Optymalizacja

Projekt 1

Jakub Lepczyński – 359 790

14.06.2017

Kod znajduje się na stronie:
https://github.com/jl359790/OptymalizacjaI/blob/master/projekt1/pivot_steps.sage

Niestety, ale część programu odpowiadająca za parsowanie nie chciała przyjąć plików z kodem w postaci LP, które bez problemu uruchamiają się w GLPK i w niczym nie odróżniają się od udostępnionych plików testowych. Z tego powodu zdecydowałem przeprowadzić testy na wybranych plikach testowych, które wnosiły coś sensownego. Wiele z nich okazało się nierozwiązywalnych, więc użyłem tylko jednego z nich „AmercianSteelProblem.lp". Ponadto użyłem: „Furniture.lp", „WhiskasModel.lp", „WhiskasModel2.lp", które dawały już jakieś sensowne wyniki.

W załączonym kodzie były już zaimplementowane funkcje „lexicographical_min_entering", „lexicographical_max_entering" , „lexicographical_min_leaving" oraz „lexicographical_max_leaving", więc nie będą omówione.

Dodane funkcje to:

- „random_entering" – wybiera losowo spośród dostępnych zmiennych wchodzących do bazy, każda z tym samym prawdopodobieństwem[1],
- „random_leaving" – odpowiednik dla zmiennej opuszczającej bazę,
- „highest_objective_coefficient_entering" – za każdym razem wybiera najwyższy współczynnik funkcji celu wśród zmiennych, które mogą wejść do bazy,
- „smallest_objective_coefficient_entering" – odpowiednik powyższej, ale wybiera zmienną z najmniejszym współczynnikiem,
- „steepest_edge_entering" – wybiera spośród możliwych zmiennych wychodzących tę, która najlepiej odpowiada gradientowi wektora *c*.

Powyższe funkcje zostały użyte w sposób wymieszany ze sobą, co zostało zaprezentowane poniżej z wynikami, które oznaczają liczbę wykonanych przez metodę sympleks kroków. Wynik 0 oznacza, że wybrany problem okazał się nierozwiązywalny.

użyte funkcje \ rozważane problemy	AmericanSteelProblem.lp	Furniture.lp	WhiskasModel.lp	WhiskasModel2.lp
lexicographical_min_entering + lexicographical_min_leaving	0	2	2	11
lexicographical_max_entering + lexicographical_max_leaving	**0**	**2**	**2**	**2**
random_entering + random_leaving	0	2	2	7
highest_objective_coefficient_entering + lexicographical_min_leaving	0	2	2	6
steepest_edge_entering + lexicographical_min_leaving	0	2	2	6

[1] O ile funkcja random.randint() faktycznie wybiera liczby z tym samym prawdopodobieństwem

smallest_objective_coefficient_entering + lexicographical_min_leaving	0	2	2	6
lexicographical_max_entering + lexicographical_min_leaving	**0**	**2**	**2**	**2**
lexicographical_min_entering + lexicographical_max_leaving	0	2	2	11
random_entering + lexicographical_min_leaving	0	2	2	4
highest_objective_coefficient_entering + lexicographical_min_leaving	0	2	2	6
steepest_edge_entering + lexicographical_min_leaving	0	2	2	6
smallest_objective_coefficient_entering + lexicographical_min_leaving	0	2	2	6
random_entering + lexicographical_max_leaving	0	2	2	3

Wydaje się, że drugi i trzeci problem są za małe, żeby dawały jakieś możliwości wyboru zmiennych. Mamy dwa wiersze zawierające najmniejsze wartości przy czym oba zawierają funkcję lexicographical_max_entering, więc można wysunąć wniosek, że akurat w tym problemie ta funkcja daje najlepsze rezultaty.