Wydział Matematyki i Nauk Informacyjnych
Politechniki Warszawskiej

# Teoria Algorytmów i Obliczeń
# Laboratorium - Etap 1

## Specyfikacja wstępna

Adrian Bednarz,
Bartłomiej Dach,
Tymon Felski

Wersja 1.0

20.10.2017

# Spis treści

# 1 Opis problemu

Niniejszy rozdział poświęcony jest dokładnemu opisaniu podstawowej wersji zadanego problemu.

Dane są zbiory:

- $E$ - **ekspertów** realizujących projekty,
- $U$ - **umiejętności** posiadanych przez ekspertów,
- $P$ - **projektów** do zrealizowania.

Każdemu ekspertowi przypisany jest wektor binarny opisujący posiadane przez niego umiejętności. Przykładowo, jeżeli ekspert posiada umiejętność $i$, to w wektorze umiejętności odpowiadającemu temu ekspertowi na $i$-tym miejscu znajduje się znak 1, w przeciwnym wypadku — 0.

**Przykład — wektory ekspertów**

Załóżmy, że liczność zbioru umiejętności $U$ jest równa 5. Wówczas ponumerujemy umiejętności rozważane w problemie liczbami z zakresu $[1, 5]$. Niech pewien ekspert ze zbioru $E$ posiada umiejętności 2, 3 i 5. Wówczas wektor opisujący jego umiejętności to:

$$[0, 1, 1, 0, 1]$$

Każdemu projektowi przypisany jest wektor liczbowy opisujący zapotrzebowanie na ekspertów posiadających dane umiejętności. Przykładowo, jeżeli do realizacji projektu potrzeba trzech ekspertów posiadających umiejętność $i$, to w wektorze umiejętności odpowiadającemu temu projektowi na $i$-tym miejscu znajduje się liczba 3.

**Przykład — wektory zapotrzebowania projektów**

Utrzymując założenie o liczności zbioru umiejętności z poprzedniego przykładu, rozważmy pewien projekt ze zbioru $P$. Niech jego zapotrzebowanie na ekspertów posiadających umiejętności 1 i 4 wynosi odpowiednio 4 i 3, a na pozostałe — 0. Wówczas wektor opisujący zapotrzebowanie tego projektu to:

$$[4, 0, 0, 3, 0]$$

Wszystkie projekty realizowane są w tym samym oknie czasowym, tzn. prace nad każdym z nich rozpoczynają się w momencie $t_0$ i kończą w późniejszym momencie $t_k$. Oznacza to, że jeżeli dany ekspert zostanie zatrudniony do pracy nad projektem $P_1$, to nie będzie mógł brać udziału w równoległym projekcie $P_2$. Ponadto każdy ekspert podczas pracy nad projektem może wykorzystywać tylko jedną z posiadanych umiejętności i nie może jej zmienić w trakcie trwania prac.

Prace nad danym projektem zostaną zakończone nawet jeżeli nie zostanie mu przydzielona wymagana liczba ekspertów posiadających potrzebne umiejętności, określona przez

wektor liczbowy odpowiadający temu projektowi. Będzie on natomiast zrealizowany gorzej niż w przypadku, gdyby przypisana została odpowiednia liczba ekspertów. Może się również zdarzyć, że najbardziej optymalne okaże się takie przypisanie ekspertów, że nad pewnym projektem nie będzie pracował nikt.

Jeżeli zapotrzebowanie projektu nie zostanie wypełnione w całości, mamy do czynienia z brakami. Poprzez braki rozumiemy różnicę pomiędzy zapotrzebowaniem projektu na ekspertów o danych umiejętnościach a rzeczywistym przydziałem. Aby wyznaczyć braki w danym projekcie, należy odjąć wektor zapotrzebowania projektu na ekspertów od wektora zawierającego informację o ekspertach przydzielonych do tego projektu i zsumować elementy uzyskanej różnicy. Dokładna definicja tego pojęcia znajduje się w rozdziale zawierającym dowód poprawności (definicja 8).

**Przykład — obliczanie liczby braków**

Niech wektorem opisującym zapotrzebowanie pewnego projektu na ekspertów będzie:

$$[4, 0, 0, 3, 0]$$

Załóżmy, że do tego projektu zostali przypisani eksperci opisani przez wektory:

$$\begin{aligned} &[1, 0, 1, 0, 1] && \text{(wykorzystuje umiejętność 1)} \\ &[1, 0, 0, 0, 1] && \text{(wykorzystuje umiejętność 1)} \\ &[1, 1, 0, 1, 0] && \text{(wykorzystuje umiejętność 4)} \\ &[0, 0, 0, 1, 1] && \text{(wykorzystuje umiejętność 4)} \end{aligned}$$

Wówczas przydział ekspertów do tego projektu można opisać wektorem:

$$[2, 0, 0, 2, 0]$$

Braki w tym projekcie obliczymy następująco:

$$\sum([4, 0, 0, 3, 0] - [2, 0, 0, 2, 0]) = \sum([2, 0, 0, 1, 0]) = 3$$

Naszym celem jest zminimalizowanie braków w obrębie wszystkich projektów (sumy wszystkich braków), czyli znalezienie optymalnego przydziału ekspertów do projektów.

# 2 Problem znajdowania maksymalnego przepływu

Okazuje się, że problem opisany w sekcji 1 można uogólnić do znanego problemu znajdowania maksymalnego przepływu w sieciach. W tej sekcji zdefiniowane są podstawowe pojęcia potrzebne do opisu tego problemu.

**Definicja 1. Siecią** nazywamy czwórkę uporządkowaną $S = (G, c, s, t)$, gdzie:

- $G = (V, E)$ jest grafem skierowanym,
- $c : E \to \mathbb{N}$ to tzw. funkcja przepustowości,

- $s, t \in V, s \neq t$ są dwoma wyróżnionymi wierzchołkami grafu $G$ — kolejno źródłem i ujściem sieci.

**Definicja 2. Przepływem** w sieci $S$ nazywamy funkcję $f : E \to \mathbb{N}$ spełniającą następujące warunki:

1. $\forall_{e \in E} \quad 0 \leqslant f(e) \leqslant c(e)$,
2. $(\forall v \in V - \{s, t\}) \sum_{u:\ uv \in E} f(uv) = \sum_{u:\ vu \in E} f(vu)$ — tzw. prawo Kirchhoffa.

W ogólniejszym przypadku funkcje przepustowości i przepływu mogą mieć wartości nieujemne rzeczywiste, lecz założenie o całkowitości zapewnia, że algorytmy wyznaczające maksymalny przepływ zawsze zakończą działanie.

Prawo Kirchhoffa stanowi, że suma wartości przepływu na krawędziach wchodzących do danego wierzchołka musi być równa sumie wartości przepływu na krawędziach wychodzących z tego wierzchołka.

**Definicja 3. Wartością przepływu** $f$ w sieci $S$ nazywamy liczbę

$$W(f) = \sum_{u:\ su \in E} f(su) - \sum_{u:\ us \in E} f(us)$$

Powyższe definicje wystarczą, aby zdefiniować problem maksymalnego przepływu.

**Definicja 4** (Problem maksymalnego przepływu)**.** Dana jest sieć $S = (G, c, s, t)$. Szukamy przepływu $f$ o maksymalnej wartości $W(f)$, zwanego również **przepływem maksymalnym**.

Zagadnienie znajdowania maksymalnego przepływu jest rozwiązywalne przez wiele zachłannych algorytmów opartych na metodzie Forda-Fulkersona, polegającej na znajdowaniu ścieżek w tzw. sieci rezydualnej. Szczegółowy opis jednego z takich algorytmów znajduje się w następnej sekcji.

# 3 Algorytm

W poniższym rozdziale precyzyjnie sformułowano algorytm pozwalający na rozwiązanie dowolnego zadania w postawionym problemie. Pseudokod został podzielony na fragmenty, z którego każdy będzie rozwiązywał pewien podproblem, w celu ułatwienia opisu głównej części algorytmu.

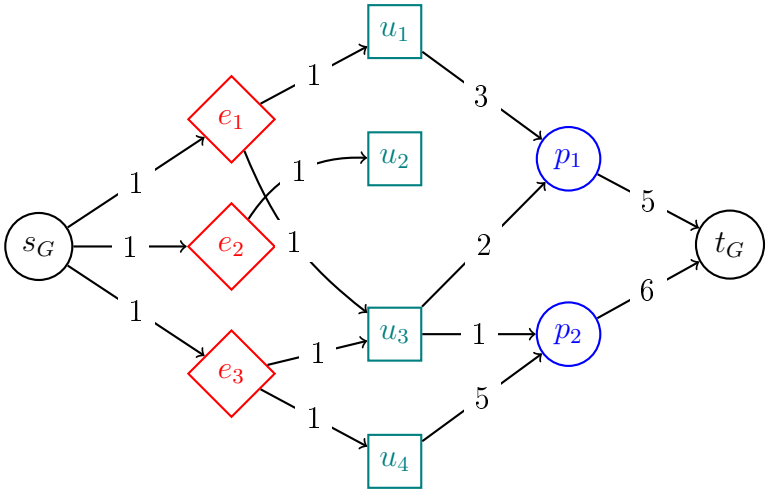
## 3.1 Konstrukcja sieci podstawowej

Niektóre podproblemy opisane w dalszej części rozdziału będą wymagać sieci reprezentowanej przez graf skierowany, który można utworzyć na podstawie danych z zadania.

Proponowany graf `G` będzie posiadał $|E| + |U| + |P| + 2$ wierzchołków, które utworzą w nim pięć warstw. Wyróżnione zostaną dwa wierzchołki — źródło `s` i ujście `t` sieci, z którego każdy będzie jedynym w swojej warstwie. Pozostałe trzy warstwy pomiędzy nimi będą zawierać wierzchołki reprezentujące odpowiednio ekspertów, umiejętności i projekty.

```
3 // Liczba ekspertów
4 // Liczba umiejętności
2 // Liczba projektów
// Wektory ekspertów
[1, 0, 1, 0]
[0, 1, 0, 0]
[0, 0, 1, 1]
// Zapotrzebowanie
// projektów
[3, 0, 2, 0]
[0, 0, 1, 5]
```

(a) Przykładowy format pliku wejściowego dla programu




(b) Wygląd sieci skonstruowanej na podstawie dostarczonych danych

Rysunek 1: Przykład skonstruowanej sieci na podstawie określonego zadania problemu

Poniżej znajduje się pseudokod pozwalający na stworzenie takiego grafu.

**Konstrukcja sieci podstawowej**

```
G <- graf skierowany o liczbie wierzchołków równej |E|+|U|+|P|+2
dla każdego wierzchołka e reprezentującego eksperta w G:
    dodaj krawędź (s, e) do G
    G.c[s, e] <- 1
dla każdego wierzchołka e reprezentującego eksperta w G:
    dla każdej umiejętności u posiadanej przez eksperta e:
        dodaj krawędź (e, u) do G
        G.c[e, u] <- 1
dla każdego wierzchołka p reprezentującego projekt w G:
    dla każdej umiejętności u wymaganej przez projekt p:
        dodaj krawędź (u, p) do G
        G.c[e, u] <- liczba wymaganych ekspertów z u
dla każdego wierzchołka p reprezentującego projekt w G:
    dodaj krawędź (p, t) do G
    G.c[p, t] <- suma przepustowości krawędzi wchodzących do p
```

## 3.2 Konstrukcja sieci rezydualnej

W celu konstruowania w sieci ścieżek rozszerzających niezbędne jest utworzenie pomocniczej sieci rezydualnej. Przepustowość krawędzi w tej sieci zależy od wartości przepływu na krawędziach oryginalnej sieci.

Niech dana będzie krawędź $uv$ i przepływ $f$. Wówczas w sieci rezydualnej istnieją krawędzie:

- $uv$ o przepustowości $c(uv) - f(uv)$,

- $vu$ o przepustowości $f(uv)$.

Sieć rezydualna będzie aktualizowana po każdym zwiększeniu przepływu wzdłuż ścieżki powiększającej. Na początku działania algorytmu (przy zerowym przepływie) będzie ona wyglądać prawie tak samo, jak wyjściowa sieć. Wystarczy stworzyć sieć opisaną w poprzednim punkcie i rozszerzyć ją w sposób następujący:

Rozszerzenie konstrukcji sieci podstawowej

```
dla każdej krawędzi (u, v) w G:
    dodaj krawędź (v, u) do G
    G.c[v, u] <- 0
```

## 3.3 Wyszukiwanie ścieżek

Do funkcjonowania algorytmu potrzebna jest podprocedura wyszukująca ścieżki między dwoma wierzchołkami grafu, co można dość prosto zaimplementować poprzez modyfikację przeszukiwania w głąb (ang. depth-first search, DFS). Poniżej znajduje się pseudokod żądanej podprocedury, wyszukującej ścieżki w grafie `G` od wierzchołka `u` do `w`.

Wyszukiwanie ścieżek w grafie

```
findPath(G, u, w):
    L <- pusta lista
    dodaj u na koniec L
    jeśli depthFirstSearch(G, L, w):
        zwróć constructPath(G, L)
    zwróć pustą listę

depthFirstSearch(G, L, w):
    u <- ostatni element listy L
    jeśli u = w:
        zwróć prawdę
    dla każdej krawędzi (u, v) wychodzącej z u w G:
        jeśli v nie należy do L:
            dodaj v na koniec L
            jeśli depthFirstSearch(G, L, w):
                zwróć prawdę
            usuń v z L
    zwróć fałsz
```

Powyższe wyszukiwanie używa pomocniczej funkcji budującej ścieżkę na podstawie listy wierzchołków `L`, która zdefiniowana jest następująco:

Budowanie ścieżki na podstawie wierzchołków

```
constructPath(L):
    E <- pusta lista
    u <- pierwszy element listy L
    dla każdego wierzchołka v poza pierwszym z L:
        dodaj krawędź (u, v) na koniec E
        u <- v
    zwróć E
```

Zaproponowany algorytm wyszukiwania jest algorytmem z powrotami. W zapisie założono, że lista L jest przekazywana przez referencję i modyfikowana we wszystkich wywołaniach rekurencyjnych.

W algorytmie Edmondsa-Karpa do wyszukiwania ścieżek wykorzystywane jest przeszukiwanie wszerz. W przypadku skonstruowanego na potrzeby zadania grafu wszystkie ścieżki mają jednak tę samą długość, a ponadto DFS znajdzie ścieżkę z mniejszą złożonością pamięciową.

## 3.4 Wyznaczanie przepływu maksymalnego

Poniżej znajduje się zapis algorytmu wyznaczającego przepływ maksymalny zgodnie z zasadą opisaną przez Forda i Fulkersona. `G_res` jest w tym przypadku siecią rezydualną wyznaczoną na podstawie sieci podstawowej skonstruowanej w pierwszym podrozdziale.

Wyznaczanie przepływu maksymalnego

```
maxFlow(G_res):
    f <- przepływ zerowy
    powtarzaj
        L <- findPath(G_res, s, t)
        df <- inf
        dla każdej krawędzi (u, v) z L:
            jeśli G_res.c[u, v] < df:
                df <- G_res.c[u, v]
        dla każdej krawędzi (u, v) z L:
            f[u, v] += df
            G_res.c[u, v] -= df
            G_res.c[v, u] += df
    dopóki lista L jest niepusta
    zwróć f
```

Algorytm ten jednak nie rozwiązuje zadania wyjściowego. Optymalną wartość braków i przyporządkowanie ekspertów do zadań należy wyznaczyć na podstawie uzyskanego przepływu.

### 3.5 Konstrukcja rozwiązania

Poniższa konstrukcja przyporządkowania ekspertów do projektów korzysta z sieci podstawowej `G` (nie z sieci rezydualnej `G_res`) i z wyznaczonego przepływu `f`.

Wyznaczanie przydziału

```
constructAssignment(G, f):
    skills <- słownik pustych kolejek dla poszczególnych umiejętności
    L <- pusta lista
    dla każdego wierzchołka e reprezentującego eksperta w G:
        dla każdej krawędzi (e, u) wychodzącej z e w G:
            jeśli f[e, u] = 1:
                skills[u].push(e)
    dla każdego wierzchołka u reprezentującego umiejętność w G:
        dla każdej krawędzi (u, p) wychodzącej z u w G:
            dopóki f[u, p] > 0:
                e <- skills[u].pop()
                dodaj krotkę (e, u, p) na koniec L
                f[u, p] -= 1
    zwróć L
```

Poniższa funkcja oblicza finalną wartość braków w wyznaczonym wyżej przyporządkowaniu `L`.

Wyznaczanie braków

```
calcLosses(G, L):
    need <- 0
    dla każdej krawędzi (p, t) wchodzącej do ujścia t w G:
        need += G.c[p, t]
    flow <- liczba elementów w liście L
    zwróć (need - flow)
```

# 4 Dowód poprawności

W tej sekcji wykażemy związek między postawionym problemem a zagadnieniem wyznaczania przepływu maksymalnego oraz równoważność rozwiązań obu zadań.

Na początek zdefiniujmy w sposób formalny pojęcia użyte w oryginalnym zadaniu. Załóżmy, że dane są następujące zbiory:

- **zbiór ekspertów**, oznaczony $E$,
- **zbiór umiejętności**, oznaczony $U$,
- **zbiór projektów**, oznaczony $P$.

**Definicja 5. Funkcją umiejętności** nazywamy funkcję

$$\text{ability} : E \times U \to \{0, 1\}$$

gdzie dla eksperta $e \in E$ oraz umiejętności $u \in U$ zachodzi $\text{ability}(e, u) = 1$ wtedy i tylko wtedy, gdy ekspert $e$ posiada umiejętność $u$, zaś 0 w przeciwnym przypadku.

**Definicja 6. Zapotrzebowaniem projektu** nazywamy funkcję

$$\text{need} : P \times U \to \mathbb{N}$$

gdzie dla projektu $p \in P$ i umiejętności $u \in U$ zachodzi $\text{need}(p, u) = n$ wtedy i tylko wtedy, gdy w projekcie $p$ liczba potrzebnych ekspertów w dziedzinie umiejętności $u$ wynosi $n$.

Zauważmy, że funkcje umiejętności i zapotrzebowania projektu są tożsame z wektorami wejściowymi zadania problemu (odpowiadają wzięciu odpowiedniej ich współrzędnej).

**Definicja 7. Przyporządkowaniem eksperta** nazywamy relację

$$\text{assign} \subsetneq E \times U \times P$$

gdzie projekt $p \in P$, umiejętność $u \in U$ oraz ekspert $e \in E$ są ze sobą w relacji assign wtedy i tylko wtedy, gdy

- ekspert $e$ posiada umiejętność $u$ (tj. $\text{ability}(e, u) = 1$),
- ekspert $e$ został przyporządkowany do pracy w projekcie $p$ w dziedzinie umiejętności $u$.

Każdy ekspert $e \in E$ może być w relacji z co najwyżej jedną parą postaci $(u, p)$, gdzie $u \in U, p \in P$.
Ponadto, dla każdego projektu $p$ i umiejętności $u$ musi zachodzić

$$\text{assigned}(p, u) \stackrel{\text{def}}{=} |\{e \in E : (e, u, p) \in \text{assign}\}| \leq \text{need}(p, u)$$

**Definicja 8. Liczbą braków w projekcie** $p$ dla danego przyporządkowania assign nazywamy liczbę

$$\text{missing}(p, \text{assign}) = \sum_{u \in U} (\text{need}(p, u) - \text{assigned}(p, u))$$

**Definicja 9. Całkowitą liczbą braków** dla danego przyporządkowania assign nazywamy liczbę

$$M(\text{assign}) = \sum_{p \in P} \text{missing}(p, \text{assign})$$

Widoczne jest, że $M$ jest parametrem minimalizowanym w postawionym problemie, zależnym od końcowego przyporządkowania.

Na podstawie powyższych definicji skonstruujemy teraz sieć, której użyjemy do wyznaczenia rozwiązań problemu.

**Definicja 10. Siecią przydziałów** nazwiemy sieć $S = (G, c, s, t)$, gdzie:

- $G = (V_G, E_G)$ jest grafem skierowanym takim, że:
  - $V_G = E \cup U \cup P \cup \{s, t\}$,
  - $E_G = \{(e, u) : \text{ability}(e, u) = 1, e \in E, u \in U\} \cup \{(u, p) : \text{need}(u, p) > 0, u \in U, p \in P\}$, tj. krawędziami połączeni są eksperci z ich opanowanymi umiejętnościami, oraz projekty z potrzebnymi do ich realizacji umiejętnościami.
- $c : E_G \to \mathbb{N}$ jest funkcją pojemności zdefiniowaną dla krawędzi $e_G$ następująco:
  - jeżeli $e_G = se, e \in E$, to $c(e_G) = 1$,
  - jeżeli $e_G = eu, e \in E, u \in U$, to $c(e_G) = \text{ability}(e, u) = 1$,
  - jeżeli $e_G = up, u \in U, p \in P$, to $c(e_G) = \text{need}(p, u)$,
  - jeżeli $e_G = pt, p \in P$, to
    $$c(e_G) = \sum_{sp \in E_G} c(up)$$
    (tj. pojemność tej krawędzi jest równa sumie pojemności krawędzi wchodzących do wierzchołka $p$).
- $s, t$ są wyróżnionymi wierzchołkami z $V_G$ — kolejno źródłem i ujściem.

**Definicja 11. Odległością** $\text{dist}(u, v)$ wierzchołka $u$ od wierzchołka $v$ w grafie $G$ nazywamy:

- liczbę krawędzi w najkrótszej ścieżce od $u$ do $v$, jeśli taka istnieje,
- 0, jeśli $u = v$,
- $\infty$, jeśli $u \neq v$ i nie istnieje ścieżka od $u$ do $v$.

**Twierdzenie 1.** *Przepływ maksymalny w sieci przydziałów wyznacza przyporządkowanie o minimalnej możliwej wartości parametru $M$.*

*Dowód.* Aby dowieść to twierdzenie, wykażemy kolejno, że:

1. Każde zadanie problemu wyjściowego jest równoważne z pewną siecią przydziałów $S$.
   - ($\Rightarrow$) Niech dane będzie pewne zadanie problemu wyjściowego (tj. dane będą zbiory $E, U, P$ oraz funkcje ability i need). Wówczas można dla tego zadania skonstruować sieć przydziałów za pomocą konstrukcji pokazanej w sekcji 3 i definicji 10.
   - ($\Leftarrow$) Niech dana będzie pewna sieć przydziałów $S = (G, c, s, t)$. Zauważmy, że wierzchołki sieci przydziałów dzielą się z definicji sieci na pięć zbiorów, określonych przez ich odległość od źródła:
     - $\text{dist}(u, v) = 0$ — singleton $\{s\}$,
     - $\text{dist}(u, v) = 1$ — zbiór ekspertów $E$,
     - $\text{dist}(u, v) = 2$ — zbiór umiejętności $U$,
     - $\text{dist}(u, v) = 3$ — zbiór projektów $P$,
     - $\text{dist}(u, v) = 4$ — singleton $\{t\}$,

co daje nam wyjściowe zbiory $E, U, P$.

Na podstawie powyższych zbiorów i funkcji przepustowości $c$ można zrekonstruować również funkcje ability i need:

- Dla każdego $e \in E$ i $u \in U$ funkcję ability możemy zdefiniować jako

$$\text{ability}(e,u) = \begin{cases} 1, & eu \in E_G \\ 0, & eu \notin E_G \end{cases}$$

- Dla każdego $u \in U$ i $p \in P$ funkcję need możemy zdefiniować jako

$$\text{need}(p,u) = \begin{cases} c(up), & up \in E_G \\ 0, & up \notin E_G \end{cases}$$

Z każdej sieci przydziałów można skonstruować więc zadanie oryginalnego problemu.

2. Dowolny przepływ w sieci przydziałów wyznacza ilość wykonanych podzadań przy danym przyporządkowaniu.

   Niech dany będzie pewien przepływ $f$ w sieci przydziałów $S$. Przyporządkowanie ekspertów do projektów $\text{assign}_f$ wyznaczamy w następujący sposób:

   (a) Pewnego eksperta $e \in E$ przypisujemy do umiejętności $u \in U$, jeżeli $f(e,u) = 1$.

   (b) Niech dana będzie pewna umiejętność $u \in U$. Oznaczmy zbiór ekspertów przypisanych do tej umiejętności w punkcie (a) jako $E_u$.
   Zbiór $E_s$ dzielimy na rozłączne podzbiory $E_{u,p}$ takie, że $|E_{u,p}| = f(u,p)$.

   (c) Dla każdego z uzyskanych podzbiorów $E_{u,p}$, gdzie $u \in U, p \in P$, do relacji $\text{assign}_f$ dodajemy krotki

$$\{(e,u,p) : e \in E_{u,p}\}$$

   Zauważmy następujące fakty:

   - Rozważmy wierzchołek $e \in E$ odpowiadający pewnemu ekspertowi.
     Z definicji zbioru krawędzi sieci i funkcji przepustowości, do wierzchołka tego wchodzi dokładnie jedna krawędź o pojemności 1, a wychodzi z niego co najwyżej $|U|$ krawędzi o pojemności 1.
     Stąd w przepływie $f$ tylko jedna z krawędzi wychodzących może mieć przepływ 1, a więc każdy ekspert może być przyporządkowany do co najwyżej jednej umiejętności.
   - Rozważmy dowolny wierzchołek $u \in U$ odpowiadający pewnej umiejętności.
     Z własności przepływu mamy

$$\sum_{wu \in E_G} f(wu) = \sum_{uv \in E_G} f(uv)$$

     Wiedząc, że wszystkie krawędzie wchodzące do $s$ wychodzą ze zbioru $E$, oraz że wszystkie krawędzie wychodzące z $s$ wchodzą do zbioru $P$, mamy

$$\sum_{e \in E} f(eu) = \sum_{p \in P} f(up)$$

Krawędzie o niezerowym przepływie wchodzące do $e$ reprezentują ekspertów przydzielonych do danej umiejętności, zaś krawędzie o niezerowym przepływie wychodzące z $e$ reprezentują zapotrzebowanie projektów na ekspertów z umiejętnością $u$.

Ponieważ suma przepływów krawędzi wchodzących i wychodzących jest taka sama, każdego eksperta przydzielonego do $u$ można przypisać do dokładnie jednego podzadania (do dokładnie jednego projektu w dziedzinie umiejętności $u$), a więc można wykonać punkt (b) konstrukcji.

- Rozważmy dowolne dwa wierzchołki $u \in U, p \in P$ takie, że $up \in E_G$. Z definicji sieci mamy $c(u,p) = \text{need}(u,p)$, a z konstrukcji rozwiązania wynika, że $f(u,p) = \text{assigned}_f(u,p)$. Stąd na mocy definicji przepływu mamy

$$\text{assigned}_f(u,p) = f(u,p) \leq c(u,p) = \text{need}(u,p)$$

- Rozważmy dowolny wierzchołek $p \in P$. Z definicji funkcji pojemności, jeśli wszystkie krawędzie wchodzące do $p$ będą wysycone przepływem (tj. $f(e) = c(e)$), to przepływ ten można przekazać w całości do ujścia krawędzią $pt$, bo

$$c(pt) = \sum_{up \in E_G} c(up)$$

Stąd pojemność krawędzi $pt$ nie ogranicza wartości maksymalnego przepływu.

Wyznaczone przyporządkowanie $\text{assign}_f$ spełnia więc wszystkie warunki prawidłowego przyporządkowania ekspertów do projektów, a ilość elementów w tej relacji odpowiada liczbie wykonanych podzadań.

3. Przepływ maksymalny wyznacza minimalną wartość parametru M.

Na mocy punktu 1., każde zadanie oryginalnego problemu jest równoważne pewnej sieci przydziałów, zaś na mocy punktu 2 dowolny przepływ w sieci przydziałów wyznacza ilość wykonanych podzadań. Wobec tego przepływ maksymalny $f_{\max}$ wyznacza maksymalną ilość wykonanych podzadań, równą $\left|\text{assign}_{f_{\max}}\right|$.

Zauważmy, że

$$\begin{aligned} M(\text{assign}) &= \sum_{p \in P} \text{missing}(p, \text{assign}) = \\ &= \sum_{p \in P} \sum_{u \in U} (\text{need}(p,u) - \text{assigned}(p,u)) = \\ &= \left( \sum_{p \in P} \sum_{u \in U} \text{need}(p,u) \right) - \left( \sum_{p \in P} \sum_{u \in U} \text{assigned}(p,u) \right) = \\ &= \left( \sum_{p \in P} \sum_{u \in U} \text{need}(p,u) \right) - |\text{assign}|, \end{aligned}$$

gdzie ostatnia równość wynika z definicji 7 (przyjęto, że jeden ekspert może być w relacji z co najwyżej jedną parą $(u,p)$).

W związku z tym maksymalizacja liczności przyporządkowania assign jest równoważna minimalizacji parametru $M$, co kończy dowód.

□