

Tudnivalók

Közelítő algoritmusok

Def: Tfh egy problémának minden I inputjához tartozik egy X_I megoldáshalmaz, a cél pedig olyan $x \in X_I$ megoldás keresése, amelyik az $f_I(x_I)$ célfüggvényt minimalizálja. Az A algoritmus *additív hibája* legfeljebb C , ha tetszőleges I inputhoz olyan y_I megoldást ad, amire $f_I(y_I) \leq \min\{f_I(x_I) + C : x_I \text{ az } I \text{ megoldása}\}$.

Példa: (1) $\chi'(G)$ meghatározása. Tetszőleges egyszerű gráf éleinek $\Delta(G) + 1$ színnel történő színezésére van gyors eljárás, így a $\chi'(G) \geq \Delta(G)$ triviális becslés miatt ennek az additív hibája legfeljebb 1.

(2) A gráf leghosszabb körének meghatározása tartalmazza a Hamilton-kör létezésének eldöntési problémáját, így NP-teljes (reménytelen rá polinomidejű algoritmust találni). Azonban konstans additív hibával sem lehet hatékonyan megoldani a feladatot, és ez abból látszik, ha az inputgráf minden élét egy C hosszúságú úttal helyettesítjük.

Def: Az A algoritmus *multiplikatív hibája* legfeljebb α , más szóval A egy α - (avagy α -*approximációs*) algoritmus, ha minden I inputra $f_I(y_I) \leq k \cdot \min\{f_I(x_I) + C : x_I \text{ az } I \text{ megoldása}\}$.

Példa: A minimális lefogó ponthalmaz $\tau(G)$ méretének meghatározása általában NP-teljes feladat, ám 2-közelítést kapunk, ha kiválasztunk egy nem bővíthető M párosítást G -ben, és a $V(M)$ lefogó ponthalmaz méretét adjuk válaszként.

Halmazfedési probléma Adott az U n -elemű alaphalmaz, az $S_1, S_2, \dots, S_k \subseteq U$ részhalmazok, és ezek $c(S_i)$ nemnegatív költsége. Cél az S_i -k egy minimális összköltségű részrendszere, ami a teljes U alaphalmazt lefedi.

Mohó algoritmus a halmazfedési feladatra Egymás után választjuk az fedésbe bekerülő részhalmazokat. Mindig azt a halmazt választjuk, amelyik fajlagosan a legolcsóbban fedi az eddig le nem fedett pontokat, azaz azt az S_i -t, amire $c(S_i)/m$ minimális, ahol m az S_i által lefedett, de az eddig választott halmazok által le nem fedett U -beli elemek száma.

Tétel: n elemű alaphalmazon futtatva a mohó algoritmus $(1 + \ln n)$ -közelítés a halmazfedési feladatra.

Ütemezési problémák

Input: J_1, J_2, \dots munkák p_i megmunkálási idők, gépek m száma.

Feladat: A munkák gépekre ütemezése. Egy S ütemezésre $S_i(i)$ és $S_m(i)$ adja meg, hogy J_i -t mikor és melyik gépen kell elkezdni. $C_i^S = S_i(i) + p(i)$ a J_i befejezési időpontja, $C_{max}^S = \max_i C_i^S$ pedig az S ütemezés átfutási ideje.

Megkötések: (1) (Itt) a gépek egyformák. (2) Egy gépen egyszerre egy munka végezhető. (3) Minden munkát megszakítás nélkül kell elvégezni.

Lehetséges feltételek J_i -kre: r_i (rendelkezésre állási idő), w_i (súly), d_i (határidő), \prec (precedencia)

Optimalizálandó mennyiségek: $C_{max}^S, (\sum_i C_i^S)/n$

Tömör jelölés: $\alpha|\beta|\gamma$, ahol $\alpha \in \{1, Pm, P\}$ (1 gép, m gép, sok párhuzamos gép)
 $\beta \in \{p = 1, r_j, \prec\}$ (feltételek megadása) $\gamma \in \{C_{max}, \sum_i C_i\}$ (célfv megadása)

Példa: : (1) $1|C_{max}$: 1 gépen kell mihamarabb végezni. $OPT = \sum_i p_i$.

(2) $1|\prec|C_{max}$. Nem mindegy a sorrend, de itt is $OPT = \sum_i p_i$.

Tétel: Az $1|\sum_i C_i$ feladatra a munkák p_i szerint növekvő (SPT) sorrendben történő elvégzése optimális ütemezés. Az $1|\sum_i w_i C_i$ feladatra a munkák p_i/w_i szerinti növekvő sorrendben történő elvégzése optimális ütemezés.

Tétel: (1) A $P2|C_{max}$ probléma optimális megoldása reménytelen. (2) a $Pm|C_{max}$ feladatra a listás ütemezést használó LS algoritmus $(2 - \frac{1}{m})$ -approximációt ad. (Itt a soron következő munkát mindig az elsőnek felszabaduló gépen végezzük.) (3) Ha az LS algoritmust p_i szerint csökkenő (LPT) sorrendben végezzük, akkor $\frac{4}{3}$ -közelítést kapunk.

Ládapakolási (bin packing) feladat

Minél kevesebb egységnyi térfogatú ládába kell bepakolni az a_1, a_2, \dots, a_n méretű tárgyakat.

FF (first fit) algoritmus minden tárgyat az első olyan ládába teszünk, amibe elfér.

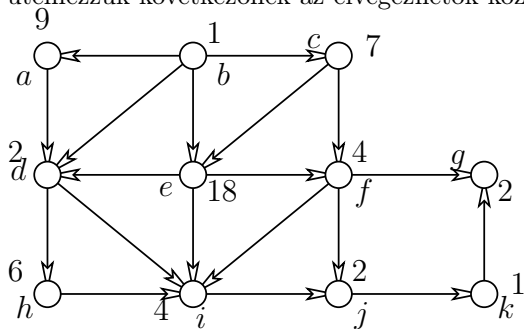
FFD (first fit decreasing) algoritmus Az FF algoritmust a méretek csökkenő sorrendjében futtatjuk.

Tétel: Az FF algoritmus legfeljebb $\frac{17}{10}OPT + 2$, az FFD legfeljebb $\frac{11}{9}OPT + 7$ ládát használ.

Gyakorlatok

1. Keressünk hatékony közelítő algoritmust a halmazfedési probléma alábbi általánosítására. Adott egy n elemű U alaphalmazon az S_1, S_2, \dots, S_k részhalmazok rendszere és adottak a $c(S_i) \geq 0$ költségek. A cél, hogy úgy válasszunk ki néhány részhalmazt a fentiek közül, hogy U minden elemét legalább két kiválasztott részhalmaz tartalmazza és a kiválasztott részhalmazok összköltsége a lehető legkevesebb legyen. Milyen multiplikatív hibával tudjuk közelíteni az optimális megoldást?
2. Keressünk hatékony közelítő algoritmust a halmazfedési probléma alábbi általánosítására. Adott egy n elemű U alaphalmazon az S_1, S_2, \dots, S_k részhalmazok rendszere és adottak a $c(S_i) \geq 0$ költségek. A cél, hogy úgy válasszunk ki néhány részhalmazt a fentiek közül, hogy a kiválasztott részhalmazok U -nak legalább k elemét tartalmazzák és a kiválasztott részhalmazok összköltsége a lehető legkevesebb legyen. Mennyire tudjuk leSORÍRTANI a közelítés multiplikatív hibáját?

- Tegyük fel, hogy G egy 2-élösszefüggő gráfnak keressük egy lehető legkevesebb élű 2-élösszefüggő feszítő részgráfját. Igazoljuk, hogy az alábbi algoritmus ennek a feladatnak egy 2-közelítését adja. Válasszuk ki G egy F feszítőfáját, majd a $G - F$ gráfnak egy F' feszítő erdejét (azaz $G - F$ minden komponensének egy feszítőfáját). Az output az $F \cup F'$ élhalmaz.
- Bizonyítsuk be, hogy minden $Pm||C_{\max}$ típusú ütemezési feladat esetén van a munkáknak olyan sorrendje, hogy amire listás ütemezés (azaz az LS algoritmus) az adott inputhoz tartozó minimális átfutási idővel ütemez.
- Tegyük fel, hogy a J_1, J_2, \dots munkákat az S ütemezés úgy ütemezni 42 gépre, hogy az átfutási idő 42, az átlagos átfutási idő pedig 24 legyen. Mutassuk meg, hogy ugyanezek a munkák ütemezhetőek 21 gépre úgy, hogy az átfutási idő legfeljebb 84, az átlagos átfutási idő pedig legfeljebb 45 legyen.
- Tegyük fel, hogy 7 munkaösszvégrehajtási ideje 42. Igazoljuk, hogy lehetséges ezeket a munkákat egy gépre ütemezni úgy, hogy az átlagos átfutási idő legfeljebb 24 legyen.
- Egy gépen 10 munkát 100 időegység alatt végeztünk el úgy, hogy az átlagos átfutási idő épp 68 volt. Mennyi idő alatt végeztük volna el a munkákat fordított sorrendben, és mennyi lett volna így az átlagos átfutási idő?
- Az ábrán látható gráf csúcsai munkákat jelentenek, és minden csúcsra az adott munka megmunkálási ideje van ráírva. A gráf egy uv élének jelentése az, hogy a v munkát csak az u munka befejezése után lehet elkezdeni. Keressünk minimális átfutási idejű ütemezést arra az esetre ugyanazon a gépen kell minden munkát elvégezni. Mennyi a minimális átfutási idő, ha tetszőlegesen sok gépet használhatunk? Határozzuk meg az átfutási időt 2 gépre történő listás ütemezés esetén, ha mindig azt a munkát ütemezzük következőnek az elvégezhetőek közül, amelyik neve az ABC-ben a legelől áll.



- Tegyük fel, hogy a ládapakolási feladat egy konkrét inputjához az FF algoritmus 42 ládát használ fel. Bizonyítsuk be, hogy ha ugyanehhez az inputhoz két és félszer akkora ládák állnak rendelkezésre, akkor a konkrét feladatot az FF algoritmus meg tudja oldani legfeljebb 21 ládával. Bizonyítsuk be, hogy 21 ládánál kevesebbet az FF algoritmus nem tud garantálni.