Robusztus ütemtervek készítése lineáris programozási eszközökkel

Hanák Dávid[1], Nagarajan Kandasamy[2] et al.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi és Információelméleti Tanszék
[1]dhanak@cs.bme.hu

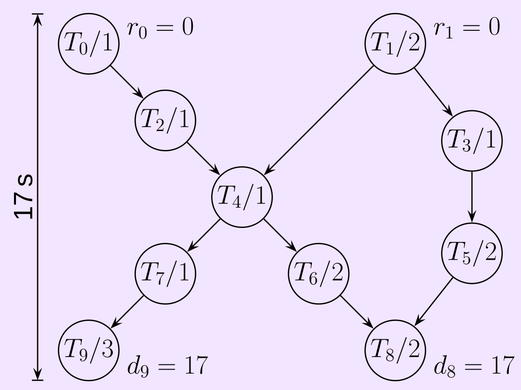
Institute for Software Integrated Systems
Vanderbilt University, Nashville TN, U.S.A.
[2]kandasamy@cbis.ece.drexel.edu

Az ismert ütemezési algoritmusok az egyes feladatok erőforrásigényének, precedenciájának és sorrendezési követelményeinek figyelembe vételével, a feladatsor végrehajtását megelőzően készítik el az időzítési elvárásoknak megfelelő ütemtervet. Dinamikusan változó, kiszámíthatatlan környezetben azonban a végrehajtás könnyen kudarcba fulladhat, ha egy-egy erőforrás kiesik, vagy az egyes feladatok végrehajtása több időt vesz igénybe az előre tervezettnél. A cél olyan robusztus, rugalmas ütemterv készítése, amely teljes újraütemezés nélkül tud alkalmazkodni az ilyen váratlan eseményekhez.
Az itt bemutatott módszer holtidő alapú megközelítést használ: az egyes feladatokhoz több időt rendel, mint amennyit a várakozások alapján igényelnek. [1] és [3] módszerei fix holtidőt adnak mindegyik feladathoz, ez pedig megnöveli a teljes ütemterv-végrehajtási időt. A mi módszerünk ezzel szemben feltételezi, hogy a teljes tervnek adott határidő- és erőforrásmegkötései vannak, és ezek figyelembe vételével igyekszik maximalizálni az egyes feladatokhoz adott holtidőt.

Az algoritmus lépései

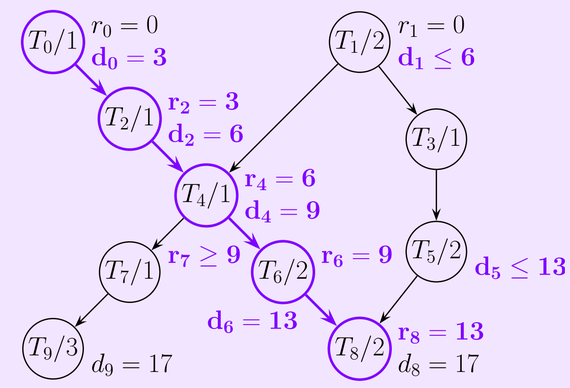
1. Globális határidő elosztása az egyes feladatok között:
   - minden feladathoz megadja azon időtartományokat, amelyekben a feladat elvégezhető;
   - kielégíti a sorrendezési követelményeket;
   - figyelmen kívül hagyja az erőforrásigényeket;
   - NP-teljes probléma – közelítő algoritmus: [2].

1. Határidőelosztás

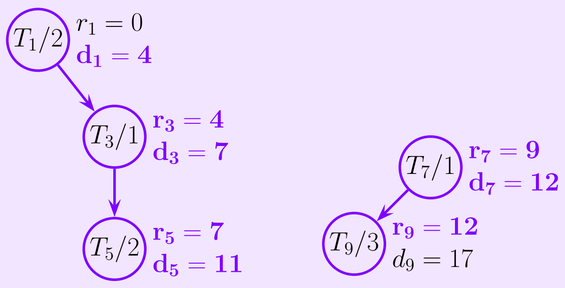


1. ábra: *Példa gráf végponti határidőkkel*

- Kezdetben csak a nyitó- és zárófeladatoknak van kezdő- és határideje.
- Útvonali holtidő: $slack_q = D_q - \sum_{i:T_i \in path_q} c_i$ (D_q : útvonal határideje, c_i : feladat futási ideje).
- Ciklus, amíg van útvonal:
  1. útvonal kiválasztása, ahol $slack_q/n$ minimális;
  2. $slack_q$ egyenletes szétosztása az útvonalon;
  3. csatlakozó kezdő- és határidők beállítása;
  4. útvonal elhagyása a gráfból.



(a) első lépés



(b) második lépés **(c)** harmadik lépés

2. ábra: *A határidőelosztási algoritmus lépései*

A kapott $[r_i, d_i)$ kezdő- és határidőkből generálható egy sor $I_{ij} = [r_{ij}, d_{ij})$ intervallum, ahol $r_i \le r_{ij} \le d_i - c_i$ és $r_i + c_i \le d_{ij} \le d_i$. Egy intervallum kiválasztási súlya $(d_{ij} - r_{ij} - c_i)/(d_{ij} - r_{ij})$.
Ezek után ütemezés ≡ intervallumkiválasztás.

2. Egyetlen időtartomány kiválasztása minden feladathoz:
   - garantálja, hogy az erőforrásigények mindig a kapacitásokon belül maradnak;
   - maximálja a holtidők súlyozott átlagát;
   - két egészértékű programozási modell.

2/a. Időszelet alapú modell

- A teljes időtartam egyforma időszeletekből áll.
- A kiválasztandó intervallum egy folytonos időszelet-csoport.
- Egy ütemtervre teljesülnie kell (ld. 3. ábra):

(1) nincs erőforráskapacitás-túllépés;
(2) a kiválasztott intervallum folytonos;
(3) a feladat nem fut az ablakán kívül;
(4) intervallum hossza $\ge$ feladat futási ideje;
(5,6) változók megválasztásából adódó konzisztenciafeltételek.

Bővebben ld. [4].

Indexhalmazok és paraméterek: (iv. = intervallum)

$L := \{i \mid i$ a T_i feladat indexe$\}$
$K_i := \{c_i, c_i + 1, \ldots, (d_i - r_i)\}$ (T_i feladat iv.-hosszai)
$w_{ik} := \frac{k - c_i}{k}, i \in L, k \in K_i$ (egy k hosszú iv. súlya)

Változók:

$x_{ij} = 1$, ha T_i akív iv.-a lefedi a j. időszeletet; különben 0
$y_{ik} = 1$, ha T_i aktív iv.-a k hosszú; különben 0

Maximalizálás: $\sum_{i \in L} \sum_{k \in K_i} y_{ik} w_{ik}$ az alábbi megkötésekkel:

Erőforráskapacitások:

$$\forall R_m, \forall j \in \{0, \ldots, D\} : \sum_{i \in \{i \mid T_i \text{ használja } R_m\text{-et}\}} x_{ij} \le \mathrm{cap}(R_m, j) \quad (1)$$

Intervallumfolytonosság:

$$\forall i \in L, \forall j \in \{r_i - 1, \ldots, d_i - 4\}, \forall l \in \{j + 2, \ldots, d_i - 2\} : x_{ij+1} - x_{ij} + x_{il+1} - x_{il} < 2 \quad (2)$$

Intervallumhossz:

$$\forall i \in L, \forall j \in \{0, \ldots, r_i - 1, d_i, \ldots, D\} : x_{ij} = 0 \quad (3)$$
$$\forall i \in L : \sum_{j \in \{r_i, \ldots, d_i\}} x_{ij} \ge c_i \quad (4)$$

Változók konzisztenciája:

$$\forall i \in L : \sum_{j \in \{r_i, \ldots, d_i\}} x_{ij} - \sum_{k \in K_i} k y_{ik} = 0 \quad (5)$$
$$\forall i \in L : \sum_{k \in K_i} y_{ik} = 1 \quad (6)$$

3. ábra: *Időszelet alapú ILP modell*

Az értékelésről

A modelleket két tiszta ILP (LP_SOLVE és GLPK) és egy SAT/ILP vegyes megoldóval (PBS) tesztelték különböző méretű gráfokon (25–150 feladat, 1000–8000 kiválasztható intervallum). Az LP_SOLVE adja a legjobb megoldásokat de a GLPK sokkal gyorsabb és skálázhatóbb.

Alkalmazási terület

A feladat katonai repülőgépek karbantartásának ütemezése [5]. A cél egy működő, de megbízhatatlan (sokszor kivárhatatlan) Mozart/Oz alapú megoldás kiváltása. A szervizelést bevetések között kell ütemezni, bizonyos feladatok (szétszedés, javítás, összerakás) egyetlen munkamenetben végzendők. A karbantartók munkaideje, feladatköre többféle.

2/b. Intervallum alapú modell

- Időszeletváltozók helyett intervallumváltozók.
- Újdonság: munka – összefüggő feladatcsoport.
- Egy munkát teljes egészében a megadott munkarésekek egyikében kell elvégezni.
- Egy ütemtervre teljesülnie kell (ld. 4. ábra):

(7) nincs erőforráskapacitás-túllépés;
(8) egy munka minden feladata ugyanabban a résben fut le;
(9) minden munkához pontosan egy rés aktív.

Indexhalmazok és paraméterek:

L : feladatok i indexei M : munkák k indexei
K_i : T_i feladat j iv.-indexei N_k : J_k munka l sávindexei
$I_{ij} := [r_{ij}, d_{ij})$ (T_i feladat j. lehetséges ütemezési iv.-a)
$H_{kl} := [s_{kl}, e_{kl})$ (J_k munka l. lehetséges munkasávja)
$w_{ij} := \frac{d_{ij} - r_{ij} - c_i}{d_{ij} - r_{ij}}$ az (I_{ij} intervallum súlya)

Változók:

$x_{ij} = 1$, ha az I_{ij} iv. aktív a T_i feladathoz; különben 0
$y_{kl} = 1$, ha a H_{kl} sáv aktív a J_k munkához; különben 0

Maximalizálás: $\sum_{i \in L} \sum_{j \in K_i} x_{ij} w_{ij}$ az alábbi megkötésekkel:

Erőforráskapacitások:

$$\forall R_m, \forall t \in \{0, \ldots, D\} : \sum_{i \in \{i \mid T_i \text{ használja } R_m\text{-et}\}} \sum_{j \in \{j \mid t \in I_{ij}\}} x_{ij} \le \mathrm{cap}(R_m, t) \quad (7)$$

Konzisztens és egyszeres intervallumkiválasztás:

$$\forall k \in M, \forall l \in N_k, \forall i \in \{i \mid T_i \in J_k\} : \sum_{j \in \{j \mid I_{ij} \subseteq H_{kl}\}} x_{ij} = y_{kl} \quad (8)$$
$$\forall k \in M : \sum_{l \in N_k} y_{kl} = 1 \quad (9)$$

4. ábra: *Intervallum alapú ILP modell*

Hivatkozások

[1] A. Davenport, C. Gefflot, and J. Beck. Slack-based techniques for robust schedules. In *Proc. of the Sixth European Conf. on Planning (ECP-2001)*, Toledo, Spain, Sept. 2001.

[2] M. Di Natale and J. A. Stankovic. Dynamic end-to-end guarantees in distributed real-time systems. In *Proc. of IEEE Real-Time Systems Symp.*, pages 216–227, San Juan, Puerto Rico, Dec. 1994.

[3] H. Gao. Building robust schedules using temporal protection—an empirical study of constraint-based scheduling under machine failure uncertainity. Master's thesis, Univ. of Toronto, Toronto, Canada, 1995.

[4] N. Kandasamy, D. Hanak, H. Neema, C. van Buskirk, and G. Karsai. Synthesis of robust task schedules for minimum disruption repair. In *Proc. of IEEE International Conference on Systems, Man and Cybernetics (SMC'04)*, pages 5056–5061, The Hague, The Netherlands, Oct. 2004.

[5] The MICANTS Project: Model Integrated Computing and Autonomous Negotiating Teams for Autonomous Logistics. http://www.isis.vanderbilt.edu/Projects/micants/micants.htm, 2004.

Intelligens Rendszerek Fiatal Kutatók Szimpóziuma, Budapest, 2006 november 23.