

Recenzja osiągnięć naukowych dra Marka Nowickiego

1. Wstęp

Niniejsza recenzja powstała w odpowiedzi na uchwałę nr 7/2026 Rady Naukowej IPI PAN. Zgodnie z art. 221 ust. 8 ustawy *Prawo o szkolnictwie wyższym i nauce*, skupię się na odpowiedzi na następujące pytanie¹:

Czy osiągnięcia naukowe dra Marka Nowickiego, pod wspólnym tytułem *Paradygmat PGAS programowania równoległego w języku Java: rozwój i analiza biblioteki PCJ na przykładzie wybranych algorytmów, w tym bioinformatycznych*, stanowią znaczny wkład w rozwój informatyki technicznej i telekomunikacji?

Pojęcie znacznego wkładu jest nieprecyzyjne. Aby choć trochę temu zaradzić, posłużę się następującymi kryteriami pomocniczymi i związaną z nimi 3-stopniową skalą ocen:

- **oryginalność** osiągnięć naukowych (bardzo wysoka, na dobrym poziomie, zbyt mała);
- **poprawność** metodyczna (wzorcowa, wystarczająca, niewystarczająca);
- **znaczenie** osiągnięć (doniosłe, niebagatelne, pomijalnie małe).

Osiągnięcia Habilitanta dotyczą obliczeń wysokiej wydajności (ang. *High Performance Computing*, w skrócie HPC). Są to obliczenia wykonywane równolegle na systemach wielordzeniowych, od pojedynczych węzłów po klastry składające się z setek lub tysięcy węzłów obliczeniowych. Jednym z paradygmatów programowania obliczeń HPC jest model pamięci PGAS (ang. *Partitioned Global Address Space*), w którym system składa się ze zbioru równolegle działających elementów przetwarzających (np. w języku Java będą one widoczne jako wątki obliczeniowe) i każdy element przetwarzający ma w swojej pamięci wyodrębniony obszar, do którego inne elementy przetwarzające mają bezpośredni dostęp (jest to rodzaj rozproszonej pamięci współdzielonej).

Choć przedstawiony do oceny cykl artykułów [1–7] jest bardzo spójny, można w nim wyodrębnić dwa osiągnięcia²:

- Opracowanie, przy współudziale nieżyjącego już prof. Piotra Bały i dra Łukasza Górskiego, biblioteki PCJ (ang. *Parallel Computing in Java*) ułatwiającej programowanie obliczeń HPC w języku Java w oparciu o paradygmat PGAS. Jest to, mówiąc językiem ustawy, oryginalne osiągnięcie konstrukcyjne. Biblioteka PCJ w wersji 5.3.4 jest dostępna na platformie GitHub³, a jej opis z punktu widzenia programisty można znaleźć m.in. w pracach [2, 6].
- Empiryczną ocenę wydajności oprogramowania budowanego z wykorzystaniem biblioteki PCJ. To osiągnięcie ma formę cyklu powiązanych tematycznie artykułów naukowych, do którego zaliczam publikacje [1, 3–5, 7].

W dalszej części recenzji przedstawiam ocenę tych osiągnięć z perspektywy wspomnianych kryteriów pomocniczych.

¹Zgodnie z przywołanym ust. 8 recenzent nie powinien oceniać aktywności naukowej, działalności dydaktycznej itp.

²Identyfikacja dwóch osiągnięć jest o tyle ważna, że pojawiają się interpretacje art. 219 ust. 1 ustawy mówiące, że Habilitant powinien mieć przynajmniej dwa osiągnięcia stanowiące znaczny wkład w rozwój dyscypliny naukowej [Gurdek-2022].

³<https://github.com/hpdcej/PCJ>

2. Oryginalność osiągnięć

Uwaga 1. Z punktu widzenia wkładu w rozwój informatyki, oryginalność biblioteki PCJ jest na **dobrym poziomie**.

Uzasadnienie. Ocena oryginalności sprowadza się do porównania danej propozycji z analogicznymi rozwiązaniami opracowanymi przez innych autorów. W przypadku biblioteki oprogramowania można wyróżnić oryginalność koncepcji (tzw. *programming model*) oraz oryginalność implementacji. Przy ocenie wkładu w rozwój dyscypliny naukowej, koncepcja narzędzia (w tym przypadku biblioteki) wydaje się ważniejsza niż jego implementacja, ale bez implementacji ocena znaczenia koncepcji jest trudna. Moja ocena oryginalności biblioteki PCJ wynika ze spostrzeżeń przedstawionych poniżej. Każde spostrzeżenie ma formę pojedynczego zdania, po którym znajduje się krótkie uzasadnienie. Możliwe, że na bazie tych spostrzeżeń ktoś oceniłby bibliotekę PCJ wyżej niż ja.

- **W sferze koncepcji, biblioteka PCJ jest bardzo podobna do biblioteki SHMEM i jej pochodnych.**

Prawdopodobnie pierwszą biblioteką implementującą paradygmat PGAS była biblioteka SHMEM. Została opracowana przez Cray Research Inc. w 1994 roku, by ułatwić programowanie superkomputerów Cray T3D i T3E [Parzyszek-2003, Chapman-2010]. Oparcie biblioteki PCJ na paradygmacie PGAS oznacza, że wiele pomysłów i decyzji projektowych, leżących u podstaw biblioteki SHMEM, znajdziemy również w bibliotece PCJ, a także w innych bibliotekach bazujących na tym paradygmacie, jak GPSHMEM [Parzyszek-2003] czy OpenSHMEM [Chapman-2010]. Do najważniejszych cech biblioteki SHMEM widocznych w bibliotece PCJ należy:

- model SPMD (ang. *Single Program Multiple Data*);
- założenie, że obliczenia są realizowane przez tzw. elementy przetwarzające (ang. *processing elements*), którym w PCJ odpowiadają wątki; jedno i drugie są ponumerowane od 0 do $N - 1$ i obie biblioteki oferują funkcje określające ich liczbę (czyli wartość N) oraz numer danego elementu/wątku (w PCJ te funkcje nazywają się `PCJ.threadCount()` i `PCJ.myId()`);
- model rozproszonej pamięci współdzielonej: każdy element przetwarzający (a w PCJ wątek) ma pamięć podzieloną na część lokalną (prywatną) i współdzieloną (dostępną innym elementom); adresując zmienną współdzieloną należy wskazać numer elementu przetwarzającego oraz adres zmiennej w danym elemencie;
- przesyłanie danych między elementami przetwarzającymi jest realizowane na zasadzie komunikacji jednostronnej (ang. *one-sided communication*): jeśli element przetwarzający A inicjuje przesłanie danych do lub z elementu B, to wykonuje całą operację sam, bez angażowania elementu B;
- włączenie do biblioteki operacji kolektywnych, takich jak *barrier*, *broadcast*, *scatter* oraz operacji redukcji (w bibliotekach SHMEM i GPSHMEM są one ograniczone do predefiniowanej listy siedmiu operacji [Parzyszek-2003]: `and`, `or`, `xor`, `sum`, `prod`, `min`, `max`).

Należy jednak podkreślić, że:

- biblioteka PCJ zawiera sporo nowych rozwiązań nieobecnych w bibliotekach SHMEM, GPSHMEM i OpenSHMEM, jak na przykład możliwość podziału wątków na grupy i ustalania ich kolejności w ramach grupy za pomocą jednej operacji, dostosowanie operacji redukcji do struktury węzeł-wątek poprzez wykonywanie redukcji najpierw w obrębie węzłów, a dopiero potem między węzłami, jak też organizację węzłów w drzewa binarne, co bardzo przyspiesza komunikację;
 - biblioteki SHMEM, GPSHMEM i OpenSHMEM są dostępne w językach C, C++ i Fortran. Nie znalazłem pracy, która wspominałaby o dostępności tych bibliotek w języku Java.
- Najprawdopodobniej **PCJ jest drugą na świecie (po APGAS) biblioteką udostępniającą paradygmat PGAS w języku Java.**

Z przeglądu literatury wynika, że pierwszą była biblioteka APGAS (*Asynchronous PGAS*), nad którą prace rozpoczęto już w 2007 roku [Saraswat-2010]. W 2011 roku ukazała się praca opisująca system *Cluster Superscalar* (*ClusterSs*) zaimplementowany w języku Java z wykorzystaniem tej biblioteki [Tejedor-2011]. O wcześniejszej dostępności biblioteki APGAS w języku Java świadczy również fakt, że język X10, dla którego została ona opracowana, był implementowany na zasadzie kompilacji do języka Java lub C++ [Tardieu-2015]. Natomiast, jak piszą autorzy biblioteki PCJ [Nowicki-2016], *pierwsze prace nad biblioteką PCJ rozpoczęły się na przełomie września i października 2011 roku*, czyli w czasie, gdy biblioteka APGAS już istniała i była używana (rok później ukazała się pierwsza praca na temat PCJ [Nowicki-2012]). Daleki byłbym jednak od stwierdzenia, że biblioteka PCJ jest wzorowana na APGAS, chociażby z tego względu, że praca [Saraswat-2010] opisuje APGAS w dość abstrakcyjny sposób (bez żadnego przykładu) i przedstawia go w ścisłym powiązaniu z językiem X10. Przed opracowaniem PCJ istniała jeszcze biblioteka *Global Arrays* bazująca na paradygmacie PGAS i w jednej z prac na jej temat jest informacja sugerująca dostępność tej biblioteki również w języku Java [Nieplocha-2006] (za pomocą narzędzia Babel i języka SIDL opracowanych w Lawrence Livermore National Laboratory), jednak

trudno znaleźć informacje potwierdzające istnienie aplikacji napisanych w języku Java i korzystających z tego rozwiązania. Co więcej, autorzy Babel/SIDL w pracy [Epperly-2012] piszą, że integracja z językami bazującymi na paradygmacie PGAS stanowi spore wyzwanie i wymieniają kilka języków programowania, ale nie wspominają o bibliotece *Global Arrays*, więc sugestią dotyczącą dostępności tej biblioteki w języku Java należy rozumieć jako *potencjalną* dostępność.

- **Sposób deklarowania współdzielonych pól w PCJ jest zdecydowanie inny niż w APGAS.**

Aby zilustrować tę tezę, posłużę się przykładem zawartym w pracy [Nowicki-2016] (pozwoliłem sobie na drobne zmiany) — jest on pokazany w formie listingu 1. Odpowiednik tego kodu dotyczący biblioteki APGAS ma postać listingu 2 — jest to uproszczona wersja kodu wygenerowanego przez ChatGPT (GPT-5.6 Sol, OpenAI). Zasadnicza różnica, która rzuca się w oczy, to korzystanie przez PCJ z mechanizmu adnotacji (@RegisterStorage i @Storage), z którego w ogóle nie korzysta APGAS.

W bibliotece PCJ obliczenia mają formę zbioru współbieżnie realizowanych *zadań*, które są implementowane jako wątki języka Java. Przyjęto model SPMD i kod wykonywany przez zadania ma formę klasy implementującej interfejs `StartPoint`. Klasa ta ma pola prywatne i niektóre z nich mogą stać się widoczne globalnie. Wskazanie nazw pól widocznych globalnie odbywa się za pomocą typu wyliczeniowego poprzedzonego adnotacją `@Storage`. Listing 1 zawiera deklarację pola `table`, którego nazwa staje się widoczna globalnie wskutek zadeklarowania jej jako elementu typu wyliczeniowego `DaneWspoldzielone` i umieszczenia nazwy tego typu (w tym przypadku `Przyklad.DaneWspoldzielone`) na liście argumentów adnotacji `@RegisterStorage`. W omawianym przykładzie każde zadanie (wątek) ma dwa pola: wspomniane pole `table` oraz pole `sum` — to ostatnie jest widoczne tylko lokalnie w danym zadaniu. Referencja do utworzonej dynamicznie tablicy znajduje się tylko w kopii pola `table` należącej do zadania nr 0; w pozostałych zadaniach pole `table` ma wartość `null`.

W bibliotece APGAS podstawowym mechanizmem są tzw. miejsca (ang. *places*), reprezentujące obszary pamięci współdzielonej (w APGAS/Java miejsce jest typowo implementowane jako osobna JVM, czyli osobny proces z własną pamięcią i wątkami). Aktywną częścią modelu są tzw. czynności (ang. *activities*), będące jednostkami współbieżnego wykonania; w danym miejscu może być wykonywanych równolegle wiele czynności, obsługiwanych przez znajdujące się w nim wątki robocze. W pamięci danego miejsca mogą znajdować się obiekty zawierające dane dostępne dla wykonywanych tam czynności (na listingu 2 jest to obiekt klasy `SharedData` z polem `table`). W analizowanym przykładzie funkcja `at(place(0), ...)` wykonuje w miejscu `place(0)` trzy instrukcje, które powodują utworzenie w tym miejscu obiektu klasy `SharedData` zawierającego referencję do utworzonej tablicy. Aby utworzyć zmienną lokalną `sum` w czynnościach wykonywanych we wszystkich miejscach (podobnie jak w wersji PCJ pole `sum` występowało we wszystkich wątkach), należy odpowiednią czynność uruchomić w pętli dla wszystkich miejsc.

Jak widać, sposoby deklaracji pól (zmiennych) w obu bibliotekach są skrajnie różne.

```
@RegisterStorage(Przyklad.DaneWspoldzielone.class)
public class Przyklad implements StartPoint {
    @Storage(Przyklad.class)
    enum DaneWspoldzielone {
        table // deklaracja nazwy pola widocznego globalnie
    }
    private int[] table; // deklaracja pola 'table' (bedzie w kazdym watku)
    private long sum; // deklaracja pola 'sum' (bedzie w kazdym watku)
    @Override
    public void main() throws Throwable {
        if (PCJ.myId() == 0) { // tworzenie tablicy table (tylko w watku nr 0)
            table = new int[PCJ.threadCount()];
        }
    }
}
```

Listing 1: Przykład deklaracji pola widocznego globalnie — wersja PCJ (na podstawie [Nowicki-2016]).

```

public class PrzykladPGAS {
    static class SharedData { // deklaracja danych dostepnych globalnie
        int[] table;
    }
    public static void main(String[] args) {
        System.setProperty(Configuration.APGAS_PLACES, "2");
        // Tworzenie tablicy 'table' w place(0)
        GlobalRef<SharedData> data =
            at(place(0), () -> { SharedData d = new SharedData();
                                d.table = new int[places().size()];
                                return new GlobalRef<>(d);
                            } );
        finish(() -> { for (Place p: places())
                        asyncAt(p, () -> { long sum; } ); } );
    }
}

```

Listing 2: Przykład deklaracji pola widocznego globalnie — wersja APGAS.

- **Biblioteki PCJ i APGAS różnią się sposobem dostępu do współdzielonych pól.**

Biblioteka PCJ implementuje tzw. komunikację jednostronną (ang. *one-sided communication*). Jeśli zadanie A chce przesłać dane do zadania B lub pobrać je z zadania B, to samo inicjuje i wykonuje odpowiednią operację, bez konieczności wykonywania odpowiadającej jej operacji przez zadanie B. W bibliotece APGAS jest inaczej. Jeśli zadanie A chce zmodyfikować dane znajdujące się w miejscu B lub odczytać ich wartość, to uruchamia w miejscu B czynność wykonującą odpowiednią operację na tych danych.

Tę różnicę dobrze widać na poziomie kodu. Zaczniemy od PCJ. Załóżmy, że tablica `table` jest w zadaniu o numerze 0 (patrz listing 1) i jakieś zadanie chce przesłać do komórki tej tablicy o indeksie 5 liczbę 777. W przypadku biblioteki PCJ instrukcja przesłania wyglądałaby następująco:

```
PCJ.put(777, 0, DaneWspoldzielone.table, 5);
```

Biblioteka APGAS wymaga w tym przypadku uruchomienia czynności w miejscu, w którym znajduje się ta tablica — zgodnie z listingiem 2 jest to miejsce o numerze 0. Dokładniej mówiąc, w miejscu o numerze 0 znajduje się obiekt wskazywany przez uchwyt `data`, zawierający tablicę `table`. Realizując wspomniane przypisanie za pomocą biblioteki APGAS, odpowiednia instrukcja, dostosowana do deklaracji z listingu 2, miałaby następującą postać (wartością `data.home()` jest `place(0)`):

```
at(data.home(), () -> { data.get().table[5] = 777; });
```

Pierwszy parametr funkcji `at()` wskazuje miejsce, w którym ma zostać wykonana czynność, natomiast drugi określa wykonywany w tym miejscu kod (zapisany za pomocą wyrażenia lambda).

- **Wkład Habilitanta w powstanie i rozwój biblioteki PCJ jest dominujący.**

Moja opinia wynika z następujących faktów:

- W autoreferacie habilitant nie wymienia żadnego istotnego pomysłu dot. biblioteki PCJ, którego autorem byłaby inna osoba.
- We wszystkich pracach wchodzących w skład cyklu powiązanych tematycznie artykułów naukowych Habilitant jest jedynym lub pierwszym współautorem, mimo że nie jest to zgodne z porządkiem alfabetycznym, co wskazuje na jego wiodącą rolę.
- Z danych dostępnych na stronie GitHub⁴ wynika, że jest tylko dwóch współautorów kodu biblioteki PCJ, Habilitant oraz promotor Jego rozprawy doktorskiej prof. Piotr Bała, przy czym Habilitant wprowadził 14 633 linii kodu a prof. Piotr Bała 270 i Habilitant usunął 9 117 linii kodu, a Jego promotor jedną.

□

Uwaga 2. *Oryginalność prac poświęconych ocenie wydajności biblioteki PCJ jest bardzo wysoka.*

Uzasadnienie. Najbardziej oryginalne i pomysłowe wydają mi się następujące rozwiązania:

HPV-KITE. Ważnym problemem bioinformatyki jest wykrycie, które spośród genotypów opisanych przez sekwencje referencyjne są obecne w badanej próbce. Specyfiką tego problemu jest fakt, że próbka jest zbiorem setek tysięcy lub nawet milionów odczytów (czyli krótkich sekwencji nukleotydowych lub białkowych).

Przedstawione w pracy [7] podejście opiera się na dwóch głównych pomysłach: masowej równoległości (osiąganej przez zastosowanie biblioteki PCJ w środowisku superkomputerów) i zastosowaniu indeksu

⁴<https://github.com/hpcj/PCJ-tests/graphs/contributors?from=10%2F01%2F2015>

Tversky'ego (jest to indeks podobieństwa dwóch zbiorów; w programie HPV-KITE zarówno sekwencje wchodzące w skład próbki, jak i pojedyncze elementy zbioru referencyjnego dzieli się na krótkie podsłowa zwane k-merami i w ten sposób powstają zbiory, do których można zastosować indeks Tversky'ego). Wydaje się, że pomysł z dzieleniem długiej sekwencji na k-mery i stosowaniem indeksu Tversky'ego można by też wykorzystać jako podstawę metody wyszukiwania sekwencji referencyjnych, stanowiącej alternatywę dla części funkcjonalności BLAST.

PCJ-BLAST. Innym ważnym problemem bioinformatyki jest dopasowanie sekwencji nukleotydowych (lub białkowych), czyli znalezienie w referencyjnym zbiorze sekwencji tych, które są najbardziej podobne (dają się dopasować) do sekwencji pełniącej rolę zapytania. Prawdopodobnie najbardziej znanym narzędziem w tym zakresie jest program BLAST opracowany w NCBI (*National Center for Biotechnology Information*). Wykorzystuje on współbieżność na poziomie wątków i jest przystosowany do obliczeń na pojedynczym węźle. Opracowany przez dra Marka Nowickiego i prof. Piotra Bałę program PCJ BLAST [1] umożliwia efektywną pracę programu BLAST w środowisku bardzo wielu węzłów bez zmiany kodu programu BLAST. Jest to, oparta na PCJ, warstwa orkiestracji obliczeń rozproszonych, w której wątek rozpraszający (ang. *scatterer*) przydziela zadania BLAST wątkom roboczym (ang. *workers*), a te uruchamiają współbieżne procesy wykonujące oryginalny program NCBI BLAST. Oceniając oryginalność PCJ-BLAST trzeba brać pod uwagę, że podobne rozwiązania, mające na celu zwiększenie skalowalności BLAST w środowisku wielu węzłów obliczeniowych, istniały już wcześniej, jak np. mpiBLAST [Darling-2003].

PCJ-BLAST można interpretować jako podejście pokrewne *Divisible Load Theory* [Robertazzi-2003]: wykorzystuje ono podzielność problemu dopasowywania sekwencji, lecz zamiast statycznego szeregowania zadań stosuje podejście dynamiczne.

Pozostałe rozwiązania również mają dużą dawkę oryginalności, ale jest ona na nieco niższym poziomie. W pracy [3] Habilitant zaproponował równoległe sortowanie dużych wolumenów danych będące alternatywą dla *Hadoop TeraSort*. Jest to ciekawa praca prezentująca wyniki nietrywialnych eksperymentów, ale algorytm *TeraSort* był już znany wcześniej. W pracy [4] przedstawiono wyniki badań nad wydajnością biblioteki PCJ w środowisku chmurowym na przykładzie Amazon AWS EC2 oraz Linaro Developer Cloud. Przygotowanie tych eksperymentów wymagało dużej wiedzy eksperckiej, ale od strony algorytmicznej jest to nieco mniej nowatorskie, gdyż w eksperymentach wykorzystano znane rozwiązania (dla łamania szyfru DES metodę "na siłę", dla szybkiej transformaty Fouriera algorytm Takahashi and Kanady, dla zliczania słów podejście *map-reduce*). Praca [5], podobnie jak wcześniej omawiane prace, ma charakter eksperymentalny i jej celem było zbadanie wydajności biblioteki PCJ. Wykorzystane w eksperymentach algorytmy obejmowały *Game of Life* (automat komórkowy), znajdowanie parametrów dla matematycznego modelu niciania *C. Elegans* z wykorzystaniem algorytmów genetycznych (na podstawie pracy [Rakowski-2017]) oraz trenowanie 3-warstwowej sztucznej sieci neuronowej z wykorzystaniem *TensorFlow*, służącej do rozpoznawania ręcznie pisanych cyfr.

□

3. Poprawność metodyczna

Uwaga 3. *Poprawność metodyczna dotycząca biblioteki PCJ jest wystarczająca.*

Uzasadnienie. Koncepcja biblioteki PCJ ewoluowała i ten proces ewolucji został elegancko opisany w autoreferacie. Kolejną zaletą od strony metodycznej jest udostępnienie kodu źródłowego biblioteki wraz z krótkimi programami testującymi za pośrednictwem platformy GitHub.

Do perfekcji brakuje udostępnienia testów regresyjnych (to mógłby być temat pracy inżynierskiej). Szkoda też, że na platformie GitHub jest tylko najnowsza wersja biblioteki (5.3.4) — przydałoby się udostępnienie wcześniejszych wersji, m.in. 4.1.0, która była wykorzystana do implementacji PCJ-BLAST (mogłaby się przydać do dalszych prac badawczych, a jest niekompatybilna z wersją 5.3.4 ze względu na inny sposób deklarowania pól dostępnych globalnie).

□

Uwaga 4. *Poprawność metodyczna prac dotyczących oceny jakości biblioteki PCJ jest wystarczająca.*

Uzasadnienie. Ocena jakości biblioteki PCJ przeprowadzono w szeregu bardzo interesujących eksperymentów wykorzystując niebanalne problemy (patrz uzasadnienie uwagi 2). Ten bardzo pozytywny obraz psuje nieco brak perfekcji metodycznej.

Aby można było powiedzieć, że praca jest wzorcowa od strony metodycznej, rozważany w niej problem powinien być jasno sformułowany. Jest to o tyle ważne, że widać wtedy granicę między problemem a algorytmem, co

daje podstawę do porównań z innymi rozwiązaniami. Ponadto takie podejście ułatwia zrozumienie zagadnienia informatykom, którzy nie są specjalistami w danej dziedzinie zastosowań (np. genetyce).

Mój "niedosyt metodologiczny" budzi przede wszystkim praca na temat algorytmu HPV-KITE [7]. Pozostałe prace dotyczą znanych problemów i algorytmów ich rozwiązywania. Praca [7] jest inna. Z jednej strony jest to kolejny przykład wykorzystania biblioteki PCJ i eksperymentalnej oceny jej wydajności, ale z drugiej strony (i to jest wielka wartość tej pracy), autorzy proponują nowe podejście do przeszukiwania referencyjnego zbioru sekwencji nukleotydowych lub białkowych na podstawie próbki. Bez klarownego opisu problemu trudno jest informatykowi zorientować się, na czym dokładnie polega problem i czym on się różni od problemu rozwiązywanego wcześniej analizowanym programem BLAST czy PCJ-BLAST. Poniżej postaram się dokładniej przedstawić moje zastrzeżenia i oczekiwania.

Praca [7] ma następującą strukturę:

- *Introduction*
- *Material and methods*
- *Sequence data indexing*
- *Datasets*
- *Experimental setup*
- *Parallelization*
- *Performance assessment*
- *Results*
- *Discussion*
- *Conclusion*

W żadnym z rozdziałów nie ma jasno sformułowanego problemu (nie ma go też w autoreferacie). W sekcji *Material and methods* jest jedynie coś w rodzaju zarysu problemu ('*rapid scanning of large genome datasets to identify HPV sequences*' i dalej '*fast and accurate HPV detection*'). Przykład precyzyjnego sformułowania problemu zawarłem w ramkach 1 i 2. Ramka 1 przedstawia ogólną wersję problemu, niezależną od sposobu, w jaki próbka pobrana od pacjenta jest dopasowywana do zbioru referencyjnego zawierającego sekwencje nukleotydów znanych wariantów wirusa.

Ramka 1. Przykładowa definicja problemu wykrywania genotypów

Niech $\Sigma = \{A, C, G, T, N\}$ oznacza alfabet nukleotydowy.

Sekwencja referencyjna jest parą (n, s) , gdzie n jest nazwą genotypu (np. HPV213), a $s \in \Sigma^*$ opisuje sekwencję nukleotydów.

Referencyjny zbiór sekwencji, R , jest niepustym skończonym zbiorem sekwencji referencyjnych (n, s) . Każda nazwa genotypu, n , występuje w R co najwyżej raz.

Nazwy genotypów, $N(R)$, jest to zbiór nazw występujących w zbiorze referencyjnym R , czyli $N(R) = \{n : (n, s) \in R\}$.

Próbka, P , jest niepustym skończonym zbiorem trójek (id, r, q) odpowiadających rekordom jednego lub większej liczby plików FASTQ. W każdej trójce $r \in \Sigma^*$ opisuje odczyt (ang. *sequencing read*), czyli krótką sekwencję nukleotydów uzyskaną w procesie sekwencjonowania, id jest identyfikatorem odczytu, natomiast q jest ciągiem ocen jakości jego poszczególnych elementów ($|q| = |r|$). W praktyce próbka może zawierać miliony elementów.

$GroundTruth(P, R)$ jest zbiorem nazw genotypów ze zbioru referencyjnego R , które faktycznie są obecne w materiale biologicznym, z którego uzyskano próbkę P ($GroundTruth(P, R) \subseteq N(R)$).

Ogólny problem wykrywania genotypów

Dane: próbka P i referencyjny zbiór sekwencji R .

Wynik: zbiór genotypów obecnych w materiale biologicznym, z którego uzyskano próbkę P , czyli zbiór $GroundTruth(P, R)$.

W pracy [7] zaproponowano heurystyczną metodę rozwiązania tego problemu. Polega ona na dzieleniu sekwencji nukleotydów próbki oraz sekwencji nukleotydów ze zbioru referencyjnego na spójne podciągi (podstowa) o tej samej długości i porównywaniu powstałych zbiorów za pomocą indeksu Tversky'ego. W ramce 2 pokazano, jak mogłaby wyglądać formalna specyfikacja tej heurystyki. Nie jest to dokładny opis podejścia opisanego przez Habilitanta. W pracy [7], z nieznanых powodów (nie podano ich ani w pracy [7], ani w autoreferacie), zdecydowano, że zbiór k-merów dla próbki P będzie tworzony na podstawie *konkatenacji* odczytów próbki, a nie jako suma zbiorów. Zabieg ten można potraktować jako *trick* implementacyjny. Ma on jednak swoje negatywne konsekwencje:

- Konkatenowanie odczytów zawartych w próbce może prowadzić do pojawienia się sztucznych k-merów, których w próbce nie ma. Na przykład ciąg GAC nie jest spójnym podciągiem ciągu ACTG ani ACTT, ale jest podciągiem powstałym z konkatenacji tych ciągów.
- Sztuczne k-mery zależą od kolejności, w jakiej odbywa się konkatenacja (podciąg GAC powstanie ze wspomnianych ciągów tylko wtedy, gdy do ciągu ACTG zostanie dołączony ciąg ACTT, a nie na odwrót). Pojawia się zatem niepożądana zależność wyniku od arbitralnej kolejności odczytów, mimo że kolejność odczytów nie ma znaczenia biologicznego.

Trzeba jednak zaznaczyć, że wyniki eksperymentalnej oceny algorytmu HPV-KITE przedstawione w pracy [7] są tak dobre, iż wpływ sztucznych k-merów na wyniki detekcji wirusa wydaje się pomijalnie mały (jest to trochę zaskakujące, ale trudno z faktami dyskutować). Zatem należy uznać, iż możliwość pojawienia się sztucznych k-merów ma bardziej znaczenie metodologiczne niż praktyczne.

Ramka 2. Specyfikacja heurystyki bazującej na biozgodności Tversky'ego

Przyjmijmy, że referencyjny zbiór sekwencji, R i próbka P są zdefiniowane jak w ramce 1. Wprowadzamy dodatkowe terminy pomocnicze:

$\text{Shingles}_k(s)$, gdzie $s \in \Sigma^*$ i k jest dodatnią liczbą całkowitą, oznacza zbiór wszystkich spójnych podciągów (podśłów) ciągu s o długości k . Element tego zbioru nazywamy *k-merem*. Jeżeli $k > |s|$, to $\text{Shingles}_k(s) = \emptyset$.

Zbiór odczytów próbki, $\Omega(P)$, jest niepustym skończonym zbiorem wszystkich odczytów zawartych w próbce P , czyli $\Omega(P) = \{r : (id, r, q) \in P\}$.

Zbiór k-merów dla próbki P , oznaczany jako $\text{SampleShingles}_k(P)$, jest zdefiniowany następująco:

$$\text{SampleShingles}_k(P) = \bigcup_{r \in \Omega(P)} \text{Shingles}_k(r).$$

Biozgodnością Tversky'ego sekwencji referencyjnej (n, s) z próbką P nazywamy wartość $BT_k((n, s), P)$ określoną następująco:

$$BT_k((n, s), P) = \frac{|\text{SampleShingles}_k(P) \cap \text{Shingles}_k(s)|}{|\text{Shingles}_k(s)|}.$$

Jest to szczególny przypadek indeksu Tversky'ego dla parametrów $\alpha = 0$ i $\beta = 1$.

Porządkowanie genotypów za pomocą biozgodności Tversky'ego

Dane: próbka P , referencyjny zbiór sekwencji R oraz dodatnia liczba całkowita k ($k \leq |s|$ dla każdego $(n, s) \in R$)

Wynik: zbiór par

$$\text{Ranking}_k(P, R) = \{(n, b) : (n, s) \in R \wedge b = BT_k((n, s), P)\}$$

uporządkowany wg nierosnących wartości biozgodności Tversky'ego, b .

Inną kwestią związaną z zaproponowaną w pracy [7] heurystyką jest niejednoznaczność opisu wyniku. W sekcji *Discussion* ostatnie zdanie brzmi następująco:

The output for each sample contains pathogens starting from the one with the highest Tversky index value and the next ones in descending order.

Zdanie to sugeruje, że wynikiem HPV-KITE jest ranking patogenów i specyfikacja HPV-KITE zawarta w ramce 2 bazuje na tej sugestii. Wątpliwości wynikają z następującego zdania zwartego w sekcji *Sequence data indexing*:

A threshold was set at 10% viral coverage.

Zgodnie z tym zdaniem wynikiem nie jest ranking obejmujący wszystkie analizowane patogeny, lecz podzbiór zbioru nazw patogenów $N(R)$, zależny od progu τ , będącego dodatkową daną wejściową, czyli zbiór $\text{Detected}_{k,\tau}(P, R)$ określony następująco:

$$\text{Detected}_{k,\tau}(P, R) = \{n : (n, s) \in R \wedge BT_k((n, s), P) \geq \tau\}.$$

Ponieważ zbiór $\text{Detected}_{k,\tau}(P, R)$ można przedstawić jako:

$$\text{Detected}_{k,\tau}(P, R) = \{n : (n, s) \in R \wedge (n, b) \in \text{Ranking}_k(P, R) \wedge b \geq \tau\}.$$

zatem możliwe, że najpierw powstaje $Ranking_k(P, R)$, a dopiero potem tworzy się zbiór $Detected_{k,\tau}(P, R)$. Szkoda, że w pracy [7] nie przedstawiono tego w jasny sposób.

Kolejną słabością metodyczną jest brak jasnego określania celu przeprowadzanych eksperymentów, klarownego formułowania wniosków i analizy zagrożeń dla poprawności tych wniosków. Jeśli chodzi o eksperymenty opisane w pracy [7], to ich cel opisany w sekcjach *Introduction* i *Material and methods* - jeśli dobrze go rozumiem - można by przedstawić na przykład w następujący sposób:

Celem eksperymentów było porównanie metody HPV-KITE z programami Diamond i Kraken2 pod względem dokładności (ang. *accuracy*) i czasu przetwarzania (ang. *time*) z wykorzystaniem:

- fińskiego zbioru danych DNA [30],
- szwedzkiego zbioru RNA-Seq [29],
- zbioru TCGA RNA-Seq
- 222 kompletnych genomów referencyjnych typów HPV pobranych z bazy PaVE i służących jako referencyjny zbiór w formacie FASTA.

Pierwsze dwa zbiory pełniły rolę próbek, a ostatni został wykorzystany jako referencyjny zbiór sekwencji.

Kolejną słabością metodyczną jest brak jasno sformułowanych spostrzeżeń (hipotez formułowanych w oparciu o wyniki eksperymentów) i analizy zagrożeń dla ich poprawności (ang. *validity threats*). Mimo wskazanych słabości, biorąc pod uwagę nietrywialność algorytmów wykorzystanych w eksperymentach, uważam że można uznać, iż te eksperymenty są na dobrym poziomie (ale nie są perfekcyjnie opisane).

Skoro mowa o poprawności metodycznej to mam jeszcze jedną drobną uwagę. Chodzi o analizę algorytmu *TeraSort* w pracy [3]. Jeśli sortujemy n elementów, to liczba porcji (ang. *buckets*), p , na które je podzielimy nie może być większa niż n . Zatem $p \log p \leq n \log n$ i złożoność najgorszego przypadku opisana w [3] jako $O(n \log n + p \log p)$ upraszcza się do zwykłego $O(n \log n)$. $\square$

4. Znaczenie osiągnięć

Uwaga 5. *Znaczenie biblioteki PCJ jest niebagatelne.*

Uzasadnienie. Biblioteka PCJ ma ważnego konkurenta, jakim jest APGAS. Ponadto sporo oprogramowania w obszarze HPC powstaje z wykorzystaniem języków C, C++ i Fortran (w różnych wersjach), a tam jest dostępna biblioteka OpenSHMEM, też oparta na paradygmacie PGAS. Konkurencja jest więc duża. Uważam jednak, że znaczenie biblioteki PCJ jest niebagatelne, co wynika z następujących faktów:

- Biblioteka PCJ w wersji 3 została wyróżniona nagrodą *HPC Challenge Class 2 Award: Best Productivity — Most Elegant Solution* na międzynarodowej konferencji *SC14 – The International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing 2014)*.
- Bibliotekę PCJ wykorzystano w eksperymentach przeprowadzonych bez udziału Habilitanta [Istrate-2017, Posner-2018], co pokazuje, że tą biblioteką potrafi się posługiwać nie tylko jej autor, a ponadto świadczy o, może jeszcze nie wielkim ale już widocznym, zainteresowaniu środowiska naukowego i jest dobrym prognostykiem na przyszłość.

$\square$

Uwaga 6. *Znaczenie prac Habilitanta, dotyczących wykorzystania biblioteki PCJ i oceny jej wydajności jest niebagatelne.*

Uzasadnienie. Wiele prac Habilitanta skupia się na ocenie wydajności biblioteki PCJ i pokazaniu jej możliwości. Wzbogacają one naszą wiedzę o możliwościach biblioteki PCJ, co jest ważne, ale wydaje się, że dla osób, które nie są zainteresowane tą biblioteką nie mają one zbyt dużego znaczenia praktycznego. Jednak dwa rozwiązania wykraczają poza ocenę biblioteki i mają spory potencjał praktyczny. Mam na myśli HPV-KITE i PCJ-BLAST.

HPV-KITE Znaczenie tego narzędzia wydaje się być znacznie bardziej ogólne niż tylko wykrywanie w próbkę wirusa HPV. Szkoda, że taką przyjęto nazwę, bo to może ograniczyć zainteresowanie tym narzędziem. Prawdą jest, że eksperymenty były prowadzone na zbiorach związanych z wirusem HPV i w oparciu o takie dane była przeprowadzana kalibracja tego algorytmu (mam na myśli np. długość k-merów). Mam jednak nieodparte wrażenie, że metodę HPV-KITE można zastosować nie tylko do wykrywania wirusa HPV. Przeprowadzone eksperymenty pokazują duży potencjał tego rozwiązania, które może być zapewne dalej doskonałone.

PCJ-BLAST Jest to bardzo ciekawe rozwiązanie, które w bardzo istotny sposób przyspiesza obliczenia realizowane przez NCBI BLAST w środowisku wielowęzłowym. Wydaje się, że w czasach intensywnego rozwoju terapii spersonalizowanych bazujących na genetyce, znaczenie takich narzędzi jak NCBI BLAST będzie rosło. Jednak oceniając znaczenie PCJ-BLAST trzeba pamiętać, że istnieją rozwiązania konkurencyjne, jak np. ElasticBLAST przyspieszający obliczenia BLAST w środowisku chmurowym⁵, czy wspomniany wcześniej mpiBLAST działający na klastrach komputerowych⁶.

□

5. Podsumowanie

Biorąc pod uwagę wyniki szczegółowej oceny biblioteki PCJ i prac omawiających przeprowadzone eksperymenty, uważam że osiągnięcia naukowe dra Marka Nowickiego pod wspólnym tytułem

Paradygmat PGAS programowania równoległego w języku Java: rozwój i analiza biblioteki PCJ na przykładzie wybranych algorytmów, w tym bioinformatycznych,

stanowią znaczny wkład w rozwój informatyki technicznej i telekomunikacji i w związku z tym **popieram** nadanie Mu stopnia doktora habilitowanego.

Literatura

Prace wchodzące w skład osiągnięcia naukowego

- [1] Marek Nowicki, Davit Bzhalava, and Piotr Bała. Massively parallel implementation of sequence alignment with Basic Local Alignment Search Tool using parallel computing in Java library. *Journal of Computational Biology*, 25(8):871–881, 2018. doi:10.1089/cmb.2018.0079.
- [2] Marek Nowicki, Łukasz Górski, and Piotr Bała. PCJ - Java library for highly scalable HPC and Big Data processing. In *Proceedings of the 2018 International Conference on High Performance Computing & Simulation (HPCS 2018)*, Orleans, France, July 16–20, 2018, pages 12–20. IEEE, 2018. doi:10.1109/HPCS.2018.00017.
- [3] Marek Nowicki. Comparison of sort algorithms in Hadoop and PCJ. *Journal of Big Data*, 7, 101(1):1–28, 2020. doi:10.1186/s40537-020-00376-9.
- [4] Marek Nowicki, Łukasz Górski, and Piotr Bała. Performance evaluation of Java/PCJ implementation of parallel algorithms on the cloud (extended version). *Concurrency and Computation—Practice & Experience*, 35(15):1–15, 2023. doi:10.1002/cpe.6536.
- [5] Marek Nowicki, Łukasz Górski, and Piotr Bała. PCJ Java library as a solution to integrate HPC, Big Data and Artificial Intelligence workloads. *Journal of Big Data*, 8, 62(1):1–21, 2021. doi:10.1186/s40537-021-00454-6.
- [6] Marek Nowicki, Łukasz Górski, and Piotr Bała. Scalable computing in Java with PCJ Library. Improved collective operations. In *Proceedings of International Symposium on Grids & Clouds 2021 – PoS (ISGC2021)*, Taipei, March 22–26, 2021, volume 378, 007, pages 1–19. Sissa Medialab Srl, 2021. doi:10.22323/1.378.0007.
- [7] Marek Nowicki, Magdalena Mroczek, Dhananjay Mukhedkar, Piotr Bała, Ville Nikolai Pimenoff, and Laila Sara Arroyo M`uhr. HPV-KITE : sequence analysis software for rapid HPV genotype detection. *Briefings in Bioinformatics*, 26(2):1–11, 2025. doi:10.1093/bib/bbaf155.

Dodatkowa literatura

- [Chapman-2010] Chapman, B., Curtis, T., Pophale, S., Poole, S., Kuehn, J., Koelbel, C., & Smith, L. (2010, October). Introducing OpenSHMEM: SHMEM for the PGAS community. In *Proceedings of the fourth conference on partitioned global address space programming model* (pp. 1–3).
- [Darling-2003] Darling, A., Carey, L., & Feng, W. C. (2003). The design, implementation, and evaluation of mpiBLAST. *Proceedings of the ClusterWorld Conference and Expo*, 2003, 13–15.

⁵<https://bio.tools/elasticblast>

⁶<https://wiki.rc.usf.edu/index.php/MpiBLAST>

- [Epperly-2012] Epperly, T. G., Kumfert, G., Dahlgren, T., Ebner, D., Leek, J., Prantl, A., & Kohn, S. (2012). High-performance language interoperability for scientific computing through Babel. *The International Journal of High Performance Computing Applications*, 26(3), 260-274.
- [Gurdek-2022] M. Gurdek, "Osiągnięcie naukowe albo artystyczne stanowiące znaczny wkład w rozwój określonej dyscypliny w rozumieniu ustawy Prawo o szkolnictwie wyższym i nauce", *Roczniki Administracji i Prawa*, vol. XXII, no. 2, 2022, s. 135–151. https://cejsh.icm.edu.pl/cejsh/element/bwmeta1.element.ojs-doi-10_5604_01_3001_0016_0953/c/articles-3200849.pdf.pdf
- [Istrate-2017] Istrate, R., Barkoutsos, P. K., Dolfi, M., Staar, P. W., & Bekas, C. (2017, September). Exploring graph analytics with the PCJ toolbox. In *International Conference on Parallel Processing and Applied Mathematics* (pp. 308-317). Cham: Springer International Publishing.
- [Nieplocha-2006] Nieplocha, J., Palmer, B., Tipparaju, V., Krishnan, M., Trease, H., & Apra, E. (2006). Advances, applications and performance of the global arrays shared memory programming toolkit. *The International Journal of High Performance Computing Applications*, 20(2), 203-231.
- [Nowicki-2012] Nowicki, M., & Bała, P. (2012, July). Parallel computations in Java with PCJ library. In *2012 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 381-387). IEEE.
- [Nowicki-2016] Nowicki, M., & Bała, P. Programowanie równoległe w języku Java z wykorzystaniem biblioteki PCJ. https://www.academia.edu/download/59932233/2016-IPB2-PCJ-rozdzial_pl.pdf
- [Pancake-2001] Pancake, C., & Lengauer, C. (2001). High-performance Java. *Communications of the ACM*, 44(10), 98-101.
- [Parzyszek-2003] Parzyszek, K. (2003). Generalized portable shmem library for high performance computing. *PhD Dissertation*, Iowa State University. <https://dr.lib.iastate.edu/bitstreams/04644d51-de65-4b6d-8e5a-9682bbdb0503/download>
- [Posner-2018] Posner, J., Reitz, M., & Fohry, C. (2018, November). Comparison of the HPC and big data Java libraries spark, PCJ and APGAS. In *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM)* (pp. 11-22). IEEE.
- [Rakowski-2017] Rakowski, F., & Karbowski, J. (2017). Optimal synaptic signaling connectome for locomotory behavior in *Caenorhabditis elegans*: Design minimizing energy cost. *PLoS computational biology*, 13(11), e1005834. <https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1005834&type=printable>
- [Robertazzi-2003] T. G. Robertazzi, Ten reasons to use divisible load theory, *Computer*, vol. 36, no. 5, 2003, s. 63–68.
- [Saraswat-2010] Saraswat, V., Almasi, G., Bikshandi, G., Cascaval, C., Cunningham, D., Grove, D., ... & Tardieu, O. (2010, June). The asynchronous partitioned global address space model. In *The First Workshop on Advances in Message Passing* (pp. 1-8).
- [Tardieu-2015] Tardieu, O. (2015, June). The APGAS library: resilient parallel and distributed programming in Java 8. In *Proceedings of the ACM SIGPLAN Workshop on X10* (pp. 25-26).
- [Tejedor-2011] Tejedor, E., Farreras, M., Grove, D., Badia, R. M., Almasi, G., & Labarta, J. (2011, June). ClusterSs: a task-based programming model for clusters. In *HPDC* (pp. 267-268). <https://www.ac.upc.edu/app/research-reports/html/RR/2011/14.pdf>

