Jakub Zbrzezny
Numer indeksu: 286689

Modelowanie matematyczne - II tercja
Dokumentacja powykonawcza
Zadanie z godziny 14
20.05.2019 r.

1 Opis rozwiązania kolokwium i analiza.

1.1 Plik.

Rozwiązanie kolokwium jest w pliku **"286689przedpoprawa.zpl"**.

1.2 Zbiory i parametry.

Program rozpocząłem od wczytania danych z plików, które były podane na kolokwium:

(1) **P** - zbiór pracowników
(2) **KD** - zbiór kodów działów (z pliku "dane.txt")
(3) **parypkd** - zbiór par postaci (p, kd) takich, że pracownik p nie ma nic przeciwko zajęciu się działem kd
(4) **profesorzy** - zbiór pracowników nadzorujących
(5) **nadzorowani** - zbiór par pracowników wymagających nadzoru
(6) **kodydzialow** - zbiór par kodów działów (z pliku "praca.txt")
(7) **czas**[**<i> in kodydzialow**] - parametr, który dla i ze zbioru kodydzialow zwraca szacowany nakład pracy dla działu i

Dodatkowo utworzyłem zbiór pomocniczy **N = P*KD - parypkd** będący zbiorem par (p, kd) takich, że pracownik p nie chce pracować w dziale kd.

Jest to różnica zbiorów P*KD i parypkd. Zbiór P*KD jest zbiorem wszystkich możliwych par (p, kd), gdzie p jest pracownikiem ze zbioru P, a kd kodem działu ze zbioru KD. Z definicji zbioru N oraz zbioru parypkd elementami zbioru N są pary (p,kd) takie, że pracownik p nie chce pracować w dziale kd.

Ponadto wziąłem zbiór dni **T = {1..sum<i> in kodydzialow do czas[i]}**. Jest to zbiór wszystkich dni od pierwszego dnia pracy do dnia w którym wykonano wszystkie działy MD.

1.3 Zmienne.

Wziąłem następujące trzy zmienne:

(1) **czyzajm[P*KD]** - zmienna binarna taka, że dla pracownika p, kodu działu kd czyzajm[p,kd] = 1 wtedy i tylko wtedy, gdy pracownik p zajmuje się działem kd.

(2) **nakladpracy[P*KD]** - dla pracownika p, kodu działu kd nakladpracy[p,kd] jest równy nakładzie pracy pracownika p w dziale kd

(3) **maksnaklpracy[P]** - dla pracownika p maksnakladpracy[p] jest maksymalnym nakładem pracy pracownika p spośród wszystkich działów ze zbioru KD

1.4 Funkcja celu.

Moją funkcją celu jest funkcja: **czaspracy: sum<p,kd> in P*KD do nakladpracy[p,kd]**, którą minimalizowałem.
Przy pomocy tej funkcji próbowałem zminimalizować maksymalny czas pracy przypadający na pracownika.
Funkcja jest sumą z nakladpracy[p, kd] po wszystkich pracownikach p i działach kd.
Nie jest poprawna, ponieważ jest całym nakładem pracy dla wszystkich pracowników i kodów działów. Minimalizacja tej funkcji nie jest równoważna minimalizacji maksymalnego czasu pracy przypadającego na jednego pracownika.

1.5 Ograniczenia.

W mojej pracy użyłem następujących ograniczeń:

(1) **forall<p,kd> in P*KD do nakladpracy[p,kd] <= czyzajm[p,kd] * card(T)**

Dane ograniczenie mówi, że jeśli pracownik p nie zajmuje się działem kd, to nakladpracy[p,kd] = 0. Natomiast, gdy nakladpracy[p,kd] = 1, pracownik p może pracować w dziale kd tak długo, ile wynosi suma z nakładów pracy w dziale k po wszystkich szacowanych nakładach pracy dla działu k.
Ograniczenie jest więc prawidłowe.

(2) **forall<p,kd> in N do czyzajm[p,kd] == 0**

Idea tego ograniczenia polega na tym, że nikt nie zajmuje się działem, którego nie chce.
Przypominamy, że zbiór N jest zbiorem par z pracownika p i kodu działu kd takich, że pracownik nie chce pracować w dziale kd.
To ograniczenie mówi, że jeśli pracownik p nie chce pracować w dziale kd, to czyzajm[p,kd] = 0. Zatem jest ono poprawne.

(3) **forall<kd> in KD do sum<p> in P do czyzajm[p,kd] >= 2**

Powyższe ograniczenie ma gwarantować, że każdy dział musi mieć przypisanych co najmniej 2 pracowników.
Jeśli dla ustalonego działu zajmuje się tylko jeden pracownik, to lewa strona nierównośći przyjjmie wartość 1, a więc ograniczenie nie jest spełnione.
A, gdy nikt się nim nie zajmuje, to lewa strona wynosi 0, czyli ograniczenie też nie jest spełnione.
Zatem ograniczenie również jest poprawne.

(4) **forall<p> in P do sum<kd> in KD do czyzajm[p,kd] <= 5**

To ograniczenie mówiło, że żaden pracownik nie może zajmować się więcej niż 5 działami.
Istotnie, jeśli dla $p \in P$ pracownik p zajmuje się k działami, gdzie $k >= 6$, to lewa strona nierówności była równa $k > 5$, więc wtedy ograniczenie nie jest spełnione.
Stąd powyższe ograniczenie jest też poprawne.

(5) **forall<p> in P do sum<kd> in KD do nakladpracy[p,kd] >= 4**

Powyższe ograniczenie polega na tym, że każdy pracownik musi przyjąć minimalny nakład pracy w liczbie co najmniej 4.
Jeśli pracownik p nie spełnia tego warunku, to oczywiście lewa strona będzie mniejsza od 4.

Stąd to ograniczenie jest też poprawne.

(6) **forall<p1,p2> in nadzorowani do forall<kd> in KD do -1 + czyzajm[p1,kd] + czyzajm[p2,kd] <= sum<prof> in profesorzy do czyzajm[prof,kd]**
Dane ograniczenie dotyczy par pracowników wymagających nadzoru. Mówi ono, że dla pary pracowników (p1, p2) należącej do zbioru nadzorowani, jeśli pracownicy p1, p2 pracują nad tym samym działem, to musi przy nich pracować jakiś pracownik nadzorujący. Jest nim pracownik ze zbioru profesorzy.
Zał, że $(p1, p2) \in nadzorowani, kd \in KD$.
Jeśli $p1, p2$ nie pracują w dziale kd, to lewa strona nierówności wynosi -1, stąd ograniczenie jest spełnione, gdyż suma zmiennych binarnych jest większa, bądź równa 0.
Gdy dokładnie jeden pracownik z pary $(p1, p2)$ pracuje w dziale kd, to lewa strona nierówności jest równa 0, a więc znowu wprost z własności zmiennych binarnych ograniczenie jest spełnione.
Gdy $p1, p2$ pracują nad działem kd, to lewa strona nierówności przyjmie wartość 1, a więc dla przynajmniej jednego pracownika $prof \in profesorzy$ musi zajść czyzajm[$prof$,kd] = 1.
Stąd to ograniczenie jest też prawidłowe.

(7) **forall<p> in P do forall <kd> in KD do maksnaklpracy[p] >= nakladpracy[p,kd]**
Dane ograniczenie mówi, że dla każdego pracownika p maksymalny nakład pracy pracownika p nie może być mniejszy niż nakład pracy w którymś dziale $kd \in KD$.
Ograniczenie jest poprawne.

(8) **forall<p,kd> in P*KD do nakladpracy[p,kd] >= 0**
To ograniczenie mówi, że dla pracownika p, kodu działu nakładpracy[p,kd] nie może być ujemny. Jest oczywiście prawidłowe, gdyż nakład pracy nie może być ujemny.

1.6 Analiza rozwiązania.

Uważam, że moje rozwiązanie nie jest zadowalające.
Wszystkie parametry, zbiory były wczytane poprawnie.
Było 634 ograniczeń. Program się kompilował, rozwiązał problem w czasie 0.01 sekundy. Wynik funkcji celu wyszedł: 44, czyli był jakąś konkretną, niezerową liczbą. Ale moje rozwiązanie nie było prawidłowe, ponieważ w rozwią-

zaniu nie jest uwzględnione, że pracownik nie może pracować nad 2 działami o łącznym nakładzie pracy mniejszym niż 3. Próbowałem to zrobić za pomocy ograniczenia: **forall<p> in P do forall<kd1,kd2> in KD*KD with kd1 != kd2 do nakladpracy[p,kd1] + nakladpracy[p,kd2]** >= **3 * (-1 + czyzajm[p,kd1] + czyzajm[p,kd2]**), lecz wtedy po uruchomieniu programu po 2 minutach liczenia nadal nie było wyniku i w czasie kolokwium już przerwałem i zrezygnowałem z użycia tego ograniczenia. Wyglądało ono poprawnie, gdyż jeśli pracownik p zajmuje się dwoma działami, to nierówność wtedy mówi, że suma nakładów pracy ma być nie mniejsza niż 3.

Głównymi błędami mojego rozwiązania są źle dobrane zmienne oraz zła funkcja celu. Wzięcie zmiennej nakladpracy[p,kd], która zależała od pracownika p i kodu działu kd jest błędem, ponieważ każdy pracownik p pracuje w dziale kd dokładnie tyle, ile wynosi szacowany nakład pracy dla działu kd. Niestety na kolokwium myślałem, że pracownik p może pracować w dziale kd więcej niż wynosi wartość parametru czas dla kodu działu kd, stąd miałem taki pomysł na zmienną.
Co więcej, wzięcie zmiennej maksnakladpracy[p] zależnej od pracownika p również nie jest dobrą drogą do uzyskania optymalnego rozwiązania. Maksymalny nakład pracy dla pracownika nie powinien zależeć od pracownika, lecz powinien być globalnym ograniczeniem górnym czasu pracy pracownika spośród wszystkich pracowników ze zbioru P. Źle również jej użyłem, gdyż dałem tylko dolne ograniczenie, a nie dałem górnego ograniczenia na tą zmienną i dlatego w rozwiązaniu dla każdego pracownika p maksnakladpracy[p] był równy nieskończoność. Poza tym nie powiązałem tej zmiennej w żaden sposób ze zmienną nakladpracy[P*KD] w ograniczeniach, tak, żeby to zmieniło wartość funkcji celu dla optymalnego rozwiązania.
Użycie zmiennej binarnej czyzajm[P*KD] jest sensowne, gdyż można w właściwy sposób dla pracownika p ze zbioru P liczyć sumę po każdym kodzie działu pracy kd, którym pracownik p się zajmuje, z czasu pracy pracownika p w dziale kd.
W przypadku funkcji celu jej wartość nie powinna być zapisana w postaci sumy, ponieważ celem w zadaniu jest zminimalizowanie **maksymalnego czasu pracy** przypadającego na **pracownika**. W gruncie rzeczy, trzeba najpierw dla każdego pracownika p ze zbioru P dobrać górne ograniczenie jego całego czasu pracy we wszystkich działach, którymi się zajmuje. Tym ograniczeniem powinna być właśnie zmienna niezależna od pracownika p, którą trzeba zminimalizować.

W dodatku, zbiór T = {1..sum<i> in kodydzialow do czas[i]} nie był nigdzie użyty w programie.

2 Poprawa rozwiązania.

Poprawione rozwiązanie znajduje się w pliku **"286689popoprawie.zpl"**.

W pracy głównie zająłem się zmianą zmiennych oraz funkcji celu. Ponadto zmodyfikowałem większość ograniczeń oraz usunąłem niepotrzebne ograniczenia. Usunąłem także zbędny zbiór T. Zrobiłem drobne zmiany w nazewnictwie parametrów.

Nazwę parametru, który zwracał szacowany nakład pracy dla kodu działu z pliku "praca.txt" zmieniłem na **naklad**.
Reszta parametrów, zbiorów pozostała bez zmian.

Z pierwotnego rozwiązania usunąłem zmienne **nakladpracy[P*KD], maksnaklpracy[P]**, gdyż tak jak opisałem wcześniej w analizie tamtego rozwiązania, nie dawały dobrej drogi do znalezienia optymalnego rozwiązania problemu zadana.
Zmienna **czyzajm[P*KD]** pozostaje z taką samą definicją.
Dodałem zmienną rzeczywistą *czas*, która jest górnym ograniczeniem czasu całej pracy pracownika p spośród wszystkich pracowników p ze zbioru P.

Funkcję celu również poprawiłem. Jest nią: **czas**, którą minimalizowałem. Znalezienie minimum tej funkcji jest równoważne znalezienie najmniejszego maksymalnego czasu pracy przypadającego na pracownika.

W przypadku ograniczeń, z starego rozwiązania usunąłem:
(1) **forall<p,kd> in P*KD do nakladpracy[p,kd] <= czyzajm[p,kd] * card(T)**
To ograniczenie wykorzystywało zmienną **nakladpracy[P*KD]**, którą już usunąłem.
(2) **forall<p> in P do forall <kd> in KD do maksnaklpracy[p] >= nakladpracy[p,kd]**
Dane ograniczenie korzystało z nieużywanych już zmiennych **nakladpra-**

cy[P*KD], maksnaklpracy[P].

(3) **forall<p,kd> in P*KD do nakladpracy[p,kd] >= 0**
Powyższe ograniczenie też korzystało z nieużywanej już zmiennej.

W nowym rozwiązaniu użyłem następujących ograniczeń:

(1) **forall<p,kd> in N do czyzajm[p,kd] == 0**
Jest to niezmienione ograniczenie (2) z pierwotnego rozwiązania.

(2) **forall<kd> in KD do sum<p> in P do czyzajm[p,kd] >= 2**
To ograniczenie takie same jak ograniczenie (3) ze starego rozwiązania.

(3) **forall<p> in P do sum<kd> in KD do czyzajm[p,kd] <= 5**
Powyższe ograniczenie jest ograniczeniem (4) z wcześniejszej wersji rozwiązania.

(4) **forall<p> in P do sum<kd> in KD do naklad[kd] * czyzajm[p,kd] >= 4**
Jest to zmodyfikowane ograniczenie (5) z pierwotnego rozwiązania.
Zmiana jest w sumowaniu po wszystkich działach kd ze zbioru KD dla ustalonego pracownika p ze zbioru P. Sumujemy nie z nieużywanej już zmiennej **nakladpracy]p,kd]**, ale z iloczynu wartości parametru naklad w kd i wartości zmiennej dla pary (p,kd).

(5) **forall<p> in P do forall<kd1,kd2> in KD*KD with kd1 != kd2 do naklad[kd1] + naklad[kd2] >= 3 * (-1 + czyzajm[p,kd1] + czyzajm[p,kd2])**
Powyższe ograniczenie mówi o tym, że pracownik nie może pracować nad 2 działami o łącznym nakładzie pracy mniejszym niż 3.
Istotnie, ustalmy pracownika p ze zbioru P oraz dwa różne kody działów kd1, kd2 ze zbioru KD.
Jeśli pracownik p pracuje w jednym z działów kd1, kd2 lub w żadnym z nich nie pracuje, to prawa strona nierówności jest mniejsza, bądź równa zero. Wtedy ograniczenie jest spełnione.
Gdy pracownik p pracuje w dziale kd1, kd2, to prawa strona nierówności przyjmie wartość równą 3. A więc wtedy ograniczenie mówi o tym, że suma nakładów pracy dla działów kd1, kd2 ma być większa, bądź równa 3. Czyli jest ono poprawne.

(6) **forall<p1,p2> in nadzorowani do forall<kd> in KD do -1 + czyzajm[p1,kd] + czyzajm[p2,kd] <= sum<prof> in profesorzy do czyzajm[prof,kd]**
Jest to ograniczenie (6) z starego rozwiązania.

(7) **forall<p> in P do sum<kd> in KD do naklad[kd] * czy-**

zajm[p,kd] <= **czas**
Powyższe ograniczenie jest warunkiem na poprawność definicji zmiennej czas. Dla ustalonego pracownika p ze zbioru P lewa strona nierówności jest sumą po wszystkich kodach działu kd ze zbioru KD z naklad[kd] * czyzajm[p,kd]. Jeśli pracownik p nie zajmuje się działem kd, to czyzajm[p,kd] = 0, czyli naklad[kd] * czyzajm[p,kd] = 0 w przeciwnym przypadku czyzajm[p,kd] = 1, a więc naklad[kd] * czyzajm[p,kd] = naklad[kd]. Stąd lewa strona nierówności jest całkowitym czasem pracy pracownika p. Ograniczenie jest zatem prawidłowe.

Dla danych otrzymanych na kolokwium miałem 1701 ograniczeń. Program rozwiązał problem w czasie 0.04 sekundy. Otrzymałem następujące rozwiązanie:
Wartość funkcji celu: 6.

(1) Pracownik 1 zajmuje się działami: 2, 4, 8.
(2) Pracownik 2 zajmuje się działami: 7, 9.
(3) Pracownik 3 zajmuje się działami: 1, 11.
(4) Pracownik 4 zajmuje się działami: 3, 6.
(5) Pracownik 5 zajmuje się działami: 5.
(6) Pracownik 6 zajmuje się działami: 5, 12.
(7) Pracownik 7 zajmuje się działami: 2, 6, 10.
(8) Pracownik 8 zajmuje się działami: 2, 4, 7.
(9) Pracownik 9 zajmuje się działami: 1, 9.
(10) Pracownik 10 zajmuje się działami: 3, 8, 12.
(11) Pracownik 11 zajmuje się działami: 10, 11.

3 Testy.

3.1 Przykład 1.

Program sprawdzam dla danych w plikach **"dane1.txt", "pary1.txt", "praca1.txt", "profesorzy1.txt"**.
Dane:
Pary (pracownik, kod działu) takich, że pracownik nie ma nic przeciwko zajęciu się danym działem:
(1, 1), (1, 2),

(2, 2).
Nie ma par pracowników wymagających nadzoru.
Szacowane nakłady pracy:
Kod działu 1: 3.
Kod działu 2: 4.
Pracownicy nadzorujący: 1.

Z ograniczenia (1): czyzajm[2, 1] = 0.
Z ograniczenia (2) musi być spełnione: czyzajm[1,1] + czyzajm[2,1] >= 2.
Ale otrzymałem wcześniej, że czyzajm[2,1] = 0, Więc czy[1,1] >= 2. Stąd z własności zmiennych binarnych nierówność jest sprzeczna.
Zatem nie ma rozwiązania problemu dla tego przypadku.
Program również wyświetla komunikat, że nie ma rozwiązania.
Zatem tu program działa prawidłowo.

3.2 Przykład 2.

Teraz sprawdzam dla danych w plikach **"dane2.txt", "pary2.txt", "praca2.txt", "profesorzy2.txt"**.
Dane:
Pary (pracownik, kod działu) takich, że pracownik nie ma nic przeciwko zajęciu się danym działem:
(1,1), (1, 2), (1,3), (1,4),
(2, 1), (2, 2),
(3, 1), (3, 4),
(4, 3).
Pary pracowników wymagających nadzoru: (2, 4)
Szacowane nakłady pracy:
Kod działu 1: 3.
Kod działu 2: 2.
Kod działu 3: 4.
Kod działu 4: 1.
Pracownicy nadzorujący: 1.

Z ograniczenia (1) otrzymuję, że:
czyzajm[2, 3] = 0;
czyzajm[2, 4] = 0;

czyzajm[3, 2] = 0;
czyzajm[3, 3] = 0;
czyzajm[4, 1] = 0;
czyzajm[4, 2] = 0;
czyzajm[4, 4] = 0.

Z ograniczenia (2) mam:
czyzajm[1, 2] = czyzajm[2, 2] = 1;
czyzajm[1, 3] = czyzajm[4, 3] = 1;
czyzajm[1, 4] = czyzajm[3, 4] = 1.

Ograniczenie (3) jest spełnione, ponieważ dla p ze zbioru P sum¡kd¿ in KD do czyzajm[p,kd] <= 4, gdyż |KD| = 4.

Sprawdzam ograniczenie (4):
czyzajm[1, 3] = 1, więc naklad[3] * czyzajm[1, 3] = 4. Stąd pracownik 1 spełnia to ograniczenie.
czyzajm[2, 2] = 1, ale czyzajm[2, 2] * naklad[2] = 2 < 4.
Ponadto czyzajm[2, 3] = czyzajm[2, 4] = 0, czyli musi zachodzić czyzajm[2,1] = 1.
czyzajm[2, 1] * naklad[1] + czyzajm[2, 2] * naklad[2] = 3 + 2 = 5 >= 4. Stąd ograniczenie jest spełnione.
czyzajm[3, 4] = 1, ale czyzajm[3, 4] * naklad[4] = 1 < 4.
Co więcej, czyzajm[3, 2] = czyzajm[3, 3] = 0.
Czyli czyzajm[3, 1] = 1.
czyzajm[3, 1] * naklad[1] + czyzajm[3,4] * naklad[4] = 3 + 1 = 4. Stąd ograniczenie jest spełnione.
czyzajm[4, 3] = 1, stąd naklad[3] * czyzajm[4, 3] = 4, a więc pracownik 4 spełnia dane ograniczenie.

Teraz sprawdzam ograniczenie (5):
Widzimy, że dla każdych dwóch różnych działów kd1, kd2 mamy: naklad[kd1] + naklad[kd2] >= 3.
Czyli jest automatycznie ono spełnione.

Sprawdzam ograniczenie (6):
Dla kd != 3 czyzajm[4, kd] = 0. Ale czyzajm[2, 3] = 0. Stąd pracownicy 2, 4 nie mogą pracować nad tym samym działem, a zatem ograniczenie jest

spełnione.

Sprawdzam ograniczenie (7):
Dla kd = 2, 3, 4: czyzajm[1, kd] = 1, stąd: sum<kd> in KD do naklad[kd] * czyzajm[1,kd] >= 2 + 4 + 1 + 3 * czyzajm[1, 1].
Czyli: czas >= 7 + 3 * czyzajm[1, 1].
Dla kd = 1, 2 czyzajm[2,kd] = 1, a dla pozostałych kd czyzajm[2, kd] = 0.
Stąd: czas >= 3 + 2 = 5.
Dla kd = 1, 4 czyzajm[3, kd] = 1, a dla pozostałych kd czyzajm[3, kd] = 0.
Zatem: czas >= 3 + 1 = 4.
Dla kd != 3 czyzajm[4, kd] = 0, a dla kd = 3 czyzajm[3, kd] = 1.
Czyli: czas >= 4.
Stąd musi być zachodzić: czas >= 5 oraz czas >= 7 + 3 * czyzajm[1, 1].

Zatem rozwiązaniem problemu jest:
Wartość funkcji celu wynosi 7.
Pracownik 1 zajmuje się działami: 2, 3, 4.
Pracownik 2 zajmuje się działami: 1, 2.
Pracownik 3 zajmuje się działami: 1, 4.
Pracownik 4 zajmuje się działami: 3.

Rozwiązanie otrzymane przez program jest identyczne.
Czyli tu również program działa prawidłowo.

3.3 Przykład 3.

Teraz sprawdzam dla danych w plikach **"dane3.txt", "pary3.txt", "praca3.txt", "profesorzy3.txt"**.
Dane:
Pary (pracownik, kod działu) takich, że pracownik nie ma nic przeciwko zajęciu się danym działem:
(1, 1), (1, 2), (1, 5),
(2, 2), (2, 3), (2, 4),
(3, 1), (3, 4),
(4, 1), (4, 2), (4, 3), (4, 4), (4, 5), (4, 6),
(5, 1), (5, 6).
Pary pracowników wymagających nadzoru: (1, 2), (2, 3).

Szacowane nakłady pracy:
Kod działu 1: 1.
Kod działu 2: 2.
Kod działu 3: 1.
Kod działu 4: 4.
Kod działu 5: 3.
Kod działu 6: 3.
Pracownicy nadzorujący: 4, 5.

Na mocy ograniczenia (1) otrzymuję, że:
czyzajm[1, 3] = czyzajm[1, 4] = czyzajm[1, 6] = 0;
czyzajm[2, 1] = czyzajm[2, 5] = czyzajm[2, 6] = 0;
czyzajm[3, 2] = czyzajm[3, 3] = czyzajm[3, 5] = czyzajm[3, 6] = 0;
czyzajm[5, 2] = czyzajm[5, 3] = czyzajm[5, 4] = czyzajm[5, 5] = 0.

Sprawdzam ograniczenie (2):
Skoro 3 działem 3 pracowników się nie zajmuje, to czyzajm[2, 3] = czyzajm[4, 3] = 1.
Wiedząc, że 5 działem 3 pracowników się nie zajmuje, dostaję, że czyzajm[1, 5] = czyzajm[4, 5] = 1.
Ponadto, skoro 6 działem 3 pracowników się nie zajmuje, mam, że czyzajm[4, 6] = czyzajm[5, 6] = 1.

Sprawdzam ograniczenie (3):
Mam, że tylko dla jednego pracownika p ze zbioru P, dla kd = 1, 2, ..., 6: pracownik p nie ma nic przeciwko zajęciu się działem kd. Jest to pracownik 4.
Na mocy tego ograniczenia istnieje kd ze zbioru KD takie, że czyzajm[4, kd] = 0.

Sprawdzam ograniczenie (4):
Pracownik 1: naklad[1] + naklad[2] = 3 < 4.
Czyli czyzajm[1, 5] = 1.
Ponadto czyzajm[1, 1] = 1 lub czyzajm[1, 2] = 1.
naklad[1] + naklad[5] >= 4 oraz naklad[2] + naklad[5] >= 4.
Czyli w obu danych przypadkach ograniczenie zachodzi.
Pracownik 2: naklad[2] + naklad[3] = 3 < 4.
Zatem czyzajm[2, 4] = 1.

Pracownik 3: naklad[1] = 1 < 4.
naklad[4] = 4
Stąd czyzajm[3, 4] = 1.
Pracownik 5: naklad[1] = 1 < 4 oraz naklad[6] = 3 < = 4
Czyli czyzajm[5, 1] = czyzajm[5, 6] = 1.
naklad[1] + naklad[6] = 4, stąd w tym przypadku ograniczenie zachodzi.

Sprawdzam ograniczenie (5):
Jest dokładnie jedna możliwość, w której dla dwóch różnych działów kd1, kd2 mamy: naklad[kd1] + naklad[kd2] < 3. Są nimi 1 i 3.
Udowodniłem, że czyzajm[4, 3] = 1, zatem czyzajm[4, 1] = 0.

Sprawdzam ograniczenie (6):
Parami pracowników wymagających nadzoru są: (1, 2), (2, 3).
Pracownikami nadzorującymi są 4 i 5.
czyzajm[3, 4] = czyzajm[2, 4] = 1, czyli czyzajm[4, 4] = 1 lub czyzajm[5, 4] = 1.
Ale wcześniej pokazałem, że czyzajm[5, 4] = 0, więc czyzajm[4, 4] = 1.

Zostało mi jeszcze znaleznie wartości zmiennych czyzajm[1, 1], czyzajm[1, 2], czyzajm[2, 2], czyzajm[2, 3], czyzajm[3, 1], czyzajm[4, 2], dla których minimum funkcji celu jest przyjmowane.

Tabela wartości zmiennych czyzajm[p, kd] dla p ze zbioru P, kd ze zbioru KD wygląda następująco:

kd/p	1	2	3	4	5	6
1	?	?	0	0	1	0
2	0	?	?	1	0	0
3	?	0	0	1	0	0
4	0	?	1	1	1	1
5	1	0	0	0	0	1

Widać z niej, że z ograniczenia (2): czyzajm[2, 3] = 1.
Ponadto, z ograniczenia (2): czyzajm[1, 1] = 1 lub czyzajm[3, 1] = 1.
Co więcej, również z ograniczenia (2): Dla dwóch pracowników p ze zbioru {1, 2, 4}: czyzajm[p, 2] = 1. (*)
Sprawdzam ograniczenie (7):

Otrzymuję, że:
Z pracownika 1: czas >= 3 + czyzajm[1, 1] + 2 * czyzajm[1, 2];
Z pracownika 2: czas >= 5 + 2 * czyzajm[2, 2];
Z pracownika 3: czas >= 4 + czyzajm[3, 1];
Z pracownika 4: czas >= 11 + 2 * czyzajm[4, 2];
Z pracownika 5: czas >= 4.

Zatem musi być spełniony warunek: czas >= 11 + 2 * czyzajm[4, 2].
Z (*) nie jest możliwe, żeby czyzajm[1, 2] = czyzajm[4, 2] = 0, więc czas >= 13.

1 przypadek. Zakł., że czyzajm[4, 2] = 0.
Wtedy z (*): czyzajm[1, 2] = czyzajm[2, 2] = 1.
Jeśli pracownicy 1 i 2, gdy pracują nad tym samym działem, to wymagają nadzoru.
Ale czyzajm[4, 2] = czyzajm[5, 2] = 0. Czyli ograniczenie (6) nie jest spełnione. Stąd odrzucam ten przypadek.

2 przypadek. Zakł., że czyzajm[1, 2] = 0.
Wtedy czyzajm[2, 2] = czyzajm[4, 2] = 1.
Czyli czas = 13.

3 przypadek. Zakł., że czyzajm[2, 2] = 0.
Wtedy czyzajm[1, 2] = czyzajm[4, 2] = 1.
Czyli czas = 13.

4 przypadek. Zakł., że czyzajm[1, 2] = czyzajm[2, 2] = czyzajm[4, 2] = 1.
Wtedy czas = 13.

Rozwiązaniem problemu jest:
Wartość funkcji celu: 13.
Pracownik 1 zajmuje się działami: 2, 5.
Pracownik 2 zajmuje się działami: 2, 3, 4.
Pracownik 3 zajmuje się działami: 1, 4.
Pracownik 4 zajmuje się działami: 2, 3, 4, 5, 6.
Pracownik 5 zajmuje się działami: 1, 6.

Rozwiązanie otrzymane przez program jest również identyczne.
Zatem w tym przypadku program także działa prawidłowo.