Simulated annealing algorithm for the resource-constrained project scheduling problem

xbilja00

April 29, 2022

1 Úvod

Projekt se zabývá naimplementování algoritmu vyhledávací metody simulated annealing pro resource-constrained project scheduling problem, který byl představen v [BL03]. Nejdříve je popsána řešený problém, poté metoda simulated annealing a algoritmus navržený v [BL03]. Nakonec jsou popsány experimenty s implementovanou metodou a jejich výsledky.

2 Resource-constrained project scheduling problem

Resource-constrained project scheduling problem (RCPSP) je kombinatorický optimalizační problém, kde cílem je najít nejkratší možný plán aktivit (schedule), kdy známe dobu trvání každé aktivity a její požadavky na zdroje. Plánování aktivit nesmí porušit 2 podmínky: 1) aktivita, která má být naplánována, nesmí vyžadovat více zdrojů, než kolik je právě dostupných a 2) aktivita musí být naplánována po skončení všech jejích předchůdců.
Formálněji, máme množinu $A = \{0, 1, ..., J, J + 1\}$ aktivit. Aktivity 0 a $J + 1$ jsou umělé aktivity znamenající začátek a konec projektu. První podmínka zajišťuje, že aktivita j nesmí začít dříve, než jsou všichni přímí předchůdci z $Pred(j)$ dokončeny. Dále máme množinu $R = \{1...R\}$ obnovitelných zdrojů, což je limit pro dostupné zdroje $Q(r)$ v určitém čase. Druhá podmínka určuje, že aktivita j může být naplánována, pokud všechny její požadavky na zdroje $q_{jr} \leq Qr$ a po dobu vykonávání jsou tyto zdroje odebrány z $Q(r)$.
Naplánování znamená přiřazení začátků časů všem aktivitám. Čas naplánování poslední aktivity je celkový čas projektu (makespan) a cílem je aby byl co nejnižší.

3 Simulated annealing

Simulated annealing (SA) je prohledávací metoda, která bere inspiraci v procesu žíhání. Metoda iterativně nachází sousední řešení a pokud má lepší objective function value (pro RCPSP nižší makespan), vychází v dalším kroku od tohoto řešení. Aby se metoda dostala z lokálních minim, s jistou pravděpodobností přijme i horší řešení.
Přesněji, v každé iteraci se pro momentální řešení x, spočítá objective function value $f(x)$. Vybere se sousední řešení x' a spočítá se $\Delta = f(x') - f(x)$. Pokud je $\Delta \leq 0$, přijme se x' jako x do další iterace. Jinak může být x' také přijmuto s pravděpodobností $P = e^{(-\Delta)/T}$. Tato pravděpodobnost je porovnána s náhodnou hodnotou $y_{random} \in [0, 1]$ a pokud je $P > y_{random}$ je x' přijmuto.
Paramer T se nazývá teplota. Čím vyšší je teplota, tím vyšší je pravděpodobnost přijetí horšího řešení. Teplota je postupem k lepšímu řešení snižována pomocí tzv. cooling scheme.

3.1 Reprezentace řešení

Reprezentace řešení je důležitá, aby bylo možné lehce vygenerovat sousední řešení. V tomto projektu je jako reprezentace řešení vybrán seřazený seznam aktivit, kdy každá aktivita v seznamu je za svými předchůdci.
Jako plánovací procedura je použita serial schedule generation scheme (SGS). SGS spočívá ve 2 operacích: plánování nové události a posunutí času. Po naplánování aukončení první umělé aktivity je

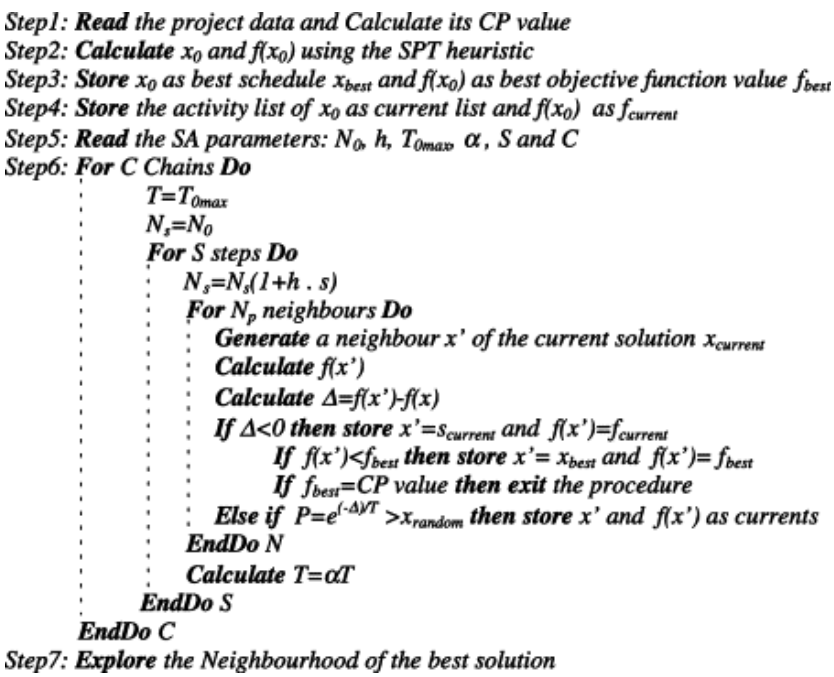

Figure 1: Proposed SA for RCPSP algorithm

testována další aktivita v pořadí. Pokud splňuje podmínky pro naplánování, je naplánována, pokud ne, provede se operace posunutí času, kdy se čas posune na nejbližší koncový čas aktivity. Toto se děje, dokud aktivita nebude splňovat podmínky a nebude naplánována. Takto se musí naplánovat všechny aktivity.

3.2 Generování sousedních řešení

Generování sousedního řešení začíná zvolením náhodné aktivity x z momentálního řešení. Pro tuto aktivitu se najdou indexy nejposlednějšího předchůdce a nejdřívějšího následníka. Tyto indexy slouží jako jakési hranice, aby se neporušily podmínky. x se potom přesune na náhodný index mezi těmito hranicemi a všechny aktivity mezi původním a novým indexem provedou přesunutí o 1 index ve směru původního indexu (operaci shift).

3.3 Cooling scheme

V projektu je použit tzv.multiple cooling chain (C), který je restartován s jiným počátečním řešením. Počet řešení v každém kroku s je zvyšován: $N_S = N_S(1 + h * s)$. Tímto je docíleno širší prohledávání lepších řešení.

4 Navržený algoritmus

Na obrázku 1 je zobrazený navržený algoritmus. Počáteční řešení je vygenerováno s heuristikou shortest process time (SPT), která z dostupných aktivit vždy naplánuje tu, která má nejkratší čas k dokončení. Funkce f(x) vypočítává makespan řešení, což vyžaduje simulaci celého řešení.

5 Experimenty

Program byl implementován v jazyce C++ a pokusy byly prováděny na jiném počítači, než v článku. Jako testovací sada byly zvoleny sady J30 a J60 dostupné na této adrese. Tyto datasety byly vygenerovány pomocí ProGen a jsou použity i v [BL03]. Datasety obsahují 480 projektů a ke každému

T_{0max}	Calculated projects	Number of optimal solutions	Cumulated variance	Avarage variance
10	30	23	15	0.5
25	30	23	16	0.53
25	100	65	165	1.65
35	30	23	23	0.76
35	100	67	167	1.67
50	30	22	20	0.67
50	100	62	174	1.74
20% x_0	30	23	20	0.67

Table 1: Table showing the effect of T_{0max} parameter on solutions.

Dataset	C	h	Searched projects	Optimal solutions found	Cumulated variance	Avarage variance
J30	1	1	30	24	10	0.80
J30	1	1	100	72	73	0.73
J30	1	2	30	28	2	0.07
J30	1	2	100	77	38	0.38
J30	2	1	30	21	12	0.40
J30	2	1	100	69	60	0.60
J30	3	1	30	25	5	0.17
J30	3	1	100	77	38	0.38
J60	1	1	30	20	30	1.0
J60	1	2	30	25	6	0.20
J60	1	2	100	70	116	1.16
J60	3	1	30	23	15	0.77
J60	3	1	100	67	167	1.67

Table 2: Table showing the effect of C and h parameters on solutions.

je dostupné i optimální řešení nebo řešení heuristikou. Pro J60 není dostupné optimální řešení, jako optimální jsem použil j60hrs.sm (heuristic solution). J30 jsou projekty s 30 aktivitami a J60 projekty s 60 aktivitami.
V experimentech jsem se zaměřil na zkoumání vlivu SA parametrů na optimálnost řešení. Nejdříve jsem spustil sadu experimentů pro nalezení nejvhodnější hodnoty počáteční teploty T_{0max}. Výsledky jsou ukázány v tabulce 1. tyto experimenty byly prováděny na části projektů z J60 s parametry C=3, S=5, h=1, $\alpha = 0.25$. V [BL03] se uvádí jako optimální 20% z SPT řešení. Z výsledků tato hodnota dopadla nejlépe spolu s podobnými fixními hodnotami (průměrně je čas SPT řešení okolo 15). Vyšší hodnoty již způsobily horší řešení. Pro další experimenty jsem tedy volil $T_{0max} = 20\%x_0$.
Pro obě omezené sady jsem spustil sadu experimentů s různými kombinacemi parametrů. Konkrétně se měnil počet cooling chains C a parametr h, který ovlivňuje počet prozkoumávaných řetězců v jednotlivých kolech s. Výsledky jsou vidět v tabulce 2. Z měření usuzuji, že parametr h má velký vliv na optimálnost řešení a měl by být zvolen alespoň na hodnotu 2. Je patrné, že je důležité prohledávat větší množství sousedů dobrých řešení. Více než 1 cooling chain pro možnost uniknout lokálnímu maximu má však také vliv.
Experimenty však nejsou prováděny v dostatečném množství, aby bylo možné perfektně porovnat výsledky s výsledky v článku.

6 Závěr

V této práci se povedlo naimplementovat algoritmus vyhledáavací metody simulated annealing pro resource-constrained project scheduling problem a ověřit správnost a vliv volby některých parametrů. V mé práci jsem nebyl schopen zopakovat stejné výsledky jako autoři práce (například jsem nedokázal nalézt všechna optimální řešení pro dataset J30). Toto bude způsobeno nejspíše nedokonalým laděním parametrů a méně celkem prohledaných řešení.

References

[BL03] K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003. Sequencing and Scheduling.