Dokumentacja do projektu z ALHE

Rafal Babinski Roman Moskalenko

1 Treść zadania

SK.ALHE.4

Złodziej ukradł X gramów złota ze skarbca i wraca do domu pociągiem. Żeby uniknąć schwytania przez policję, musi zamienić złoto na banknoty, więc postanawia sprzedać złoto pasażerom pociągu. Zainteresowanych kupnem jest N pasażerów, każdy z nich zgadza się kupić a_i, $i \in (1, 2, ..., N)$ gramów złota za v_i, $i \in (1, 2, ..., N)$. Złodziej chce uciec przed policją, jednocześnie maksymalizując zysk. Zaimplementuj program bazujący na algorytmie ewolucyjnym, który wskaże pasażerów, którym powinien sprzedać złoto oraz sumę wartości banknotów, którą zarobi. Zastosowanie i porównanie z innym algorytmem będzie dodatkowym atutem przy ocenie projektu.

2 Projekt wstępny

2.1 Założenia ogólne

Ewidentnie mamy do czynienia z problemem plecakowym zero-jedynkowym. Zakładamy, że $a_i, v_i, X \in \mathbb{R}^+$.

W projekcie będziemy nawiązywać się do artykułu[1], w którym badano efektywność algorytmów genetycznych dla problemu plecakowego zero-jedynkowego. Dalej będziemy odnosić się do niego po prostu jako “artykuł”.

Preferowanym językiem programowania jest Python.

2.2 Planowane rozwiązanie

W ramach projektu planujemy porównać działanie trzech algorytmów:

- **Prosty algorytm genetyczny** – algorytm genetyczny, w którym odrzuca się rozwiązania niespełniające ograniczenia problemu.
- **Algorytm genetyczny zmodyfikowany** – algorytm genetyczny bazowany na artykule. Polega on na poradzeniu sobie z rozwiązaniami niespełniającymi ograniczenia. Do tego zaimplementujemy dwie metody, które najlepiej się sprawdziły wg. wspomnianego artykułu: *zastosowanie w funcji celu logarytmicznej funkcji kary* oraz *zachłanne naprawianie rozwiązań* (do wyboru odpowiednio do pojemności plecaka).
- **Branch-and-Bound** – algorytm, który powinien zapewnić nam rozwiązanie optymalne. Znając rozwiązanie optymalne będziemy w stanie dokładniej oszacować efektywność algorytmów genetycznych.

Oczekujemy, że zmodyfikowana wersja algorytmu genetycznego okaże się efektywniejsza niż wersja prosta.

3 Realizacja projektu

3.1 Generowane danych testowych

Na potrzeby projektu generowano instancje problemu plecakowego w sposób opisany w artykule.

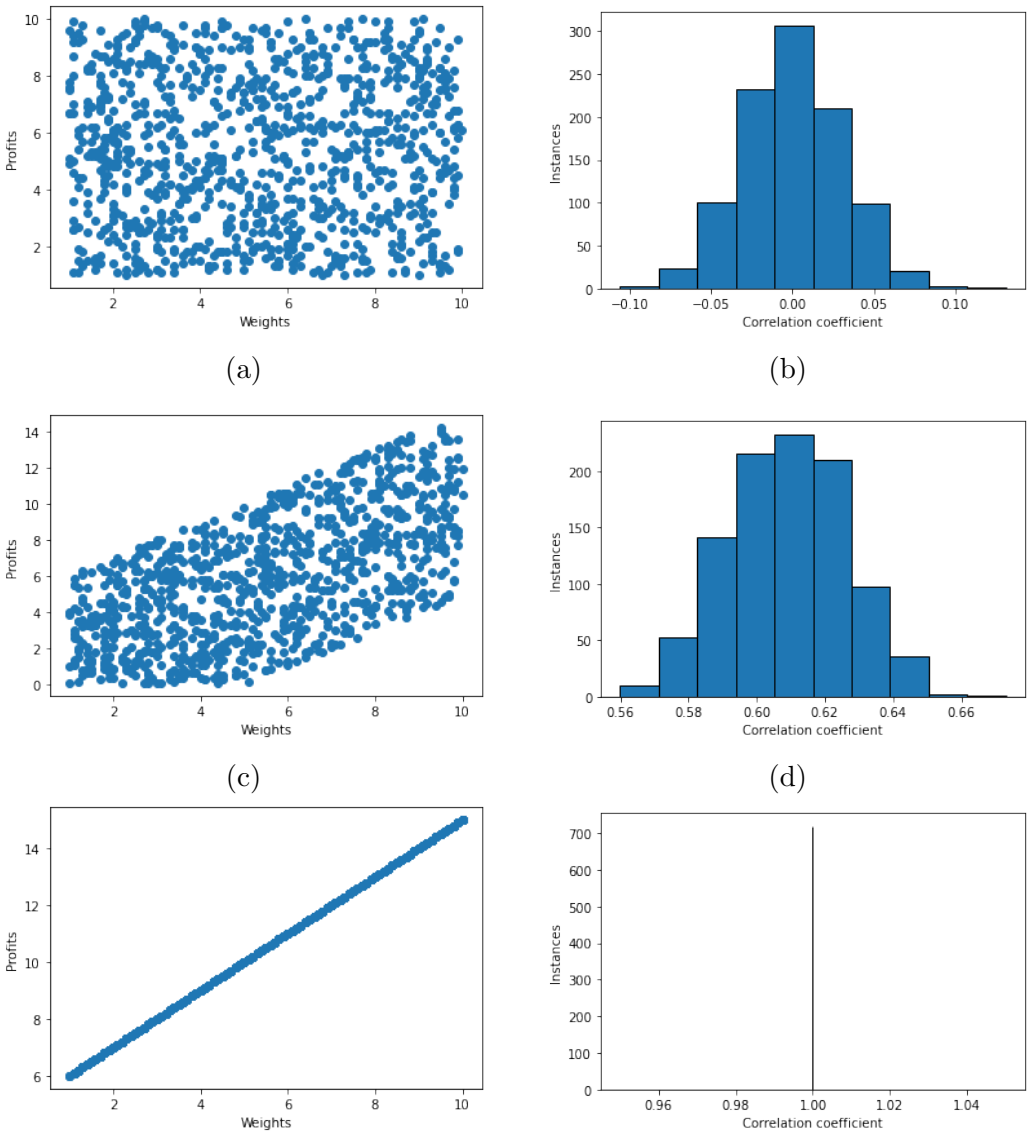
3.1.1 Korelacja

Rozróżnia się instancje, gdzie wagi i zyski są:

- nieskorelowane – współczynnik korelacji w przybliżeniu równy 0,
- skorelowane słabo – współczynnik korelacji w przedziale (0, 1),
- skorelowane mocno – współczynnik korelacji równy 1.

Większa korelacja, jak wspomniano w artykule, implikuje większą trudność problemu.

Duży wpływ na współczynnik korelacji mają parametry v – wartość maksymalna wag oraz r – odchylenie zysku od wagi. Nie będziemy szczególnie badać tej zależności. W dalszych rozdziałach wykorzystane dane zostały wygenerowane z parametrami $v = 10$, $r = 5$.



Rysunek 1: 1000 próbek danych (przy $v = 10, r = 5$) nieskorelowanych (a), skorelowanych słabo (c) i mocno skorelowanych (e). Współczynnik korelacji w 1000 wygenerowanych instancji bez korelacji (b), z korelacją słabą (d) i mocną (f).

Zwrócimy uwagę, że dane skorelowane "słabo" mają współczynnik korelacji w przybliżeniu równy 0.61 co możemy uznać za słabą korelację.

3.1.2 Pojemność plecaka

W artykule opisano 2 typy pojemności:

- Ograniczona – $2v$, gdzie v jest wartością maksymalną wag.
- Średnia – równa połowie sumy wszystkich wag.

Nasz generator umożliwia również ustawienie pojemności wybranej arbitralnie przez użytkownika.

Dla wygody posługiwania generowane dane zostają posortowane według stosunków zysków do wag.

3.2 Branch-and-Bound

Metoda przeglądu drzewa przestrzeni rozwiązań z heurystycznym odcinaniem gałęzi, które nie mają rozwiązania optymalnego. Nasza implementacja tej metody wykorzystuje przegląd drzewa wgłąb.

3.2.1 Wykorzystanie zasobów czasowych i pamięciowych

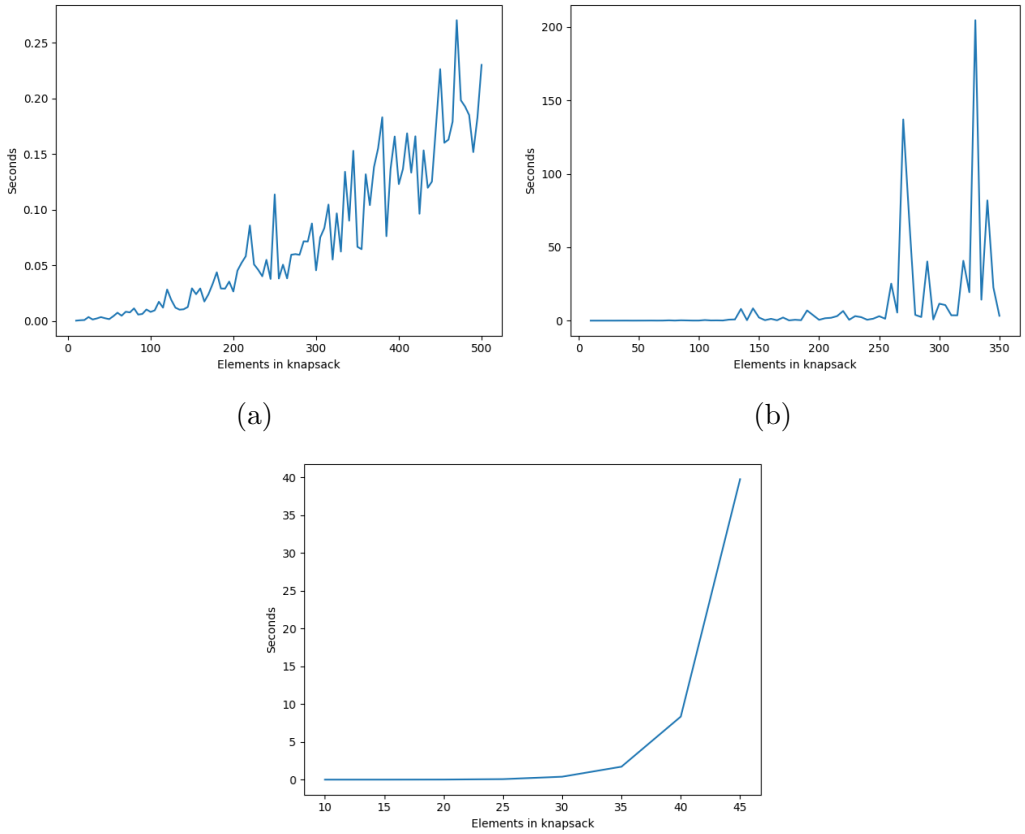
Przedstawimy kilka wykresów opisujących zależność czasu trwania algorytmu od liczby elementów w plecaku przy różnych korelacji i pojemności.

Rysunek 2 dobrze opisuje jak bardzo czas trwania algorytmu zależy od korelacji danych i typu pojemności plecaka. Dla każdego punktu na wykresie algorytm został odpalony 5 razy i został zachowany czas uśredniony.

Jak widać dla słabej korelacji i ograniczonej pojemności algorytm działa bardzo szybko (mniej niż 1 sekunda). Tak się dzieje dla tego, że ograniczona pojemność pozwala algorytmowi dość często odrzucać gałęzie drzewa.

Dla silnej korelacji czas trwania jest rzędu dziesiątek sekund, czasami sięga kilku minut.

Dla powiększonej pojemności plecaka wykres czasu trwania zaczyna przypominać funkcję wykładniczą. Także zauważyliśmy wykorzystanie przez algorytm zbyt dużej ilości pamięci (czasami system operacyjny zabijał program) dla tego postanowiliśmy nie kontynuować pomiaru czasu dla większych wartości.



Rysunek 2: Wykresy zależności czasowej działania algorytmu Branch and Bound od liczby elementów w plecaku przy różnych właściwościach zbioru: (a) słaba korelacja i ograniczona pojemność plecaka, (b) silna korelacja i ograniczona pojemność, (c) słaba korelacja i średnia pojemność.

3.2.2 Ocena algorytmu

Czas trwania i pochłanianie pamięci robią algorytm bardzo nieatrakcyjny. Z tego powodu dalej jeśli zajdzie potrzeba będziemy wykorzystywali ograniczoną wersję algorytmu. Polega to na tym, że ograniczymy maksymalną głębokość drzewa dla algorytmu, poniżej której algorytm nie ma prawa zejść. Wartościami zwracanymi będą oszacowania wartośći górnej i dolnej zysku optymalnego.

3.3 Algorytm genetyczny

Referencje

[1] Zbigniew Michalewicz, Jarosław Arabas. "Genetic Algorithms for the 0/1 Knapsack Problem." 1994.