СОФИЙСКИ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ“
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

КУРСОВ ПРОЕКТ
ПО СИСТЕМИ, ОСНОВАНИ НА ЗНАНИЯ

Тема:

Програма за съставяне на график на работни смени като задача за удовлетворяване на ограничения (CSP)

Студент(и):

Мария Стефанова Маргаритова, 1MI0700004

Мая Деянова Денева, 2MI0700013

София, януари 2024 г.

1. Формулировка на задачата

Задачата се състои в съставяне на график на работни смени като задача за удовлетворяване на ограничения (CSP). Проблемите с планиране събират популярност през последните години, тъй като се основават на концепцията за "малък ход - голям ефект", която максимизира печалбата на различни организации чрез оптимално използване на работна сила и оборудване. През последните две десетилетия са публикувани над 300 статии, обсъждащи различни проблеми, свързани с планирането.

Работа като крупие в казино подсигурява динамична, постоянно променяща се работна среда с атрактивно заплащане. Работата обаче има недостатъци: работят се нощни смени, работният ден е дълъг - 12 часа, а постоянният контакт с клиенти (често ядосани поради загуби) може да доведе до преумора у крупиетата, което не е благоприятно нито за работника, нито за работодателя.

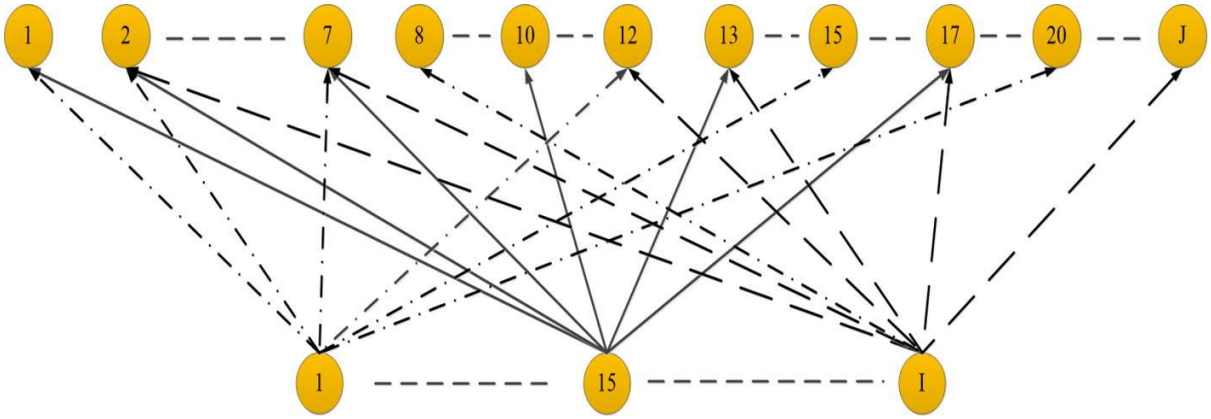
За да разрешим този проблем, създадохме план за месечен график на работни смени на крупиета в казина, като поставихме най-различни ограничения, които да подсигурят оптимални резултати за работния процес.

Ограничения:

- Всеки ден се състои от дневна и нощна смяна.
- Продължителност на една смяна е 12 часа.
- Не може да има дневна след нощна смяна.
- Не може да има повече от 4 последователни смени за даден служител.
- В месеци с 31 дни - един служител има 16 смени.
- В месеци с 30 дни - един служител има 15 смени.
- В месеци с 28 или 29 - един служител има 14 смени.
- Един човек не може да работи повече от една смяна за един ден.
- Всеки ден трябва да има поне един човек за дневна и поне един човек за нощна смяна.
- Дневните и нощни смени, чийто общ брой е четен се разпределят: (16 смени - 8 дневни, 8 нощни)
- Дневните и нощни смени, чийто общ брой е нечетен се разпределят: (15 смени - 8 дневни, 7 нощни).
- Има n брой служители между които трябва да се разпределят смените, така че всеки да е работил необходимите смени.
- За всяка смяна трябва да има не повече от 40% служители, които да работят (с цел да не се получава прекалено натрупване в някои дни)

2. Използвани алгоритми

За решението на задачата е използван модел за линейно програмиране с помощта на библиотеката PuLP. Целта е да се минимизират разходите на казиното за заплатите на работниците, и същевременно да максимизира тяхната работоспособност, като ги разпредели на подходящ брой смени, спрямо часовете, които трябва да изработят.



Фигурата показва мрежовото представяне на предложения линеен модел.
В мрежата има определен брой крупиета (I) и определен брой смени (J). За всяко крупие (i) дъгите представляват всички възможни назначения на крупието за всички смени. Крупиетата представляват възли-източници, които се свързват чрез дъги с възлите за смени.

След като имаме описание на задачата, формулираме програмата.

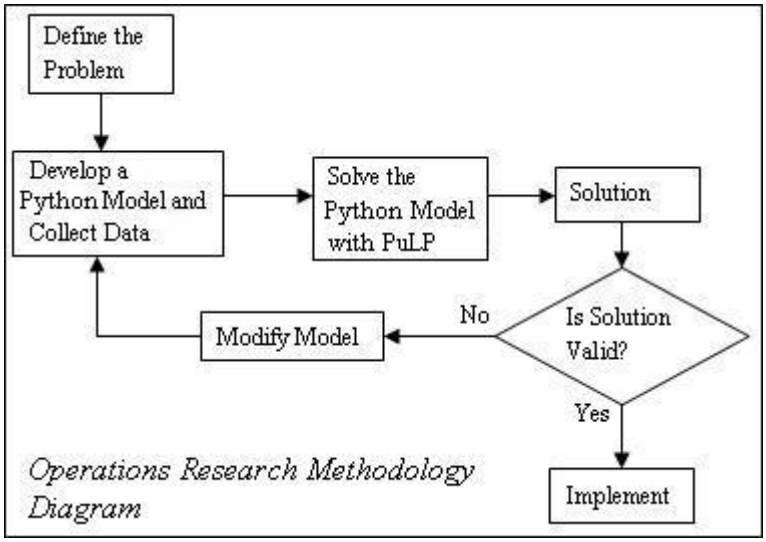
В този етап идентифицираме ключовите количествени решения, ограничения и цели от описанието на проблема и засичаме техните зависимости в математически модел. Процесът на формулиране може да се раздели на 4 основни стъпки:

- Дефиниране на ключовите променливи, като обърнем внимание на мерните единици (например: трябва да решим колко работни дни ще има в месеца).

- Формулиране на целевата функция: Използвайки ключовите променливи, можем да построим минимизираща или максимизираща целева функция.

- Формулиране на ограниченията, които могат да бъдат логически, или явни в описанието на проблема (например, не може човек да работи дневна смяна веднага след нощна смяна). Ограниченията се изразяват чрез променливите.

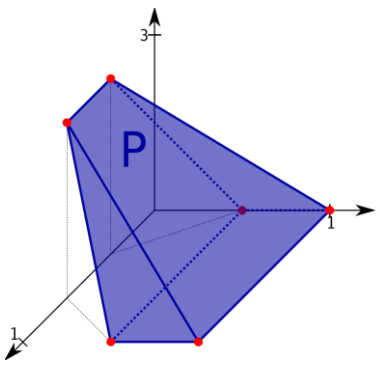
- Формулиране на данните, необходими за целевата функция и ограниченията. За да решим програмата ни трябват конкретни числа като граници на променливите и/или коефициенти на променливите в целевата функция и/или ограниченията.



Следва да се реши самия линеен модел.

Линейното програмиране (ЛП), наричано още линейна оптимизация, е метод за постигане на най-добрия резултат (като максимална печалба или най-ниска стойност) в математически модел, чиито изисквания и цел са представени чрез линейни взаимоотношения. Линейното програмиране придобива все по-голяма важност в последните години поради приложимостта си в програмирането и изкуствения интелект.

По-формално, линейното програмиране е техника за оптимизация на линейна целева функция, подчинена на линейни равенства и линейни неравенства. Неговата област на допустими решения е конвексен политоп, който е множество, дефинирано като сечение на крайно много полупространства, всяко от което е дефинирано чрез линейно неравенство. Целевата функция е реалнозначна афинна (линейна) функция, дефинирана върху този политоп. Алгоритъмът за линейно програмиране намира точка в политопа, където тази функция има най-голяма (или най-малка) стойност, ако такава точка съществува.



Нашата задача е задача с три променливи - служители, дни, и тип смяна.Затворената област от допустими решения на проблем с три променливи е конвексен полиедър. Повърхностите, задаващи фиксирана стойност на целевата функция, са равнини. Задачата по линейно програмиране е да се намери точка в полиедъра, която лежи на равнината с най-висока възможна стойност.

Линейното програмиране се извършва чрез Метода на ревизирания симплекс.

Симплекс-методът е универсален метод, с който може да се решава всяка задача на линейното оптимиране. Основната идея на симплекс-метода е в преминаване от едно базисно решение на задачата към по-добро базисно решение, за да се намери оптималното решение или да се установи, че не съществува такова. Симплекс-методът е един от десетте най-важни алгоритми, разработени през 20-ти век.

Етапи на симплекс-метода:

1) Намиране на **начално базисно решение (план)** или установяване, че областта на решения е празно множество.
2) Установяване дали дадено базисно решение е оптимално или не според **критерия за оптималност**.
3) Намиране на **подобрено базисно решение (подобряване на плана)** или установяване, че целевата функция няма краен оптимум.

Нека общата задача на линейно оптимиране е зададена в стандартна форма:

– *Целева функция*

$$\max(\min) Z(X) = c_1x_1 + c_2x_2 + ... + c_nx_n = C_БX ,$$

където $C_Б = (c_1, c_2, ..., c_n)$ е целеви базисен вектор;

– *Система ограничителни условия:*

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = 1, 2, \text{L} , m$$

– *Условия за неотрицателност:*

$$x_j \geq 0, \quad j = 1, 2, \text{L} , n$$

Избираме за **начално базисно решение с базисни неизвестни** $x_{m+1}, x_{m+2}, ..., x_n$. В този случай **началния базис** е $X = (b_1, b_2, ..., b_m, 0, 0, ..., 0)$ и **стойността на целевата функция** е $Z(X_1) = C_БB = c_1b_1 + c_2b_2 + ... + c_mb_m$. Ще

3. Описание на програмната реализация

Основният метод solve_scheduling_problem приема от конзолата входни параметри: брой дни (за месец) и брой служители. За броят служители последователно се въвеждат имената им.

Дефинират се променливите на решението x[i][j][k], където i е индексът на служителя, j е индексът на деня, а k представлява типа на смяна (0 за дневна, 1 за нощна). Тези променливи са бинарни (0 или 1) и показват дали даден служител е разпределен за конкретна смяна в даден ден.

Дефинира се целевата функция за минимизация: Тя представлява сумата от всички променливи на решението и отразява общия брой разпределени смени.
Добавят се ограниченията, описани в т.1, към линейния модел. Те се изразяват чрез ключовите променливи. Извиква се функцията на библиотеката - model.solve(), която извиква solver, който да намери решение на линейния модел. Функцията връща първото валидно решение, което намери, и спира търсенето по-нататък.
Резултатът се отпечатва в конзолата, първо като се изпишат смените за всеки служител, а накрая излиза ревизирана таблица за всички смени на всички служители.

4. Примери, илюстриращи работата на програмната система

Приложението е конзолно. Въвежда се от колко дни се състои месеца, за който да се подготви график. Следва да се въведат брой служители, след което се въвеждат и имената им. Имената се използват само във визуалната част на проекта - с цел подобрение на потребителското изживяване.

```
D:\desktop\pythonProject2\.venv\bin\python.exe D
Въведете колко дни е месеца: 31
Въведете броя на служителите (максимум 10): 5
Въведете името на служител 1:

Мария
Въведете името на служител 2: Елена
Въведете името на служител 3: Люба
Въведете името на служител 4: Дора
Въведете името на служител 5: Стефка
```

Пример 1: Месец - 31 дни, служители - 5 и съответните им имена

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Мария		Н	Н						Д	Д	Н			Н	Н	Н	Н			Д		Д		Н		Д	Д			Д	Д
Елена	Н		Д	Д	Н	Н	Н				Д		Д								Д		Д	Д	Н			Д	Н	Н	Н
Люба	Д	Д	Д	Н								Н		Д	Д	Д	Д	Н	Н			Н			Д				Н	Н	Н
Дора	Н	Н	Н	Н		Д	Д	Д		Н		Д							Д	Н		Д				Н				Д	Д	Н
Стефка	Д	Д			Д		Д	Н	Н				Н					Д		Д	Н		Н		Н	Н	Н		Д		Д

Резултат 1: Таблица на смените

```
D:\desktop\pythonProject2\.venv\bin\python.exe D
Въведете колко дни е месеца: 30
Въведете броя на служителите (максимум 10): 4
Въведете името на служител 1:

Мая
Въведете името на служител 2: Стели
Въведете името на служител 3: Кони
Въведете името на служител 4: Коци
```

Пример 2: Месец - 30 дни, служители - 4 и съответните им имена

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Мая	Д		Д							Н			Н	Н			Д	Д	Д	Д	Д		Н		Н		Д	Н		Н
Стели				Д		Н				Д			Д	Д			Н	Н	Н		Н	Д	Д		Д	Д	Н		Н	
Кони	Н		Н	Н			Д	Д	Д		Д	Д			Д	Д				Н		Д		Н		Н			Д	Д
Коци	Д	Д	Д		Д	Д	Н	Н	Н		Н	Н			Н	Н						Н		Д				Д		Д

Резултат 2: Таблица на смените

```
D:\desktop\pythonProject2\.venv\bin\python.exe D
Въведете колко дни е месеца: 29
Въведете броя на служителите (максимум 10): 5
Въведете името на служител 1:

Елвис
Въведете името на служител 2: Мартин
Въведете името на служител 3: Драгомир
Въведете името на служител 4: Николета
Въведете името на служител 5: Виктория
```

Пример 3: Месец - 29 дни, служители - 5 и съответните им имена

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Елвис	Д	Н	Н				Н			Н		Н		Д					Д	Д		Н		Д	Н		Д	Д	
Мартин	Н		Д	Н			Д									Д	Д	Д		Н	Н		Н		Д		Д	Н	Н
Драгомир				Д			Д	Д	Н		Д	Д	Н		Д		Н			Н						Д	Н	Н	Н
Николета		Д	Д		Д	Д		Н					Д	Н	Н	Н		Н					Д	Н			Н		Д
Виктория			Н		Н	Н		Д	Д	Д	Н								Н		Д	Д			Н	Н		Д	Д

```
Process finished with exit code 0
```

Резултат 3: Таблица на смените

```
Въведете колко дни е месеца: 28
Въведете броя на служителите (максимум 10): 6
Въведете името на служител 1:

Димитър
Въведете името на служител 2: Радослав
Въведете името на служител 3: Симеон
Въведете името на служител 4: Веселин
Въведете името на служител 5: Галин
Въведете името на служител 6: Никола
```

Пример 4: Месец - 28 дни, служители - 6 и съответните им имена

```
          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Димитър   Н     Н     Н  Н     Д  Н     Д     Н     Д                    Д  Н        Д  Д  Д
Радослав  Д  Д  Д        Д     Н     Н  Н        Д     Д     Н  Н  Н                 Д        Н
Симеон    Н  Н  Н           Д     Д        Н                 Д     Д  Д     Д     Д  Н  Н  Н
Веселин   Н  Н     Н     Н  Н     Д        Д        Н     Д        Н     Д           Д  Д  Д
Галин     Д  Н  Н        Д           Д     Д  Д  Н     Н  Н  Н        Д        Д           Н
Никола    Д  Д  Д  Д  Д  Д                    Н                    Д     Н  Н     Н  Н  Н  Н

Process finished with exit code 0
```

Резултат 4: Таблица на смените

```
Въведете колко дни е месеца: 30
Въведете броя на служителите (максимум 10): 29
Грешка: Не може да има повече от 10 служители.
```

Пример 5: Невалиден вход за брой служители

```
Въведете колко дни е месеца: 10
Грешка: Няма месец с толкова дни!
```

Пример 6: Невалиден вход за брой дни в месеца

5. Литература

В проекта ни е използвана най-различна литература и също така се консултирахме с хора, които работят на подобен график относно това как се чувстват и какво биха подобрили. Получихме от тях примерни графици, които да ни запознаят с предметната област. Установихме, че нашата програма ефективно се справя с дадената задача и получихме положителна обратна връзка от източниците ни. *(крупиета в казина)*

- Linear Programming
- PuLP Documentation
- Nurse Scheduling problem
- Работа на крупие
- Линейно оптимиране