1. Paralelní návrh pro řešení Resource-Constrained Project Scheduling Problem

Martin Lukeš

Kombinatorická optimalizace

2013

* 1. Popis:

Resource Constraint Project Sheduling problém (RCPSP) je instanciací obecného Resource Constraint Scheduling Problem(RCSP), který je podtřídou problémů scheduling problémů, jež řadíme do kategorie NP složitých problémů. Kromě RCPSP jsou v této kategorii problémy job-shop flow-shop a production scheduling.

RCSP je v obecné formě definován takto: Mějme množinu aktivit, množinu úloh, množinu úloh, množinu zdrojů a míru vykonávání rozvrhu. Jakým způsobem nejlépe přiřadit aktivity zdrojům (nebo zdroje aktivitám) v dané časy, aby byl rozvrh co nejlepší?

RCPSP potom zavádí další omezení, kterým je nepreemptivnost jednotlivých aktivit a následnost jednotlivých aktivit. Nepreemptivnost vyjadřuje, že po započetí dané aktivity tato aktivita nemůže být pozastavena před jejím dokončením. Následnost aktivit vyjadřuje, že daná aktivita nemůže být v rozvrhu dříve než jsou dokončeny všechny její předchůdci. Následnost aktivit je nejčastěji zobrazována pomocí acyklického grafu aktivit, kde uzly symbolizují jednotlivé aktivity a hrany následnost. Existují dvě varianty RCPSP : multi – mode a single – RCPSP. Single mode varianta předpokládá v jediný způsob vyřešení aktivity, kdežto multi-mode metoda umožňuje vykonávání aktivity ve více módech oddělených například podle délky trvání aktivity a jí přiřazených zdrojů.

* 1. :

Cílem této práce je nalezení paralelního single-mode algoritmu pro řešení tohoto problému. Problém RCPSP byl řešen například pomocí enumerativní metody, Branch and bound algoritmu, metody CPM(Critical Path method), genetických algoritmů a Ant Colony Optisation přístupu.

**Model**

Před řešením úlohy je nutné definovat si struktury/model instance tasků, zachytit precedenční vztahy mezi tasky, množství zdrojů Kj potřebných k provedení tasku *i* pro každý task *j*, délku trvání tasku a množství obnovitelných zdrojů jednotlivých typů, které jsou dostupném v každém časovém okamžiku.

Nejprve reprezentujeme statické prvky úlohy. Precedenční závislost tasků se dá namodelovat jak již bylo zmíněno acyklickým grafem. V implementaci budeme tento graf reprezentovat pomocí pole seznamů následníků, protože matice sousednosti takového grafu může být maximálně horní trojúhelníková a typicky je tento graf řidší. Pro grafy s horní trojúhelníkovou maticí sousednosti existuje pouze jedno řešení – viz sekce inicializace, pro hustější precedenční grafy se počet řešení limitně blíží jedné.

Množství obnovitelných zdrojů v daném čase se dá reprezentovat pomocí celočíselného pole hodnot o délce *n*.

Délku trvání jednotlivých tasků P – tzv. processing time můžeme taktéž reprezentovat pomocí pole celočíselných hodnot o délce *m.*

Závislosti tasků na zdrojích se typicky reprezentují maticí K o rozměrech *n x m* celočíselných hodnot, kdy závislost tasku *k* na zdroji *l* najdeme v K na pozici [l, k], přičemž *n* je počet zdrojů a *m* počet tasků. V literatuře jsou typicky spojeny matice K s polem délek trvání tasků P.

Jsou dvě možné reprezentace instance dané úlohy. První z nich reprezentuje úlohy pomocí uspořádané k-tice celočíselných hodnot, reprezentující počáteční časy jednotlivých tasků, druhý způsob je reprezentace také uspořádanou k-ticí reprezentující pořadí jednotlivých tasků v rozvrhu. Tato k-tice na rozdíl od první je jen permutací hodnot *1*... *n* *,* kdy n je počet tasků.Vybral jsem si druhý způsob, protože se mi s ním lépe pracuje – viz sekce genetický algoritmus.

**Paralení algoritmy**

V rámci této práce jsem prostudoval materiály profesora Tvrdíka k předmětu MI-PAR - Paralelní algoritmy a systémya materiály k předmětu z FJFI [Paralelní algoritmy a architektury](http://geraldine.fjfi.cvut.cz/~oberhuber/node/paa-cs)**.** Profesor Tvrdík uvádí měřítka výkonnosti sekvenčních algoritmů – asymptotickou dobu výpočtu algoritmu - *T(n)*, paměťovou náročnost *Sp(n)* při výpočtu algoritmu a porovnává je s měřítky výkonnosti paralelních algoritmů.

Sekvenční algoritmus:

T(n)

Sp(n)

Paralelní algoritmy mají kromě asymptotické doby výpočtu a paměťové náročnosti i počet procesorů, které vykonávají danou úlohu. Délka trvání výpočtu je potom závislá nejen na velikosti vstupních dat, ale i na počtu procesorů *p*, které úlohu počítají.

T(n, p)

Formální definice prof Tvrdíka: T(n, p) je čas, který uplynul od začátku paralelního výpočtu do okamžiku, kdy poslední (nejpomalejší) procesor skončil výpočet.

T(n, p) je měřen čítáním:

1. výpočetních kroků: aritmetické, logické, paměťové operace

+

2. komunikačních kroků: přenos a výměna dat mezi procesory

Pokud si definujeme horní mez výpočetní složitosti *SU(n)*, pak můžeme definovat paralelní zrychlení *S(n, p)*

S(n, p) =SU(n)/T(n, p) <= p

Pokud je toto zrychlení asymptoticky těsné vůči p, potom

S(n, p) = Q(p)