

Wprowadzenie do Rust i Algorytmów Genetycznych

Prezentacja omawia podstawy języka Rust, jego mechanizmy bezpieczeństwa pamięci, wydajność oraz ekosystem narzędzi. Następnie przedstawia algorytmy genetyczne jako heurystyczną metodę optymalizacji inspirowaną biologią. Na koniec omawiamy klasyczny problem plecaka i przedstawione zostanie jego rozwiązanie przy użyciu algorytmu genetycznego, napisane w języku Rust..

Wojciech Trapkowski

Rust: Wprowadzenie

Systemowy język programowania

Rust został opracowany przez Mozillę i jest komplikowany do kodu maszynowego.

Główne mechanizmy

- Ownership (własność)
- Borrowing (pożyczanie)
- Lifetimes (czasy życia)

Bezpieczeństwo pamięci

Zapewnia bezpieczeństwo pamięci bez użycia garbage collectora.

Zastosowania

- systemy operacyjne
- gry
- przeglądarki
- IoT



```
st
elect
← Fitcowering
▼ ddatafs
<intgociles: projectif:
    nro090 "intains:
        Watl0Srete.aCi_scards:
    PastOmmeriew: WessaMetagoffer:
        inckistatts: ffitchald tech-cuatocigus:
            <neFlanny,
            coerast-ippps- ietbanl79:
            fergens semi(),
            costrat-wuppold,hestmalade,
            tnosates: fergoactisefer<(overtrpvPatā)"
        <raiztaly:
        <douttes: felghtes/thanepurfarsk>
            *ates
            <iates:
            ereclascstip: "ioatti/lestingis.detlesgiran" at),
            <apariangUagh>
        >
        Fuealarte: ficersaly hntecta(auhtnjortals.-4e f1969) -dsul-isng,
        <se_peropersful>
        fe socrand-latlafeale-ictes:
        fori Aincall-deanings:
        teclifive-not_rete):
        taclinscuneplisalle:
        <vartinge:
            violurte, tach- tsb6-6. (tapr:
            vsicles_pate)>
        >
        fruttress: fergosctls instover(pł21)
        <amytinage:
            cinersctakes,ropperasteyfs,JK;
            cistinrigg:
            inecractor-we sear- "and pedatleg/opidk"
        >
        "factOhsinkley>
        feccierile: /accsaberforil inccude"
        feestaiter (ictalsllately, orcontooter"|,
        cutnotiyle:
        forcecastaine:lTindopies:
        conrise-restentiae:
        mayatirat- EFF-cnspaces>
        fałla>
        żnrliline:
        <convnising W.ladge> {
```

Rust: Ownership

Jeden właściciel zmiennej

Każda zmienna ma dokładnie jednego właściciela, co zapobiega błędem pamięci.

Przeniesienie własności

Po przeniesieniu własności poprzednia zmienna staje się nieważna, eliminując błędy typu "use after free".

Rust: Borrowing

Pożyczanie zmiennych

Rust umożliwia pożyczanie zmiennych do odczytu (&T) i zapisu (&mut T).

Bezpieczeństwo wątków

Nie pozwala na jednocześnie mutowalne i niemutowalne pożyczanie, chroniąc przed wyścigami danych.



Rust: Lifetimes

Śledzenie czasu życia zmiennych

Kompilator analizuje, jak długo zmienne istnieją, aby zapobiec używaniu danych po ich zwolnieniu.

Bezpieczne operacje

Przykład błędu: odwołanie do zmiennej, która przestała istnieć po zakończeniu bloku.

Rust: Współbieżność

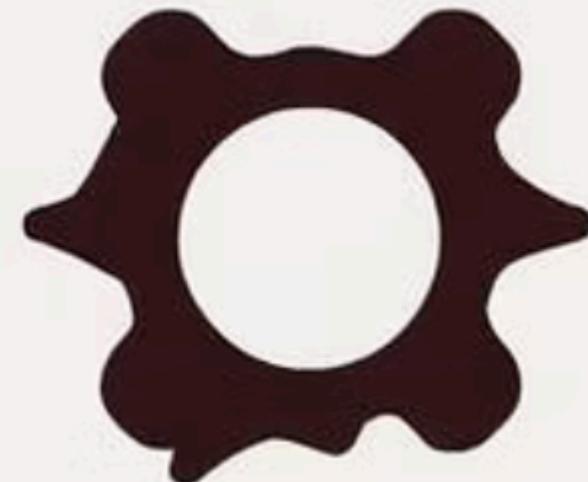
Brak data races

Na poziomie kompilatora data races są niemożliwe dzięki regułom Send i Sync.

Bezpieczeństwo wątków

Kompilator wymusza zasady eliminujące błędy wielowątkowe, zapewniając stabilność aplikacji.





Rust: Ekosystem narzędzi

Cargo

Zarządzanie projektami
i zależnościami



Rustfmt

Automatyczne
formatowanie kodu



Clippy

Lintowanie i sugestie
poprawy



Rust Analyzer

Inteligentne
podpowiedzi w
edytorach

Algorytmy Genetyczne



1

Algorytmy Genetyczne

Heurystyczna metoda optymalizacji inspirowana biologią, stosowana przy dużych przestrzeniach rozwiązań bez wzoru matematycznego.



2

Zastosowania

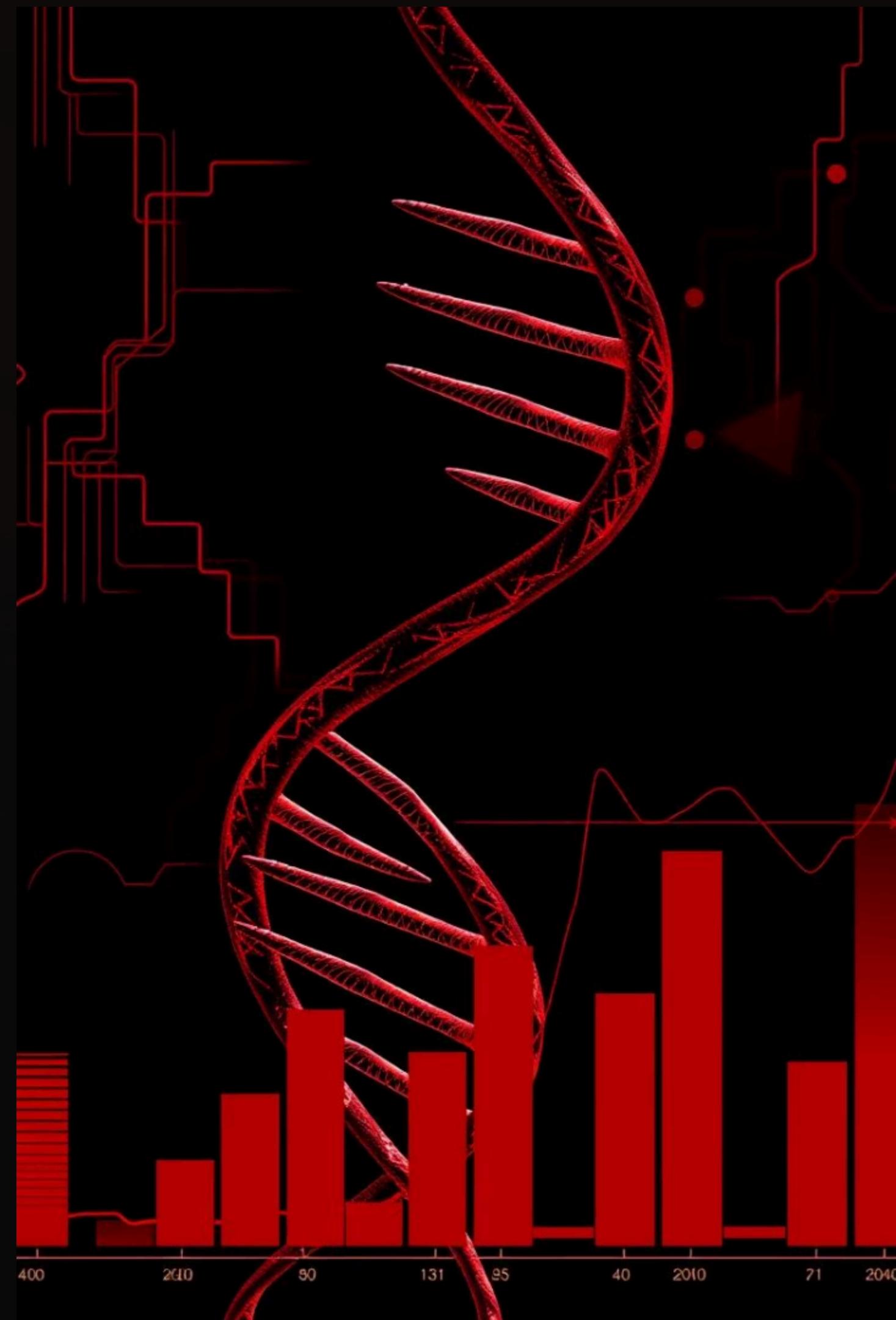
- Planowanie tras i harmonogramów
- Sztuczna inteligencja w grach
- Optymalizacja sieci neuronowych



3

Kluczowe pojęcia

- **Osobnik** – jedno rozwiązanie problemu (np. 101011)
- **Populacja** – zbiór osobników
- **Fitness** – jakość rozwiązania (np. liczba jedynek)
- **Selekcja** – wybieramy lepszych osobników
- **Krzyżowanie** – łączenie osobników w nowe
- **Mutacja** – losowe zmiany dla różnorodności



Algorytm genetyczne: krok po kroku

Zakodowanie problemu

Losowa populacja

Przykładowo 20 osobników

Selekcja

Wybór najlepszych osobników

Oblicz fitness

Ocena jakości każdego osobnika

Krzyżowanie

Tworzenie nowych osobników

Mutacja

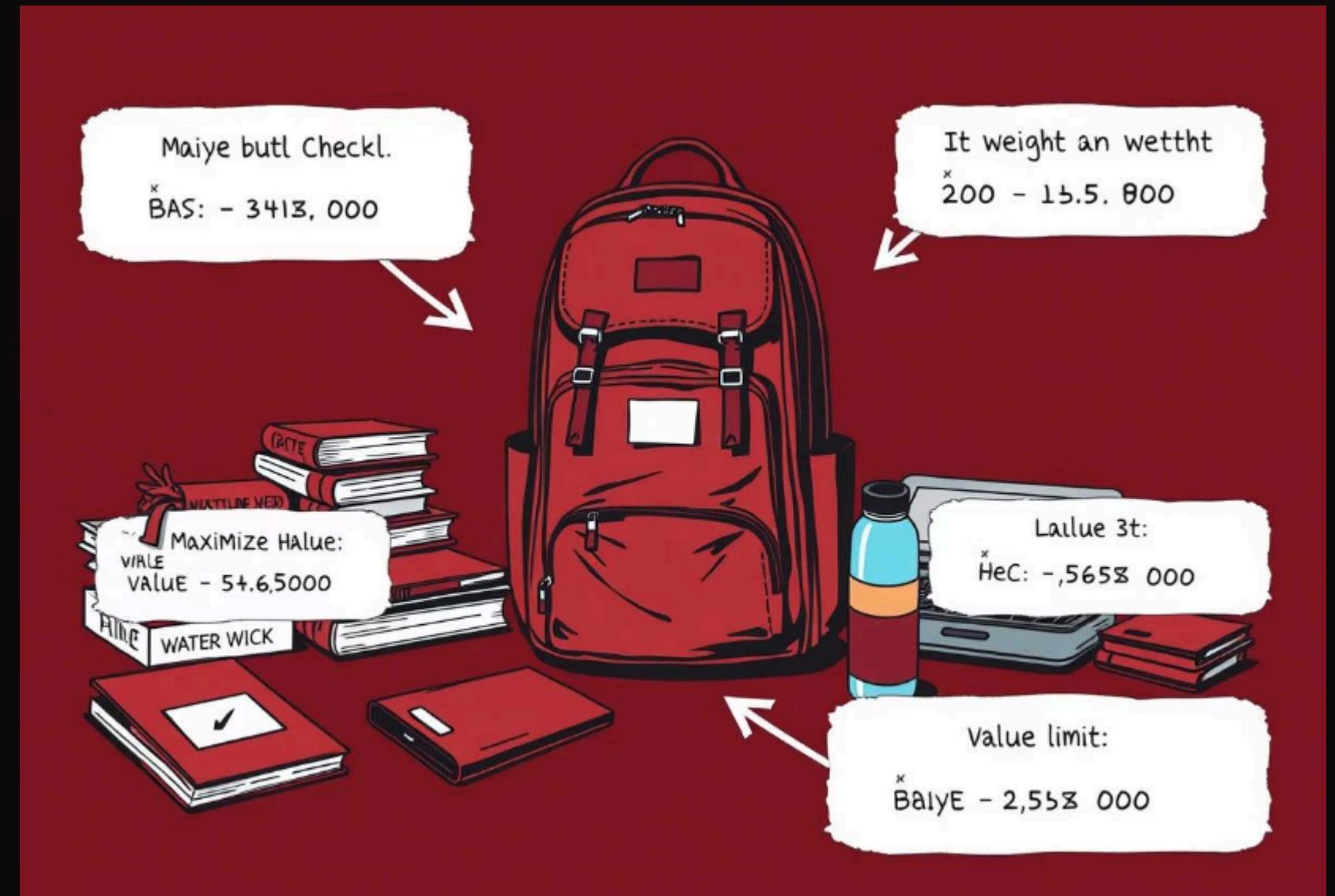
Losowe zmiany dla różnorodności

Powtarzaj

Do osiągnięcia celu lub limitu iteracji

Problem plecaka

Problem plecaka to klasyczne zagadnienie optymalizacyjne polegające na wybraniu takiego zestawu przedmiotów o maksymalnej łącznej wartości, aby ich łączna waga nie przekroczyła pojemności plecaka. Jest to problem NP-trudny, często stosowany do testowania efektywności algorytmów heurystycznych, takich jak algorytmy genetyczne. W praktyce znajduje zastosowanie w logistyce, planowaniu transportu, alokacji zasobów oraz w analizie decyzji w warunkach ograniczeń.



Rozwiążanie problemu plecaka przy użyciu algorytmu genetycznego w języku Rust

https://github.com/wojciechtrapkowski/genetic_knapsack_rust

Koniec

Dziękuję za uwagę!