Załącznik 2

Analiza danych genetycznych. Autoreferat.

Spis treści

1 Dane osobowe i przebieg zatrudnienia w jednostkach naukowych

1. Imię i Nazwisko.
   Robert Nowak
2. Posiadane dyplomy, stopnie naukowe – z podaniem nazwy, miejsca i roku ich uzyskania
   - tytuł magistra inżyniera w zakresie Informatyki, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska, 1999 r.,
   - stopień naukowy doktora nauk technicznych w zakresie Informatyki, Wydział Elektroniki i Technik Informacyjnych, Politechnika Warszawska, 2004 r., rozprawa doktorska pt: „Zagadnienia implementacji urządzeń przetwarzających informację przy użyciu cząsteczek DNA".
3. Informacje o dotychczasowym zatrudnieniu w jednostkach naukowych
   - od 2004-11-01 do 2005-09-30 asystent w Instytucie Systemów Elektronicznych Politechniki Warszawskiej, Samodzielna Pracownia Sztucznej Inteligencji,
   - od 2005-10-01 adiunkt w Zakładzie Sztucznej Inteligencji Instytutu Systemów Elektronicznych Politechniki Warszawskiej.

2 Tytuł osiągnięcia naukowego

Jako osiągnięcie naukowe uzyskane po otrzymaniu stopnia doktora, stanowiące znaczny wkład autora w rozwój dyscypliny naukowej, wskazuję cykl publikacji powiązanych tematycznie zatytułowany „Analiza danych genetycznych". W skład osiągnięcia wchodzą publikacje [1–11].

3 Wykaz publikacji stanowiących osiągnięcie naukowe

Publikacje są przedstawione w kolejności pojawiania się ich opisu w referacie.

[1] R. Nowak, "Application to estimate haplotypes for multiallelic present-absent loci," in *Information Technologies in Biomedicine, Advances in Soft Computing*, vol. 47, pp. 357 – 364, Springer, 2008. doi:10.1007/978-3-540-68168-7_40, `http://dx.doi.org/10.1007/978-3-540-68168-7_40`.

[2] R. M. Nowak and R. Ploski, "Nullhap - a versatile application to estimate haplotype frequencies from unphased genotypes in the presence of null alleles," *BMC Bioinformatics*, vol. 9:330, pp. 1–8, 2008. doi:10.1186/1471-2105-9-330, `http://dx.doi.org/10.1186/1471-2105-9-330`.

Moim wkładem było opracowanie modelu matematycznego uwzględniającego nieme warianty, konstrukcja algorytmu, implementacja i testowanie oprogramowania, wykonanie badań dla dostarczonych danych, utworzenie tekstu. Mój udział szacuję na 90%. ***IF=3.781***

[3] R. M. Nowak, A. Wojtowicz-Krawiec, and A. Plucienniczak, "Dnasynth: a computer program for assembly of artificial gene parts in decreasing temperature," *BioMed Research International*, vol. 2015, pp. 1 – 8, 2015. doi:10.1155/2015/413262, `http://dx.doi.org/10.1155/2015/413262`.

Moim wkładem było opracowanie modelu, dobór algorytmów, implementacja, uruchomienie, testowanie, poprawa błędów w kodzie, wykonanie badań dla dostarczonych danych, opracowanie wyników eksperymentu biologicznego, utworzenie tekstu. Mój udział szacuję na 75%. Praca była rozpatrywana przez BMC Bioinformatics, ale została wycofana na naszą prośbę, ponieważ przez 18 miesięcy nie podjęto decyzji. ***IF=2.706***

[4] R. Nowak, "Genome assembler for repetitive sequences," in *Information Technologies in Biomedicine*, vol. 7339 of *Lecture Notes in Computer Science*, pp. 422–429, Springer, 2012. doi:10.1007/978-3-642-31196-3_42, `http://dx.doi.org/10.1007/978-3-642-31196-3_42`.

[5] R. M. Nowak, "Assembly of repetitive regions using next-generation sequencing data," *Biocybernetics and Biomedical Engineering*, 2015. doi:10.1016/j.bbe.2014.12.001, `http://dx.doi.org/10.1016/j.bbe.2014.12.001`.

IF=0.157

[6] R. Nowak and M. Godlewski, "Analiza mieszanin DNA z uwzględnieniem stężeń roztworów [analysis of DNA mixture including concentrations of the solutions]," *Elektronika*, vol. 50, no. 8, pp. 279 – 284, 2009. `http://www.sigma-not.pl/publikacja-46385-analiza-mieszanin-dna-z-uwzgl%C4%99dnieniem-st%C4%99%C5%BCe%`

C5%84-roztwor%C3%B3w-elektronika-konstrukcje-technologie-zastosowania-2009-8.html.

Moim wkładem było opracowanie koncepcji, dobór architektury, nadzór nad implementacją, analiza wyników, utworzenie tekstu. Mój udział szacuję na 70%.

[7] K. Nałęcz-Charkiewicz and R. Nowak, "Algorithm of detecting structural variations in dna sequences," in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, vol. 9290, pp. 92901H 1–8, International Society for Optics and Photonics, 2014. doi: 10.1117/12.2074123, http://dx.doi.org/10.1117/12.2074123.

Moim wkładem był udział w opracowaniu koncepcji rozwiązania problemu i doborze algorytmów, nadzór nad implementacją i testowaniem, analiza poprawności dostarczanych wyników oraz edycja części tekstu. Mój udział szacuję na 30%.

[8] R. Nowak and A. Plucienniczak, "Sposob identyfikacji czasteczek DNA opisanych wyrazeniem regularnym," in *Patent Rzeczypospolitej Polskiej*, 2006. PL 216751.

Moim wkładem było opracowanie koncepcji automatów na DNA, wykonanie eksperymentów biologicznych zgodnie z wytycznymi prof. Płucienniczaka, opracowanie wyników eksperymentu, utworzenie tekstu. Mój udział szacuję na 70%.

[9] R. Nowak and A. Plucienniczak, "Sposob identyfikacji czasteczek DNA opisanych wyrazeniem regularnym," in *Patent Rzeczypospolitej Polskiej*, 2006. PL 216761.

Moim wkładem było opracowanie koncepcji automatów na DNA, wykonanie eksperymentów biologicznych zgodnie z wytycznymi prof. Płucienniczaka, opracowanie wyników eksperymentu, utworzenie tekstu. Mój udział szacuję na 70%.

[10] R. Nowak and A. Plucienniczak, "Finite state automata built on DNA," *Biocybernetics and Biomedical Engineering*, vol. 28, no. 4, pp. 3 – 19, 2008. http://ibib.waw.pl/bbe/bbefulltext/BBE_28_4_003_FT.pdf.

Moim wkładem było opracowanie koncepcji automatów skończonych wykorzystujących cząsteczki DNA, wykonanie eksperymentów biologicznych zgodnie z wytycznymi prof. Płucienniczaka, opracowanie wyników eksperymentu, utworzenie tekstu. Mój udział szacuję na 80%. ***IF=0.208 (5-Year)***

[11] R. M. Nowak, "Polyglot programming the applications to analyze genetic data," *BioMed Research International*, vol. 2014, pp. 1 – 7, 2014. doi:10.1155/2014/253013, http://dx.doi.org/10.1155/2014/253013.

IF=2.706

4 Opis osiągnięcia naukowego

Prowadzone przeze mnie badania dotyczą stosowania informatyki i jej narzędzi do analizy danych genetycznych. Cechą charakterystyczną analizowanych problemów jest konieczność analizy dużej ilości danych złożonymi algorytmami, co sprawia, że wydajność systemu przetwarzającego dane jest niezwykle istotna. Prace obejmują tworzenie modeli, badanie ich właściwości, wykorzystanie ich do tworzenia algorytmów, realizację algorytmów oraz analizę rzeczywistych danych. Algorytmy

są realizowane na komputerach elektronicznych oraz w roztworach, gdzie wykorzystywane są specyficzne reakcje pomiędzy cząsteczkami DNA.

Dane genetyczne są najczęściej reprezentowane przez zbiór napisów, gdzie każdy napis jest sekwencją symboli nad skończonym alfabetem. Taka reprezentacja, nazywana strukturą pierwszorzędową, odzwierciedla fakt, że cząsteczki przechowujące informację genetyczną są biopolimerami. W niektórych algorytmach cząsteczka przechowująca informację genetyczną jest opisywana bardziej złożoną strukturą niż sekwencja symboli, struktura drugorzędowa jest grafem, uwzględnia ona oddziaływania pomiędzy monomerami, struktura trzeciorzędowa zawiera położenie poszczególnych atomów we współrzędnych trójwymiarowych. Oprócz reprezentacji cząsteczek, dane genetyczne zawierają inne informacje, m.in. podobieństwo do innych cząsteczek, funkcje cząsteczki, strukturę cząsteczki.

Algorytmy przetwarzające dane genetyczne wykorzystują wszystkie znane techniki oprócz algorytmów siłowych (*brute force*), które nie nadają się ze względu na dużą przestrzeń rozwiązań. Szczególne znaczenie mają algorytmy programowania dynamicznego, ponieważ pozwalają one rozwiązywać problemy w czasie wielomianowym, pomimo tego, że przestrzeń poszukiwań rośnie wykładniczo. Algorytmy uliniawiania oparte o programowanie dynamiczne są stosowane do badania podobieństw, do tworzenia i badania profili oraz w poszukiwaniu pod-sekwencji za pomocą ukrytych modeli Markowa (*Hidden Markov Model, HMM*). Algorytmy rekurencyjne z nawrotami są stosowanie do znajdowania motywów, algorytmy zachłanne w badaniu rearanżacji genomów, algorytmy typu dziel i zwyciężaj m.in. w efektywnym pamięciowo algorytmie uliniowienia sekwencji. Problemy odtwarzania sekwencji na podstawie losowych odczytów wykorzystują algorytmy grafowe, w szczególności oparte o grafy de Bruijna. Zastosowanie mają także algorytmy redukcji wymiarów, algorytmy drzewiaste (np. do analizy drzew filogenetycznych), algorytmy maszynowego uczenia się (klasyfikacja, aproksymacja i inne) oraz algorytmy optymalizacji lokalnej i globalnej.

W moich pracach wykorzystuję strukturę pierwszorzędową oraz drugorzędową do opisywania danych, zaś przy analizie haplotypów i mieszanin DNA używam opisu uproszczonego, warianty są reprezentowane przez ich etykiety. Posługiwałem się algorytmami programowania dynamicznego do badania podobieństwa, odnajdowania re-aranżacji, tworzenia struktury drugorzędowej na podstawie sekwencji oraz do odtwarzania stanów dla sekwencji opisanych HMM. Algorytmy optymalizacji, w szczególności algorytm ewolucyjny, algorytm wspinaczkowy oraz algorytm maksymalizacji wartości oczekiwanej (*Expectation-Maximization, EM*) wykorzystywałem do analizy haplotypów, analizy mieszanin DNA, estymacji ukrytego modelu Markowa na podstawie obserwacji, tworzenia zbioru cząsteczek do syntezy sztucznych genów. Algorytmy klasyfikacji używałem do określania funkcji cząsteczek RNA oraz do analizy pokrewieństw, zaś algorytmy grafowe do odtwarzania sekwencji cząsteczki na podstawie odczytów jej fragmentów. Metoda typu dziel i zwyciężaj została użyta w analizie podobieństw oraz w analizie haplotypów.

Moje badania koncentrują się na analizie haplotypów, analizie mieszanin DNA, badaniu podobieństw, odtwarzaniu sekwencji cząsteczki na podstawie odczytów jej fragmentów oraz na wspomaganiu procesu syntezy sztucznych genów. Cząsteczki DNA były wykorzystywane do przetwarzania informacji na nich zakodowanej. Dobór tematów wynikał z potrzeb ośrodków, z którymi współpracowałem, wymienionych w załączniku 5 sekcje 4 i 5. Prace rozszerzają znane algorytmy, pozwalając lepiej analizować dostępne dane. Dostarczone programy komputerowe są bardziej wydajne lub bardziej elastyczne niż znane rozwiązania. Większość z nich jest używana w praktyce.

Szczegółowy opis badanych przeze mnie problemów, dotyczących analizy danych genetycznych, jest zamieszczony poniżej.

- Analiza haplotypów

  Haplotyp jest to zespół sprzężonych wariantów genów (alleli). Powszechnie stosowane techniki genotypowania, czyli odczytywania alleli w ustalonych miejscach DNA (loci), nie pozwalają jednoznacznie przyporządkować alleli do haplotypów[1]. Można obliczyć prawdopodobieństwo wystąpienia danego haplotypu wykorzystując genotypy zbioru osobników, które pochodzą z tej samej populacji znajdującej się w stanie równowagi Hardy'ego-Weinberga. Metoda ta, opisana przez Excoffiera i Slatkina[2] jest stosowana w badaniach klinicznych ze względu na niskie koszty, aby przyporządkować allele do haplotypów, nie potrzebuje ona genotypów rodziców, ani sekwencji nukleotydowych poszczególnych chromosomów.

  Dostarczyłem program komputerowy realizujący tę metodę. Rozszerzyłem istniejący model uwzględniając nieme warianty (*null allele*). Takie allele nie są obserwowane podczas genotypowania u heterozygot, występują m.in. u ludzi w genach z rodziny KIR (*killer cell inhibitory/immunoglobulin-like receptors*). Powstał program komputerowy, wykorzystujący nowe podejście, który został przetestowany na danych rzeczywistych, które zostały udostępnione przez autorów innych aplikacji. Pokazałem, że dla danych bez niemych alleli wyniki dostarczane przez moją aplikację są porównywalne z wynikami innych programów komputerowych, natomiast dla danych zawierających nieme warianty, wyniki są znacznie lepsze, ponieważ uwzględniam specyficzne cechy takich wariantów. Wyniki zostały opublikowane w pracy [1]. Następnie została wykonana ponowna analiza danych dotyczących genów KIR dla 200 Irlandczyków oraz genów KIR dla 99 Polaków chorych na łuszczycę. Nasza aplikacja dostarczyła zbiór haplotypów, który można powiązać z wynikami publikowanymi dla populacji chińskiej. Nasze badania opisaliśmy w pracy [2], która potwierdza użyteczność wprowadzonego modelu i jego implementacji w praktyce. Praca ta zawiera ponadto ocenę wpływu wielkości grupy oraz odstępstw od równowagi Hardy'ego-Weingerga na wyniki obliczeń.

- Synteza sztucznych genów

  Biosynteza białek wymaga utworzenia cząsteczek DNA kodujących gen metodami chemicznymi. Cząsteczki takie są nazywane sztucznymi genami. Istniejące metody nie pozwalają na tworzenie długich nici DNA, dlatego sztuczne geny uzyskuje się łącząc szereg krótszych fragmentów. Nadmiarowość kodu genetycznego pozwala modyfikować sekwencję nukleotydów sztucznego genu. Zmiana sekwencji jest niezbędna, gdy chcemy syntezować białko, które ma ulegać ekspresji w organizmie innym, niż organizm gospodarza. Po utworzeniu nowej sekwencji musimy wyeliminować sekwencje regulacyjne oraz obszary, które silnie oddziałują ze sobą tworząc struktury hamujące ekspresję.

  W pracy [3] połączyliśmy algorytm projektowania sekwencji DNA dla sztucznego genu z algorytmem podziału sekwencji na zbiór fragmentów o długości mieszczących się w zadanym przedziale. Podejście to jest nowe, dotychczas zadania te były wykonywane niezależnie. Połączenie obu warunków pozwala uzyskać lepsze sekwencje i lepsze podziały na fragmenty, gdzie miarą była energia struktur drugorzędowych, które hamują ekspresję oraz zgodności częstości używanych kodonów z tabelami uzyskanymi doświadczalnie dla danego organizmu.

[1] Przykład: genotyp $A_1A_2B_1B_2$ może być utworzony przez haplotypy A_1B_1 i A_2B_2 albo A_1B_2 i A_2B_1.

[2] L. Excoffier, M. Slatkin, Maximum-likehood estimation of molecular haplotype frequencies in a diploid population, *Mol. Biol. Evol.*, 1995

Wykorzystaliśmy algorytm ewolucyjny do optymalnego doboru fragmentów, które będą syntezowane. Algorytm dobiera miejsca rozdzielające fragmenty w sekwencji docelowej,minimalizując siłę oddziaływania fragmentów, które nie są sąsiednie. Miarą siły oddziaływania pomiędzy cząsteczkami jest algorytm wzorowany na algorytmie Nussinov, wprowadzona przez nas modyfikacja pozwala na analizę struktur drugorzędowych dla dwóch, a nie jednej, cząsteczki.

Zaproponowana w naszej pracy technika łączenia fragmentów pozwala umieścić ich wiele w tej samej probówce i łączyć je w jednym doświadczeniu, co znacznie przyspiesza i upraszcza proces tworzenia sztucznego genu, w porównaniu z techniką łączenia każdej pary fragmentów niezależnie. Opisana w literaturze technika łączenia fragmentów za pomocą polimerazy i ligazy[3] została przez nas zmodyfikowana. W metodzie pierwotnej temperatura w roztworze jest stała, zaś w naszym rozwiązaniu temperatura w roztworze się zmienia: roztwór ogrzewa się powyżej temperatury denaturacji, a następnie wolno chłodzi. Nowa metoda wymaga, aby sekwencje łączące kolejne fragmenty różniły się temperaturą równowagi termodynamicznej. Podczas zmniejszania temperatury następuje przyłączanie kolejnych fragmentów. Fragmenty te mogą się łączyć bardziej precyzyjne niż w metodzie pierwotnej, co pozwala zwiększyć ich liczbę w probówce. Miarą siły oddziaływania jest własna implementacja algorytmu Zukera, zaś optymalizację, w tym wypadku wielokryterialną, wykonuje algorytm ewolucyjny połączony z algorytmem wspinaczkowym.

Jeżeli istnieją cząsteczki, które łączą się niepoprawnie, reakcję można rozbić na trzy: rekcje syntezujące części cząsteczki wynikowej zawierające fragmenty, które są w konflikcie oraz reakcję łączenia tych części. Aplikacja opisana w naszej pracy uwzględnia takie sytuacje.

Program komputerowy wykorzystuje architekturę klient-serwer, użytkownik potrzebuje jedynie przeglądarki www. Obliczenia są rozproszone (CORBA), dodatkowo używamy procesora karty graficznej, jeżeli jest dostępny. W laboratorium inżynierii genetycznej przeprowadziliśmy dodatkowe doświadczenie pokazujące, że utworzone białko *Ubp4p* jest aktywne w organizmie *E.Coli*, następnie wyizolowaliśmy to białko i odczytaliśmy sekwencję dla 20 plazmidów. 30% z nich miało poprawną sekwencję, zaś 80% miała poprawną kolejność fragmentów, a odczytane sekwencje różniły się od zadanej na pojedynczych nukleotydach, co oznacza, że błędy wynikały z niskiej jakości fragmentów syntezowanych chemicznie, a nie z innego, niż zaprojektowany, przebiegu reakcji. Dotychczas najdłuższą cząsteczką jest sztuczny gen *CSF3R* o długości $2336\,bp$, który został zbudowany z 42 fragmentów, reakcja syntezy przebiegała w trzech krokach.

- Sekwencjonowanie

  Najbardziej wydajną metodą odczytu sekwencji nukleotydów w materiale genetycznym jest izolacja cząsteczek DNA, następnie podział tych cząsteczek na fragmenty położone losowo, później odczyt sekwencji nukleotydów dla tych fragmentów, na koniec użycie programów komputerowych, nazywanych assemblerami DNA, które dostarczają sekwencji wynikowej. Assemblery DNA wykorzystują algorytmy grafowe, odczyty (sekwencje fragmentów DNA) reprezentują wierzchołki w grafie, zaś miara podobieństwa odczytów jest krawędzią. Wynik obliczeń, sekwencja całej cząsteczki, jest ścieżką w grafie. Sekwencjonowanie drugiej generacji (*next generation sequencing*) dostarcza bardzo dużej ilości odczytów, dlatego w assemblerach

[3] M. Wiedmann, at al., Ligase chain reaction (LCR)–overview and applications, *PCR Methods and Applications*, 1994

stosuje się podejście oparte o graf de Bruijna, co pozwala pominąć czasochłonny proces obliczania miary podobieństwa odczytów, gdzie rozważa się wszystkie pary. Wadą tego podejścia jest używanie sekwencji o długościach równych rzędowi grafu, rząd grafu musi być mniejszy niż długość odczytu.

Dla rzeczywistego zbioru odczytów uzyskanego z danej cząsteczki DNA, dążymy do uzyskania pojedynczej sekwencji wynikowej. Niestety, często uzyskujemy zbiór wielu sekwencji, nazywanych kontigami, ponieważ graf reprezentujący odczyty nie jest spójny, czyli sekwencja wejściowa nie jest pokryta w całości, albo graf posiada wiele alternatywnych ścieżek Eulera, czyli w odczytywanych sekwencjach występują sekwencje powtarzające się dłuższe niż stopień grafu de Bruijna. Aby zmniejszyć prawdopodobieństwo braku pokrycia sekwencji wejściowej w całości, odczytuje się większą liczbę fragmentów, niż wynika z ilorazu ilości odczytanych symboli do długości badanej sekwencji. Współczynnik nadmiarowości $c = \frac{LN}{G}$, wynosi od 5 do 20, gdzie L jest średnią długością fragmentu, N ilością fragmentów, G długością badanej sekwencji.

W pracy [4] rozszerzyłem algorytm oparty o graf de Bruijna. Nowy model pozwala zmniejszyć liczbę kontigów, czyli dostarcza lepszych danych wyjściowych w algorytmie assemblera DNA niż znane metody. W istniejących rozwiązaniach dodatkowe kontigi są tworzone, gdy przy tworzeniu ścieżki Eulera w grafie skierowanym analizujemy wierzchołek, który ma więcej niż jedną krawędź wychodzącą. W nowym modelu dla wierzchołka, który ma dokładnie dwie krawędzie wychodzące oraz jedna z tych krawędzi jest mostem, czyli łączy różne silnie spójne składowe w grafie, nie tworzymy nowego kontiga. W takim przypadku algorytm tworzący ścieżkę Eulera musi wykorzystać najpierw krawędź, która nie jest mostem, zaś podczas kolejnego odwiedzania tego wierzchołka krawędź, która jest mostem. Analiza danych generowanych sztucznie, na podstawie genomów *Escherichia coli* oraz *Saccharomyces cerevisiae* pokazała istnienie kilkunastu takich wierzchołków, co daje ok. 5% poprawę wyników. Zaprezentowany algorytm znajdowania ścieżki Eulera ma wyższą, bo kwadratową (w funkcji liczby wierzchołków) złożoność obliczeniową, algorytmy stosowane powszechnie mają złożoność liniową.

W pracy [5] dostarczam model matematyczny, opisujący właściwości przedstawionego assemblera, dla idealizowanych danych. Z modelu wynika, że błąd estymacji długości sekwencji powtarzających się, dłuższych niż rząd grafu de Bruijna, jest liniowo zależny od długości takich sekwencji oraz jest odwrotnie proporcjonalny do współczynnika nadmiarowości. Model ten pokazał, że lepsze rezultaty osiąga się dla małych rzędów grafu. Wyniki przedstawione w tej pracy pozwalają wykorzystać istniejące wyniki sekwencjonowania, aby zmniejszyć liczbę obszarów o nieznanej sekwencji.

Dalsze prace nad naszymi aplikacjami to połączenie analizy sekwencji sparowanych końców (*paired end tags, PET*) z assemblerem oraz uwzględnienie jakości sekwencjonowania i błędów. Algorytmy analizy sekwencji będą wykorzystane do ponownej analizy odczytów genomu ogórka odmiany Borszczagowskiego (*Cucumis sativus L.*) oraz do badania różnic genetycznych pomiędzy grupami osób zdrowych oraz chorych. W tym drugim przypadku zamierzamy pominąć referencyjny genom człowieka, ponieważ spodziewamy się, że pozwoli nam to lepiej uwzględniać insercje i delecje.

- Analiza mieszanin DNA i podobieństw sekwencji

  Mieszaniną DNA nazywa się wynik genotypowania materiału, który pochodzi od dwu lub więcej osób. Analiza takich danych jest wykorzystywana w kryminalistyce, polega ona na obliczaniu prawdopodobieństw hipotez[4]. Ponieważ standardowe testy używają markerów dziedziczonych niezależnie (wariaty nie są sprzężone), algorytm analizy mieszanin bada każde loci osobno, stosując np. algorytm z nawrotami. Aby zmniejszyć liczbę rozpatrywanych przypadków wykorzystuje się informację o ilości materiału dla poszczególnych wariantów. Proporcje składników w mieszaninie, o ile nie są znane, można wyznaczyć uśredniając proporcje obliczone na podstawie analizy poszczególnych markerów. W pracy [6] opisujemy nowy program komputerowy analizujący mieszaniny DNA. Cechą charakterystyczną naszego rozwiązania jest wykorzystanie architektury klient-serwer, na komputerach klienta wykorzystywana jest jedynie przeglądarka internetowa. Wykorzystujemy informację o ilości materiału, wyniki analiz są takie same jak wyniki najlepszych programów dostępne na rynku.

  Badanie podobieństw dwóch genomów, uwzględniające re-aranżacje wewnątrz-chromosomowe zostało przedstawione w pracy [7]. Wykorzystano algorytm programowania dynamicznego, algorytm Knutha-Morrisa-Prata oraz algorytm bazujący na porównaniu sekwencji w przestrzeni ich transformat Fouriera. Algorytmy te testowano na re-aranżacjach generowanych sztucznie dla genomów *Escherichia coli*, *Salmonella enterica* oraz *Arabidopsis thaliana* dodając szum. Nasze wyniki były lepsze niż wyniki innych badań, ale nie zostały jeszcze przetestowane na sekwencjach rzeczywistych, które mogą zawierać błędy. Aby przyspieszyć obliczenia wykorzystujemy markery genetyczne, które są dostarczane przez nasz program. Pozwala to analizować sekwencje całych genomów w akceptowalnym czasie.

- Obliczenia realizowane na cząsteczkach DNA

  Cząsteczki DNA, ze względu na występowanie selektywnych oddziaływań wodorowych, mogą przetwarzać informacje. Urządzenia wykorzystujące te cząsteczki charakteryzują się bardzo dużą równoległością, nieosiągalną dla komputerów elektronicznych, niskim zapotrzebowaniem na energię oraz dużą stopą błędów. Moja praca doktorska była poświęcona obliczeniom molekularnym realizowanym na cząsteczkach DNA (*DNA computing*). Po otrzymaniu stopnia doktora kontynuowałem ten wątek badawczy, zbudowałem i przetestowałem urządzenie, działające w probówce, które jest w stanie rozpoznawać cząsteczki DNA, posiadające sekwencję opisaną danym wyrażeniem regularnym. Urządzenia takie jest molekularną realizacją automatu skończonego, może być ono wykorzystywane do wykrywania cząsteczek DNA należących do pewnej klasy, bez potrzeby odczytu sekwencji i jej obróbki przez komputer klasyczny. Rozwiązanie zostało uznane za wynalazek i uzyskało prawną ochronę w Urzędzie Patentowym Rzeczypospolitej Polskiej. Patent [8] opisuje automat molekularny, który jest w stanie analizować sekwencje DNA posiadające ściśle ustaloną sekwencję na początku i na końcu, dodatkowo sekwencje reprezentujące symbole są rozdzielone pewną stałą i z góry określoną sekwencją reprezentującą separator. Patent [9] pozwala analizować sekwencje bez ustalonej sekwencji reprezentującej separator, ale wydajność reakcji jest znacznie mniejsza niż w poprzednim rozwiązaniu, więc stopa błędów podczas analizy jest większa. Opisywane rozwiązanie oraz wyniki doświadczeń przeprowadzonych w laboratorium inżynierii genetycznej potwierdzających poprawność koncepcji opisujemy w pracy [10].

[4]Przykład: materiał genetyczny oskarżonego, który ma genotyp $A_1A_2B_1B_2$ nie jest obecny w mieszaninie $A_1A_2A_3A_4B_1B_3B_4$.

- Narzędzia i języki programowania używane do tworzenia aplikacji przetwarzających dane genetyczne

  Moje badania obejmowały dobór architektury aplikacji, narzędzi i języków programowania. W wyniku tych badań okazuje się, że warto tworzyć aplikacje złożone z modułów tworzonych w różnych językach oprogramowania. Zaproponowałem tworzenie rozwiązań w językach JavaScript i HTML5, Python oraz C++, co jest alternatywą do używania PHP, .NET czy Javy, zapewniając wiele korzyści, np. wydajność i elastyczność rozwiązań. Kosztem stosowania tego podejścia jest konieczność opanowania kilku języków programowania oraz poznania narzędzi umożliwiających komunikację pomiędzy nimi.

  Ponieważ nie istnieje środowisko, przeznaczone do tworzenia aplikacji przetwarzających dane genetyczne, wykorzystujące wspomniane języki, zaproponowałem i utworzyłem takie środowisko [11]. Jest ono przeznaczone dla aplikacji charakteryzujących się dużym zapotrzebowaniem na moc obliczeniową, gdzie użytkownicy preferują graficzny interfejs użytkownika oraz prosty sposób aktualizacji oprogramowania. Zaproponowana architektura jest warstwowa, oddzielnymi warstwami są: interfejs użytkownika, przetwarzanie danych oraz przechowywanie danych. Część kliencka wykorzystuje przeglądarkę www do realizacji warstwy prezentacji oraz do częściowego przetwarzania danych, m.in. sprawdzania poprawności i generowania wykresów. Przetwarzanie i przechowywanie danych jest realizowane na serwerze. Warstwa trwałości używa systemu zarządzania relacyjną bazą danych, zaś warstwa przetwarzania, zawierająca implementację algorytmów, ma strukturę złożoną: zawiera moduły wydajnie przetwarzające dane oraz pewne fragmenty kodu, które pośredniczą pomiędzy warstwą trwałości oraz warstwą interfejsu użytkownika.

  Do implementacji algorytmów proponuję wykorzystywać C++ w wersji określonej przez standard C++11 i C++14 oraz zbiór bibliotek Boost. Pozwala to tworzyć wydajne implementacje, ponieważ kod C++ jest translowany do kodu maszynowego, ponadto rozwiązania mogą być przenośne na poziomie kodu źródłowego i mogą wykorzystać się w pełni możliwości współczesnych komputerów, czyli używać wielu procesorów (lub wielu rdzeni) i procesora karty graficznej (GPU). Dla kodu, który pośredniczy przy dostarczaniu danych do algorytmów, oraz kodu dostarczającego wyniki dla użytkownika istotna jest elastyczność, możliwość łatwej konfiguracji, dostępność wielu formatów wymiany danych, obsługa systemów zarządzania bazą danych oraz dostępność protokołów transmisji. W tej roli, moim zdaniem, znakomicie sobie radzi język Python. Ze względu na to, że jest to interpreter, bardzo łatwo dostosowywać rozwiązania do potrzeb konkretnych użytkowników, nie ma potrzeby re-kompilacji kodu, zaś mnogość bibliotek i obsługiwanych formatów znacznie przyspiesza pracę przy opracowywaniu prototypów. Ponadto język Python posiada biblioteki pozwalające korzystać z publicznie dostępnych genetycznych bazy danych.

  Warstwa klienta jest tworzona w JavaScript i HTML5, ponieważ język ten jest interpretowany przez przeglądarkę. Alternatywnie możemy w tej warstwie wykorzystać Apache Flex (dawniej Adobe Flex), który dla starszych przeglądarek dostarcza podobnych udogodnień (za pomocą wtyczki Flash Player). Część kliencka jest ładowana podczas uruchamiania aplikacji, więc aktualizacja oprogramowania po stronie klienta jest banalna, wystarczy odświeżenie strony. Dodatkowo można uruchamiać część kliencką na różnych platformach, jej wydajność nie ma wpływu na czas obliczeń.

  Moje badania obejmowały także sposoby zarządzania pracami zespołu programistycznego.

Prowadzone pod moim kierownictwem projekty charakteryzowały się tym, że nie mogłem liczyć na finansowanie zespołu programistów przez odpowiednio długi okres, więc wspierałem się pomocą studentów, którzy dysponowali niezbyt dużą ilością czasu, a poza tym nie mieli dużego doświadczenia w projektowaniu i programowaniu. Okazało się, że najlepsze rezultaty uzyskiwały projekty, które były tworzone w metodologiach zwinnych (*agile*), ponieważ zapewniały współpracę przyszłego klienta aplikacji, pozwalały na transfer wiedzy biologicznej i medycznej, zaś unikanie powielania informacji i niezbyt duża ilość dokumentacji umożliwiała stosunkowo szybko podejmować pracę nowym osobom.

5 Omówienie pozostałych osiągnięć naukowo-badawczych

Moje badania wymagają budowania programów komputerowych, dlatego interesuję się technikami stosowanymi w tworzeniu oprogramowania. Monografia [13] omawia techniki stosowane w języku C++, we współczesnych aplikacjach, gdzie istotna jest wydajność oraz przenośność (dla C++ na poziomie kodu źródłowego). Techniki te to specyficzne wzorce projektowe oraz udogodnienia, które w momencie publikowania były dostępne jako elementy zbioru bibliotek eksperymentalnych Boost, zaś obecnie są częścią standardu C++11. Mojego autorstwa są rozdziały omawiające techniki stosowane w programowaniu generycznym (klasy cech, klasy wytycznych, meta-programowanie - przetwarzanie informacji w czasie kompilacji i statyczne asercje), zarządzanie obiektami na stercie (sprytne wskaźniki), wzorce projektowe (fabryki, singleton, opóźnione tworzenie i kopiowanie obiektów, iteratory, adaptery, wizytator, wielometoda, komenda, obserwator, kompozyt, dekorator), funkcje anonimowe, poprawne oznaczanie obiektów jako stałe, warianty, kolekcje dostępne w bibliotekach Boost (tablice wielowymiarowe, grafy), obsługa daty i czasu, obsługa strumieni, współbieżność (wątki), asynchroniczna obsługa wejścia i wyjścia, wyrażenia regularne oraz łączenie modułów tworzonych w języku C++ z modułami tworzonymi w języku Python.

Opublikowałem cykl artykułów popularnonaukowych w czasopismach Software Developers Journal oraz Programista, które zajmują się zarządzaniem obiektami na stercie [67], technikami programowania generycznego [68, 69], wzorcami projektowymi w kontekście języka C++ [70, 71, 72, 73, 74, 75, 90], poprawnym oznaczaniem obiektów jako stałe w danym kontekście [76], wykorzystywaniem współbieżności [77], obsługą urządzeń wejścia-wyjścia [78, 85], oraz omawiając ciekawe biblioteki ze zbioru Boost [79, 80, 81, 82, 83]. Powstał także artykuł dotyczący testów jednostkowych [84] oraz aplikacji internetowych [86, 87]. Jestem autorem dwóch artykułów, opublikowanych na portalu `http://it.pwn.pl` [88, 89].

W czasie studiów doktoranckich zajmowałem się obliczeniami molekularnymi, z tego okresu pochodzą publikacje [12, 38, 39, 40, 41, 42, 43, 64, 65]. Kontynuowałem ten wątek badawczy publikując nowy sposób utworzenia pamięci asocjacyjnej na DNA [15] oraz realizację automatu skończonego na DNA, omówionego w poprzednim punkcie.

Badałem sposoby uwzględnienia ryzyka w zarządzaniu pulą kontraktów na towarowej giełdzie energii. Prace te wynikały z udziału w projekcie 8decision (patrz załącznik 4 sekcja 2). Jestem współtwórcą aplikacji, która wykorzystuje zmienne losowe, predykcję szeregów czasowych oraz algorytm ewolucyjny i wspinaczkowy do optymalnego zarządzania pulą kontraktów na rynku energii z uwzględnieniem ryzyka. Wyniki badań są zawarte w pracy [20] oraz w raportach technicznych [45, 46, 47, 48, 49, 50, 51]. Praca [21] zawiera badanie różnych strategii handlowych w symulowanym, wielo-agentowym środowisku giełdy towarowej.

Fuzja danych jest dziedziną, którą chciałbym wykorzystać do przetwarzania danych genetycz-

nych. Brałem udział w projekcie DAFNE (patrz załącznik 4 sekcja 2), w którym badałem algorytmy wykorzystywane w fuzji danych w zastosowaniach obronnych. Prace [22, 23] zawierają wyniki moich badań, zaś raporty techniczne [52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63] szczegóły realizacji systemu.

Niektóre prace, dotyczące analizy danych genetycznych, nie weszły w skład cyklu publikacji, wymienionego w punkcie 3, ponieważ opisują wykorzystanie zaprezentowanych wcześniej algorytmów, rozszerzenia tych algorytmów lub wstępne wyniki nowych badań. Do publikacji takich zaliczam:

- pracę [14] pokazującą wstępne wyniki dla automatów skończonych zbudowanych na cząsteczkach DNA;
- pracę [16], gdzie pokazałem nowy model do obliczania haplotypów uwzględniający nieme warianty;
- pracę [17] pokazującą wstępne wyniki dla algorytmu syntezy sztucznego genu;
- pracę [18], która pokazuje wykorzystanie algorytmów sztucznej inteligencji do wykrywania pokrewieństw, gdzie dane obejmujące etykietę wariantu dla 10000 markerów genetycznych dla członków 8 osobowej rodziny były badane za pomcą sieci bayesowskich;
- artykuł [19], gdzie pokazaliśmy wstępne wyniki dotyczące syntezy sztucznych genów;
- pracę [24], w której opisaliśmy program komputerowy do automatycznego znajdowania typu cząsteczki RNA przechowywanych w bazie RNA Strand[5]; wykorzystaliśmy naiwny klasyfikator bayesowski, drzewo decyzyjne oraz klasyfikator oparty o algorytm k najbliższych sąsiadów; baza zawierała 3557 sekwencji przypisanych do 36 klas; nasza praca jest pierwszą automatyczną klasyfikacją danych z tej bazy;
- pracę [25] opisującą nasz algorytm do odwrotnej translacji wykorzystujący ukryty model Markowa; przeprowadzone badania dla modelu pierwszego rzędu dały zgodność utworzonej sekwencji na poziomie 37%; ten wynik jest nieznacznie lepszy od wyników aplikacji zakładających wykorzystanie jednego, najbardziej częstego kodonu dla każdego aminokwasu;
- pracę [26] opisującą aplikację do przeglądania genomu, która powstała na zlecenie Polskiego Konsorcjum Sekwencjonowania Genomu Jądrowego Ogórka;
- pracę [27] pokazującą algorytm, który ogranicza przestrzeń przeszukiwanych rozwiązań naszego algorytmu do rozkładu haplotypów [2] poprzez analizę ustalonej liczby blisko położonych loci; skraca to czas obliczeń kosztem pogorszenia jakości wyników;
- pracę [28] opisującą nowy algorytm do określania funkcji biologicznych dla danej sekwencji; używamy algorytmu BLAST we wskazanych bazach danych dostępnych online i poszukujemy sekwencji podobnych do danej sekwencji wejściowej, następnie pobieramy opisy funkcji biologicznych za pomocą słowników zgodnych z Gene Ontology, na koniec łączymy te opisy wykorzystując reguły Dempstera-Shafera;

[5]M. Andronescu, at al., RNA STRAND: the RNA secondary structure and statistical analysis database, *BMC Bioinformatics*, 2008

- prace [29, 30], gdzie pokazano syntezę sztucznego genu i produkcję białka za pomocą naszego algorytmu opisanego w [3]; opisaliśmy syntezę cząsteczki kodującej proteazę *Ubp4p* o długości 276 *bp*, która powstała poprzez połączenie 11 fragmentów w jednej reakcji oraz testy otrzymanych klonów;
- pracę [31], która tworzy skafold, czyli sekwencję zawierającą poprawnie ułożone kontigi oraz przerwy pomiędzy nimi, o znanej długości; wykorzystaliśmy wyniki odczytu sekwencji sparowanych końców, zaś problem rozbiliśmy na dwa pod-problemy: znajdowanie porządku kontigów i znajdowanie długości przerw pomiędzy kontigami, co zmniejszyło złożoność obliczeniową; użyto technik do przeszukiwania przestrzeni z więzami (*constraint satisfaction problem, CSP*);
- pracę [32], w której opisaliśmy aplikację rozszerzającą mój algorytm assemblera DNA [4] o odczyty pochodzące z dwu komplementarnych nici DNA; tworzone są dwa grafy opisujące obie nici;
- pracę[33], która opisuje wstępne wyniki algorytmu do wykrywania rearanżacji bazującego na markerach genetycznych;
- abstrakt i plakat [34] demonstrujący wykorzystanie środowiska Bioweb do tworzenia aplikacji analizujących dane genetyczne;
- pracę [35] zawierającą porównanie narzędzi do automatycznego wyszukiwanie struktury sekwencji;
- pracę [36], gdzie opisujemy wstępne wyniki grupowania danych mikromacierzowych przez zmodyfikowany algorytm k-najbliższych sąsiadów;
- pracę [37] zawierającą porównanie istniejących asemblerów DNA.

Architektura aplikacji przetwarzającej sekwencje genetyczne, po modyfikacjach, została wykorzystana w projektach ZSM, MWSSSG oraz projektach realizowanych we współpracy z firmą BRASTER, patrz załącznik 4 sekcja 2.

Do osiągnięć badawczych zaliczam także twórcze prace zawodowe wymienione w załączniku 4 sekcja 2 i 8 oraz załączniku 5 sekcja 12.

Robert Nowak