Eksperymenty obliczeniowe z algorytmami ewolucyjnymi i porównania algorytmów.

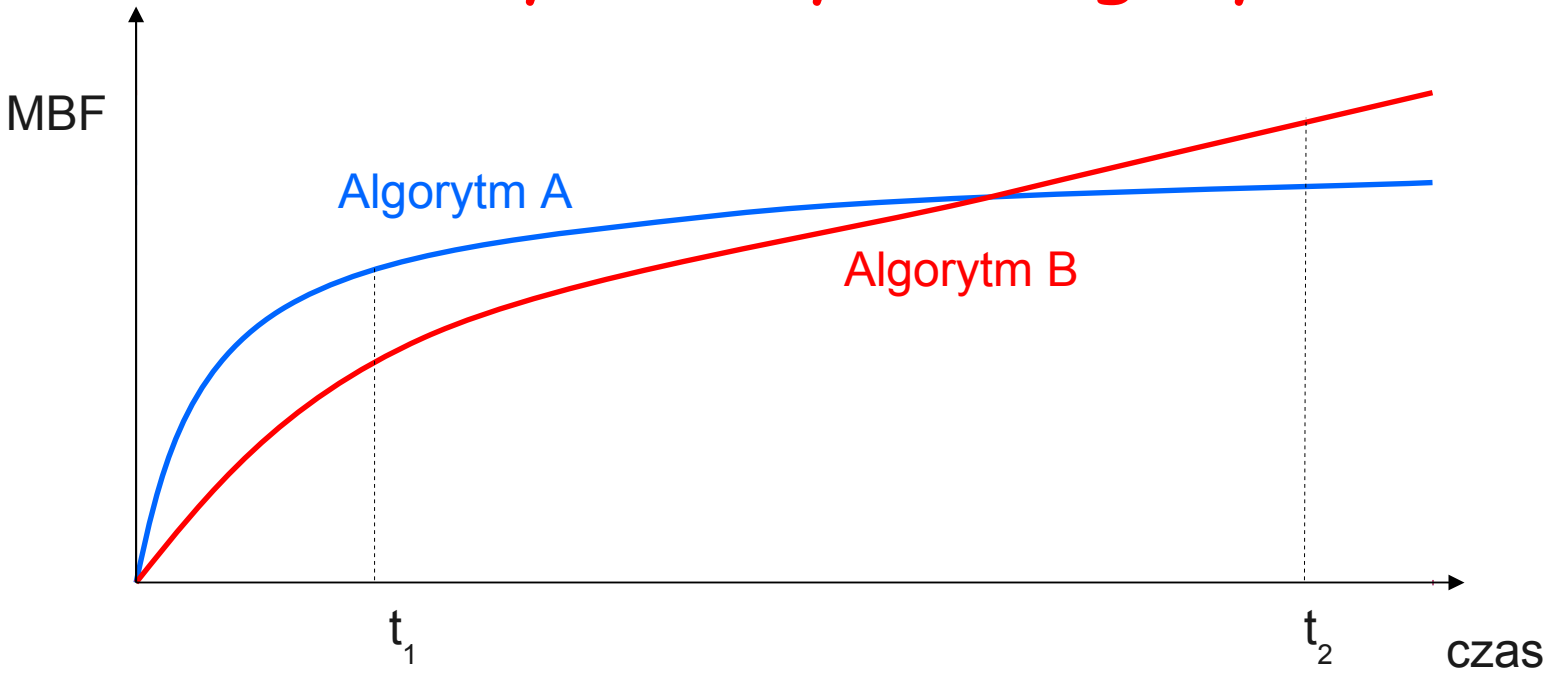
Wprowadzenie

- Do tej pory nie rozważaliśmy odpowiedzi na pytanie, „Po co uruchamiam algorytm ewolucyjny” Kilka możliwych odpowiedzi:
- Mam bardzo trudny problem optymalizacji dyskretnej lub ciągłej i chciałbym go rozwiązać (np. projekt układu scalonego VLSI).
  - Algorytm musi znaleźć bardzo dobre rozwiązanie tego problemu przynajmniej jeden raz.
  - Inne kryteria: czas pracy, powtarzalność wyników są drugorzędne (możemy uruchomić algorytm wiele razy i zapamiętać najlepszy wynik; algorytm może pracować tygodniami).
- Chcę uruchomić algorytm ***wielokrotnie***, dla ***różnych instancji*** problemu, i za każdym razem ***szybko*** otrzymać dobre wyniki. (firma transportowa musi rozwiązać wersję problemu komiwojażera każdego dnia w ciągu 30 minut)
  - potrzebna jest powtarzalność wyników (algorytm ewolucyjny jest algorytmem probabilistycznym).
- Wymyśliłem nowy algorytm ewolucyjny i chcę udowodnić, że mój algorytm jest istotnie lepszy niż klasyczna heurystka x oraz istniejące algorytmy ewolucyjne y oraz z, spełniając standardy publikacji naukowych (dotyczy również prac magisterskich).
  - Istotnie lepszy dla jednej albo wielu instancji problemu.

Miary osiągów algorytmów ewolucyjnych

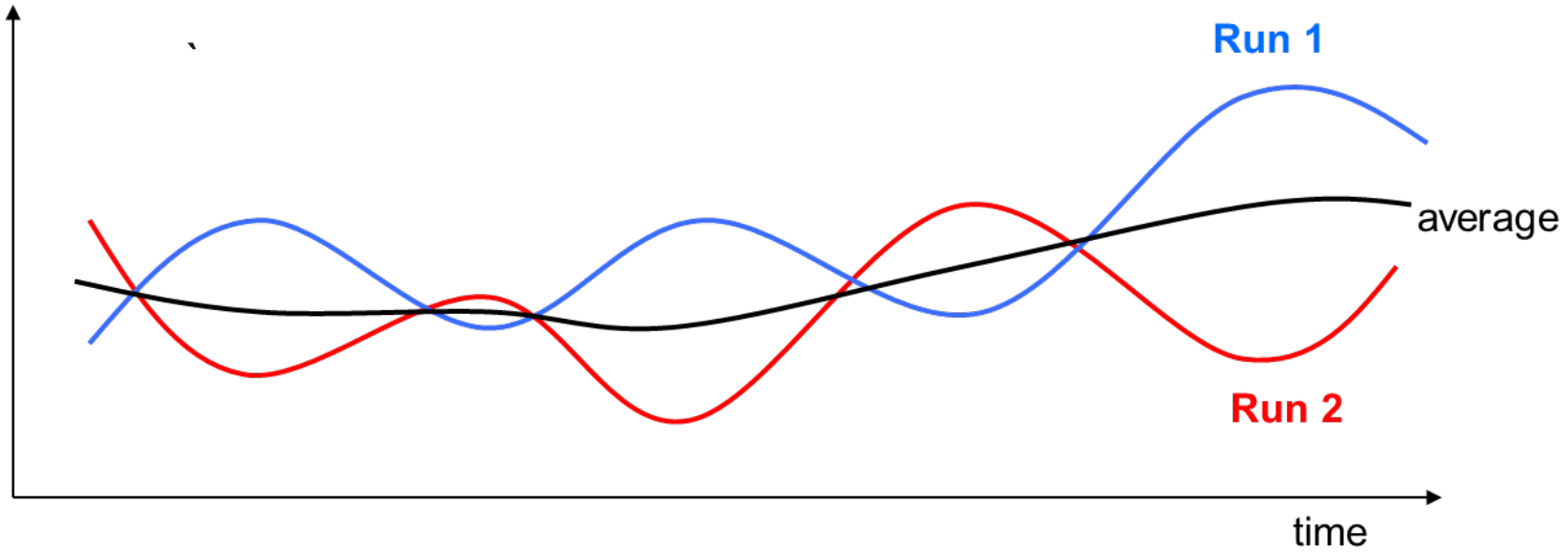
- Probabilistyczna natura algorytmów ewolucyjnych sprawia, że nie możemy polegać na pojedynczym przebiegu algorytmu. Musimy wykonać ich wiele, nawet jeśli mamy tylko jedną instancję problemu.
- Jeżeli znamy położenie optimum, to możemy posłużyć się ***częstością sukcesów*** (ang. success rate - SR) zdefiniowaną jako stosunek liczby przebiegów w których algorytm odniósł sukces do całkowitej liczby przebiegów. Za sukces uważamy sytuację, w której algorytm znajdzie rozwiązanie optymalne lub bliskie optymalnemu (np. o wartości dopasowania nie mniejszej niż ε%).
- Możemy również posłużyć się uśrednionym najlepszym dopasowaniem (ang. mean best fitness - MBF). Wynikiem pracy algorytmu ewolucyjnego jest osobnik o najlepszym dopasowaniu. MBF to średnia z najlepszych dopasowań, liczona po wszystkich przebiegach.
- Zarówno MBF, jak i SR zakładają że wynik algorytmu zostanie osiągnięty po pewnym skończonym czasie. Są one wskaźnikami ***efektywności*** (ang. effectiveness) algorytmu, określającymi jak dobre rozwiązanie może on znaleźć w przydzielonym mu limicie mocy obliczeniowej.

Kiedy zatrzymać algorytm ?



- Krzywe na wykresie, to MBF uśredniony z np. 30 przebiegów (dla każdego z dwóch algorytmów).
- Jeżeli algorytmy zatrzymamy po czasie t_1 wygrywa algorytm A, jeżeli po czasie t_2 to wygrywa algorytm B.
  - Efekt „żółwia i zająca” (ang. turtle and hare).

Problemy z uśrednianiem



- Uśrednianie może „zamaskować” ciekawe informacje.
- Inny przykład wysoki SR bardzo słaby MBF, ponieważ algorytm jeden raz (np. z 30 prób) znalazł bardzo kiepskie rozwiązanie.
  - Dlatego obok MBF mogą być przydatne miary BBF (best best fitness najlepsze z 30 prób) oraz WBF (worst best fitness najgorsze z 30 prób).

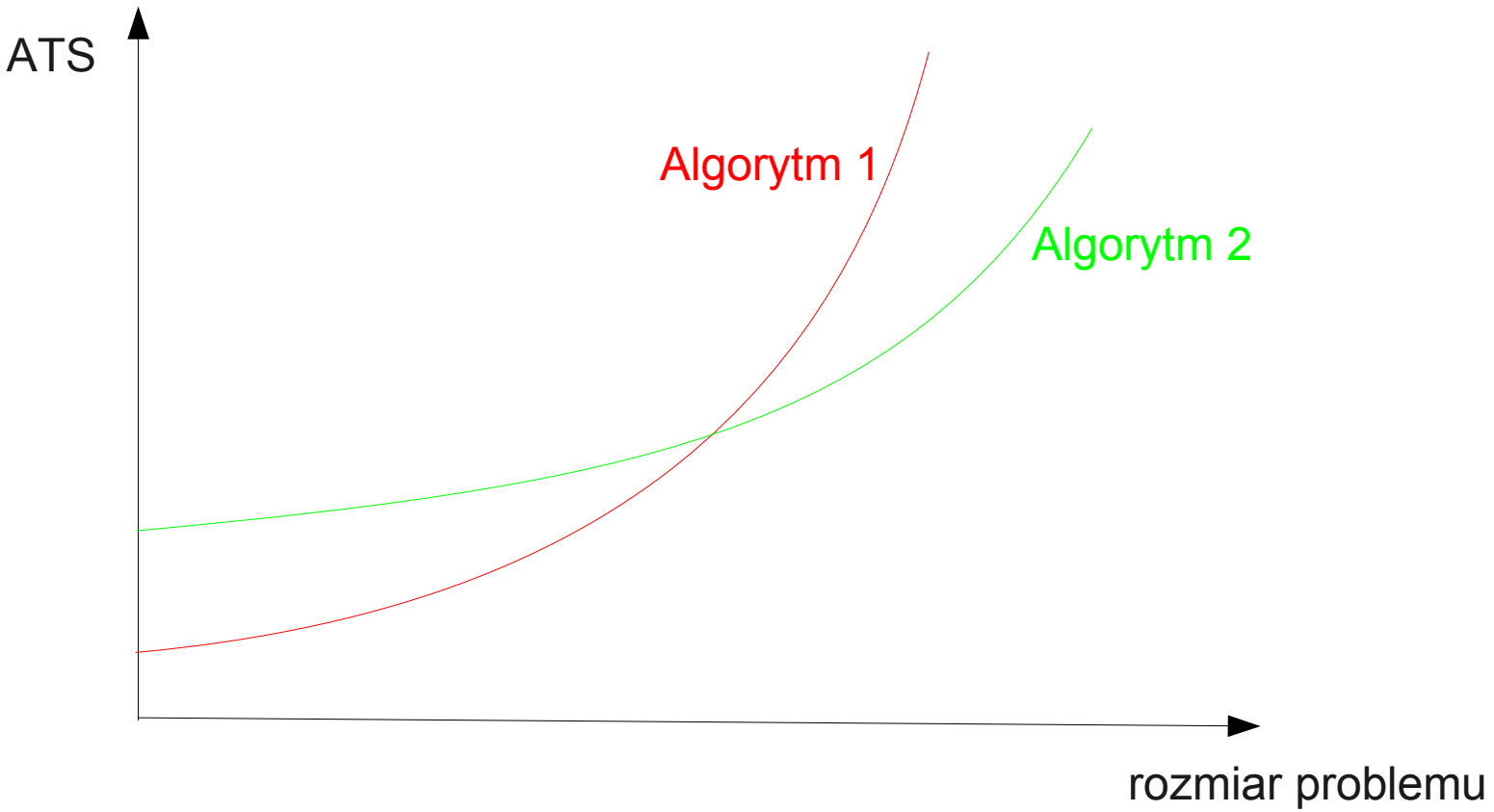
Uczciwe porównania algorytmów

- Podstawowe dwie reguły:
  - Porównuj algorytmy na tych samych problemach.
  - Alokuj te same zasoby (pamięć, zasoby obliczeniowe) różnym algorytmom.
- Jak mierzyć przydzielone zasoby obliczeniowe
  - Mierz rzeczywisty czas pracy programu (ang. elapsed time). Podatny na wpływ sieci oraz obciążenia systemu innymi procesami.
  - Mierz wykorzystany czas pracy procesora (CPU time). Zależy od umiejętności programisty, implementacji (np.. ten sam algorytm zaimplementowany w C++ oraz Pythonie), opcji kompilatora. Ponadto Linux sumuje ten czas po wszystkich procesorach - rdzeniach (nie wykaże korzyści ze zrównoleglenia na maszynie ze wspólną pamięcią).
  - Zliczaj liczbę punktów odwiedzonych w przestrzeni poszukiwań (co jest równoważne ze zliczaniem obliczeń funkcji dopasowania). Nie uwzględnia kosztu innych operacji jak krzyżowanie mutacja i selekcja. Uwaga na algorytmy hybrydowe ! (należy uwzględnić liczbę kroków algorytmu optymalizacji lokalnej).

Miary wydajności (ang. efficiency)

- Typową miarą jest AES (average number of evaluations to a solution) - średnia liczba obliczeń funkcji celu niezbędnych do znalezienia rozwiązania.
  - Może to wymagać to precyzji z jaką ma być znalezione rozwiązanie optymalne, np. jeżeli $f_{min}=0$, to za znalezienie rozwiązania uważamy sytuację, w której algorytm znajdzie rozwiązanie o wartości funkcji celu $<\varepsilon$, gdzie np. $\varepsilon=0.00001$. Jest to jednoczesne kryterium stopu algorytmu.
  - Można również użyć miary ATS (average time to solution), co jest szczególnie pożądane w:
    - porównaniach z heurystykami nieewolucyjnymi, operującymi na rozwiązaniach cząstkowych.
    - sytuacjach w których inne operacje algorytmu np. heurystyczne operatory krzyżowania zajmują sporo czasu.
- Dla wielu problemów np. funkcji testowych w optymalizacji numerycznej możemy zmieniać ich rozmiar (np. liczbę zmiennych). Pozwala to na wyznaczenie charakterystyki skalowalności algorytmu, określającej AES (albo ATS) w funkcji rozmiaru problemu.

Porównanie wydajności



- Wynik ATS dla każdego rozmiaru problemu powinien być uśredniony (np. średnia z 30 przebiegów).
- Który algorytm jest lepszy i dlaczego ?

Porównania statystyczne algorytmów - przykład

Eksperyment	EA_1	EA_2
1	500	657
2	600	543
3	556	654
4	573	565
5	420	654
6	590	712
7	700	456
8	472	564
9	534	675
10	512	643

- Algorytmy ewolucyjne są probabilistyczne z natury.
- Który z algorytmów jest lepszy i dlaczego ?

Przykład (2)

Eksperyment	EA_1	EA_2
1	500	657
2	600	543
3	556	654
4	573	565
5	420	654
6	590	712
7	700	456
8	472	564
9	534	675
10	512	643
Średnia	545.7	612.3
Odchylenie Standardowe	77.58	77.53

- Wartości MBF uzyskane z 10 przebiegów pokazują. że EA_2 może być lepszy niż EA_1.
- Ale proszę zauważyć że wynik EA_2 jest mniej więcej odległy o jedno odchylenie standardowe od EA_1.
- Czy EA_2 jest istotnie większy czy po prostu miał „więcej szczęścia”

Przykład (3)

Eksperyment	EA_1	EA_2
1	500	657
2	600	543
3	556	654
4	573	565
5	420	654
6	590	712
7	700	456
8	472	564
9	534	675
10	512	643
Średnia	545.7	612.3
Odchylenie Standardowe	77.58	77.53
Test t-Studenta (p-value)	0.071	

- p-value (empiryczny współczynnik istotności) – dla testu t-Studenta szansa na zaobserwowanie takiej (lub większej) różnicy w średnich, przy założeniu że wartości oczekiwane w obydwu próbkach są takie same.
- Zazwyczaj jeżeli p-value<0.05 przyjmuje się że różnica jest statystycznie istotna. Przy tym postępowaniu, mamy 1/20 szansę na pomyłkę.
- Uwaga: jeżeli wynik testu nie uprawnia do stwierdzenia że EA_2 jest istotnie lepszy niż EA_1, to również nie uprawnia do stwierdzenia że brak różnicy pomiędzy algorytmami.

Test t-Studenta - założenia

- Rozkład wyników uzyskanych przez, zarówno jeden jak i drugi algorytm jest rozkładem normalnym.
  - Należy sprawdzić, czy test statystyczny (np. Kołmogorowa-Smirnowa) nie odrzuci hipotezy o normalności rozkładów.
  - Jeżeli tak, to zamiast testu t-Studenta należy posłużyć się testem nieparametrycznym Manna-Whitneya, zwanym także testem sumy rang Wilcoxona.
- Wariancje w obydwu populacjach są równe lub próbki mają podobne rozmiary.
  - istnieje wersja testu dla nierównych wariacji i/lub różnych rozmiarów próbek.
- Dane mierzone na skali przedziałowej.
- W przypadku porównania więcej niż dwóch algorytmów należy użyć analizy wariancji (ANOVA) lub w wersji nieparametrycznej testu Friedmana.

Dobór danych testowych – standardowe benchmarki

- Użycie problemów z akademickiego zbioru benchmarków (TSPLib, standardowe funkcje unimodalne i multimodalne).
- Inni badacze też wykorzystują te dane, mamy więc możliwość porównania z szerokim spektrum algorytmów bez konieczności ich re-implementacji.
  - Wszyscy używają tych funkcji więc istnieje ryzyko rozwoju nie lepszych algorytmów ale algorytmów działających lepiej na tych funkcjach.
- Należy zadbać, aby wśród problemów testowych były zarówno problemu unimodalne (i testować na nich wydajność, np.. AES) jak i multimodalne, z bardzo dużą liczbą optimów lokalnych (np. rosnącą wykładniczo wraz z liczbą zmiennych).
- Należy skoncentrować się na funkcjach wielowymiarowych. Użycie wymiaru problemu n=2 to zdecydowanie zły pomysł.

Dobór danych testowych - generatory

- Np. dla problemu komiwojażera użyj zestawu miast, których współrzędne wierzchołków zostały wygenerowane losowo na płaszczyźnie a wagi krawędzi obliczone jako odległość euklidesowa pomiędzy wierzchołkami.
  - Dla tak wygenerowanych danych istnieje przybliżenie, oparte na statystyce, długości najkrótszego cyklu Hamiltona w grafie pełnym.

- Dla problemu plecakowego generuj losowo wartości i wagi przedmiotów w plecacku

- Generalnie idea sprawdza się w przypadku problemów optymalizacji kombinatorycznej, w których możemy generować losowo parametry problemu.

- Ale porównanie z innymi algorytmami może wymagać ich re-implementacji.

Dobór danych testowych - dane rzeczywiste

- Algorytm testowany na problemach rzeczywistych, „wziętych z życia”.
- Wynik jest na pewno ***bardzo istotny*** z punktu widzenia dostawcy danych.
  - Ważne z punktu widzenia komercjalizacji badań naukowych.
- Problemy:
  - Praktyczne zadania mogą być nadmiernie skomplikowane.
  - Dostawca danych może się nie zgadzać na opublikowanie danych/wyników co utrudnia innym badaczom porównanie z naszymi wynikami.
  - Trudno uogólnić wyniki na inne problemy.
  - Trudno uzyskać dane (wymaga kontaktów z przemysłem, etc.)

Podsumowanie

- Wybieraj starannie problemy testowe.
- Uruchom algorytm ewolucyjny wiele razy dla tych samych danych i agreguj wyniki (MBF,SR,AES).
- Porównuj algorytmy uczciwie (te same problemy, ten sam czas obliczeń).
- Wykonaj plan eksperymentów (np. ile zajmą czasu ?).
- Użyj analizy statystycznej (testy hipotez, R).
- Czytelnie przekazuj swoje wyniki (wykresy,tabele).
- Przechowuj wyniki eksperymentalne a nie podsumowania statystyczne (i nigdy ich nie kasuj !).
- Publikuj swój kod i dane (np. w sieci) aby umożliwić innym porównanie ze swomi wynikami.

W projekcie

- Na ocenę najwyższą proszę o zweryfikowanie czy rozkład wyników z 30 przebiegów algorytmu jest normalny i w zależności od wyników testu statystyczne porównanie dwóch algorytmów przy pomocy
  - testu t-Studenta (brak odrzucenia hipotezy o rozkładzie normalnym).
  - testu Manna - Whitneya (hipoteza o rozkładzie normalnym odrzucona).
- Uwaga nie dotyczy osób, które już oddały sprawozdanie.
- Przy okazji przypomnicie Państwo wiadomości z waszego ulubionego przedmiotu, to jest statystyki matematycznej ☺

Literatura dodatkowa

- http://faculty.vassar.edu/lowry/webtext.html. Internetowy podręcznik statytyki i kalkulator on-line, także testów statystycznych.
- http://www.r-project.org/ znany doskonale Państwu, z zajęć statystyki pakiet R (Open Source).
- A,. E. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer, 2003, rozdz. 14 – na podstawie tej książki powstał dzisiejszy wykład.
- A. E. Eiben, M. Jelasity, A critical note on experimental research methodology in EC, CEC '02. Proceedings of the 2002 Congress on Evolutionary Computation, 2002 – krytyczny artykuł na temat aktualnej praktyki eksperymentowania z algorytmami ewolucyjnymi.