ALHE

Wykorzystanie VNS do zadania optymalnego doboru publikacji - sprawozdanie

Gabriel Rębacz 283753
Michał Belniak 283663

Przestrzeń poszukiwań i reprezentacja rozwiązania

Przestrzeń poszukiwań X została zdefiniowana jako zbiór wektorów, których poszczególne komórki oznaczają binarną decyzję czy wkład danego autora do danej publikacji zostanie wzięty pod uwagę w trakcie wyliczania ostatecznej wartości punktowej dla Politechniki Warszawskiej.

Przykład: W zbiorze jest 5 publikacji, więc wektor oznaczający wybrane publikacje danego autora ma długość 5 i może mieć przykładową wartość: [1, 0, 1, 1,0]. Oznacza to że w trakcie wyliczania punktów dla politechniki wybrany został wkład autora 1. do jego 1., 3. i 4. publikacji. Analogicznie reprezentujemy wkłady kolejnych autorów.

Zatem przestrzenią poszukiwań dla naszego zadania jest zbiór kombinacji wszystkich wektorów dla każdego autora o długościach będących ilością publikacji i wartościach komórek 0 lub 1.

Funkcja celu

1. Algorytm kary

$$q : X \to R, q = \sum_i \left(\sum_j a_{i,j} w_{i,j} - K\hat{f}\left(\sum_j a_{i,j} u_{i,j} - 4d_i \right) \right) - K\hat{f}\left(\sum_{i,j} a_{i,j} u_{i,j} - 3\sum_i d_i \right)$$

gdzie:

$$\hat{f}(x) = \begin{cases} 1 + x, & \text{jeśli } x > 0 \\ 0, & \text{w p.p.} \end{cases}$$

$a_{i,j}$ - przedstawienie do ewaluacji wkładu autora *i* w dzieło *j*; może mieć wartość 0 albo 1

$w_{i,j}$ - kwant punktowej wartości przeliczeniowej autora *i* w dziele *j*

$u_{i,j}$ - udział jednostkowy autora *i* w dziele *j*; ułamek z zakresu (0,1]

d_i - średnia wartość iloczynu wymiaru czasu pracy i udziału w dyscyplinie autora *i*.

K - kara za udział przekraczający ograniczenie 4 udziałów na współpracownika oraz 3 * N dla uczelni, gdzie N jest sumą etatów wszystkich autorów. Początkowo współczynnik K ma wartość 250, jako że jest to więcej niż autor może maksymalnie wnieść do sumy.

2. Algorytm naprawy

$$q : X \to R, q = \sum_i (\sum_j a_{i,j} w_{i,j}) - \sum_{k,l} w_{k,l}$$

gdzie:

$a_{i,j}$ - przedstawienie do ewaluacji wkładu autora *i* w dzieło *j*; może mieć wartość 0 albo 1

$w_{i,j}$ - kwant punktowej wartości przeliczeniowej autora *i* w dziele *j*

$w_{k,l}$ - kwant punktowej wartości przeliczeniowej autora *k* w dziele *l*, który zostanie usunięty z rozwiązania z powodu przekroczenia ograniczeń (naprawa). Najpierw publikacje będą usuwane tak, aby spełnić ograniczenie dotyczące autora. Następnie, usunięte zostaną publikacje tak, aby spełnić ograniczenie dotyczące uczelni. Dzieło, które ma zostać usunięte z rozwiązania będzie obliczane na podstawie stosunku jego wartości przeliczeniowej do udziału autora w publikacji.

W obu przypadkach problem będzie polegał na maksymalizacji funkcji celu.
W naszych rozważaniach uwzględniliśmy tylko 2 podstawowe ograniczenia, o których była mowa na spotkaniu wprowadzającym.

Określenie sąsiedztwa dla algorytmu VNS.

W algorytmie Variable Neighbourhood Search należy przede wszystkim zdefiniować sąsiedztwo rozwiązania. Reprezentacją rozwiązania jest zbiór wektorów binarnych, więc do jego sąsiedztwa będą należeć takie zbiory wektorów, w których liczba różnic w wartościach elementów na tej samej pozycji będzie nie większa niż parametr r. Wartość tego parametru będzie musiała zostać odpowiednio dobrana, jako że od niej zależy rozmiar sąsiedztwa, który w tym przypadku rośnie ze złożonością kwadratową względem r.

Problemy do rozstrzygnięcia

W przypadku funkcji celu z algorytmem kary warto się zastanowić nad modyfikacją parametrów kary oznaczonych K we wzorze. Zbyt niskie wartości mogą umożliwiać dobranie nieprawidłowych rozwiązań, a zbyt duże może przedłużyć proces wyszukiwania rozwiązań bliskich optymalnym.

Problemy wydajnościowe

W trakcie implementacji pojawiły się problemy wydajnościowe, które sprawiały, że program wykonywał obliczenia bardzo wolno. Analiza wykonania programu za pomocą profilera CProfile, wykazała że najbardziej obciążającą częścią była funkcja *cost_value_function* i *value_function*.

Początkowo funkcje te obliczały wartości funkcji celu poprzez sumowanie wszystkich wartości macierzy punktu i odpowiednich wartości ze zbioru danych w podwójnej pętli *for*. Po zastosowaniu biblioteki numpy i wykorzystania funkcji wbudowanych do jej operacji na macierzach oraz zamiany zmiennych globalnych przechowujących dane wejściowe na parametry funkcji, obliczenia znacznie przyspieszyły.

Zysk wynikał z tego, że operacje macierzowe w numpy zostały napisane w języku C i są one przetwarzane wielowątkowo. Było to nadal zbyt mało by obliczyć w rozsądnym czasie plik "*informatyka_techniczna_telekomunikacja-input.txt*" z takimi samymi progami na liczbę iteracji co w przypadku przetwarzania plików "*filozofia-input.txt*" oraz "*matematyka-input.txt"*. Limit na liczbę iteracji przy przetwarzaniu "*informatyka_techniczna_telekomunikacja-input.txt*" został więc ograniczony do 10*n*. Jeśli chodzi o wartość *n*, uznaliśmy, że jest to wartość *A * P*, gdzie *A* to liczba autorów, a *P* liczba publikacji, lecz okazało się, że wydajniej byłoby przyjąć, że wynosi ona tyle, co liczba par <publikacja, autor> gdzie udział jest różny od 0. Wtedy w punkcie z pojedynczej iteracji można byłoby zawrzeć tylko pary <publikacja, autor>, których udział był różny od 0, co zredukowało by znacząco ilość obliczeń. Jednakże nie zdążyliśmy wprowadzić poprawek i przeprowadzić kolejnych obliczeń.

Ostatecznym rozwiązaniem, które najprawdopodobniej rozwiązałoby problemy wydajnościowe byłoby przechowywanie w pamięci macierzy potrzebnych do wyliczenia wartości funkcji celu i kosztów w algorytmie kary i aktualizacja ich wartości o różnicę wynikającą ze zmiany danych bitów nowo wygenerowanego punktu względem starego, zamiast liczyć funkcję celu od nowa.

Wyniki

Pliki filozofia i matematyka były uruchamiane z parametrami
Maksymalny promień sąsiedztwa = 5,
Obliczanie do progu = 1000n,
Część rozmiaru sąsiedztwa przy iteracji = 1/10,
Początkowy promień = 1,

Plik informatyka był uruchamiany z parametrami
Maksymalny promień sąsiedztwa = 20,
Obliczanie do progu = 10n,
Część rozmiaru sąsiedztwa przy iteracji = 1/300
Początkowy promień = 10

Zmiana parametrów w przypadku pliku informatyka spowodowała znacznie szybsze osiąganie wyższych wartości w przypadku informatyki, gdyż w trakcie generowania pojedynczego zbioru sąsiedztwa algorytm przeglądał punkty, które mogły się różnić między

sobą o większą ilość punktów i dobierał szybciej punkt, gdyż generował mniejszą część sąsiedztwa.
Wartości funkcji celu dla algorytmów kary i naprawy punktów dla danych progów w obliczanych plikach prezentują się następująco:

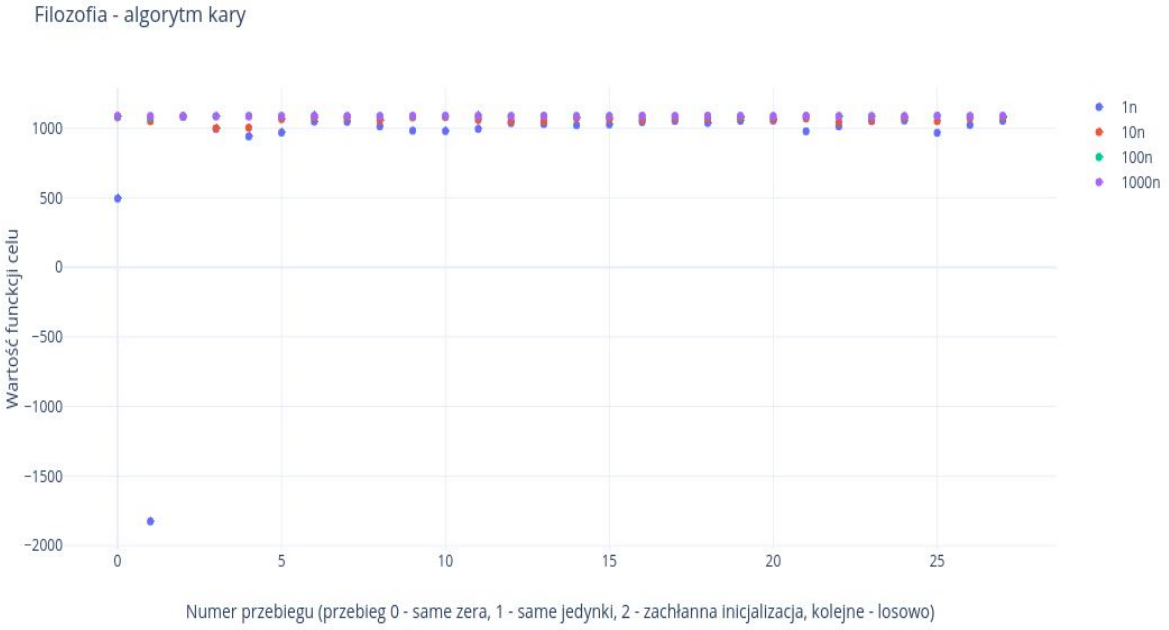
Filozofia - algorytm kary	1n	10n	100n	1000n
same zera	494.7	1078.25	1083.9	1089.15
same jedynki	-1826.55	1050.5	1078	1088.25
zachłanny dobór	1081.05	1086.85	1086.85	1086.85
losowo	994.7	1000.5	1085.75	1086.85
	941.35	1004.35	1086.85	1086.85
	968.6	1065.6	1086.85	1086.85
	1047.95	1074.25	1088.25	1089.15
	1046.4	1072.2	1086.85	1086.85
	1013.85	1056.85	1086.85	1086.85
	982.8	1077.8	1086.85	1089.15
	979.95	1079.95	1088.25	1089.15
	995.3	1060.3	1088.25	1089.15
	1037.45	1049.95	1086.85	1086.85
	1030.8	1051.4	1088.25	1089.15
	1022.15	1071.85	1086.85	1086.85
	1027.35	1066.85	1086.85	1086.85
	1044.75	1061.75	1088.25	1089.15
	1052.5	1068	1086.85	1086.85
	1038.45	1068.45	1088.25	1089.15
	1053.4	1072.55	1085.75	1089.15
	1055.05	1065.05	1086.85	1086.85
	978.4	1070.9	1088.25	1089.15
	1013.95	1043.45	1085.75	1086.85
	1050.4	1058.9	1085.75	1086.85
	1055.6	1071.4	1083.9	1088.25
	967.25	1050.95	1088.25	1089.15
	1023.6	1071.4	1086.85	1086.85
	1053.4	1073.9	1085.75	1089.15

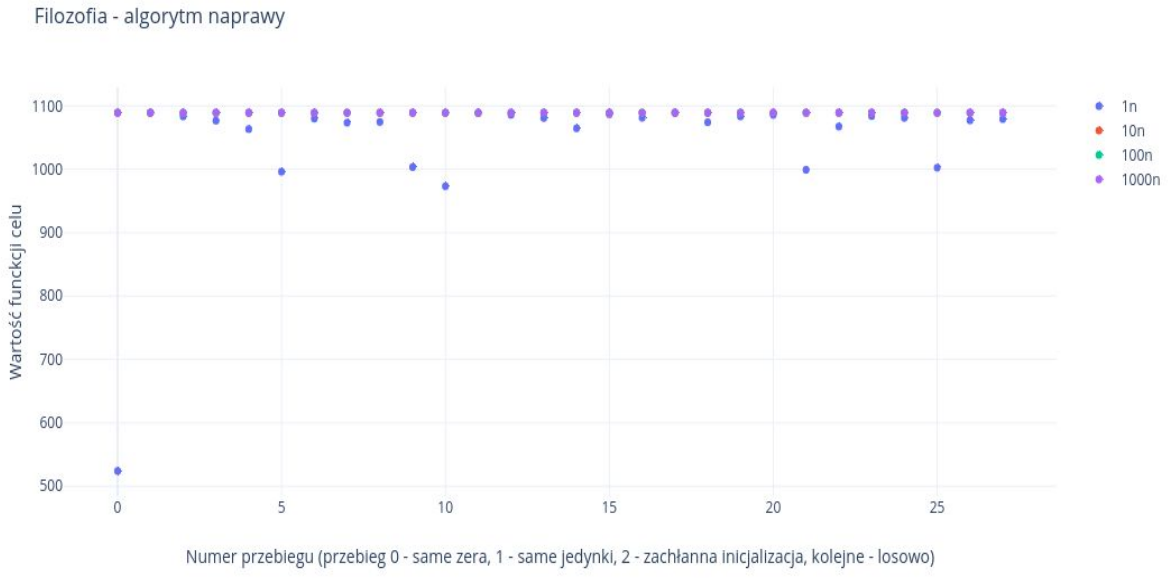
Filozofia - algorytm naprawy	1n	10n	100n	1000n
same zera	524.2	1089.15	1089.15	1089.15
same jedynki	1089.15	1089.15	1089.15	1089.15
zachłanny dobór	1083.35	1089.15	1089.15	1089.15
losowo	1076.65	1089.15	1089.15	1089.15
	1063.35	1089.15	1089.15	1089.15
	996.15	1089.15	1089.15	1089.15
	1079.95	1089.15	1089.15	1089.15
	1073.45	1089.15	1089.15	1089.15
	1074.6	1089.15	1089.15	1089.15
	1003.35	1089.15	1089.15	1089.15
	973.25	1089.15	1089.15	1089.15
	1089.15	1089.15	1089.15	1089.15
	1085.85	1089.15	1089.15	1089.15
	1080.85	1089.15	1089.15	1089.15
	1064.6	1089.15	1089.15	1089.15
	1086.85	1089.15	1089.15	1089.15
	1081.25	1089.15	1089.15	1089.15
	1089.15	1089.15	1089.15	1089.15
	1074.15	1089.15	1089.15	1089.15
	1083.35	1089.15	1089.15	1089.15
	1085.75	1089.15	1089.15	1089.15
	999.15	1089.15	1089.15	1089.15
	1067.45	1089.15	1089.15	1089.15
	1083.9	1089.15	1089.15	1089.15
	1081.05	1089.15	1089.15	1089.15
	1002.45	1089.15	1089.15	1089.15
	1077.1	1089.15	1089.15	1089.15
	1079.15	1089.15	1089.15	1089.15

Matematyka - algorytm kary	1n	10n	100n	1000n
same zera	820	3488.9333	3815.9166	3827.1749
same jedynki	-134.2501	3519.8499	3659.8999	3766.4999
zachłanny dobór	3800.0999	3800.0999	3805.9499	3816.7999
losowo	3458.7749	3468.7749	3654.5249	3810.0166
	3452.3416	3535.6249	3559.9249	3812.8749
	3550.5499	3564.6499	3646.6499	3801.9999
	3569.9749	3594.9749	3675.7749	3815.8499
	3562.7499	3567.6999	3589.7499	3802.5999
	3345.6666	3357.2666	3388.0499	3759.1249
	3500.8416	3540.8416	3616.2166	3821.0499
	3695.6666	3695.6666	3700.9666	3812.6999
	3628.8999	3628.8999	3660.2999	3802.6999
	3626.1166	3626.1166	3655.1666	3818.7666
	3472.7416	3472.7416	3633.5916	3805.4916
	3650.2999	3662.7999	3691.6999	3812.9999
	3580.3166	3587.8166	3609.3166	3812.0999
	3601.6749	3604.6749	3708.7749	3804.9499
	3646.5499	3646.5499	3728.0999	3756.6999
	3522.2749	3533.8749	3554.2999	3778.1999
	3579.7499	3579.7499	3659.7499	3691.8999
	3533.6166	3533.6166	3793.9499	3816.6999
	3551.6916	3551.6916	3683.7749	3796.0999
	3530.3249	3541.9249	3667.3249	3725.1999
	3527.3249	3589.3999	3599.3999	3741.5499
	3555.7749	3571.7999	3714.2499	3814.5999
	3359.0666	3359.0666	3543.8166	3707.6499
	3615.6999	3643.8999	3718.1499	3816.7999
	3457.7499	3457.7499	3589.6999	3732.1499

Matematyka - algorytm naprawy	1n	10n	100n	1000n
same zera	735	3531.025	3834.9749	3834.9749
same jedynki	3825.1249	3828.3749	3834.9749	3834.9749
zachłanny dobór	3801.7499	3824.7499	3833.9749	3834.9749
losowo	3759.5999	3829.0999	3832.3499	3833.3499
	3686.0749	3827.3249	3834.9749	3834.9749
	3447.1666	3820.5166	3832.2666	3833.3499
	3776.1249	3821.9499	3826.7499	3833.3499
	3661.1666	3815.5666	3824.0416	3834.9749
	3740.1999	3830.0999	3833.3499	3833.3499
	3720.5999	3826.8499	3833.3499	3833.3499
	3641.5249	3829.2249	3834.9749	3834.9749
	3676.8499	3818.2499	3833.9749	3834.9749
	3677.7499	3821.4999	3833.3499	3833.3499
	3661.7666	3823.4999	3833.3499	3833.3499
	3664.8749	3823.2999	3833.3499	3833.3499
	3580.7166	3825.7999	3834.9749	3834.9749
	3752.0916	3820.8416	3833.3499	3833.3499
	3633.0416	3831.7249	3834.9749	3834.9749
	3793.3749	3830.8749	3834.9749	3834.9749
	3679.9666	3822.7166	3833.3499	3833.3499
	3618.0999	3818.0999	3824.2499	3833.3499
	3640.6749	3831.3999	3832.3999	3834.9749
	3638.7749	3830.7249	3834.9749	3834.9749
	3565.1749	3823.5749	3834.9749	3834.9749
	3685.2999	3817.7499	3834.9749	3834.9749
	3580.4416	3820.0416	3824.0416	3834.9749
	3746.9166	3826.1166	3833.3499	3833.3499

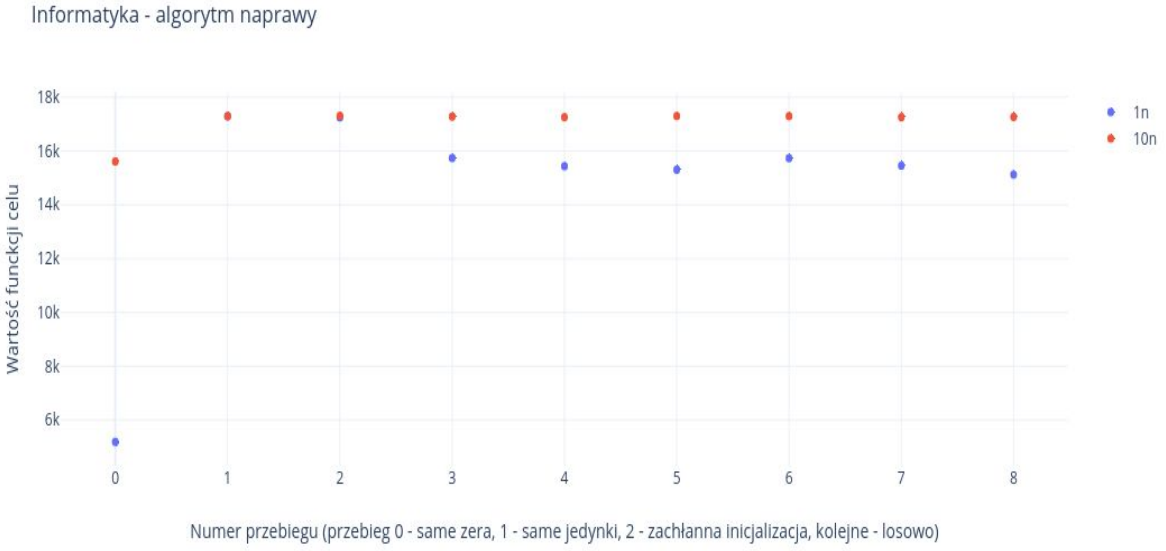
Informatyka - algorytm kary	1n	10n	100n	1000n
same zera	5318.7	15357.3924		
same jedynki	-97514.5301	8025.7971		
zachłanny dobór	17151.3529	17156.6862		
losowo	12313.6726	12484.1552		
	12758.0468	12808.1868		
	12699.3432	12774.1849		
	12180.3192	12484.1109		
	12393.0993	12617.0326		
Informatyka - algorytm naprawy	1n	10n	100n	1000n
same zera	5183.4333	15612.5858		
same jedynki	17295.1912	17297.6662		
zachłanny dobór	17252.782	17310.1153		
losowo	15740.1692	17280.8026		
	15434.4691	17261.5026		
	15304.939	17302.1401		
	15735.6582	17298.3233		
	15455.5809	17259.3409		
	15124.5508	17268.4866		


Filozofia - algorytm kary
Wartość funckcji celu
1000
500
0
−500
−1000
−1500
−2000
0
5
10
15
20
25
Numer przebiegu (przebieg 0 - same zera, 1 - same jedynki, 2 - zachłanna inicjalizacja, kolejne - losowo)
1n
10n
100n
1000n


Filozofia - algorytm naprawy
Wartość funckcji celu
1100
1000
900
800
700
600
500
0
5
10
15
20
25
Numer przebiegu (przebieg 0 - same zera, 1 - same jedynki, 2 - zachłanna inicjalizacja, kolejne - losowo)
1n
10n
100n
1000n


Matematyka - algorytm kary
Wartość funckcji celu
4000
3000
2000
1000
0
0
5
10
15
20
25
Numer przebiegu (przebieg 0 - same zera, 1 - same jedynki, 2 - zachłanna inicjalizacja, kolejne - losowo)
1n
10n
100n
1000n


Matematyka - algorytm naprawy
Wartość funckcji celu
4000
3500
3000
2500
2000
1500
1000
0
5
10
15
20
25
Numer przebiegu (przebieg 0 - same zera, 1 - same jedynki, 2 - zachłanna inicjalizacja, kolejne - losowo)
1n
10n
100n
1000n



Podsumowanie

Pod względem kolejnych rozważanych progów dla liczby iteracji, wyniki są zgodne z oczekiwaniami - większy limit skutkował lepszymi wynikami w każdym rozważanym problemie.

Ciekawe różnice wystąpiły między sposobem inicjalizacji przypisań samymi zerami a samymi jedynkami. Przy wykorzystaniu algorytmu kary, inicjalizacja samymi jedynkami dawała dużo gorsze wyniki dla progu 1*n* niż inicjalizacja samymi zerami. Z pewnością wiąże się to z bardzo dużą karą nałożoną na początku przebiegu przy wybraniu wszystkich prac (same jedynki). Natomiast przy wykorzystaniu algorytmu naprawy, to inicjalizacja samymi zerami dawała dużo gorsze rezultaty, a inicjalizacja samymi jedynkami przynosiła bardzo dobre wyniki. Również jest to uzasadnione, gdyż przy przypisaniu samych zer, algorytm

naprawy nie wykonuje żadnej pracy, ponieważ wszystkie limity są spełnione. Skutkuje to niską wartością funkcji celu osiągniętą przy limicie 1n (w zasadzie w zerowej iteracji funkcja celu wynosi 0). W obydwu przypadkach, różnice zanikają wraz ze zwiększeniem limitu na liczbę iteracji.

Warto zwrócić uwagę na inicjalizację wektorów metodą zachłannego doboru publikacji. Nawet dla limitu 1*n* uzyskano bardzo dobre wyniki. W przypadku informatyki, taki sposób inicjalizacji okazał się najlepszy spośród rozważanych.