UKŁADANIE GRAFIKU DLA PRACOWNIKÓW

Projekt wykonał: Grzegorz Huk

1. Model matematyczny

Dane wejściowe:

- n **-** liczba pracowników
- g - norma godzin do wypracowania - zgodnie z kodeksem pracy obliczamy liczbę godzin pracy w każdym miesiącu.
- M_{dost} - macierz dostępności pracowników - dajemy możliwość pracownikom wyboru dziewięciu zmian w okresie miesiąca, w których mogą zadeklarować swoją nieobecność w pracy z różnych powodów.
- M_{pref} - macierz preferencji pracowników - każdy z pracowników preferuje inną zmianę (poranną/popołudniową/nocną)
- $M_{dośw}$ - macierz doświadczenia pracowników - pracownicy różnią się obyciem w zawodzie, efektywnością i szybkością rozwiązywania problemów. Każdemu pracownikowi został przyporządkowany współczynnik doświadczenia z zakresu 0.1-0.9
- $M_{wDośw}$ - macierz wymaganego doświadczenia na każdą zmianę - w zależności od ilości pracy na pojedynczej zmianie wymagane jest konkretne doświadczenie, które powinno być mniejsze niż suma doświadczeń osób pracujących na tej zmianie.
- d - liczba dni w miesiącu

Postać rozwiązania:

$$M_{n\,x\,z}\ ,\ a_{ij} \in \{0,1\}\ ,$$
$$i = 0...n-1\ \wedge\ j = 0...z-1$$

Gdzie:

- n - liczba pracowników
- z - liczba zmian w miesiącu (dzień liczy trzy zmiany)

Macierz wynikowa jest macierzą o ilości wierszy równej liczbie pracowników pracujących w firmie oraz ilość kolumn równej liczbie zmian w miesiącu. Każdy element macierzy a_{ji} ma przyporządkowaną wartość logiczną. Wartość *True* oznacza wpisanie do grafiku osoby na konkretny dzień i zmianę.

Funkcja celu to minimalizacja wszystkich kar :

$$\sum_{i=0}^{7} K_i \Rightarrow min$$

gdzie :

K_i to odpowiednio **kary** za :

- **wpisanie do grafiku osoby pomimo jej niedostępności**, dodatkowo sprawdzamy czy każdy z pracowników nie dał statusu “niedostępny” na jednej zmianie.
- **brak przerwy** - zgodnie z regulaminem pracy osoba po skończonej zmianie musi mieć przynajmniej dwie zmiany przerwy.
- **brak odpowiedniej ilości pracowników na zmianie** - zakładamy grupy dwu lub trzyosobowe w zależności od ilości pracy do wykonania oraz doświadczenia pracowników.
- **niedogodziny** - każdy z pracowników ma określoną liczbę godzin do przepracowania w miesiącu.
- **brak odpowiednio doświadczonego personelu na zmianie** - każdy z pracowników cechuje się innym doświadczeniem zdobytym w branży oraz innym poziomem rozwiązywania problemów zarówno doczesnych jak i losowych. Zależy nam zatem aby zapobiegać umieszczeniu na jednej zmianie np. trzech ‘’nowo upieczonych’’ pracowników.
- **nadgodziny** - o ile nie chcemy pozwolić na niedogodziny tak na nadgodziny w pewnym stopniu pozwalamy. Dążymy jednak do tego aby każdy z pracowników posiadał w miesiącu podobną ilość przepracowanych godzin.

- **brak zadowolenia pracowników** - W naszym algorytmie dajemy możliwość wyboru preferencji zmian (poranna,popołudniowa,nocna). Przytaczając słowa Terencjusza - komediopisarza rzymskiego “Ile głów ludzi, tyle zdań” niemożliwe będzie zminimalizowanie tej kary. Chcemy zatem aby poziom zadowolenia lub niezadowolenia pracowników był u wszystkich na podobnym poziomie minimalizując odchylenie standardowe pomiędzy niezadowoleniem.
- **brak sprawiedliwej częstotliwości pracy każdego pracownika** - tutaj również minimalizujemy odchylenie standardowe, ale między średnią ilością dni pracy z rzędu każdego pracownika

2. Opis algorytmu

Przejście z jednego rozwiązania do drugiego jest realizowane przez tzw. funkcję przejścia i polega na znalezieniu rozwiązania sąsiedniego. Zaletą algorytmu symulowanego wyżarzania jest możliwość wyboru przez niego gorszego rozwiązania. Wybór taki jest dokonywany z pewnym prawdopodobieństwem. Dzięki temu algorytm symulowanego wyżarzania może w określonych warunkach wyjść ze znalezionego minimum lokalnego i dalej podążać w kierunku rozwiązania optymalnego. Parametrem algorytmu, który ma wpływ na prawdopodobieństwo wyboru gorszego rozwiązania jest parametr przeniesiony bezpośrednio z podstaw termodynamicznych algorytmu, czyli temperatura. Im wyższa, tym prawdopodobieństwo wyboru gorszego rozwiązania jest większe. Im niższa, tym algorytm jest bardziej zbliżony w działaniu do typowych metod iteracyjnych. To właśnie znajduje odzwierciedlenie w drugim ważnym aspekcie algorytmu symulowanego wyżarzania czyli w powolnym chłodzeniu. Na początku działania algorytmu temperatura jest wysoka, dzięki czemu algorytm może bardzo często zmieniać konfigurację rozwiązania, niejednokrotnie wybierając rozwiązanie gorsze. Wraz z kolejnymi iteracjami algorytmu temperatura spada i wybierane są częściej rozwiązania lepsze. Pod koniec pracy algorytmu, temperatura jest na tyle niska, że prawdopodobieństwo wyboru gorszego rozwiązania jest bliskie zeru. Algorytm zachowuje się wówczas, jak typowy algorytm iteracyjny i stara się maksymalnie ulepszyć rozwiązanie.

Schemat algorytmu:

Wyznaczyć rozwiązanie początkowe s i s_b

```
Wyznaczyć temperaturę początkową t

for i = 0 to L
        Wyznaczyć losowo sąsiednie rozwiązanie s' ∈ N(s)
        if (f(s') < f(s_b)) then s_b = s'
        δ = f(s') − f(s)
        if δ < 0 then s = s'
        else
                Wylosować x z zakresu (0,1)
                if (x < exp(−δ/t)) then s = s'
t = α(t)
Until warunek zatrzymania = true
Zwrócić rozwiązanie s_b
```

Elementy adaptowalne do rozmiaru problemu:

Na działanie algorytmu symulowanego wyżarzania mają wpływ jego parametry i sposób ich wyznaczania. Do ogólnych parametrów algorytmu, niezależnych od problemu w jakim jest on stosowany, należą: temperatura początkowa, długość epoki (liczba wewnętrznych iteracji algorytmu), funkcja zmiany temperatury i kryterium zatrzymania.

Istnieje wiele różnych wariantów wyznaczania temperatury początkowej. Jedną z bardziej rozbudowanych metod jest przyjęcie pewnego prawdopodobieństwa, z jakim ma być przyjęte gorsze rozwiązanie w pierwszej iteracji algorytmu. Na tej podstawie i w wyniku działania pewnej liczby kroków funkcji zmieniającej konfigurację wyznacza się temperaturę początkową. Dobrym i prostszym sposobem jest uwarunkowanie temperatury początkowej od kosztu rozwiązania początkowego. Mając określoną funkcję kosztu rozwiązania (zależną od implementacji algorytmu w konkretnym problemie) wyznaczamy T_0 przez pomnożenie wartości kosztu rozwiązania początkowego przez przyjęty z góry współczynnik.

Parametry algorytmu:

Długość epoki ma znaczący wpływ na działanie algorytmu. Im jest ona dłuższa, tym algorytm może dokonywać większej liczby zmian w konfiguracji znajdując się na określonym poziomie temperatury i tym samym większe są szanse na znalezienie lepszych rozwiązań. Teoretycznie jej długość jest wyznaczana jako określony procent wielkości sąsiedztwa konfiguracji. Jednakże w praktyce trudno jest czasami określić wielkość sąsiedztwa, bądź też zależy ona w znacznym stopniu od konkretnej implementacji. Dlatego też jej długość najczęściej zostaje wyznaczona eksperymentalnie dla konkretnego problemu. Generalnie im dłuższa epoka tym algorytm działa dokładniej, ale jednocześnie zwiększa się czas jego wykonania.

Funkcja zmiany temperatury prawie zawsze jest sprowadzana do operacji $T_{k+1} = \alpha T_k$, gdzie α jest z góry określonym współczynnikiem. Ten sposób zmiany temperatury jest w zupełności wystarczający, natomiast wielkość współczynnika α waha się zwykle pomiędzy 0.85 a

0.98. Innym sposobem jest na przykład funkcja: $T_{k+1} = \alpha^k T_0$ W trakcie działania algorytmu można zmieniać temperaturę początkową a także funkcję jej zmiany.

Kryterium zatrzymania jest parametrem, który w największym stopniu podlega modyfikacjom w konkretnych implementacjach algorytmu symulowanego wyżarzania. Najczęstszym i najprostszym sposobem jest przyjęcie określonej liczby kroków (zewnętrznej pętli algorytmu). Innym często stosowanym sposobem jest zatrzymywanie algorytmu w momencie, gdy od pewnej (ustalonej) liczby kroków nie dokonano poprawy rozwiązania. Jeszcze innym rozwiązaniem jest zatrzymywanie algorytmu w momencie, gdy liczba zaakceptowanych ruchów spośród wszystkich ostatnio wykonanych spadła poniżej określonego procenta. Generalnie im dłużej algorytm może działać zanim zostanie osiągnięte kryterium zatrzymania tym lepsze może uzyskać wyniki. Trzeba jednak pamiętać, że ma to także wpływ na znaczne wydłużenie czasu jego działania.

Wyznaczanie rozwiązań sąsiednich:

W naszym algorytmie zastosowaliśmy 6 funkcji pozwalających nam znaleźć rozwiązanie sąsiednie. Wybór, która z nich zostanie użyta był zdefiniowany przez rodzaj kary która była największa.

Funkcje znajdowania sąsiedztw są następujące:

- Zamiana wierszy
- Zamiana kolumn
- Dodanie godzin
- Obcięcie godzin
- Przesunięcie zmian
- Zamiana zmian

3. Oprogramowanie aplikacji

Wymagania:

Aplikacja została przetestowana na systemach operacyjnych:

- Windows 10
- MAC OS

Do poprawnego działania aplikacji wymagane jest posiadanie:

- Języka programowania Python w wersji 3 lub nowszej

- Interpretera języka (zalecany Jupyter Notebook)
- Bibliotek:

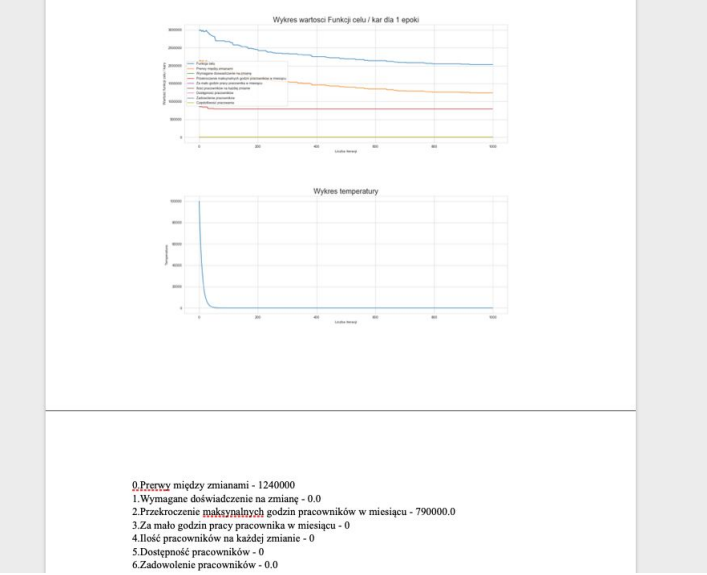
- Numpy - podstawowa biblioteka do obliczeń naukowych w Pythonie. Zapewnia wysokiej jakości wielowymiarowy obiekt macierzy i narzędzia do pracy z tymi tablicami.
- Pandas - to otwarta biblioteka licencjonowana BSD zapewniająca wydajne, łatwe w użyciu struktury danych i narzędzia do analizy danych dla języka programowania Python.
- Seaborn - biblioteka wizualizacji danych Pythona oparta na matplotlib. Zapewnia interfejs wysokiego poziomu do rysowania atrakcyjnej i informacyjnej grafiki statystycznej.
- Matplotlib.pyplot - to zbiór funkcji stylu poleceń, które sprawiają, że matplotlib działa jak MATLAB. Każda funkcja Pyplot wprowadza pewną zmianę do figury
- Docx - biblioteka pozwalająca na eksport danych to tego właśnie rodzaju pliku, korzystamy z niej przy raportowaniu o którym w dalszej części

Dane wejściowe możemy wczytywać z plików CSV. Do tego używana jest biblioteka Seaborn która pozwala na łatwe konwertowanie DataFrame'ów do plików. Same dane na których operujemy przy obliczeniach to tablice (array) z biblioteki Numpy które pozwalają na zoptymalizowane działania macierzowe.

4. Metodyka testów

W części testowej zdefiniowaliśmy dwa główne zadania: **Dobranie wag dla kar i dobranie parametrów algorytmu**.

Jak było wspomniane wcześniej, w celu ułatwienia testów stworzyliśmy moduł testowy, który używając biblioteki Docx zbiera najważniejsze informacje o algorytmie w jednym dokumencie. Część przykładowego raportu możemy zobaczyć na zdjęciu poniżej:



Na potrzeby naszej aplikacji stworzyliśmy 8 funkcji obliczających kary za brak spełnienia warunków wymaganych do uzyskania optymalnego rozwiązania. Każdej z funkcji nadaliśmy priorytet.

Kary o najwyższym priorytecie:
- wpisanie do grafiku osoby pomimo jej niedostępności
- brak przerwy
- brak odpowiedniej ilości pracowników na zmianie.
- niedogodziny.

Kary o średnim priorytecie:
- brak odpowiednio doświadczonego personelu na zmianie
- Nadgodziny

Kary o niskim priorytecie:
- brak zadowolenia pracowników
- brak sprawiedliwej częstotliwości pracy każdego pracownika

Z powodu wymaganej wysokiej mocy obliczeniowej, a więc długiego czasu potrzebnego na testy zdecydowaliśmy się zastosować kryterium zamykania na podstawie ilości iteracji, zawsze przy jednej epoce algorytmu. Parametry które będziemy modyfikować to ilość iteracji na dany poziom temperatury, funkcję zmiany temperatury. Będziemy sprawdzać jak algorytm radzi sobie z różnymi danymi początkowymi oraz grafikiem początkowym na podstawie którego wyznaczane są rozwiązania sąsiednie.

Z powodu po części losowego charakteru wyznaczania rozwiązań sąsiednich i skali naszego problemu, przy ilości iteracji którą wybraliśmy niemożliwe będzie wyeliminowanie każdej kary, albo nawet zbliżenie się do osiągnięcia tego celu. Dlatego skupimy się na tym jak algorytm przy różnych konfiguracjach zdoła poprawić rozwiązanie początkowe.

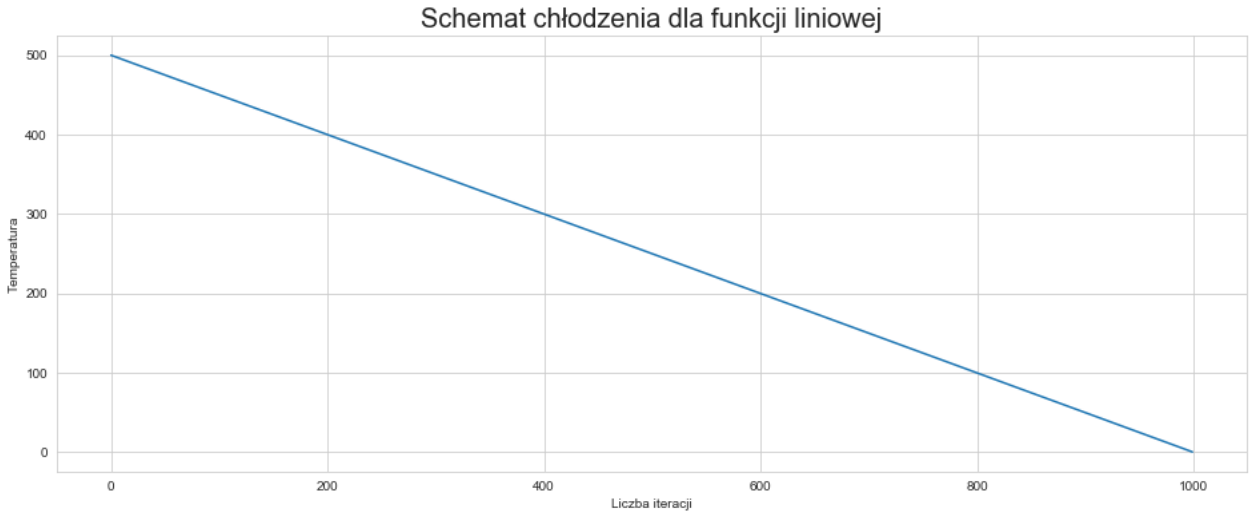
5. Testy

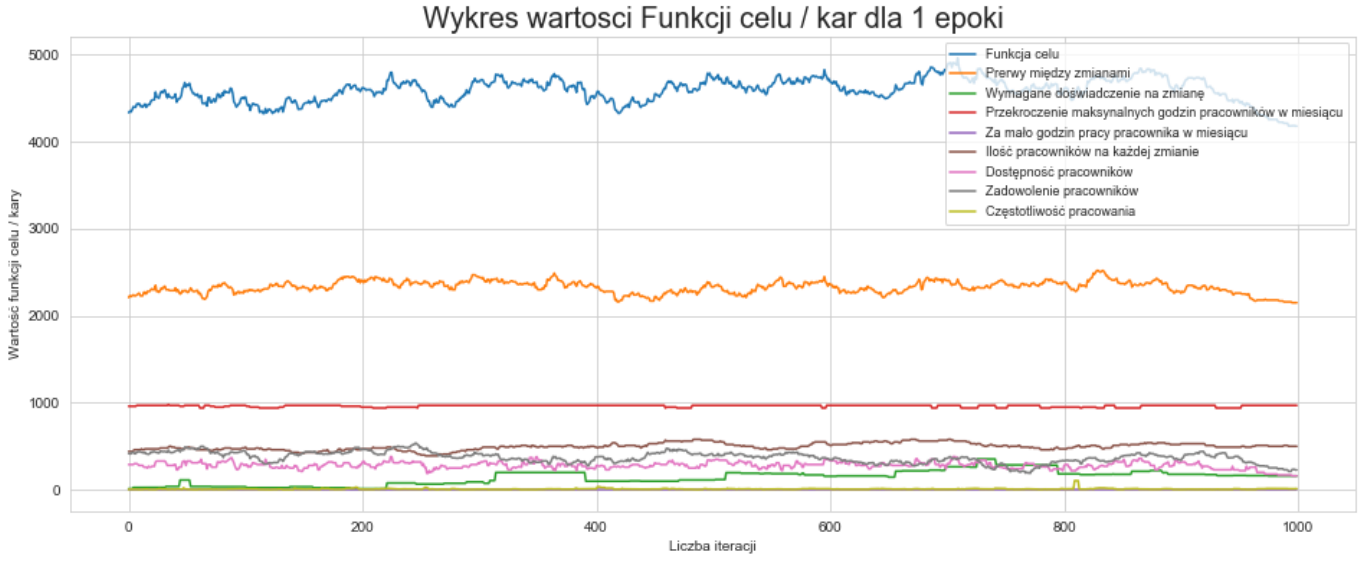
Celem testów jest porównanie działania algorytmu dla różnych danych początkowych oraz parametrów algorytmu, konkretnie te które sprawdzimy:

- Funkcja zmiany temperatury:
  - Wykładnicza - $T(t) = T_0\alpha^t$
  - Liniowa - $T(t) = T_0 - \eta t$

    Gdzie α i η to współczynniki zmiany temperatury, a t to liczba iteracji
- Liczba iteracji na każdy poziom temperatury $l \in \{0, 1, 2, 3\}$

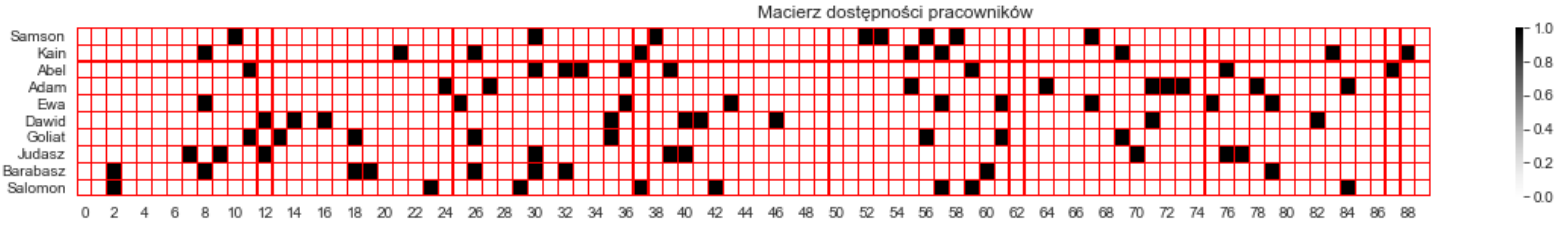
Na początku sprawdziliśmy jak **liniowy schemat chłodzenia** poradzi sobie z optymalizacją problemu, i otrzymując wyniki jak poniżej, postanawiamy odrzucić go w dalszych testach - nie radzi on sobie z naszym problemem, nawet przy stosunkowo niskiej temperaturze początkowej:

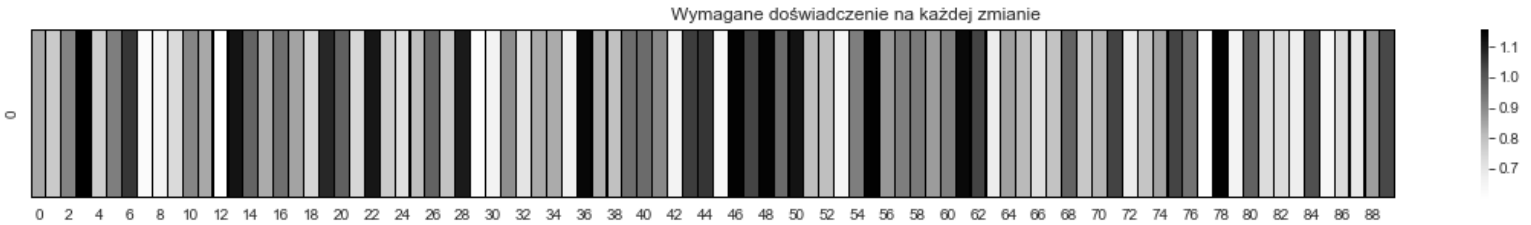


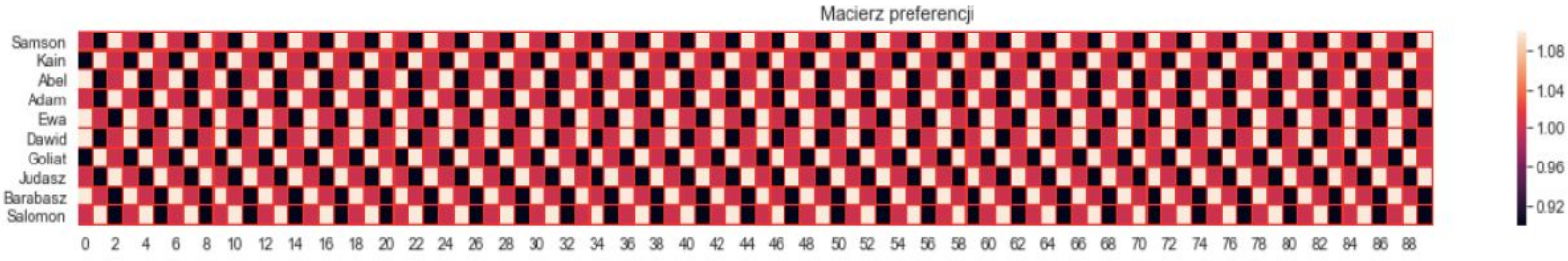


Dlatego będziemy pracować tylko z **wykładniczym schematem chłodzenia**:

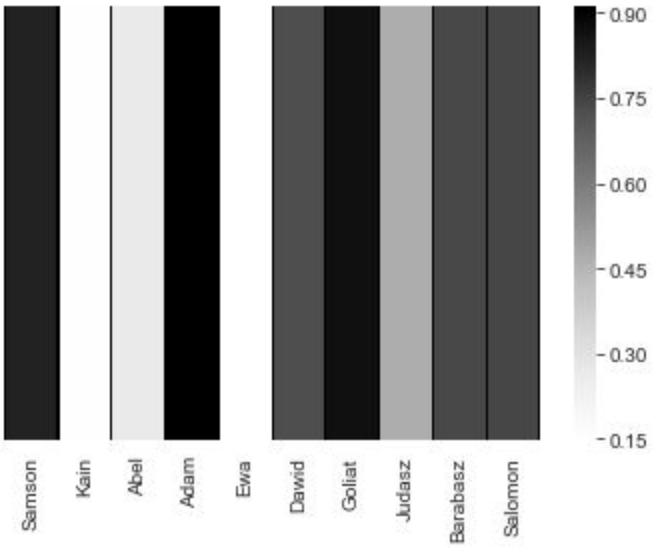
Przypadki testowe będą wykonywane dla tych samych macierzy dostępności, preferencji, doświadczenia pracowników oraz wymaganego doświadczenia na zmianie (wygenerowane losowo):



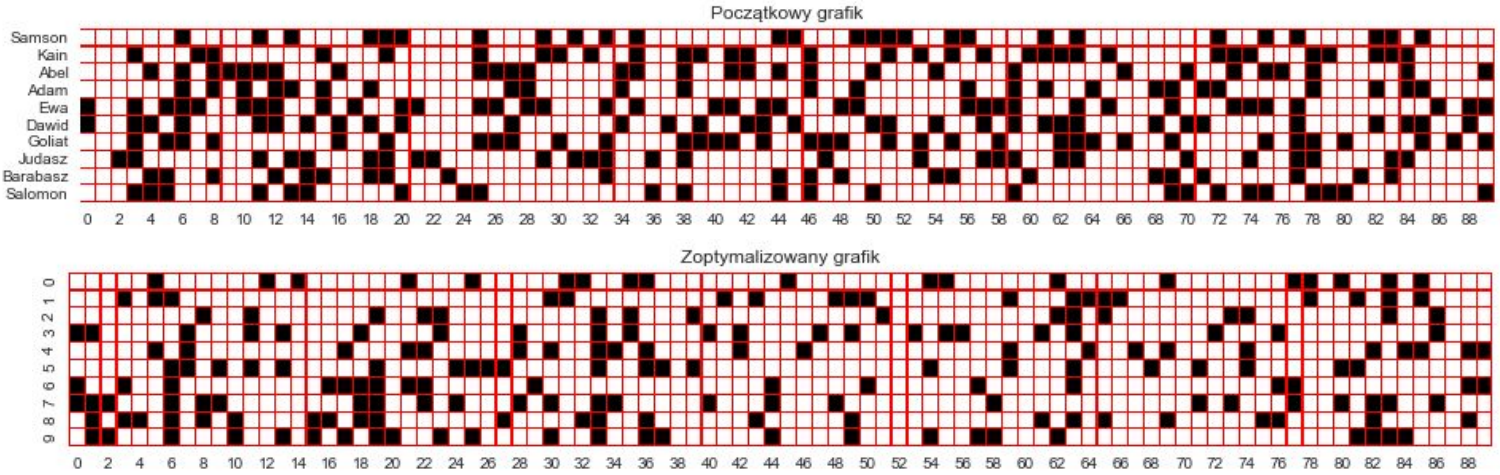




Doświadczenie pracowników:



Pierwszy przypadek sprawdzimy dla całkowicie losowego grafiku początkowego. Poniżej zobaczymy przykładowe rozwiązanie:



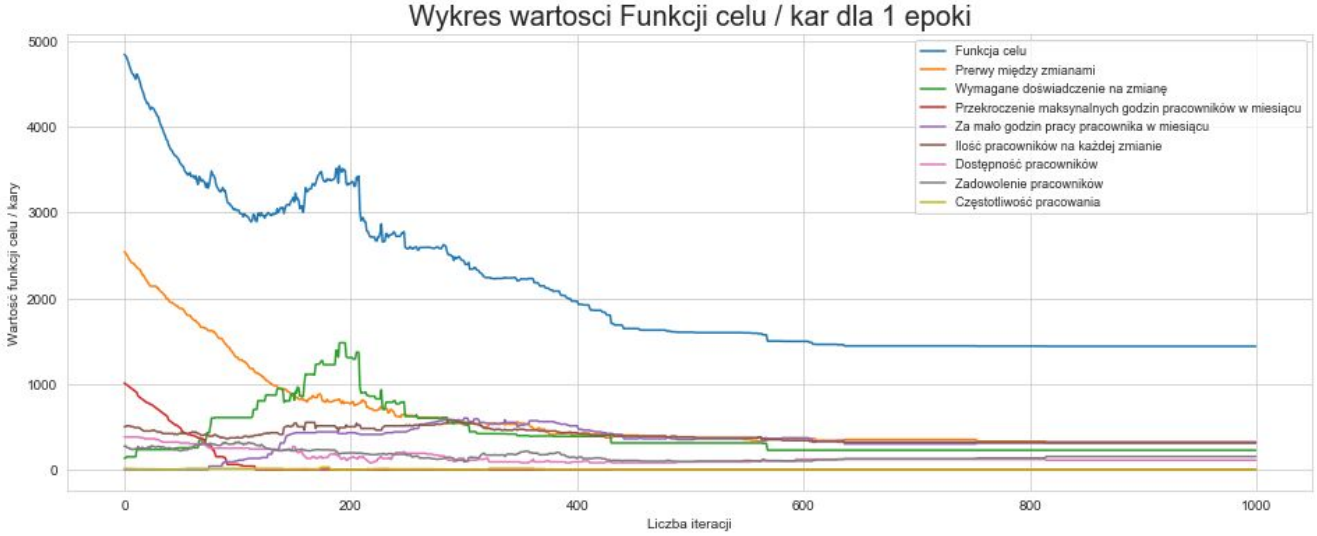
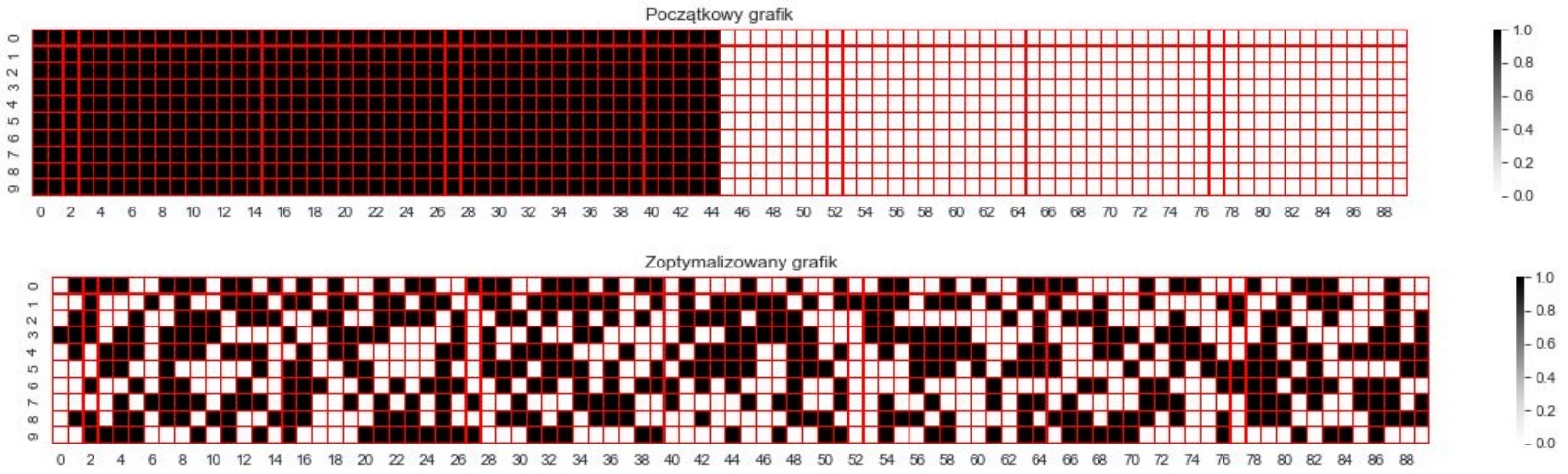


Tabela 1. - Wyniki dla przypadku pierwszego

Nr. Przypadku	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Wartość alfa	0,96	0,96	0,96	0,96	0.9	0.9	0.9	0.9	0.9	0.9
Ilość iteracji na zmianę temperatury	1	2	5	10	2	5	10	2	5	10
Te początkowa temperatura	5000	5000	5000	5000	2500	2500	2500	1000	1000	1000
Początkowa wartość funkcji celu	4911	4911	4911	4911	4911	4911	4911	4911	4911	4911
Końcowa wartość F Celu	1699	1551	1781	3811	1538	1438	1384	1942	1397	1559
Poprawa bezwzględna	3212	3360	3130	1100	3373	3473	3527	2969	3514	3352
Poprawa względna	65.4%	68.4%	63.7%	22.4%	68.7%	70.7%	71.8%	60.5%	71.6%	68.3%

W **drugim przypadku** grafik początkowy będzie bardzo skupiony w jednym miejscu:



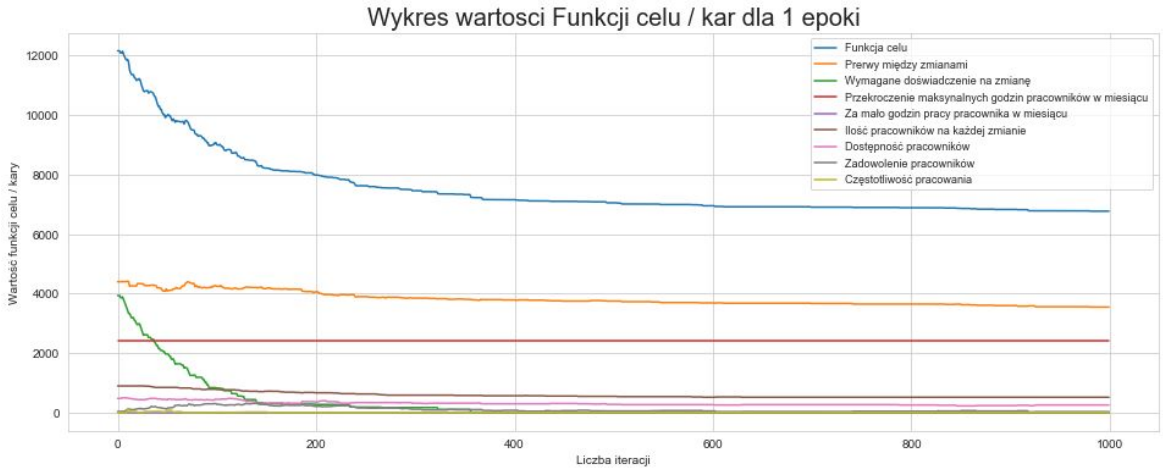
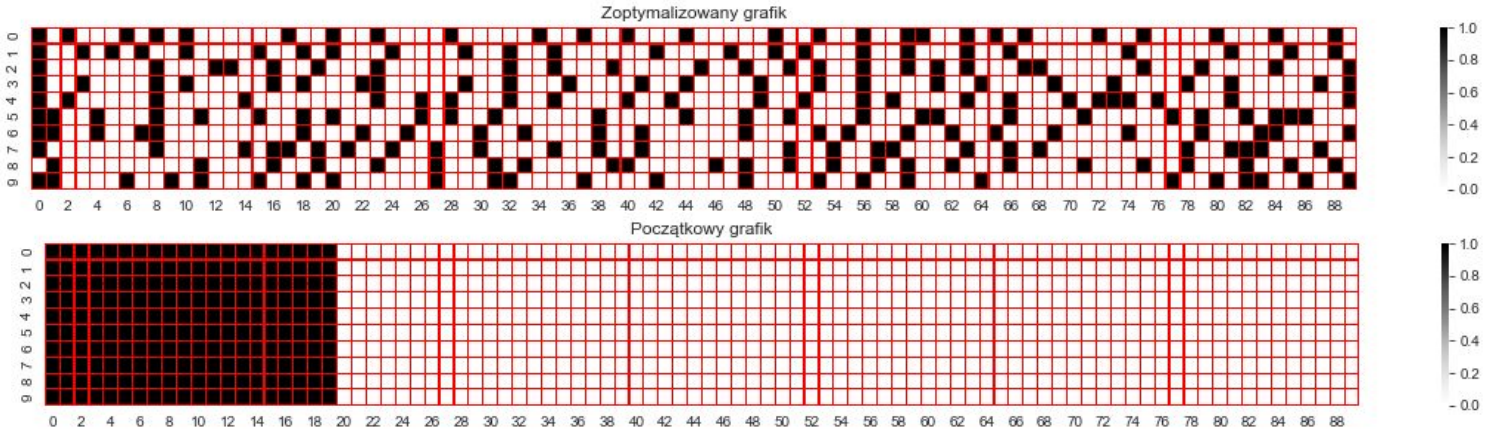


Tabela 2. - Wyniki dla przypadku drugiego

Nr. Przypadku	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Wartość alfa	0,96	0,96	0,96	0,96	0.9	0.9	0.9	0.9	0.9	0.9
Ilość iteracji na zmianę temperatury	1	2	5	10	2	5	10	2	5	10
Te początkowa temperatura	5000	5000	5000	5000	2500	2500	2500	1000	1000	1000
Początkowa wartość funkcji celu	12215	12215	12215	12215	12215	12215	12215	12215	12215	12215
Końcowa wartość F Celu	6824	6928	7313	8388	6959	6934	7096	6989	6790	7182
Poprawa bezwzględna	5421	5287	4902	3827	5256	5281	5119	5226	5425	5033
Poprawa względna	44.4%	43.3%	40.1%	31.3%	43.0%	43.2%	41.9%	42.8%	44.4%	41.2%

W **trzecim przypadku** grafik początkowy będzie gęsto rozłożony, ale w jednym małym miejscu:



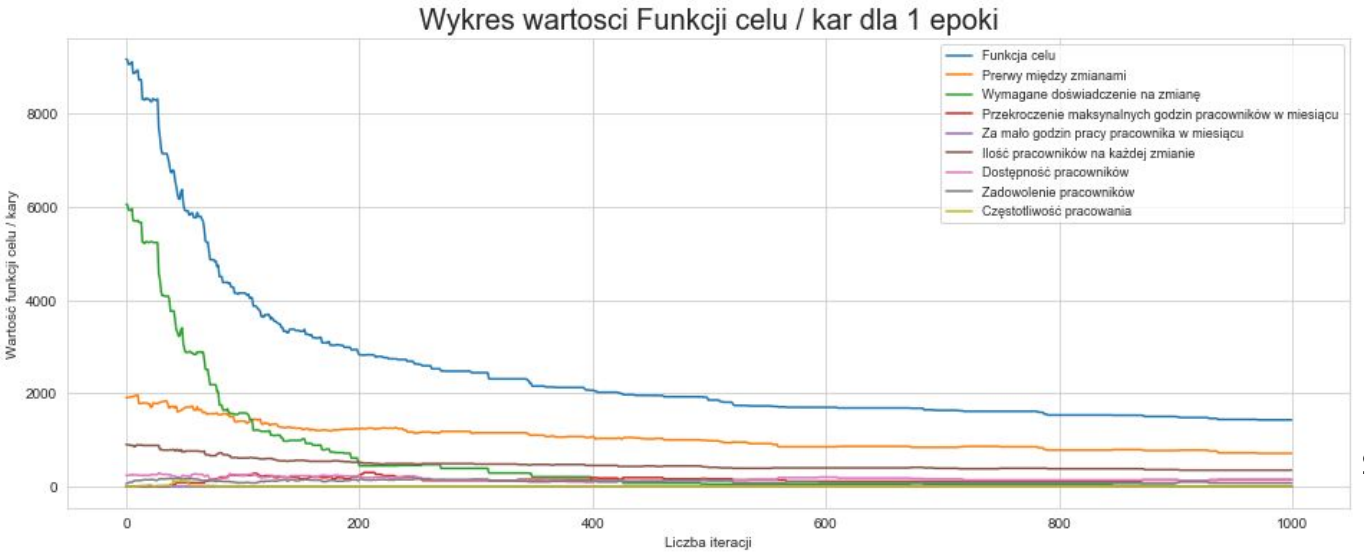
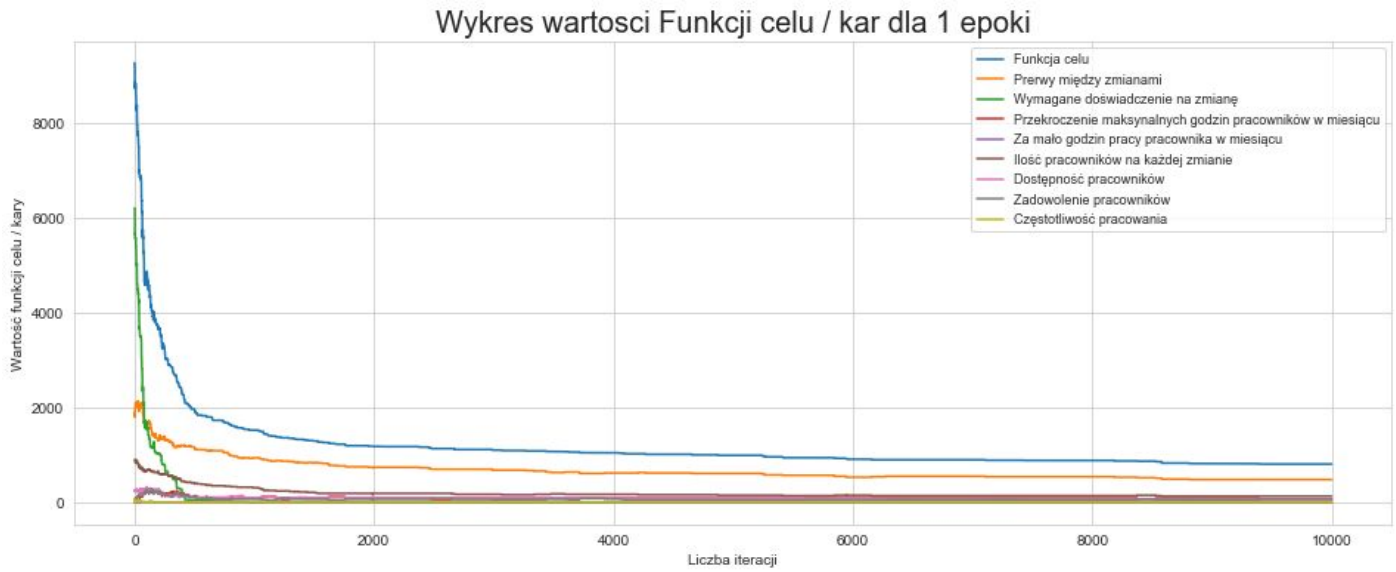


Tabela 3. - Wyniki dla przypadku trzeciego

Nr. Przypadku	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Wartość alfa	0,96	0,96	0,96	0,96	0.9	0.9	0.9	0.9	0.9	0.9
Ilość iteracji na zmianę temperatury	1	2	5	10	2	5	10	2	5	10
Te początkowa temperatura	5000	5000	5000	5000	2500	2500	2500	1000	1000	1000
Początkowa wartość funkcji celu	9296	9296	9296	9296	9296	9296	9296	9296	9296	9296
Końcowa wartość F Celu	1297	1854	1835	3438	654	822	1600	1390	1175	1256
Poprawa bezwzględna	7999	7442	7461	5858	8642	8474	7696	7906	8121	8040
Poprawa względna	86.0%	80.1%	80.3%	63.0%	93.0%	91.2%	82.8%	85.0%	87.4%	86.5%

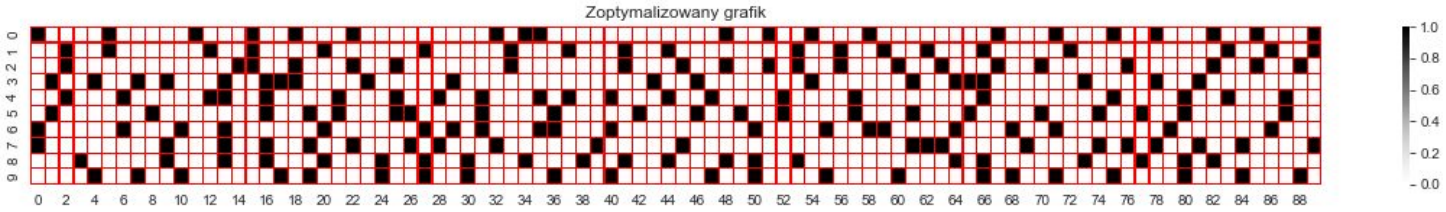
Dodatkowy test: Przeprowadziliśmy dodatkowy test, wykonując algorytm symulowanego wyżarzania 10000 razy - 10 razy więcej niż każdy poprzedni przypadek - dla najlepszych parametrów znalezionych w tabeli 3 (T0 2500, alfa = 0.9, 5 iteracji na zmianę temperatury) aby mieć najlepszą szansę na znalezienie maksimum globalnego. Jednak widzimy że przez mocną losowość algorytmu, nie udało się znaleźć lepszego rozwiązania niż dotychczas - funkcja celu wyniosła 808, największą trudność jak w innych przypadkach sprawiło zapewnienie przerw między zmianami:



Dodatkowy test 2: Przeprowadziliśmy również drugi dodatkowy test, zapewniając 5 epok algorytmu symulowanego wyżarzania dla parametrów takich jak w poprzednim podpunkcie. Pierwsze 3 epoki poprawiały rozwiązanie, lecz nadal nie było lepsze od dotychczas znalezionego (miało wartość 702), niestety kolejne były już gorsze.

6. Analiza danych i wnioski:

Na początek na podstawie najlepszego znalezionego pokażemy wartości kar:



Przerwy między zmianami	360
Wymagane doświadczenie na zmianę	7
Przekroczenie maksymalnych godzin pracowników w miesiącu	0
Za mało godzin pracy pracownika w miesiącu	0
Ilość pracowników na każdej zmianie	180
Dostępność pracowników	70
Zadowolenie pracowników	37
Częstotliwość pracowania	0

Należy też dodać że każdy pracownik ma 20 lub 21 zmian w tym miesiącu. 36 razy nie została zapewniona przerwa między zmianami, 18 razy na zmianie nie było odpowiedniej ilości pracowników, 7 zmian pracownicy musieli przyjść gdy byli niedostępni. Możemy z tego wyciągnąć wnioski że należy dopracować wagi odpowiednich kar oraz poprawić znajdowanie sąsiednich rozwiązań.

Najgorzej algorytm wydaje się pracować przy najwyższej alfie - 0.96, i 10 iteracjach na zmianę temperatury. Nie należy wykluczać spowodowania tego przez losowość algorytmu, lecz wynik wydaje się konsekwentny dla każdego scenariusza testowego. Trudno też zauważyć zależność

między pozostałymi parametrami - być może dałoby się wyciągnąć konkretniejsze wnioski po powtórzeniu testów wiele razy i wyciągnięciu średniej.

Łatwo zauważyć że największy wpływ na znalezienie lepszego rozwiązania ma początkowe ustawienie grafiku który będziemy modyfikować - aktualny sposób wybierania sąsiednich rozwiązań ma wtedy ‘najłatwiej’.