Tomasz Dzięcioł 132216 tomasz.dzieciol@student.put.poznan.pl
Ewa Fengler 132219 ewa.fengler@student.put.poznan.pl

Bioinformatyka, laboratoria środa godz. 11:45 – 13:15

Sprawozdanie 2
Zaimplementowanie i przetestowanie heurystyki

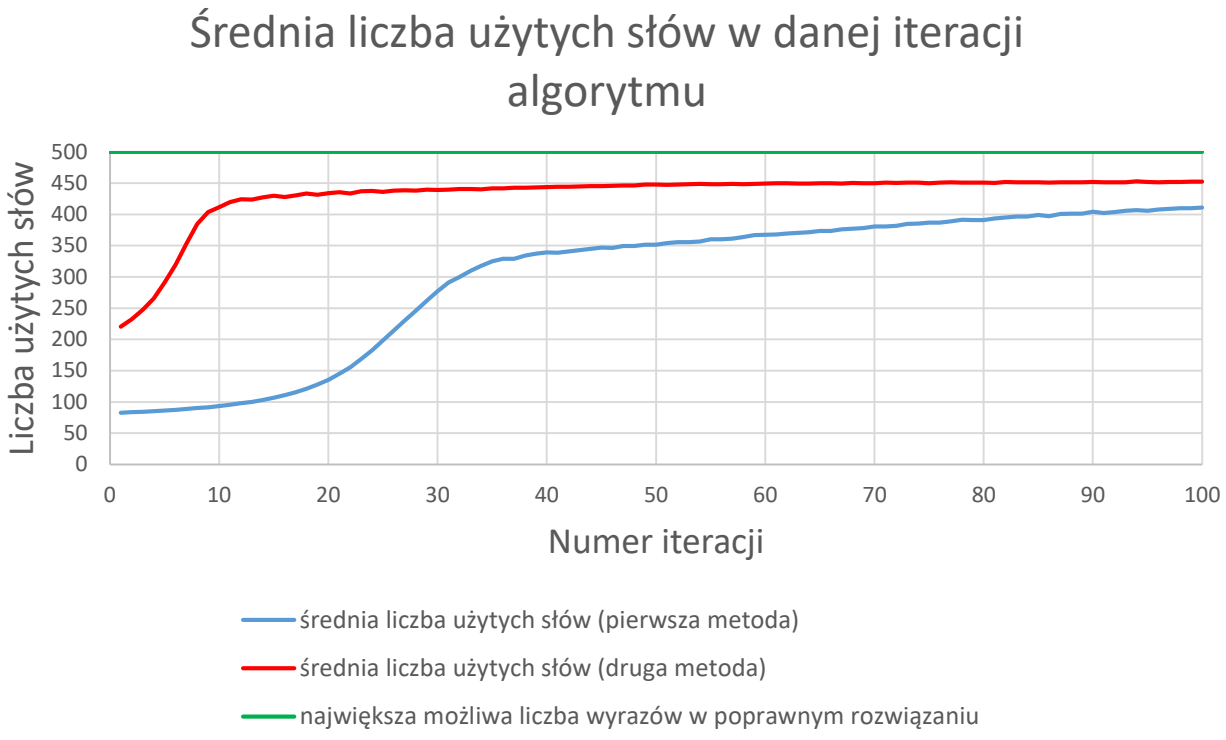
1. Wybór metody inicjalizowania początkowych szans dla wyboru kolejnych wyrazów.

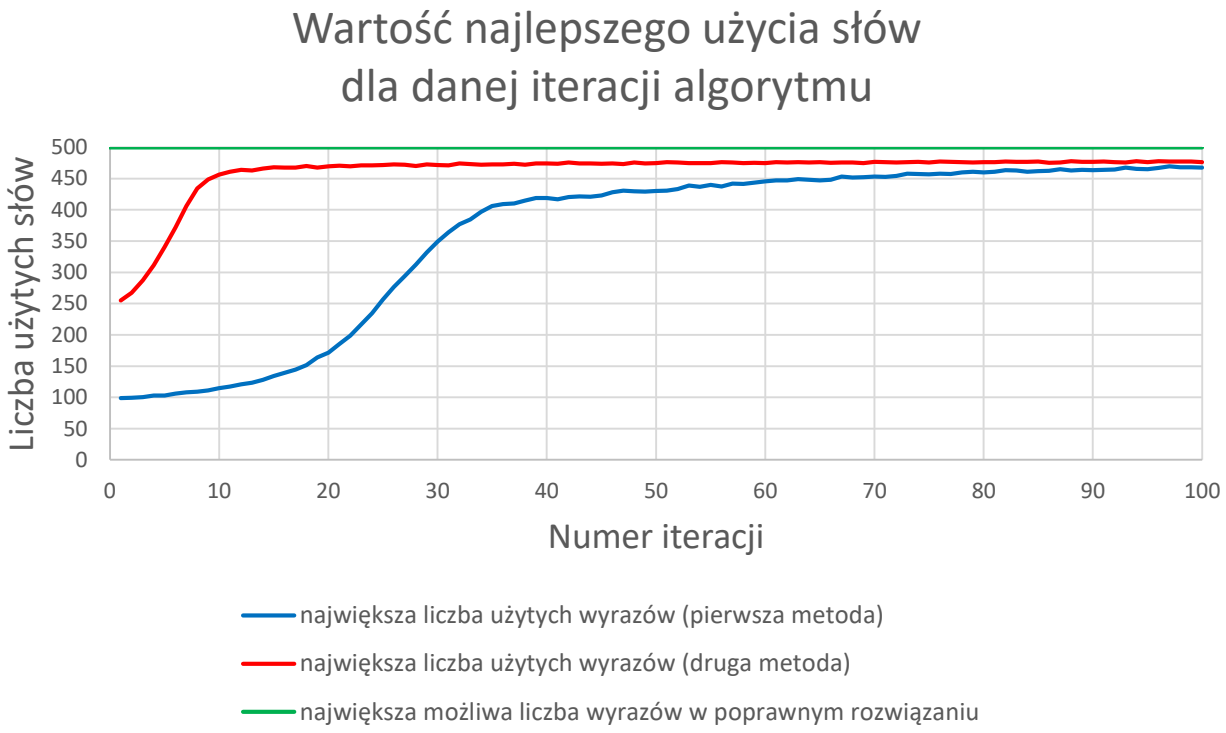
Zaprezentowany przez nas w pierwszej części sprawozdania algorytm został zaimplementowany zgodnie z przedstawioną ideą. Zanim został on ostatecznie przystosowany do zwracania jak najlepszych wyników, algorytm ten musiał przejść szereg testów parametryzujących. Pierwszym problemem wymagającym rozstrzygnięcia był szeroko opisywany wcześniej sposób początkowego przypisania wag prawdopodobieństwa dla wyboru kolejnych wyrazów podczas tworzenia sekwencji.

W obu strategiach wykorzystany miał być pomysł k-krotnie większej szansy wyboru dla kolejnego wyrazu przyłączanego do sekwencji, który to by miał wspólny początek z końcem sekwencji o jednym znaku dłuższym. Tak oto skonstruowana wartość prawdopodobieństwa wyboru w pierwszym przypadku miała zostać przypisana każdej parze wyrazów odpowiednio w zależności od długości wspólnych końcówek. Drugie podejście brało dodatkowo pod uwagę liczbę możliwości doboru kolejnego słowa o danej długości wspólnej końcówki. Wówczas dane prawdopodobieństwo nie było przypisywane, lecz rozdzielane równomiernie na wszystkie kombinacje tworzące zazębiające się końcówki o danej długości.

Kwestia ustalenia początkowych szans wyboru jest znacząca dla całego przetwarzania heurystyki. Ich wpływ najlepiej powinien być widoczny w jakościach otrzymywanych rozwiązań na przestrzeni kolejnych iteracji. W związku z tym zostanie przeprowadzone zebranie średnich wartości rozwiązań otrzymywanych w każdej kolejnej iteracji. Do eksperymentu posłuży duża instancja z pozytywnymi losowymi błędami – 59.500+200. Tak duży zbiór wyrazów powinien uwypuklić ewentualne różnice między opisanymi strategiami.

Algorytm w trakcie tego badania będzie wykonywać 100 iteracji, próbując w każdej z nich 100 razy odtworzyć sekwencję. Pozostałe parametry, które zostaną jeszcze przeanalizowane, są ustawione na domyślne wartości uznane za odpowiednie na etapie opisu stosowanej heurystyki. Ważne jest też, aby dla danej metody inicjalizacji wag nie wykonywać algorytmu mrówkowego tylko raz, który swoje działanie opiera na losowości. Aby móc zauważyć dane właściwości i móc wyciągnąć z nich poprawne wnioski, należy powtarzać całą heurystykę wielokrotnie. Dopiero po uśrednieniu otrzymywanych wyników można pozwolić sobie na analizę wykorzystanej wersji algorytmu. Taka metoda wielokrotnego powtarzania algorytmu będzie również zastosowana przy doborze odpowiednich wartości parametrów.





Na zaprezentowanych powyżej wykresach wyraźnie widać, która metoda inicjalizacji wag rozkładu prawdopodobieństwa działa lepiej. To dla drugiej metody przeciętnie otrzymywane wyniki w kolejnych iteracjach są zdecydowanie wyższe. Widać to zarówno w średniej liczbie wykorzystanych słów we wszystkich próbach, jak i w najlepszej próbie z użyciem największej ilości wyrazów. Już po pierwszych 10 iteracjach druga strategia równomiernego rozdzielenia danej wagi prawdopodobieństwa daje najlepsze wyniki na poziomie przekraczającym 90%, podczas gdy dla pierwszej metody poziom ten jest osiągany dopiero 7 razy później. Otrzymane wyniki wyraźnie pokazują jak znacznie lepsza jest druga metoda przypisania początkowych szans prawdopodobieństwa wyboru kolejnego wyrazu.

2. Dobór odpowiednich wartości parametrów.

W zaimplementowanej przez nas wersji algorytmu mrówkowego można wyróżnić kilka parametrów, którym można by nadać różne wartości. Od ich wartości będzie zależeć przebieg działania algorytmu – odpowiednie ich dobranie skutkować będzie otrzymywaniem jak najlepszych rozwiązań. Parametry te to:

- *iterations_number* - liczba iteracji,
- *attempts_number* - liczba prób w iteracji,
- *k_set_starting_wages* - wartość k stosowana przy k-krotnie większej szansie wyboru kolejnego wyrazu,
- *part_of_the_best_sequences* - część zbioru najlepszych rozwiązań podlegający nagrodzeniu (wartość domyślna: $0{,}1$),
- *best_price_for_sequence* - wartość zwielokrotnienia szansy wyboru połączenia par wyrazów będących częścią nagradzanych rozwiązań (wartość domyślna: $1{,}2$),
- *compensation_coef* - współczynnik dominacji jednego połączenia nad pozostałymi, określający moment przeprowadzania wygładzania wag prawdopodobieństwa (wartość domyślna: $0{,}98$),
- *base_of_logarithm* - podstawa logarytmu używanego we wzorze wygładzającym dominację najlepszego połączenia (wartość domyślna: 2).

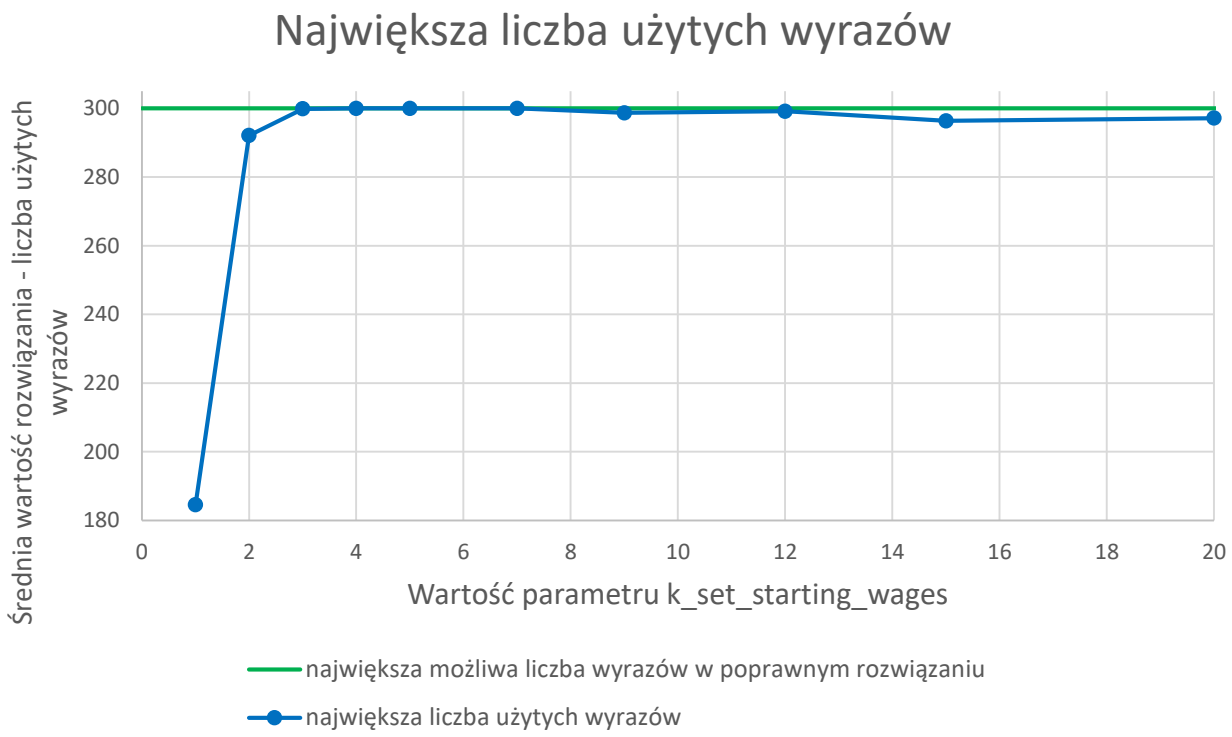
Kolejnym etapem jest więc ustalenie najlepszych wartości dla tych parametrów. Pierwsze dwa z nich są dość specyficzne.. Od liczby iteracji i ilości powtórzeń zależy bowiem długość działania algorytmu. Jednocześnie przy dłuższej pracy programu losowe szukanie rozwiązania doprowadzi kiedyś do coraz to lepszych rozwiązań. Mając to na uwadze można dojść do wniosku, że aby uzyskać najlepsze wykorzystanie wyrazów w sekwencji, powinno uruchomić program na nieskończenie długi czas. W związku z tą zależnością przyjęty został następujący plan działania algorytmu, który można uznać za kompromis pomiędzy pozwoleniem na jak najdłuższą pracę programu, a koniecznością podania ostatecznych wyników w rozsądnym czasie. Ustalona została bowiem stała liczba 100 powtórzeń prób utworzenia sekwencji w danej iteracji, podczas gdy sam algorytm zostaje przerwany, gdy od czasu wykonywania pewnej stałej liczby iteracji nie zostaje otrzymane lepsze rozwiązanie. Wystarczająca wydaje się być liczba 30 iteracji bez wzrostu funkcji celu, która to wyznaczać będzie koniec pracy algorytmu mrówkowego.

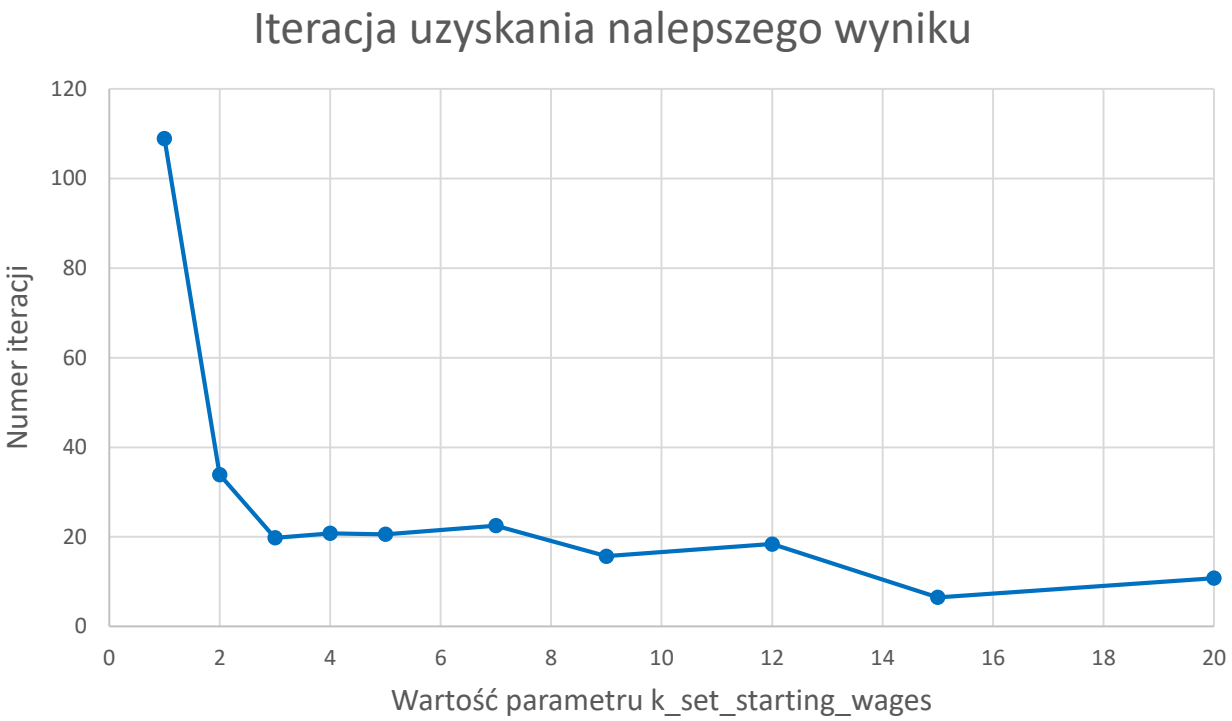
Dzięki wprowadzonym ograniczeniom pozostaje pięć parametrów wymagających ustalenia na podstawie analizy otrzymywanych rezultatów algorytmu. Sposób przeprowadzenia parametryzacji jest kluczowy w kwestii przeprowadzenia go w skończonym czasie. Chcąc bowiem sprawdzić jak działa heurystyka przy 10 różnych wartościach dla każdego z 5 parametrów, trzeba by zebrać wyniki dla wszystkich 10^5 kombinacji przypisania parametrom wartości. W świetle wykonywania jednorazowego przebiegu algorytmu w czasie liczonym w sekundach, przeprowadzenie algorytmu dla wszystkich kombinacji dla jednej instancji testowej staje się praktycznie nie do wykonania.

Konieczne okazuje się więc dokonywanie testów wszystkich rozpatrywanych wartości danego parametru przy domyślnych wartościach pozostałych niezbadanych jeszcze współczynników. Wówczas wybierana jest wartość parametru, dla którego uzyskiwane są najlepsze wyniki, po czym można przejść do badania kolejnego parametru. Tym sposobem powinno zostać odkryte zestawienie parametrów składających się na bardzo dobre sprawowanie się algorytmu.

Aby móc wykonać omówiony proces parametryzacji, należy jeszcze tylko obrać instancje problemu, przy której dokonywana będzie analiza otrzymywanych przeciętnie wyników. Ze względu na wielokrotnie wykonywanie całego algorytmu, nie można pozwolić na obranie zbyt dużej instancji wykorzystującej co najmniej 500 wyrazów do odtworzenia wynikowej sekwencji. Odpowiednią instancją do tego zadania powinien być problem z rodziny 300+100 – problem z błędami pozytywnymi. Dodatkowym atutem takiej instancji oprócz nie dużego rozmiaru jest stosunkowo największa trudność w odnalezieniu oryginalnej sekwencji. Własność ta została zaobserwowana przy wstępnie wykonywanych obserwacjach uzyskiwanych wyników algorytmu mrówkowego w zależności od rodzaju instancji problemu. Trenowanie parametrów pod trudny przypadek powinno przynieść również lepsze rezultaty w późniejszym rozwiązywaniu potencjalnie łatwiejszych problemów.

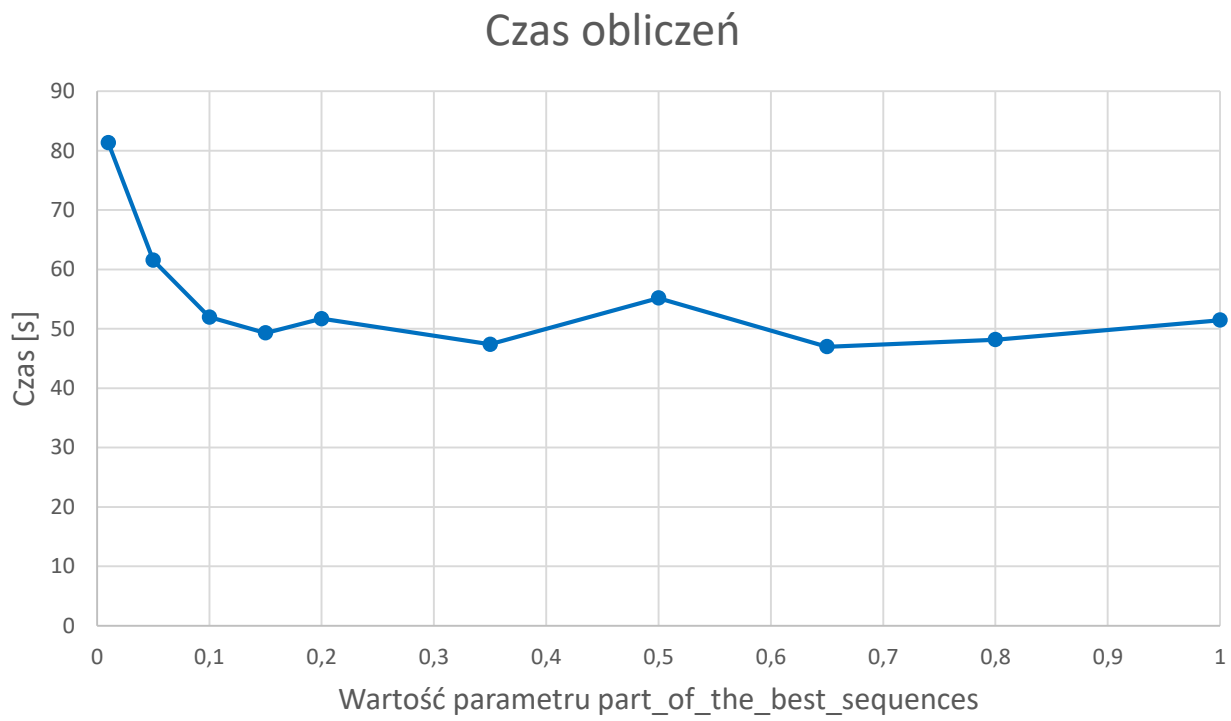
a) Wartość k stosowana przy k-krotnie większej szansie wyboru kolejnego wyrazu.

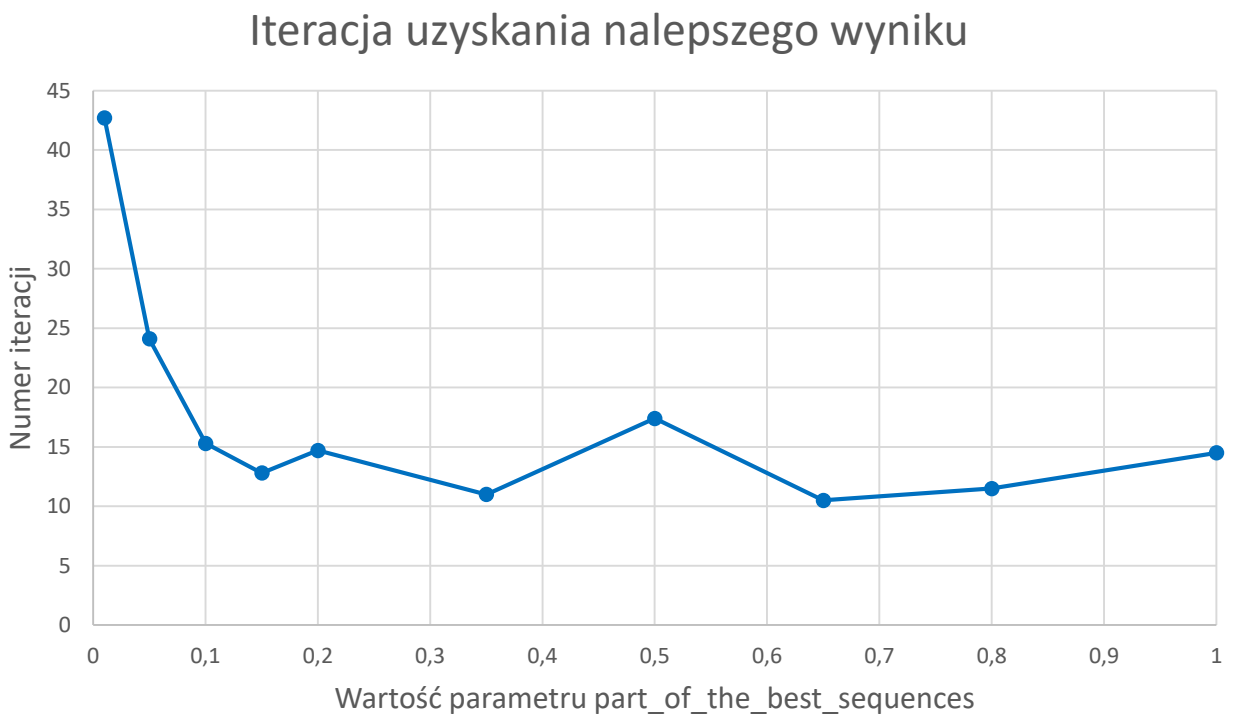




Pierwszy wykres przedstawiający średnią jakość otrzymywanych rozwiązań pokazuje, że dla wartości parametrów większego niż 2 heurystyka znajduje już praktycznie najlepsze rozwiązanie. Co więcej, najlepsze wyniki zostają znalezione już w około dwudziestej iteracji. Odstającym przypadkiem jest tu wartość współczynnika równa 1, dla której otrzymywane wyniki są bardzo słabe. Taka wartość parametru oznacza równe szanse połączenia danych par wyrazów, bez względu na długość zachodzących na siebie końcówek. Przypadek ten pokazuje więc, że nie uwzględnienie długości wspólnego sufiksu-prefiksu skazuje jakiekolwiek próby poszukiwania rozwiązania na całkowitą porażkę. Najlepszą wartością dla tego współczynnika będzie 5, dla której znajdywane wcześnie rozwiązania osiągały w trakcie testów zawsze maksymalną wartość.

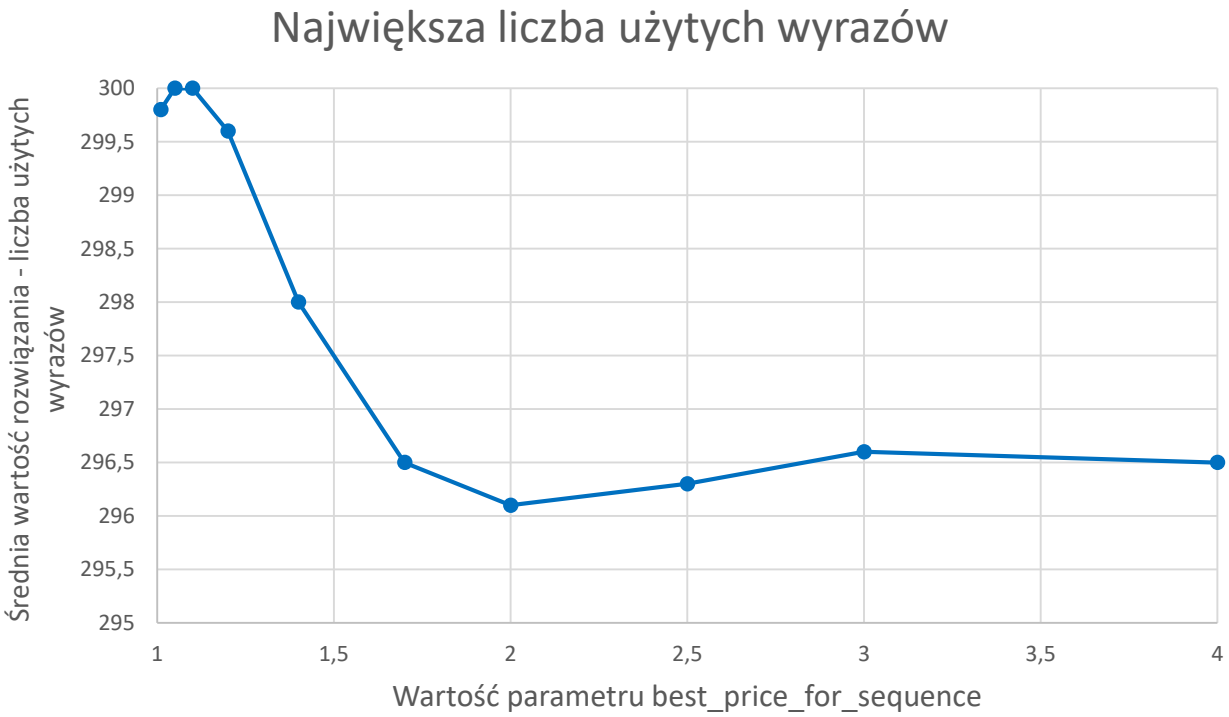
b) Część zbioru najlepszych rozwiązań podlegający nagrodzeniu.

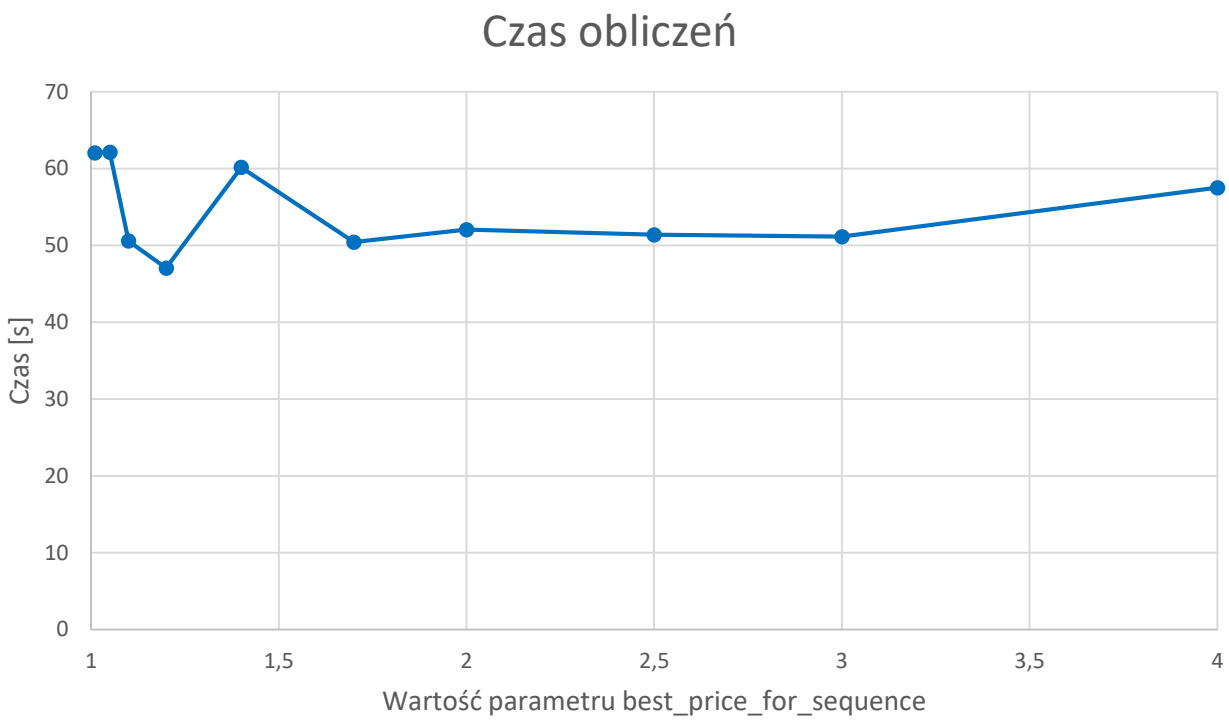




Dzięki wybraniu już odpowiedniego współczynnika w poprzednim badaniu, teraz podczas obserwowania działania algorytmu otrzymywane wyniki niemal zawsze były maksymalne. Warto tutaj zatem skupić swoją uwagę na innych pomiarach - czas obliczeń oraz iterację uzyskania najlepszego wyniku. Słusznie można tutaj zauważyć, że wczesne znalezienie najlepszego ułożenia wyrazów skutkuje szybkim zakończeniem pracy heurystyki. Związane jest to z poczynionym założeniem odnośnie liczby wykonywanych iteracji, które są dopiero kończone, gdy po wykonaniu 30 kolejnych kolejek algorytmu nie następuje poprawa funkcji celu. Uzyskane wykresy wskazują na wybieranie 65% $(0{,}65)$ najlepszych utworzonych sekwencji, które posłużą do nagrodzenia wykorzystywanych w nich połączeń par wyrazów.

c) Wartość zwielokrotnienia szansy wyboru połączenia par wyrazów będących częścią nagradzanych rozwiązań.





Podczas dobierania odpowiedniej wartości czynnika zwielokratniającego należy zwrócić uwagę na średnią wartość uzyskiwanych najlepszych rozwiązań. Praktycznie są one niemal jednakowe, bliskie maksymalnej wartości wykorzystania trzystu wyrazów. Wymagając jednak od algorytmu rekonstrukcji najlepszych sekwencji, trzeba postawić na niskie wartości badanego parametru. Najlepszym wówczas rozwiązaniem staje się wartość 1,1.

d) Współczynnik dominacji jednego połączenia nad pozostałymi, określający moment przeprowadzania wygładzania wag prawdopodobieństwa.

compensation_coef	czas obliczeń	największa liczba użytych wyrazów	iteracja uzyskania najlepszego wyniku
0,5	53,73755	300	16,4
0,7	52,90251	300	15,8
0,8	52,51591	300	15,3
0,85	51,92842	300	14,8
0,9	48,75428	300	12,3
0,92	47,51673	299,5	11,2
0,95	53,36137	300	16,2
0,97	54,85823	300	17,4
0,98	49,74286	300	13
0,99	48,92409	300	12,3

Wyniki badań dla współczynnika dominacji zostały przedstawione w powyższej tabeli. Różnice pomiędzy poszczególnymi wartościami współczynnika prawie nie występują. Wyróżnić można warianty, dla których praca algorytmu zakończyła się przed upływem 50 sekund. To dla nich odnalezienie najlepszego rozwiązania miało miejsce średnio już w pierwszych kilkunastu iteracjach. Wartość parametru compensation_coef=$0{,}92$ budzi zastrzeżenia ze względu na nie osiąganie maksymalnego użycia wyrazów. W tych okolicznościach najbardziej atrakcyjną staje się wartość $0{,}9$.

e) Podstawa logarytmu używanego we wzorze wygładzającym dominację najlepszego połączenia.

base_of_logarithm	czas obliczeń	największa liczba użytych wyrazów	iteracja uzyskania najlepszego wyniku
2	53,84044	300	16,5
3	47,52177	300	11,2
4	51,02738	300	14,2
5	47,49789	300	11,1
7	50,53964	300	13,9
10	51,42452	300	14,6
13	50,05657	299,6	13,4
16	50,33539	299,6	13,8
20	51,31465	300	14
25	47,77991	299,2	11,3

Zmiana ostatniego z parametrów nie wprowadza drastycznych zmian w procesie przetwarzania heurystyki. Jednakże, zbyt wysoki dobór podstawy logarytmu potrafi przeszkadzać w przeciętnym osiąganiu maksymalnego rozwiązania. W celu ustalenia ostatecznego zestawu wartości parametrów, wartość ostatniego współczynnika zostanie dopasowany do czwartego wariantu: 5.

3. Działanie algorytmu na obowiązkowych zbiorach instancji testowych.

Po przeprowadzeniu szerokich testów parametrów, stworzony algorytm jest gotowy do zastosowania go na zbiorze instancji testowych. Wyniki dla poszczególnych zestawów poddanych działaniu stworzonego algorytmu mrówkowego prezentują się w poniższej tabeli.

Instancje z błędami negatywnymi losowymi					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
9.200-40	160	160	100,00%	2	12,72682
9.200-80	120	120	100,00%	2	9,265264
10.500-100	400	400	100,00%	3	38,85622
10.500-200	300	299	99,67%	3	26,22421
18.200-40	160	160	100,00%	3	12,87104
18.200-80	120	117	97,50%	3	8,754714
20.300-120	180	178	98,89%	3	14,70514
20.300-60	240	240	100,00%	3	20,59669
25.500-100	400	399	99,75%	3	37,56611
25.500-200	300	299	99,67%	38	53,29983
35.200-40	160	159	99,38%	3	12,65795
35.200-80	120	119	99,17%	8	10,06328
53.500-100	400	391	97,75%	3	37,48876
53.500-200	300	300	100,00%	5	27,91457
55.300-120	180	180	100,00%	4	14,71313
55.300-60	240	237	98,75%	3	20,17207
55.400-160	240	236	98,33%	40	39,83382
55.400-80	320	320	100,00%	8	32,60532
58.300-120	180	178	98,89%	3	14,22303
58.300-60	240	236	98,33%	3	20,16897
62.400-160	240	240	100,00%	34	38,11009
62.400-80	320	320	100,00%	4	29,07889
68.400-160	240	235	97,92%	16	26,84904
68.400-80	320	320	100,00%	4	29,71222

Instancje z błędami negatywnymi wynikającymi z powtórzeń					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
28.500-18	482	482	100,00%	5	50,54745
34.500-32	468	468	100,00%	6	50,00396
59.500-2	498	498	100,00%	3	50,05488
113.500-8	492	492	100,00%	17	70,35809
144.500-12	488	488	100,00%	3	49,51438

Instancje z błędami pozytywnymi losowymi					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
10.500+200	500	500	100,00%	8	66,37451
18.200+80	200	198	99,00%	26	27,48189
20.300+120	300	300	100,00%	22	43,70716
25.500+200	500	500	100,00%	24	99,10164
35.200+80	200	200	100,00%	17	24,67859
53.500+200	500	495	99,00%	7	71,94066
55.300+120	300	300	100,00%	12	36,28849
55.400+160	400	398	99,50%	7	48,96291
58.300+120	300	300	100,00%	25	47,63339
62.400+160	400	400	100,00%	26	72,33793
68.400+160	400	400	100,00%	17	61,72615
9.200+80	200	200	100,00%	5	17,97404

Instancje z błędami pozytywnymi, przekłamania na końcach oligonukleotydów					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
9.200+20	200	200	100,00%	5	15,81208
10.500+50	500	484	96,80%	61	129,7307
18.200+20	200	192	96,00%	7	15,59897
20.300+30	300	293	97,67%	50	58,43479
25.500+50	500	472	94,40%	5	51,00759
35.200+20	200	200	100,00%	5	16,08857
53.500+50	500	476	95,20%	25	78,42609
55.300+30	300	291	97,00%	8	28,24194
55.400+40	400	394	98,50%	52	87,11342
58.300+30	300	291	97,00%	33	47,69526
62.400+40	400	388	97,00%	10	42,59624
68.400+40	400	384	96,00%	33	64,6048

Zaprezentowane wyniki są ogólnie mówiąc zadawalające. Dla około połowy przypadków odnaleziona została sekwencja wykorzystująca maksymalną możliwą liczbę wyrazów. Dla pozostałych instancji otrzymany wynik jest prawie idealny, gdzie skuteczność sięga poziomu ponad 95%. Oglądając wyniki można się przekonać jak duże znaczenie ma proces parametryzacji na jakość otrzymywanych rozwiązań. W zaprezentowanych przygotowaniach algorytmu parametry były dobierane na podstawie rezultatów otrzymywanych dla instancji '20.300+120'. Jest to przykład problemu zawierającego błędy pozytywne losowe i to właśnie dla tego typu problemu uzyskiwane wyniki bardzo często osiągają optimum. Można by uznać tą klasę problemu za najbardziej przystępną dla utworzonej heurystyki, jeśli zostanie wzięty pod uwagę fakt, że tylko 3 z 12 instancji nie uzyskała pełnej skuteczności, nie spadając jednocześnie poniżej 99%.

Znakomicie prezentują się rezultaty dla instancji z błędami negatywnymi wynikającymi z powtórzeń. Tu dla wszystkich instancji odnalezione zostało maksymalne rozwiązanie. Sukces ten wynika najprawdopodobniej z rodzaju postawionego problemu. Brak dziesięcioznakowego wyrazu w danym miejscu sekwencji nie będzie miał dużego wpływu, jeśli istnieje wyraz spajający ciąg być może na mniejszej ilości znaków, ale będący za to najlepszym dostępnym wyborem. Argument ten równie dobrze można by przytoczyć dla losowych błędów negatywnych. Kluczowa staje się natomiast ilość brakujących wyrazów. Algorytm mrówkowy otrzymał idealne rezultaty dla instancji z powtórzeniami, gdyż liczba utraconych wyrazów stanowiła kilka procent całego zestawu. Instancje z losowymi brakującymi wyrazami były z kolei pozbawione większej ilości wyrazów, rzędu kilkudziesięciu procent. Jest to bardzo znacząca różnica. Prawdopodobnie w tym tkwi tajemnica stuprocentowej skuteczności heurystyki dla badanych instancji zawierających jedynie błędy negatywne wynikające z nielicznych powtórzeń.

Istotną kwestią wymagającą omówienia jest jeszcze czas działania algorytmu. W przedstawionych tabelach można zaobserwować czas trwania przetwarzania heurystyki liczący od 10 do 100 sekund. Jest to spowodowane przede wszystkim liczbą używanych słów oraz liczbą przeprowadzonych iteracji algorytmu. Główną czynnością wykonywaną przez program jest układanie wyrazów w pewnej kolejności otrzymując wynikową sekwencję. Naturalne staje się więc, że im więcej słów zostaje używanych przy tworzeniu sekwencji, tym dłużej musi odbywać się cały proces. Ważna jest też wspomniana liczba iteracji, która uzależniona została od momentu znalezienia najlepszego rozwiązania. Dla szybko znalezionego ostatecznego wyniku algorytm wykonuje nieco ponad 30 iteracji. W pesymistycznych przypadkach wartość ta jak się okazuje może sięgać do 80 iteracji. Przypominając sobie o 100 próbach utworzenia sekwencji w danej iteracji, uzasadniony staje się tak długi czas oczekiwania na zakończenie pracy algorytmu. Wykonując odpowiednie obliczenia można też udowodnić, że jednorazowa próba losowego ułożenia sekwencji zajmuje jedynie około jednej setnej sekundy, w zależności od wielkości zbiorów wyrazów używanych do odtworzenia sekwencji. Algorytm dokonujący takiej więc próby ułożenia kilka tysięcy razy potrzebuje już niestety więcej czasu, który trzeba mierzyć w sekundach.

Działanie algorytmu można ostatecznie uznać za poprawne. Wyniki osiągają ponad 95%, gdzie dosyć często można się spodziewać najlepszych rezultatów. Oparcie działania na losowym tworzeniu sekwencji pozwala uwolnić się od rozważań na temat rodzaju błędów w badanych instancjach. Jedynym minusem może być fakty zbyt dobrego dopasowania się do przykładu instancji służącego do ustalenia optymalnych wartości współczynników. Zaprezentowane wyniki pokazują jednak, że wzorowanie się na jednej instancji nie powstrzymuje heurystyki przed uzyskiwaniem niemal idealnych wyników dla pozostałych zestawów instancji.

4. Działanie algorytmu na własnych zbiorach instancji testowych.

Stworzony algorytm został poddany jeszcze dodatkowym testom, które mają sprawdzić jakość działania utworzonej heurystyki dla instancji innego rodzaju. W tym celu ustanowione zostały jeszcze 4 dodatkowe typy problemów. Są to:

- instancje z błędami negatywnymi losowymi – mało błędów.
- instancje z błędami negatywnymi wynikającymi z powtórzeń – dużo błędów
- instancje z błędami pozytywnymi, przekłamania na końcach oligonukleotydów – dużo błędów
- instancje z losowymi błędami negatywnymi i pozytywnymi

Największym problemem było wygenerowanie instancji zawierających błędy negatywne wynikające z powtórzeń. Maksymalnie możliwe było wygenerowanie liczby błędów na poziomie około 10%, co i tak stanowi większą część niż w przypadku obowiązkowych instancji tego typu. W poniższych tabelach można zapoznać się z otrzymanymi wynikami.

Instancje z błędami negatywnymi losowymi					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
01.100-5	95	95	100,00%	2	4,437204
02.100-10	90	90	100,00%	2	4,140331
03.200-10	190	190	100,00%	3	9,691942
04.300-20	280	280	100,00%	3	14,98329
05.300-10	290	290	100,00%	3	15,78013
06.400-20	380	380	100,00%	3	21,01413
07.500-30	470	470	100,00%	3	27,804
08.500-10	490	490	100,00%	3	29,34087

Instancje z błędami negatywnymi wynikającymi z powtórzeń					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
09.200-10	190	190	100,00%	3	9,04623
10.200-20	180	180	100,00%	3	8,596341
11.300-20	280	280	100,00%	8	16,13948
12.300-32	268	268	100,00%	4	13,76465
13.400-30	370	370	100,00%	3	20,23294
14.400-45	355	355	100,00%	5	20,19425
15.500-40	460	460	100,00%	26	44,71142
16.500-52	448	445	99,33%	4	26,12824

Instancje z błędami pozytywnymi, przekłamania na końcach oligonukleotydów					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
17.100+40	100	87	87,00%	37	6,015214
18.200+40	200	184	92,00%	37	14,99894
19.200+80	200	148	74,00%	17	8,827498
20.300+60	300	249	83,00%	9	14,78018
21.300+120	300	206	68,67%	41	22,00614
22.400+80	400	320	80,00%	37	35,59083
23.400+160	400	294	73,50%	43	35,32783
24.500+150	500	370	74,00%	38	45,80128

Instancje z losowymi błędami negatywnymi i pozytywnymi					
Instancja	**Maksymalna liczba wyrazów**	**Uzyskana najlepsza liczba wyrazów**	**Skuteczność**	**Numer iteracji uzyskania najlepszego wyniku**	**Czas obliczeń**
25.100-20+20	80	80	100,00%	3	3,359153
26.200-20+20	180	180	100,00%	5	8,39003
27.200-20+40	180	180	100,00%	5	8,546251
28.200-40+20	160	160	100,00%	12	9,644516
29.200-40+40	160	160	100,00%	9	8,940391
30.300-80+80	220	220	100,00%	4	12,19383
31.300-100+100	200	200	100,00%	13	13,24906
32.400-100+100	300	300	100,00%	4	17,70404
33.400-150+150	250	250	100,00%	9	16,17073
34.500-150+150	350	350	100,00%	4	23,12336
35.500-150+200	350	348	99,43%	27	37,75591
36.500-200+150	300	299	99,67%	42	38,70132
37.500-200+200	300	291	97,00%	6	20,04763

Pierwsze dwie tabele prezentują jak działa algorytm dla instancji z małą liczbą błędów negatywnych. Uzyskana skuteczność pokazuje, że dla heurystyki nie ma znaczenia, czy są to błędy losowe, czy wynikające z powtórzeń. Gdy błędów negatywnych jest względnie mało, to oryginalna sekwencja jest z łatwością rekonstruowana już w pierwszych iteracjach. Nieodnalezienie pełnego rozwiązania przytrafiło się jedynie ostatniej instancji z błędami powtórzenia. Liczba błędów jest tam bowiem wyższa niż w pozostałych przypadkach, stanowi ponad 10% całego zestawu wyrazów.

Kolejną grupą problemów są liczne błędy pozytywne wynikające z przekłamań na końcach oligonukleotydów. Tutaj bardzo widoczny jest spadek skuteczności. Heurystyka potrafi zbudować rozwiązanie używające od 70% do 90% możliwie najlepszego wykorzystania wyrazów. Instancje te stanowią bowiem znacznie większe wyzwanie. Duża liczba drobnych przekłamań na końcach wyrazów zwiększa liczbę atrakcyjnych połączeń par wyrazów. Wybieranie kolejnego z pozoru dobrego wyrazu może wówczas prowadzić do słabych następnych możliwości wyboru, ze względu na zniekształcenia końcówki tego wyrazu. W ten sposób trudnym okazuje się być odnalezienie oryginalnej sekwencji. Na zaradzenie tego problemu nasuwa się pomysł spoglądania nie o jeden lecz o dwa wyrazy wprzód. Analiza wyboru dwóch kolejnych wyrazów mogła by wyeliminować przypadki, w których wybierany zostaje kolejny wyraz o pasującym prefiksie, lecz o zupełnie niesprzyjającym sufiksie.

Ostatnim typem problemu są instancje zawierające losowe błędy zarówno negatywne jak i pozytywne. Okazuje się, że algorytm podołał zadaniu i poradził sobie w każdym z przypadków. Wymieszanie obu typów błędów nie wpłynęło na gorszą skuteczność problemu. Główną trudność stanowi tutaj jak zwykle ilość występujących błędów. Im więcej występuje błędów, tym trudniejsze staje się odbudowanie wzorcowej sekwencji. Własność tą widać w ostatnich badanych instancjach, gdzie nie udało się całkowicie przywrócić oryginalnych sekwencji.