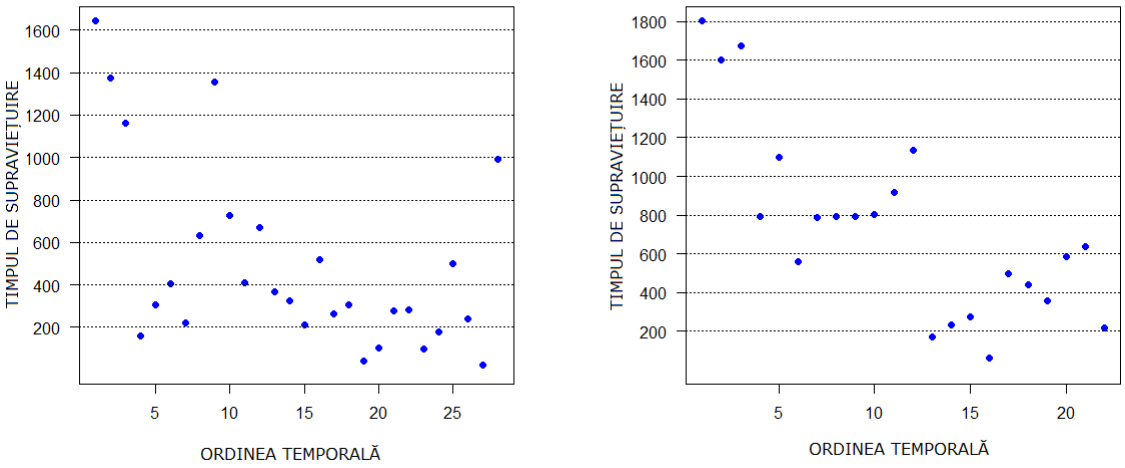


Figura 4.3: Ordinile obținute în raport cu timpii de supraviețuire pentru setul de date asociate gliomurilor. Figura din stângă corespunde setului de glioblastome, iar ce-a din dreapta ilustrează rezultatele pentru setul de oligodendrogliome anaplastice.

4.2.2.2 Date de Expresie Genică Asociate Cancerului

Am testat algoritmul nostru RL pe un set de date de expresie genică asociate cancerului [NMB+03], constând în probe prelevate de la pacienți suferind de gliom: 28 de glioblastome și 22 oligodendrogliome anaplastice (anaplastic oligodendrogliomas). Pentru fiecare probă din aceste seturi avem nivelurile expresiei genice a 12625 de gene, precum și timpul de supraviețuire al pacienților după diagnosticul inițial [NMB+03]. Pentru fiecare din cele două seturi am aplicat pasul de preprocesare. În urma acestui pas, dimensionalitatea datelor de intrare a fost redusă semnificativ: pentru glioblastom au rămas 28 de atribute (gene), iar pentru oligodendrogliom anaplastic au rămas 41 de atribute.

Agentul a fost antrenat folosind mecanismul inteligent de selecție a acțiunilor utilizat cu $\epsilon = 0.8$, de-a lungul a $3 \cdot 10^5$ episoade. Figura 4.3 [CBC13] prezintă soluțiile obținute și indică corelația dintre ordini și timpii de supraviețuire ai pacienților. Ordinile sunt ilustrate astfel încât prima probă este întotdeauna în partea stângă și ultima în dreapta. Se poate observa că în ambele cazuri ordinile sunt, într-o anumită măsură, bine corelate cu timpul de supraviețuire: probele din partea dreaptă a fiecărui grafic aparțin pacienților a căror rata de supraviețuire este mai mică, iar cele din jumătatea stângă aparțin pacienților cu timpi mai mari de supraviețuire.

4.2.3 Variații ale abordării RL

Folosind una dintre seriile temporale prezentate în Subsecțiunea 4.2.2.1, am evaluat experimental alți doi algoritmi de tip Q-learning, care folosesc urme de eligibilitate (Secțiunea 3.1.3): $Q(\lambda)$ [Wat89] și *naive* $Q(\lambda)$ [SB98]. Pentru teste am utilizat două tipuri de politici de selecție a acțiunilor și anume mecanismul inteligent (procedura look-ahead) - Secțiunea 3.2.4 și politica softmax - Secțiunea 3.1.2).

Algoritmii sunt comparați examinând precizia ordinilor obținute, numărul de episoade de care au nevoie pentru a converge înspre soluție, precum și timpul de execuție. Algorimul tradițional Q-learning, fără urme de eligibiltate dovedește a avea o performanță foarte bună, obținând soluția foarte rapid (mai puțin de 2 secunde), atunci când este utilizat cu mecanismul inteligent de selecție a acțiunilor. Politica softmax conduce, de asemenea, către ordinea corectă, dar în acest caz procesul de convergență este mai lent. De îndată ce sunt introduse urmele de eligibilitate, comportamentul algoritmului Q-learning se modifcă radical: pentru anumite valori ale parametrilor politicilor convergența nu este atinsă deloc, în timp ce pentru alte valori se obține ordinea corectă, dar în perioade îndelungate de timp. O posibilă explicație pentru acest comportament ar fi faptul că în reprezentarea noastră a mediului nu se cunoaște niciodată complet mulțimea stărilor și atunci urmele de eligibilitate pot fi actualizate numai pentru o submulțime cunoscută a spațiului. Concluzionăm astfel că, pentru problema TO, algoritmul Q-learning care nu folosește deloc urme de eligibilitate este mai adecvat.

4.2.4 Discuție

În acest subcapitol am abordat problema TO prin adaptarea modelului RL care determină un drum optim (Secțiunea 3.2.1). Pentru evaluarea experimentală a abordării noastre, am selectat în primul rând o serie de seturi de date care au fost deja utilizate în literatura de specialitate [GBJ08, MLK03]. Rezultatele pe care le-am obținut sunt comparabile și în unele cazuri, chiar mai bune decât rezultatele care au fost raportate în literatură până în prezent. În al doilea rând, am evaluat metoda noastră RL pe două seturi, conținând date de expresie genică asociate cancerului: unul format din 28 de probe prelevate de la pacienți suferind de glioblastom si al doilea conținând 22 de probe ale unor pacienți suferind de oligodendrogliom anaplastic [NMB+03]. Intuitiv, bazându-ne pe corelația între progresul bolii și timpul total de supraviețuire a pacienților, am ales timpul de supraviețuire ca măsură de validare pentru ordinile temporale obținute. Deși pentru datele asociate oligodendrogliomului anaplastic corelația este mai puternică, putem afirma că, în ambele cazuri, ordinile obținute sunt bine corelate cu timpul de supraviețuire.

Soluția problemei ordonării temporale a datelor biologice poate fi folosită pentru un dublu scop. Pe de o parte o ordonare temporală a datelor asociate unui proces biologic ar putea oferi noi perspective despre dezvoltarea acestui proces. Pe de altă parte, abordarea noastră ar putea fi, de asemenea, folosită pentru a determina pozițiile corecte ale unor probe noi într-un set de date ordonat. O aplicație practică este în domeniul de cercetare al cancerului. Presupunem că avem deja un set ordonat de pacienți, pentru care se știu și timpii de supraviețuire. În momentul în care apar noi pacienți, pentru care nu se cunoaște timpul de supraviețuire, poziționarea corectă a acestora în șirul ordonat existent ar putea dezvălui informații importante cu privire la speranța lor de viață.

În ceea ce privește dezavantajele abordării noastre, menționăm că singura modalitate de reducere a zgomotului este etapa de preprocesare, care utilizează cunoștințe a priori asupra problemei. Un alt dezavantaj ar putea fi faptul că este nevoie de un număr mare de episoade de antrenare pentru probleme cu multe instanțe, pentru a obține rezultate precise. Dar, după cum au arătat rezultatele experimentale, procesul de convergență poate fi accelerat folosind mecanisme performante de căutare locală. Credem că direcția folosirii de tehnici RL în rezolvarea acestei probleme merită să fie în continuare studiată, iar îmbunătățiri suplimentare pot duce către rezultate mai valoroase.

4.3 Framework de Programare folosind Învățarea prin Întărire

În acest subcapitol prezentăm o interfață de programare pentru rezolvarea problemelor de optimizare combinatorială folosind tehnici RL. Interfața a fost introdusă în publicațiile originale [CCB11a, CCB11b]. Ea a fost dezvoltată în principal pentru a oferi o modalitate simplă și rapidă de a evalua experimental modelele RL propuse în Subcapitolul 3.2. Cu toate acestea, interfața a fost concepută suficient de generic pentru a fi utilizată în dezvoltarea de aplicații pentru probleme de optimizare.

Propunem o interfață de programare care permite dezvoltarea cu ușurință a unor aplicații pentru rezolvarea problemelor de optimizare combinatorială folosind tehnici RL. În framewok-ul introdus facem abstracție de modul în care problema de optimizare este modelată pentru a putea fi abordată cu metode RL. Acesta este avantajul major al interfeței noastre RL: algoritmul RL este definit independent de modul cum sunt definite mediul, stările și acțiunile. Prin urmare, un utilizator care dorește să rezolve o problemă de optimizare specifică trebuie doar să definească funcțiile de tranziție și recompensă, precum și spațiul stărilor și al acțiunilor, deoarece restul componentelor RL sunt deja definite și pot fi utilizate.

Interfața a fost implementată folosind JDK (Java Development Kit) 1.6 și are patru module de bază: agent, mediu, RL, și simulare. Ca în orice sistem bazat pe agenți [Wei99], *agentul* este entitatea care interacționează cu *mediul*, care primește percepții și selectează *acțiuni*. Agentul învață să își atingă obiectivul, mai exact el învață să determine o soluție optimă a problemei, folosind RL. În general, datele de intrare ale agentului sunt percepțiile privind mediul (în cazul nostru, stările mediului), iar datele de ieșire sunt acțiunile. Mediul oferă recompense în urma interacțiunilor cu agentul. Interacțiunea dintre agent și mediu este controlată de o entitate numită *simulare*.

Figura 4.4 [CCB11a] ilustrează o diagramă UML [Gro13] simplificată a interfeței, în care putem observa nucleul interfeței RL. Este important fapul că toate clasele din interfață rămân neschimbate în toate aplicațiile de rezolvare a problemelor de optimizare combinatoriale modelate folosind RL.

Framework-ul RL a fost evaluat experimental aplicându-l pentru rezolvarea celor trei probleme de optimizare din bioinformatică, ce au fost modelate folosind tehnici RL: ordonarea temporală a datelor biologice (Secțiunea 4.2.2), asamblarea fragmentelor ADN (Secțiunea 3.3.2) și predicția structurii

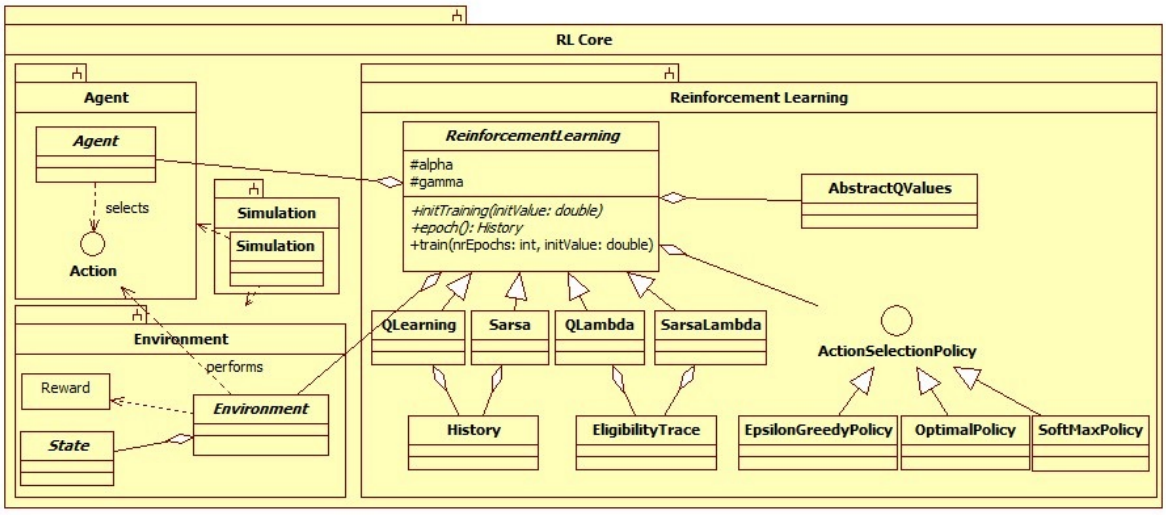


Figura 4.4: Diagrama UML a interfeței de programare bazată pe tehnici de învățare prin întărire.

terțiare a proteinelor (Secțiunea 3.4.2). Framework-ul RL poate fi ușor folosit pentru a dezvolta aplicații pentru rezolvarea acestor probleme. După selectarea modelelor conceptuale pentru spațiile stărilor și acțiunilor și după alegerea unui algoritm RL și a politicii de selecție a acțiunilor, tot ce trebuie să facă utilizatorul este să dezvolte clase noi, care extind clasele abstracte definite în interfață.

4.4 Concluzii și Cercetări Ulterioare

Am introdus în acest capitol două abordări diferite pentru problema ordonării temporale a unor date biologice: prima propune o metodologie pentru obținerea unor ordini temporale folosind date de tip CNA, iar a doua folosește un model RL propus în capitolul anterior. Am propus și o nouă interfață de programare pentru rezolvarea problemelor de optimizare folosind tehnici RL. Metodele prezentate în acest capitol au fost publicate în [CBC13, Boc12a, BCG+12, BCG+13, CCB11a, CCB11b].

Prima abordare propune o nouă metodologie care permite aplicarea unei soluții propuse anterior pentru date de expresia genică [GBJ08] la un set de date asociate cancerului colorectal, care conține date de tip CNA. Sunt definite mai multe măsuri cromozomiale și anumite filtre care vizează porțiuni semnificative de cromozomi. Experimentele sunt realizate pe un set de date prelevate de la pacienți afectați de cancer colorectal, în diferite etape de progresie a bolii. În ceea ce privește cercetările ulterioare, vom aborda problema zgomotului în date, vom extinde evaluarea și pe alte seturi de date CNA reale și vom investiga modul în care alte măsuri cromozomiale ar putea influența rezultatele.

A doua metodă pe care am propus-o abordează problema TO prin aplicarea unui model RL care determină un drum optim. Pentru evaluarea experimentală am folosit mai multe seturi de date de expresie genică (serii de timp conținând celule umane și ale organismului drojdie, precum și seturi conținând date asociate cancerului). Au fost aplicați mai mulți algorimti Q-learning, folosind diferite mecanisme de selecție a acțiunilor, diferite tipuri de urme de eligibilitate și diferite valori ale parametrilor algoritmilor. Performanța bună a modelului RL ne conduce la concluzia că astfel de modele de învățare sunt capabile să detecteze anumite tipare în datele de intrare care variază de-a lungul timpului. Vom investiga posibile îmbunătățiri ale modelului RL prin: folosirea altor funcții recompensă, adăugarea unor diverse mecanisme de căutare locală, folosirea de metode de aproximare a funcțiilor, pentru cazurile în care spațiul de stări este foarte mare, considerarea modificării parametrului din strategia ϵ-Greedy pe parcursul procesului de învățare. Vom considera, de asemenea, o extindere a modelului RL pentru problema TO înspre o abordare RL distribuită (Secțiunea 3.2.1).

În ceea ce privește framework-ul de programare care folosește tehnici RL introdus în acest capitol, remarcăm că acesta este general și a fost conceput pentru a facilita cercetarea în direcția de rezolvare a problemelor de optimizare combinatorială folosind tehnici RL. Cercetări ulterioare vor fi întreprinse pentru a investiga alte modele conceptuale pentru spațiile stărilor și al acțiunilor în cazul problemelor de optimizare din bioinformatică pe care le-am studiat până acum. Dorim să extindem evaluarea framework-ului propus și pentru alte probleme de optimizare combinatorială.

Concluzii

Principalele obiective ale activității de cercetare în științele biologiei moderne sunt înțelegerea modului de funcționare a organismelor, proceselor celulare și căilor metabolice, cu scopul de a recunoaște de ce și în ce fel apar disfuncționalități în aceste procese. Un prim pas este analiza și extragerea de cunoștințe din imensele cantități de date biologice și medicale. În acest scop, informatica oferă metodele, instrumentele și algoritmii necesari. Bioinformatica s-a dovedit astfel a fi o nouă disciplină importantă în era post-genomică.

Cercetările prezentate în această teză au avut drept scop găsirea de soluții pentru mai multe probleme dificile din bioinformatică, folosind modele bazate pe instruirea automată. Ne-am concentrat în special pe două direcții principale de cercetare. Prima este aplicarea *regulilor de asociere relaționale* pentru a rezolva problemele de clasificare din bioinformatică, iar modelul propus a fost folosit pentru a aborda problema *predicției regiunilor promotor* în molecule ADN. Cea de a doua direcție se referă la aplicarea unor *tehnici bazate pe învățarea prin întărire* în scopul de a rezolva problemele NP-complete de optimizare combinatorială din bioinformatică. Modelele de învățare propuse au fost aplicate pe trei probleme importante: *asamblarea fragmentelor ADN*, *predicția structurii terțiare a proteinelor* și *ordonarea temporală a datelor biologice*. În plus față de aceste două direcții de cercetare primare am prezentat o metodologie nouă care vizează o problemă specifică în bioinformatică și un anumit tip de date biologice, dezvoltată în colaborare cu grupul de cercetare BIMIB de la Universitatea Milano-Bicocca. În cele din urmă, am prezentat contribuțiile noastre originale la dezvoltarea sistemelor software prin introducerea unei interfațe de programare pentru rezolvarea problemelor de optimizare folosind tehnici de învățare prin întărire.

Performanțele modelului de clasificare bazat pe reguli de asociere relaționale pentru predicția regiunilor promotor în ADN ne conduc înspre concluzia că modelele de instruire automată și tehnicile de extragere inteligentă a datelor sunt instrumente importante capabile să recunoască tipare în date biologice, care sunt greu de identificat folosind tehnici convenționale.

Am propus trei modele care folosesc tehnici de învățare prin întărire pentru un anumit tip de probleme de optimizare combinatorială, care pot fi modelate ca probleme de identificare a unor permutări generalizate. Aceste modele au fost modificate în mod corespunzător, adaptate și aplicate pe trei probleme în bioinformatică. Evaluările experimentale au fost efectuate pe seturi de date reale din domeniul problemelor considerate și rezultatele obținute s-au dovedit a fi performante, demonstrând astfel potențialul propunerilor noastre.

Am oferit comparații cu abordări similare din literatura de specialitate pentru modelele pe care le-am propus. În multe cazuri, abordările noastre originale au surclasat metode similare, evidențiind astfel eficiența modelelor noastre. În plus, pentru toate cazurile în care sunt propuse diferite modele (care se bazează pe aceeași idee), oferim comparații și analize ale acestora.

Framework-ul de programare care folosește tehnici de învățare prin întărire introdus în această teză a fost conceput pentru a facilita cercetarea în direcția de rezolvare a problemelor de optimizare combinatorială. Genericitatea acestuia permite dezvoltarea simplă a aplicațiilor pentru rezolvarea problemelor de optimizare cu ajutorul învățării prin întărire. Am folosit acest framework pentru a dezvolta aplicații ce abordează toate cele trei probleme din bioinformatică pe care le-am modelat folosind învățarea prin întărire.

În ceea ce privește direcții ulterioare de cercetare, dorim să investigăm posibile îmbunătățiri ale abordările propuse, pentru a extinde evaluarea lor folosind diferite seturi de date din domeniul problemelor considerate. Vom aplica versiunile fuzzy a abordărilor propuse (unde este posibil) și ne vom concentra și asupra hibridizării modelelor noastre prin combinarea lor cu alte tehnici bazate pe instruirea automată. Mai mult vom aborda noi probleme semnificative din bioinformatică, fie prin utilizarea modelelor deja propuse fie prin dezvoltarea de modele noi.

Cuvinte cheie

- bioinformatica
- biologie computațională
- instruire automată
- predicția regiunilor promotor
- reguli de asociere
- învățare prin întărire
- Q-învățare
- asamblarea fragmentelor ADN
- predicția structurii proteinelor
- ordonare temporală
- alterări cromozomiale
- expresie genică
- date de tip microarray
- optimizare combinatorială
- framework de programare

Bibliografie

[AAdFG99] E. Angeleri, B. Apolloni, D. de Falco, and L. Grandi. DNA Fragment Assembly Using Neural Prediction Techniques. *International Journal of Neural Systems*, 9(6):523–544, 1999.

[ACB+10] C.S. Attolini, Y.K. Cheng, R. Beroukhim, G. Getz, O. Abdel-Wahab, R.L. Levine, and F. Michor. A mathematical framework to determine the temporal sequence of somatic genetic events in cancer. *Proc Natl Acad Sci USA*, 107:17604—-17609, 2010.

[AHO03] C. Arima, T. Hanai, and M. Okamoto. Gene Expression Analysis Using Fuzzy K-Means Clustering. *Genome Informatics*, 14:334–335, 2003.

[Anf73] C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[ASD+10] H. Ashktorab, A.A. Schäffer, M. Daremipouran, D.T. Smoot, E. Lee, and H. Brim. Distinct genetic alterations in colorectal cancer. *PLoS ONE*, 5(1):e8879, 2010.

[BCC11a] Maria Iuliana Bocicor, Gabriela Czibula, and Istvan Gergely Czibula. A distributed Q-learning approach to fragment assembly. *Studies in Informatics and Control*, 20(3):221–232, 2011.

[BCC11b] Maria Iuliana Bocicor, Gabriela Czibula, and Istvan Gergely Czibula. A reinforcement learning approach for solving the fragment assembly problem. In *In Proceedings of the 3th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC '11)*, pages 191–198. IEEE Computer Society, 2011.

[BCG+12] Iuliana M. Bocicor, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri, and Marco Antoniotti. Ordering Copy Number Alteration Data to Analyze Colorectal Cancer Progression. *EMBnet.journal*, 18(Suppl. B (NETTAB 2012)):84–86, 2012.

[BCG+13] Iuliana M. Bocicor, Giulio Caravagna, Alex Graudenzi, Claudia Cava, Giancarlo Mauri, and Marco Antoniotti. Reconstructing Colorectal Cancer Progression through Copy Number Alteration Data. *Complex Systems Models in Biology and Medicine: Generic Properties and Applications*, 2013. (Will be submitted).

[BJK+05a] N. Beerenwinkel, Rahnenfuhrer J., R. Kaiser, D. Hoffmann, J. Selbig, and T. Lengauer. Learning multiple evolutionary pathways from cross-sectional data. *J Comput Biol*, 12(6):584–598, 2005.

[BJK+05b] N. Beerenwinkel, Rahnenfuhrer J., R. Kaiser, D. Hoffmann, J. Selbig, and T. Lengauer. Mtreemix: a software package for learning and using mixture models of mutagenetic trees. *Bioinformatics*, 21(9):2106–2207, 2005.

[BL98] B. Berger and T. Leighton. Protein folding in hp model is np-complete. *Journal of Computational Biology*, 5:27–40, 1998.

[Boc10a] Maria Iuliana Bocicor. Algoritmi evolutivi aplicaţi în chemoterapie. In *National Symposum "Interferente", 1st Edition*, pages 145–148. Ceconi Baia Mare, 2010.

[Boc10b] Maria Iuliana Bocicor. Bioinformatica si aplicatiile ei. *Scoala Maramureseana*, (41-45):222–223, 2010.

[Boc11a] Maria Iuliana Bocicor. Invatarea automata pentru identificarea regiunilor promotor in adn. In *National Symposum "Interferente", 3rd Edition*, pages 68–70. Universitatea de Nord Baia Mare, 2011.

[Boc11b] Maria Iuliana Bocicor. Modele pentru problema plierii proteinei. In *National Symposum "Interferente", 2nd Edition*, pages 191–193. Universitatea de Nord Baia Mare, 2011.

[Boc12a] Iuliana M. Bocicor. A Study on Using Reinforcement Learning for Temporal Ordering of Biological Samples. *Studia Universitatis "Babes-Bolyai" Informatica*, LVII(4):62–74, 2012.

[Boc12b] Maria Iuliana Bocicor. Experiments on promoter sequences prediction using association rules. In *In Proceedings "Zilele Academice Clujene 2012, Departamentul de Informatica"*, pages 32–35. Presa Universitara Clujeana, 2012.

[Boc12c] Maria Iuliana Bocicor. A study on using association rules for predicting promoter sequences. *Studia Universitatis "Babes-Bolyai", Informatica*, LVII(2):32–42, 2012.

[BPSS01] A. Brazma, H. Parkinson, T. Schlitt, and M. Shojatalab. A quick introduction to elements of biology - cells, molecules, genes, functional genomics, microarrays. `http://www.ebi.ac.uk/microarray/biology_intro.html`, 2001. Accessed: 30 August 2010.

[BS06] E. Bindewald and B.A. SHAPIRO. RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers. *RNA*, 12:342–352, 2006.

[BVBT02] David N. Baldwin, Veena Vanchinathan, Patrick O. Brown, and Julie A. Theriot. A gene-expression program reflecting the innate immune response of cultured intestinal epithelial cells to infection by Listeria monocytogenes. *Genome Biology*, 4(1):4241–4257, 2002.

[BWTB08] E. Bolton, Y. Wang, P.A. Thiessen, and S.H. Bryant. PubChem: Integrated Platform of Small Molecules and Biological Activities. In *Annual Reports in Computational Chemistry*, volume 4, chapter 12. American Chemical Society, Washington DC, 2008.

[CBC11a] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. A distributed reinforcement learning approach for solving optimization problems. In *In Proceedings of the 5th International Conference on Communications and Information Technology (CIT '11)*, pages 25–30, 2011.

[CBC11b] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. An experiment on protein structure prediction using reinforcement learning. *Studia Universitatis "Babes-Bolyai", Informatica*, LVI(1):25–34, 2011.

[CBC11c] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. A reinforcement learning model for solving the folding problem. *International Journal of Computer Technology and Applications*, 2(1):171–182, 2011.

[CBC11d] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Solving the protein folding problem using a distributed Q-learning approach. *International Journal of Computers*, 5(3):404–413, 2011.

[CBC12] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Promoter sequences prediction using relational association rule mining. *Evolutionary Bioinformatics*, 8:181–196, 2012.

[CBC13] Gabriela Czibula, Maria Iuliana Bocicor, and Istvan Gergely Czibula. Temporal ordering of cancer microarray data through a reinforcement learning based approach. *PloS ONE*, 8(4):e60883, 2013.

[CBL+12] Y.K. Cheng, R. Beroukhim, R.L. Levine, I.K. Mellinghoff, E.K. Holland, and F. Michor. A Mathematical Methodology for Determining the Temporal Order of Pathway Alterations Arising during Gliomagenesis. *PLOS Computational Biology*, 8(1):1–15, 2012.

[CCB11a] Istvan Gergely Czibula, Gabriela Czibula, and Maria Iuliana Bocicor. A Software Framework for Solving Combinatorial Optimization Tasks. *Studia Universitatis "Babes-Bolyai", Informatica*, Special Issue, LVI(3):3–8, 2011.

[CCB11b] Istvan Gergely Czibula, Gabriela Czibula, and Maria Iuliana Bocicor. A reinforcement learning based framework for solving optimization problems. In *In Post proceedings of Knowledge Egineering Principles and Techniques*, pages 235–246. Presa Universitara Clujeana, 2011.

[CCB13] Gabriela Czibula, Istvan Gergely Czibula, and Maria Iuliana Bocicor. A Comparison of Reinforcement Learning Based Models for the DNA Fragment Assembly Problem. *Studia Universitatis "Babes-Bolyai" Informatica*, LVIII(2):90–102, 2013.

[Chi10] C. Chira. Hill-Climbing Search in Evolutionary Models for Protein Folding Simulations. *Studia*, LV:29–40, 2010.

[CK91] David Chapman and Leslie Pack Kaelbling. Input generalization in delayed reinforcement learning: an algorithm and performance comparisons. In *Proceedings of the 12th international joint conference on Artificial intelligence - Volume 2*, pages 726–731, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

[CL05] Shyi-ming Chen and Chung-hui Lin. Multiple dna sequence alignment based on genetic algorithms and divide-and-conquer techniques. *International Journal of Applied Science and Engineering*, 3:89–100, 2005.

[Coo11] William Cook. Concorde TSP solver, 2011. http://www.tsp.gatech.edu/concorde.html.

[CPCC06] Y. Chen, Y. Pan, L. Chen, and J. Chen. Partitioned optimization algorithms for multiple sequence alignment. *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, pages 618–622, 2006.

[CS94] Mark W. Craven and Jude W. Shavlik. Machine learning approaches to gene recognition. *IEEE Intelligent Systems*, 9(2):2–10, 1994.

[CSS99] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[CSTM06] A. Câmpan, G. Serban, T.M. Truta, and A. Marcus. An Algorithm for the Discovery of Arbitrary Length Ordinal Association Rules. In *DMIN*, pages 107–113, 2006.

[DD01] Chris H. Q. Ding and Inna Dubchak. Multi-class Protein Fold Recognition Using Support Vector Machines and Neural Networks. *Bioinformatics*, 17:349–358, 2001.

[Dil85] K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[DJK+99] Richard Desper, Feng Jiang, Olli P. Kallioniemi, Holger Moch, Christos H. Papadimitriou, and Alejandro A. Schaffer. Inferring Tree Models for Oncogenesis from Comparative Genome Hybridization Data. *Journal of Computational Biology*, 6(1):37–51, 1999.

[DJK+00] Richard Desper, Feng Jiang, Olli P. Kallioniemi, Holger Moch, Christos H. Papadimitriou, and Alejandro A. Schaffer. Distance-based reconstruction of tree models for oncogenesis. *J Comput Biol*, 7(6):789—-803, 2000.

[EB96] M.L. Engle and C. Burks. Artificially generated data sets for testing dna fragment assembly algorithms. *Genomics*, 16(1):286–288, 1996.

[FA10] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Fog05] G.B. Fogel. Gene expression analysis using methods of computational intelligence. *Pharmaceutical Discovery*, 5:12–18, 2005.

[Fra07] S.A. Frank. *Dynamics of Cancer*. Princeton University Press, 2007.

[FV90] E.R. Fearon and B. Volgestein. A genetic model for colorectal tumorigenesis. *Cell*, 61:759–767, 1990.

[GBF+07] Thiago Gonzaga, Cristina Bentes, Ricardo Farias, Maria Clicia Castro, and Ana Cristina Garcia. Using distributed-shared memory mechanisms for agents communication in a distributed system. In *Proceedings of the Seventh International Conference on Intelligent Systems Design and Applications*, pages 39–46, Washington, DC, USA, 2007. IEEE Computer Society.

[GBHB09] M. Gerstung, M. Baudis, Moch. H, and N. Beerenwinkel. Quantifying cancer progression with conjunctive Bayesian networks. *Bioinformatics*, 25(21):2809–2815, 2009.

[GBJ08] Anupam Gupta and Ziv Bar-Joseph. Extracting Dynamics from Static cancer Expression Data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(2):172–182, June 2008.

[GEL+11] Moritz Gerstung, Nicholas Eriksson, Jimmy Lin, Bert Volgestein, and Niko Beerenwinkel. The Temporal Order of Genetic and Pathway Alterations in Tumorigenesis. *PLoS ONE*, 6(11):1–9, 2011.

[GP08] Ashish Ghosh and Bijnan Parai. Protein secondary structure prediction using distance based classifiers. *International Journal of Approximate Reasoning*, 47:37–44, 2008.

[Gro13] Object Management Group. UML Resource Page. `http://uml.org/`, 2013. Accessed: 10 May 2013.

[GSK+00] Audrey P. Gasch, Paul T. Spellman, Camilla M. Kao, Orna Carmel-Harel, Michael B. Eisen, Gisela Storz, David Botstein, and Patrick O. Brown. Genomic Expression Programs in the Response of Yeast Cells to Environmental Changes. *Molecular Biology of the Cell*, 11(12):4241–4257, 2000.

[HCP+00] K.B. Hwang, D.Y. Cho, Wook Park, Kim S.W., Zhang S.D., and B.Y. Applying machine learning techniques to analysis of gene expression data: Cancer diagnosis. *In: Proc. 1st Conf. on Critical Assessment of Microarray Data Analysis*, 2000.

[HHL06] M. Hjelm, M. Hoglund, and J. Lagergren. New probabilistic network models and algorithms for oncogenesis. *Journal of Computtational Biology*, 13(4):853–865, 2006.

[HL03] C. J. Huang and W.-C. Liao. A comparative study of feature selection methods for probabilistic neural networks in cancer classification. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 451–458, 2003.

[HMTA08] A.E. Hassanien, M.G. Milanova, Smolinski T.G., and Abraham A. Computational Intelligence in Solving Bioinformatics Problems: Reviews, Perspectives, and Challenges. *Computational Intelligence in Biomedicine and Bioinformatics Studies in Computational Intelligence*, 151:3–47, 2008.

[HYYY02] D. N. Hung, I. Yoshihara, K. Yamamori, and M. Yasunaga. A parallel hybrid genetic algorithm for multiple protein sequence alignment. *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 309–314, 2002.

[JDH99] T. Jaakkola, M Diekhans, and D Haussler. Using the sher kernel method to detect remote protein homologies. *In ISMB*, pages 149–158, 1999.

[JSXW10] A. Jemal, R. Siegel, J. Xu, and E. Ward. Cancer statistics 2010. *CA Cancer J. Clin.*, 60:277–300, 2010.

[KC06] Satoko Kikuchi and Goutam Chakraborty. Heuristically Tuned GA to Solve Genome Fragment Assembly Problem. *IEEE CEC*, pages 1491–1498, 2006.

[Klo72] A.H. Klopf. Brain function and adaptive systems. A heterostatic theory. *Technical Report AFCRL-72-0164*, 1972.

[Kos07] Walter Kosters. Bioinformatics: Fragment Assembly. *IPA–Algorithms and Complexity - course*, 2007.

[KP04] Nikola Kasabov and Shaoning Pang. Transductive support vector machines and applications in bioinformatics for promoter recognition. *Neural Information Processing - Letters and Reviews*, 3(2):31–37, 2004.

[LAR+10] Geer L.Y., Marchler-Bauer A., Geer R.C., Han L., He J., He S., Liu C., Shi W., and Bryant S.H. The ncbi biosystems database. *Nucleic Acids Research*, (38(Database issue)), 2010.

[LATK06] G. Luque, E. Alba Torres, and S. Khuri. Assembling DNA Fragments with a Distributed Genetic Algorithm. *Parallel Computing for Bioinformatics and Computational Biology*, pages 285–302, 2006.

[LEN02] Christina Leslie, Eleazar Eskin, and William Stafford S. Noble. The spectrum kernel: a string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[LHJYFHB04] Wang Long-Hui, Liu Juan, Li Yan-Fu, and Zhou Huai-Bei. Predicting protein secondary structure by a support vector machine based on a new coding scheme. *Genome Informatics*, 15:181–190, 2004.

[LK04] Lishan Li and Sami Khuri. A comparison of DNA fragment assembly algorithms. In *Proceedings of the International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences*, pages 329–335. CSREA Press, 2004.

[MC03] P. Meksangsouy and N. Chaiyaratana. DNA fragment assembly using an ant colony system algorithm. In *Proceedings of CEC'03 - vol.3*, pages 1756–1763. IEEE Press, 2003.

[MGR06] J. Martin, J. F. Gibrat, and F. Rodolphe. Analysis of an optimal hidden markov model for secondary structure prediction. *BMC Structural Biology*, 6:25–45, 2006.

[Mit97] T.M. Mitchell. *Machine Learning.* New York: McGraw-Hill, 1997.

[MLK03] Paul M. Magwene, Paul Lizardi, and Junhyong Kim. Reconstructing the temporal ordering of biological samples using microarray data. *Bioinformatics*, 19(7):842–850, 2003.

[MML01] A. Marcus, J.I. Maletic, and K.I. Lin. Ordinal association rules for error identification in data sets. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 589–591, New York, NY, USA, 2001. ACM.

[Net12] The Cancer Genome Atlas Network. Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330–337, 2012.

[NMB+03] Catherine L. Nutt, D. R. Mani, Rebecca A. Betensky, Pablo Tamayo, J. Gregory Cairncross, Christine Ladd, Ute Pohl, Christian Hartmann, Margaret E. McLaughlin, Tracy T. Batchelor, Peter M. Black, Andreas von Deimling, Scott L. Pomeroy, Todd R. Golub, and David N. Louis. Gene Expression-based Classification of Malignant Gliomas Correlates Better with Survival than Histological Classification. *Cancer Research*, 63(7):1602–1607, 2003.

[NT88] Qian Ning and Sejnowski Terrence. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202:865–884, 1988.

[NVNM07] S. Nasser, G.L. Vert, M. Nicolescu, and A. Murray. Multiple sequence alignment using fuzzy logic. *Proceedings IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, pages 304–311, 2007.

[OP09] O. Okun and H Priisalu. Dataset complexity in gene expression based cancer classification using ensembles of k-nearest neighbors. *Artificial Intelligence in Medicine*, 45(2-3):151–162, 2009.

[PE95] Anders Gorm Pedersen and Jacob Engelbrecht. Investigations of Escherichia coli Promoter Sequences With Artificial Neural Networks: New Signals Discovered Upstream of the Transcriptional Startpoint. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 3:292–299, 1995.

[Pev00] Pavel A. Pevzner. Computational molecular biology: An algorithmic approach. 2000.

[PFB95] Rebecca J. Parsons, Stephanie Forrest, and Christian Burks. Genetic Algorithms, Operators, and DNA Fragment Assembly. In *Machine Learning*, pages 11–33. Kluwer Academic Publishers, 1995.

[PPN09] Amiya Kumar Patel, Seema Patel, and Pradeep Kumar Naik. Binary Classification of uncharacterized uncharacterized proteins int DNA binding/non-DNA binding proteins from sequence derived features using ANN. *Digest Journal of Nanomaterials and Biostructures*, 4:775–782, 2009.

[PS94] Dayan P. and T.J. Sejnowski. Td(lambda) converges with probability 1. *Machine Learning*, 14:295–301, 1994.

[PSBM09] S. Pathare, A. Schaffer, N. Beerenwinkel, and M Mahimkar. Construction of oncogenetic tree models reveals multiple pathways of oral cancer progression. *International Journal of Cancer*, 124(12):2864–2871, 2009.

[PU98] A. Perez-Uribe. Introduction to reinforcement learning, 1998. http://lslwww.epfl.ch/∼anperez/RL/RL.html.

[RGS+09] J.F. Reid, M. Gariboldi, V. Sokolova, P. Capoblanco, A. Lampis, F. Perrone, S. Signoroni, A. Costa, E. Leo, S. Pilotti, and M.A. Pierotti. Integrative Approach for Prioritizing Cancer Genes in Sporadic Colon Cancer. *Genes, Chromosomes and Cancer*, 48:953–962, 2009.

[RK03] T.K. Rasmussen and T Krink. Improved hidden markov model training for multiple sequence alignment by a particle swarm optimization-evolutionary algorithm hybrid. *BioSystems*, 72:5–17, 2003.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[SCC06] G. Serban, A. Câmpan, and I.G. Czibula. A Programming Interface for Finding Relational Association Rules. *International Journal of Computers, Communications & Control*, I(S.):439–444, 2006.

[SH05] A. Shmygelska and H.H. Hoos. An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6, 2005.

[Spe04] C. Spearman. The proof and measurement of association between two things. *Amer. J. Psychol.**15***, pages 72–101, 1904.

[SS96] S. P. Singh and R. S. Sutton. Reinforcement Learning with Replacing Eligibility Traces. *Machine Learning*, 22:123–158, 1996.

[SS05] M. Sevaux and K. Sorensen. Permutation distance measures for memetic algorithms with population management. In *Proceedings of the The Sixth Metaheuristics International Conference*, MIC'05, 2005.

[SSZ+98] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive Identification of Cell Cycle–regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization. *Mol. Biol. Cell*, 9(12):3273–3297, 1998.

[Ste93] Evan W. Steeg. Neural networks, adaptive optimization, and RNA secondary structure prediction. In *In Artificial Intelligence and Molecular Biology*, pages 121–160, 1993.

[Thr92] Sebastian Thrun. The Role of Exploration in Learning Control. In *Handbook for Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold, Florence, Kentucky, 1992.

[TMM08] T. Thalheim, D. Merkle, and M. Middendorf. Protein Folding in the HP-Model Solved With a Hybrid Population Based ACO Algorithm. *IAENG International Jurnal of Computer Science*, 35, 2008.

[TPAG11] Uma Devi Tatavarthi, Venkata Nageswara Rao Padmanbhuni, Appa Rao Allam, and Ramachandra Sridhar Gumpeny. In silico promoter prediction using grey relational analysis. *Journal of Theoretical and Applied Information Technology*, 24(2):107–112, 2011.

[TSN90] Geoffrey G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. *In Proceedings of the Eighth National Conference on Artificial Intelligence(AAAI-90)*, pages 861–866, 1990.

[Tuf11] Stephane Tuffery. *Data Mining and Statistics for Decision Making.* John Wiley and Sons, 2011.

[TYA08] N.T. Tung, E. Yang, and I.P Androulakis. Machine Learning Approaches in Promoter Sequence Analysis. *In Machine Learning Research Progress*, 2008.

[UM93] R. Unger and J. Moult. Genetic Algorithms for Protein Folding Simulations. *Journal of Molecular Biology*, 231:75–81, 1993.

[VFH+88] B. Vogelstein, E.R. Fearon, S.R. Hamilton, S.E. Kern, A.C. Preisinger, M. Leppert, Y. Nakamura, R. White, A.M. Smits, and J.L. Bos. Genetic alterations during colorectal-tumor development. *N. Engl. J. Med.*, 319:3526–3535, 1988.

[Wat89] C. J. C. H. Watkins. Learning from Delayed Rewards. *PhD thesis*, 1989.

[WCT06] W. Wetcharaporn, N. Chaiyaratana, and S. Tongsima. DNA Fragment Assembly by Ant Colony and Nearest Neighbour Heuristics. *ARTIFICIAL INTELLIGENCE AND SOFT COMPUTING – ICAISC 2006*, pages 1008–1017, 2006.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[Wei99] Gerhard Weiß. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1999.

[WLD03] D. Wang, N.K. Lee, and T.S. Dillon. Extraction and optimization of fuzzy protein sequences classification rules using GRBF neural networks. *Neural Information Processing - Letters and Reviews*, 1:53–57, 2003.

[XDE+03] X. Xiao, E.R. Dow, R.C. Eberhart, Z.B. Miled, and R.J. Oppelt. Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. *In: Proc. 17th Intl. Symposium on Parallel and Distributed Processing*, 2003.

[YLGW02] Y. Yuhui, C. Lihui, A. Goh, and A. Wong. Clustering gene data via associative clustering neural network. *In: Proc. 9th Intl. Conf. on Information Processing*, pages 2228–2232, 2002.

[YV04] B-Y Yoon and P.P. Vaidyanathan. Hmm with auxiliary memory: a new tool for modeling RNA secondary structures. In *Proc. 38th Asilomar Conference on Signals, Systems, and Computers*, pages 1651–1655, 2004.

[ZCY+11] Wenyu Zhang, Jiajia Chen, Yang Yang, Yifei Tang, Jing Shang, Bairong Shen, and I. King Jordan. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PLoS ONE*, 6(3):e17915, 2011.

[ZWL+09] X. Zhang, T. Wang, H. Luo, Y.J. Yang, Y. Deng, J. Tang, and M. Q. Yang. 3D Protein structure prediction with genetic tabu search algorithm. *BMC Systems Biology*, 4, 2009.