Мађарски метод

Мађарски алгоритам је оптимизациони алгоритам који решава проблем доделе који има сложеност субекспоненцијалног времена и који ишчекује касније примал-дуал методе. Развио га је и објавио 1955. године Харолд Кан, који га је назвао Мађарски алгоритам, јер је он свој рад засновао на раду двојице мађарских математичара:  Денес Кониг and Џенуо Егрбери.

Проблем задатка настаје као резултат различитих ситуација доношења одлука које се односе на посао или задатак у нашим свакодневним активностима. У основи, модел додељивања има два циља или минимизирање или максимизирање. Проблем задатка игра значајну улогу у решавању проблема из стварног живота. Прихватљив и добро коришћен алат широм света.

У моделу додељивања, сви задаци који се обављају морају бити додељени на бази један на један. Два задатка или посла не могу бити додељена једној машини или особљу. Овај модел је техника која одговара на питање како можемо задати н (број објеката) на м (број објеката) на најбољи могући (оптималан) начин. Уколико је број агената и задатака једнак, онда је то проблем балансиране доделе. У супротном, назива се проблем небалансиране доделе.

Главни циљ проблема задавања је минимизација трошкова, времена, простора и максимизација профита, ефективности, ефикасности и др. Може се користити за додељивање послова машини, оператера машини, продајног особља територијама, радника надзорнику, курсева предавањима, инжењера градилиштима између осталог са фокусом на минимизирање или максимизирање.

Формална дефиниција проблема доделе је следећа: Дата су два скупа, А и Т који су једнаке величине, заједно са тежинском функцијом C: A x T → R. Наћи бијекцију f : A → T такву да функција цене

буде минимална.

Проблем доделе представља специјални случај транспортног проблема, који представља специјални случај проблема протока са минималном ценом, који представља специјални случај линеарног програмирања. Иако је могуће решити било који од ових проблема користећи симплекс метод, свака специјализација оставља простора за додатне оптимизације које би одређени алгоритми могли да искористе.

Мађарски метод одвија се у 7 корака, а то су:

1. **Трансформација коефициената матрице**

У случају максимизације ћемо множити целокупну матрицу са -1. Функција која то извршава јесте *MupltiplyWithMinusOne*. Налазимо најмањи елемент по свакој врсти и тај елемент одузмемо од свих елемената те врсте. Исти поступак урадимо за сваку колону. Ову функционалност решили смо функцијом *CalculateZerosByAxis* која за параметре прима матрицу и осу по којој ће да се одузму најмање вредности.

1. **Одређивање независних нула у матрици**

За сваку врсту се одређује број нула и креће се од реда који има минималан број нула. У тој врсти се означава једна као независна, а остале нуле у тој врсти и колони се означавају као зависне. За крај корака узимамо ситуацију када прођемо кроз све врсте у матрици. Посао функције *FindIndependentZeros* јесте да проласком кроз све врсте налази врсте где се налази једна нула. Такве нуле проглашава за независне, и прецртава све остале нуле по врсти и колони где је нашао независну нулу. Након тога ће пронаћи остале независне нуле поновним проласком кроз врсте након прецтравања колона и врста где су биле независне нуле. Функција коју смо користили за прецртавање колона и врста где се налазе независне нуле јесте *ScratchZerosInTheSameRowAndColumn.*

1. **Означавање врста које немају независне нуле.**

За сваку од врста оређиваћемо број независних нула и означаваћемо само оне редове које немају независне нуле. Функција *MarkRowsWithoutIndependentZero* убацује бројеве врста које немају независне нуле у листу.

1. **Прецртавање колона које имају нулу у означеним врстама**

У назначеним врстама где нисмо имали независне нуле тражимо број колоне нуле која није независна функцијом *MarkColsWhereRowIsZero* која за параметре прима матрицу и озачене редове који немају независне нуле. Прецртавање колона које имају нулу у означеним врстама извршава функција *ScratchColsWhereRowIsZero*.

1. **Означавање врста**

Означавају се све врсте које имају независну нулу у прецртаним колонама функцијом *MarkRowsWithIndependentZerosInScratchedCols* која враћа листу бројева свих означених врста.

1. **Прецртавање свих неозначених врста**

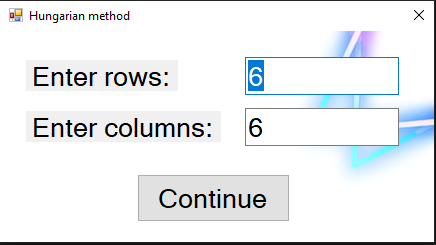
Ова функционалност је решена уз помоћ функције *ScratchUnmarkedColumns* где смо прецтрали неозначене врсте које смо идентификовали на основу означених. За параметре прима матрицу и означене врсте.

1. **Трансформација коефициената матрице**

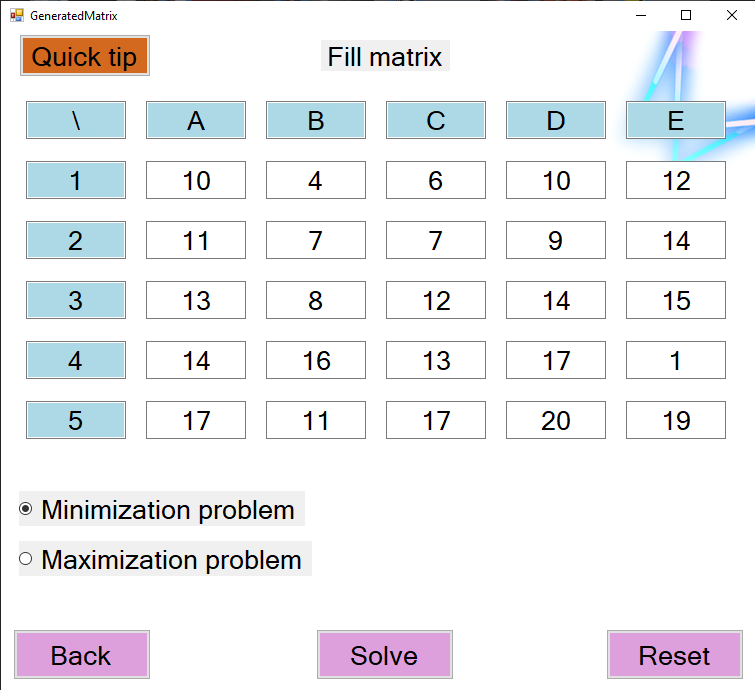
Све непрецртане смањујемо за вредност најмањег броја од непрецртаних, док вредности на пресеку прецртане колоне и врсте повећавамо за такође вредност најмањег од непрецртаних. Остали бројеви се само преписују. Функција која трансформише коефицијенте матрице јесте *TransformationWithMinimunValue*.

Сви остали кораци понављаће се све док у сваком реду и у свакој колони не постоји тачно једна независна нула. Ова провера ће се извршити функцијом *CheckIfAllRowsHaveIndependentZero*.

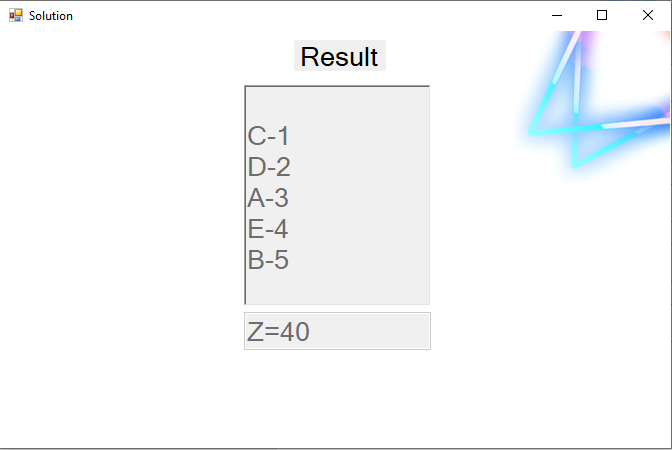
Сви претходно наведени кораци покрећу се кликом на дугме *Solve,* која ће за улаз узети матрицу коју корисник уноси у направљени *GUI*, а вратиће нам крајњи резултат у у коме ће за сваки ред и сваку колону бити означена тачно једна независна нула. Ова интерпертација представља оптималан начин да се на основу дате матрице цена направи тражени распоред у односу на циљ да максимизујемо или да минимизујемо.



**Слика 1** Корисник уноси димензије матрице



**Слика 2** Корисник уноси произвољне вредности у матрицу и бира да ли жели да решава проблем максимизује или минимизације



**Слика 3** Након клика на дугме *Solve* кориснику се приказује решење Мађарског метода

За тестирање саме имплементације алгоритма коришћени су примери са предавања и вежби, као и примери настали као резултат насумичног убацивања вредности у табелу.