Scheduler

Martin Makuch

18. júna 2024

Úvod

V tejto práci predstavujem výsledky svojho ročníkového projektu s názvom "Scheduler,"ktorý bol vyvíjaný pre firmu MicroStep spol. s r. o. Cieľom projektu je navrhnúť a implementovať algoritmus na efektívne plánovanie rozvrhov práce v prostredí typu 'Job Shop'. Tento typ problému je charakterizovaný množinou prác a strojov, kde každá práca pozostáva z postupnosti operácií, ktoré musia byť vykonané na konkrétnych strojoch v stanovenom poradí. Projekt sa zameriava na optimalizáciu časového rozvrhu tak, aby boli všetky práce vykonané v čo najkratšom čase.

Cieľom projektu je nasadenie v reálnom prostredí výroby v Hríňovej a Partizánskom, kde sa projekt bude testovať. To umožní overiť efektívnosť navrhnutého riešenia v praxi a identifikovať prípadné nedostatky či oblasti na zlepšenie.

Klasifikácia problému

Nech m, n sú ľubovoľné nezáporné premenné. Potom množina $M = \{M_1, M_2, \ldots, M_m\}$ označuje stroje. Množina $J = \{J_1, J_2, \ldots, J_n\}$ označuje práce. Pre každú prácu J_j z množiny J definujeme čísla: m_j označujúce počet operácii; $p_{j1}, p_{j2}, \ldots, p_{jm_j}$ označujúce doby spracovania; d_j označujúce čas ukončenia práce; w_j označujúce relatívnu dôležitosť.

Nastavenie strojov

Problém, ktorý riešime je v literatúre označovaný ako 'Job Shop'. Každá práca J_j z množiny J pozostáva z usporiadanej m_j-tice operácii $(O_{j1}, O_{j2}, \ldots, O_{jm_j})$, kde číslo m_j reprezentuje počet operácii, z ktorých práca J_j pozostáva, a kde sa operácia O_{ji} musí v poradí vykonať na stroji μ_{ji} za čas p_{ji}. Pritom pre $i = 1, 2, \ldots, m_j$ platí, že $\mu_{i-1} \neq \mu_i$.

Charakteristika prác

V našom prostredí zakazujeme prerušovanie vykonávania jednotlivých operácii. Žiadne ďalšie obmedzenia nateraz nekladieme.

Cieľ

Cieľom je naplánovať rozvrh pre optimálne vykonanie prác (za optimálne riešenie budeme považovať také, ktoré v najkratšom čase vykoná všetky práce).

Brute-force algoritmus

Keďže každá práce pozostáva z nejakej postupnosti operácii, kde každá z nich musí byť vykonaná na práve jednom stroji - môžeme usporiadať práce podľa toho na akom stroji sa majú vykonať.

Zjavne v každom rozvrhu platí, že ak sa ľubovoľné dve operácie musia vykonať na jednom stroji, tak niektorá z nich sa musí vykonať skôr. Výmenou poradia ľubovoľných dvoch operácii teda dostaneme nový rozvrh. (Výmeny poradia jednotlivých strojov uvažovať nebudeme vzhľadom k tomu, že stroje pracujú nezávisle.)

Vo všeobecnosti by sme počet všetkých rôznych rozvrhov (teda takých, ktoré sa líšia poradím nejakých dvoch operácii prislúchajúcich jednému stroju) mohli zapísať následovne: nech $[M_i]$ označuje triedu ekvivalencie na množine všetkých operácii $O = O_1 \cup \ldots \cup O_n$; potom $\prod_{i=1}^{m} |[M_i]|!$ vyjadruje hľadaný počet.

Výsledný rozvrh aj s časmi vykonania jednotlivých operácii zostrojíme následovne:

1. rozdelíme operácie do radov podľa strojov, na ktorých sa majú vykonať, v poradí, v ktorých sa majú vykonať;
2. opakujeme v cykle:
   - 2.1. počínajúc ľubovoľným strojom iterujeme všetky stroje; môžu nastať dva prípady:
     - 2.1.1. operácia na začiatku rady sa môže vykonať t.j. všetky predchádzajúce operácie sa už vykonali:
       - 2.1.1.1 vyberieme opráciu z rady a naplánujeme ju,
       - 2.1.1.2 pokračujeme na ďalšou opráciou v rade;
     - 2.1.2. operácia na začaitku rady sa ešte vykonať nemôže t.j. niektorá z predchádzajúcich operácii sa ešte nevykonala,
       - 2.1.2.1 prejdeme na ďalší stroj;
   - 2.2. ak sa nenaplánovala žiadna operácia, ukončíme cyklus;
3. po skončení cyklu môžu nastať dva prípady:
   - 3.1 všetky operácie sú naplánované, vrátime maximum z časov ukončenia všetkých operácii;

3.2 nejaká operácia nie je naplánovaná; vrátime nekonečno;

Algoritmu postupne prechádza cez rady operácii prislúchajúce jednotlivým strojom a operáciu vyberie (naplánujeme) ak operácia, ktorá jej má predchádzať je už naplánovaná.

(Stačí si pri tom pamätať iba poslednú naplánovanú operáciu pre každú prácu.) Počiatočný čas vykonania operácie sa vypočíta ako maximum z počiatočného času vykonania poslednej operácie zväčšeného o dĺžku vykonania tej operácie a počiatočného času poslednej operácie vykonanej na danom stroji zväčšeného o dĺžku vykonania tej operácie. (Pre druhé uvedené nám stačí udržiavať pre každý stroj čas ukončenia poslednej operácie na ňom.)

Nedostatky

S uvažovaným modelom sa síce pracuje príjmne, pre jeho jednoduchosť, je však v praxi len sotva využiteľný.

V ďalších sekciách si predstavíme iné, všeobecnejšie, spôsoby pohľadu na operácie. Potešujúce je, že mnohé myšlienky, použité pre tento model, budeme vedieť aplikovať aj na všeobecenjšie verzie problému.

Zovšeobecnenie

Máme výrobnú linku obsahujúcu štyri pracoviská: rezanie, vŕtanie, ohýbanie a skladanie. Proces výroby prebieha nasledovne:

1. Na **prvom pracovisku** (rezanie) vyrežeme z kusu typu **A** dva kusy typu **B**.
2. **Druhé pracovisko** (vŕtanie) spracuje jeden z kusov typu **B**, čím vznikne kus typu **B'**.
3. **Tretie pracovisko** (ohýbanie) spracuje druhý kus typu **B**, čím vznikne kus typu **B"**.
4. Na **štvrtom pracovisku** (skladanie) sa z kusov **B'** a **B"** zostaví jeden kus typu **C**.

Malo by byť vidieť, že takýto problém nevieme v základnom modeli vyjadriť. konkrétne nám robí problém vyjdariť fakt, že niektoré joby, môžu medzi sebou zdieľať operácie.

Úpravený brute-force algoritmus

Pri priamočiarej úprave reprezentácie následností operácii ako orientovaného acyklického grafu sa pôvodný algoritmus zmení iba v bode, kedy zisťuje, či je možné naplánovať ďalšiu opráciu. O upravenom algoritmne potom vieme argumentovať o jeho správnosti analogicky.

Pozorovanie

Upravený algoritmus, ako sme aj ukázali, funguje naďalej. Otázka je ako by sme vedeli znížiť počet všetkých vyskúšaných rozvrhov?

(...)

Operácie ako vstupno-výstupné relácie

Ďalšie zovšeobecnenie, ktoré vieme spraviť je zmena pohľadu na operácie ako na čisto matematický objekt. Toto zovšeobecenenie sa môže ukázať ako kľúčové pri nasadzovaní techník lineárneho programovania.

(...)