Optymalizacja
Projekt 3 - Spy Union

Autorzy: Michalina Wilk, Agnieszka Sękalska

Opis projektu:

Celem projektu było stworzenie algorytmu generującego program liniowy dla problemu Spy Union (polegającego na znalezieniu największej liczby zwolnionych pracowników dwóch organizacji, dla której zachowują one swoją strukturę oraz wskazaniu tych pracowników) i sprawdzenie jego działania na jedenastu danych plikach testowych w formacie .in.

Projekt zawiera dwa programy, oba bazujące na funkcjach generujących drzewa pracowników obu organizacji, ale dające różne outputy: jeden z nich tworzy i rozwiązuje bezpośrednio szukany program liniowy, drugi wypisuje program liniowy w Sage'u (przykład poniżej).

Wyniki testów (tabela):

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
all	5	20	100	200	500	1000	2000	5000	10000	20000	50000
max	2	9	53	115	302	585	1174	2754	5761	11492	26428
frac	40	45	53	57,5	60,4	58,5	58,7	55,08	57,61	57,46	52,86

gdzie: **numery kolumn** (w pierwszym wierszu) odpowiadają numerom plików testowych .in, natomiast dla poszczególnych plików testowych:

all - łączna liczba wszystkich pracowników WSA i Union,
max - maksymalna liczba zwolnionych przy zachowaniu struktur obu organizacji,
frac - % zwolnionych osób (spośród wszystkich pracowników).

Dodatkowe wnioski:

1. Dla testów z liczbą pracowników > 100 (3.in - 10.in) zwolnionych zostało średnio 57,26% pracowników (min: 55.08% dla 7.in, max: 60.4% dla 4.in).
2. Dla dużych danych istotne jest, ze względu na złożoność czasową, zminimalizowanie liczby powtarzających się ograniczeń w problemie liniowym.

Przykład:

Dla programu wejściowego 0.in (pierwszy wiersz zawiera liczbę wszystkich pracowników, kolejne odpowiednio numery szefów danego pracownika w WSA i Union oraz wymagane liczby osób w obu departamentach zarządzanych przez daną osobę):

```
5
1 0 1 2
2 0 1 2
2 1 2 0
2 1 0 1
1 3 0 0
```

drugi program generuje program liniowy:

```
N = 5
p = MixedIntegerLinearProgram(maximization=False)
v = p.new_variable(real=False)
p.set_objective(v[0] + v[1] + v[2] + v[3] + v[4])
p.add_constraint(v[0] >= 1)
p.add_constraint(v[0] + v[1] + v[4] >= 1)
p.add_constraint(v[0] + v[1] + v[2] + v[3] + v[4] >= 2)
p.add_constraint(v[0] + v[1] + v[2] + v[3] + v[4] >= 2)
p.add_constraint(v[1] + v[2] + v[3] + v[4] >= 2)
p.add_constraint(v[3] + v[4] >= 1)
for i in xrange(N):
    p.add_constraint(0 <= v[i] <=1)
print('Liczba wszystkich osob to:')
print(N)
print('Maksymalna liczba osob, ktore mozna zwolnic to:')
print(N - p.solve())
print('Nalezy zwolnic osoby o numerach:')
for i in xrange(N):
    if p.get_values(v[i]) == 0:
        print(i)
print('(indeksujac od 0)')
```

Powyższy program daje następujący output:

```
Liczba wszystkich osob to:
5
Maksymalna liczba osob, ktore mozna zwolnic to:
2.0
Nalezy zwolnic osoby o numerach:
2
```

4
(indeksujac od 0)