IP feladat megoldó speciális esetekre

Tanulók: Karcag Dávid - egyedi munka

1 Bemutatás

Egy IP feladat megoldó algoritmust szeretnék fejleszteni ami speciális IP feladatokra ad megengedett megoldást, vagy bizonyítékot a nem megoldhatóságra, lehetőleg gyors időben. Az algoritmus Branch and Bound gondolatmentet követne, ahol az egyes részfeladatokat változók értékének rögzítésével kapnánk és az egyes részfeladatok megoldhatóságát LP relaxációk és egyéb gyorsan ellenőrizhető megoldáshoz szükséges követelmények ellenőrzésével tervezem megoldani.

2 Célok

A cél egy $Ax = b\ x \geq 0$ alakú feladat megoldása, azzal a plusz megkötéssel hogy $x \in \{0, 1, \ldots, k\}^n$ ahol k nem túl nagy egész, azaz x minden koordinátája kellően kicsi. Formálisabban:
INPUT: $A \in \mathbb{R}^{m \times n}$ és egy $k \in \mathbb{N} : k < K$
OUTPUT: $x \in \mathbb{R}^n : Ax = b, x \in \{0, 1, \ldots, k\}^n$ vagy bizonyíték arra, hogy a a feladat nem megoldható.
Futási idő: A feladat NP-nehéz ezért a cél a k-ban exponenciális futási idejű Brute-force módszernél, gyakorlati példákon sok esetben lényegesen gyorsabb módszer előállítása

3 Alkalmazhatóság

Míg a feladat a k-ra vonatkozó megszorítások nélkül elég általános, kérdés hogy ezek bevezetésével továbbra is megfontolásra érdemes problémát kapunk-e. Az alábbiakban leírok néhány példát érdekes, gyakorlatban vizsgált feladatokra amikre az algoritmus megoldással szolgálhat.

3.1 Gráfok speciális tulajdonságú él vagy csúcshalmazainak kiválasztása

Párosítások, független ponthalmazok, lefogó csúcs illetve élhalmazok, ν_j élhalmazok (élek amik legfeljebb j-szer illeszkednek egy-egy csúcsra), egyéb részhalmazok kiválasztása. Ezek szinte minden esetben megoldhatóak az incidencia mátrix és $x \in \{0, 1\}^n$ tartalmazást binárisan indikáló vektorok segítségével.

3.2 Skálázási, javító utas, illetve egyéb hasonló algoritmusok

Sok esetben bonyolultabb feladatoknál, ha valamilyen formában részfeladatokat vagy lokális javításokat végzünk, visszavezethetjük a problémát az előző alcímben tárgyalt feladatokra pl.: Párosítás keresése a pontos élek részgráfján.

3.3 Legfeljebb k színű színezések, számhalmazok k partícióra osztása

Itt kevés színre vagy partícióra már egész érdekes feladatok oldhatók meg, például a Two-Way Partitioning Problem (számok két egyenlő részhalmazra osztása) és egyéb NP-nehéz problémák eldöntése

4 Lehetséges optimalizációk

Branching: Az algoritmus a részfeladatotokat, nem egyértelmű hogy milyen sorrendeben nézi át, itt kérdés lehet melyikek a "problémás" változók és ezekre fókuszálva haladni. (Például megnézzük a mátrixban hol vannak relatíve nagy számok, avagy melyik oszlop tartalmaz sok illetve kevés nem nulla elemet) **Újraindítások:** A logikai formulákkal foglalkozó előadáson is láttuk, hogy vannak esetek amikor nem mindegy honnan kezdjük el a feladat megoldását, néha megérheti bizonyos információk segítségével újra kezdeni a feladatot.

5 Implementáció menete

Az algoritmust Pythonban tervezem megvalósítani, Főleg a Branch and Bound, illetve a Szimplex módszerre összpontosítva. Sok részfeladatra már, létezik beépített solver, vagy elérhető könyvtár ami a problémát megoldja ezek közül szeretnék minél kevesebbet használni, de tekintve, hogy a most felvázolt projekt helyeken túlmutat a kurzus elvárásain, fokozatosan fogok haladni a következő sorrendben:

5.1 Algoritmus

Be tud olvasni és meg tud oldani egy problémát, szinte minden előre megírt függvény.

5.2 Algoritmus,példák

Az előző fázist kibővítem egy-két kidolgozott megoldott gyakorlati feladattal a **(3.)** pontban említettek közül

5.2.1 Algoritmus, példák, megengedettségi szimplex

A feladat ezen a ponton már egy általam írt megengedettségi szimplex módszert használ. ez a lépés már önmagában is kifejezetten hosszadalmas, lehetséges, hogy idáig már nem jutok el teljesen.

5.2.2 Algoritmus, példák, megengedettségi szimplex, ötletek optimalizálásra

A kész program mellett bemutatok, néhány ötletet arra, hogy a lehetne a programot gyorsabbá tenni, idei taroznak **(4.)** pontban említet módszerek, illetve pár egyéb.