

Egylépéses gyártási feladatok költség optimális ütemezése időzített automatával

Vida Judit, gazdaságinformatikus Bsc.

Témavezető: Dr. Hegyháti Máté, tudományos főmunkatárs
Széchenyi István Egyetem

2018. március 8.

Tartalomjegyzék

1. Bevezetés	2
2. Irodalmi áttekintés	4
2.1. Gyártási feladatok ütemezése	4
2.2. Megoldó módszerek	5
2.2.1. MILP modellek	5
2.2.2. S-gráf	6
2.2.3. Petri háló és automaták	6
2.3. Az egylépéses ütemezés ábrázolása	7
3. Időzített automaták	8
3.1. Időzített automaták	8
3.2. UPPAAL Cora	9
4. Problémadefiníció	12
5. Egylépéses feladat LPTA modellje	13
5.1. Leírás	13
5.2. Template-ek és deklaráció	13
5.3. Lekérdezések	15
6. Teszteredmények, összehasonlítás	17
7. Összefoglalás és jövőbeli tervek	19
Irodalomjegyzék	20
A. Jelölések	21

1. fejezet

Bevezetés

Ütemezési problémákkal az élet számos területén találkozhatunk, például hétköznapi feladatainkat is be kell osztanunk, az iskolában el kell osztani az órákat termekbe és a számítógép is ütemezi az elvégzendő folyamatait. Gyártórendszereknél jellemzően el kell osztani az adott erőforrásokat az azokon műveletet végző egységek között, így a gyártásütemezési feladatok az ütemezési problémák jelentős hányadát teszik ki.

A gyártásütemezés az iparra jellemző, a berendezéseken megadott idő alatt hajtának végre műveleteket a termékeken. A végcél szempontjából az ütemezés célja lehet makespan minimalizálás vagy átviteli kapacitás (throughput) maximalizálás. Fontos a feladatokon elvégzett műveletek sorrendje, melyeknek a megadott módon kell egymást követniük.

Az irodalomban több lehetséges megoldó módszerről esik szó, gyártási probléma ütemezését elvégezték már többféle technika segítségével. A leggyakoribbak a MILP megoldó módszerek, melyek vegyes-egész lineáris programozáson alapulnak, de az S-gráf alapú megoldások is jellemzőek, valamint Petri-hálóval is oldottak már meg hasonló ütemezési feladatot. Dolgozatomban egy gyártási probléma ütemezését választottam, amelyet költségfüggvénnyel kiterjesztett időzített automata használatával optimalizálok, és elemzem az eredményeket. Az időzített automata egy olyan automata, ahol órák segítségével modellezhetőek és korlátozhatóak az események.

Időzített automatával még nem járták körül bővebben a problémát, de érdemes vele foglalkozni, mert más problémaosztályok vizsgálata során hatékony módszernek bizonyult.

A feladat egy irodalmi példa, amelyben a meghatározott számú feladatot a gépek egy lépésben végzik el. A feladat célja, hogy minél több feladatot elvégezzenek a gépek, és minél kevesebb legyen a kész termékek tárolási költsége.

A további megoldó módszerekkel elvégzett ütemezés eredményeiről szó esik a kapcsolódó irodalomban is, amelyeket fel tudunk használni arra, hogy az automatával elvégzett optimalizálás eredményeivel összehasonlítsuk őket.

Több modellt vizsgálunk meg, melyek eltérő korlátozási paraméterekkel rendelkeznek, így teszteljük az ütemezési folyamat lefutási idejét.

Dolgozatom második fejezetében az irodalmi háttérrel foglalkozom, a harmadik fejezetben részletezem a problémát, majd az időzített automatákról, a használt szoftverről és az LPTA alapú ütemezésről teszek említést. A negyedik fejezetben a teszteredményekről lesz szó, amelyeket korábbi megoldásokkal hasonlítok össze. A végén található az összefoglalás az ütemezés eredménye alapján, majd a dolgozat végére kerülnek a hivatkozások és a függelék.

2. fejezet

Irodalmi áttekintés

2.1. Gyártási feladatok ütemezése

Bármely gyártási ütemezési feladat rendelkezik közös vonásokkal, hasonló paraméterekkel, bármilyen módszert használunk a megoldáshoz. Közös bennük, hogy adottak a feladatok, a feladatokat elvégezni képes berendezések vagy eszközök, egy adott végrehajtási idő minden feladathoz, a cél pedig, hogy a kijelölt szempontok mellett megtaláljuk a lehető legjobb megoldást. Az egyes feladatok végrehajtási ideje az az időtartam, ami a munka elvégzéséhez szükséges, ez általában minden esetben egyedien meghatározott, tehát minden különböző feladatnak eltérő mennyiségű időre van szüksége. A feladatoknak lehet elvégzési határideje is, amit szintén figyelembe kell venni az ütemezés folyamán. Az irodalomban a feladatokat task megnevezéssel is használják, a munkát elvégző gépeket unit-ként, a termékeket pedig product-ként.

A problémákat különböző szempontok alapján lehet kategorizálni, az egyik csoportosítás alapján megkülönböztetünk sztochaikus és determinisztikus ütemezési feladatokat, ahol a sztochaikus típus azokat a problémákat jelöli, ahol a paraméterek futás közben kapnak értéket. A determinisztikus esetében az értékek előre be vannak állítva.

Egy másik csoportosítás szerint egy feladat a megadott paraméterek alapján lehet offline illetve online, ahol offline esetben a szükséges bemeneti adatok elérhetőek az ütemezés időpontjában, online esetben pedig a döntéseket meg kell hozni, mielőtt néhány paraméter értéke kiderülne. Az itt elemzett probléma az offline, determinisztikus feladatok körébe tartozik.

A különböző problémaesetek megoldhatóság szempontjából szintén kétféleképpen alakulhatnak. Ha az ütemezés nem elégít ki legalább egy korlátozást, akkor az nem megoldható (infeasible), minden egyéb esetben megoldható, tehát feasible.

A gyártási ütemezési feladatok egyik altípusa az egylépéses ütemezési prob-

léma (single stage probléma), ahol a feladatokon egy tevékenységet kell végrehajtani, hogy azt befejezetté lehessen nyilvánítani. Az egylépéses problémák mellett vannak más gyakori típusok is, például a simple multiproduct, ahol a feladatot lineárisan több lépésben kell elvégezni, a general multiproduct, ami a simple multiproduct-hoz hasonló, de ki lehet hagyni lépéseket. A multipurpose (többcélú) problémátípusban a lépéseknek nincs meghatározott sorrendje, tetszőlegesen hajthatóak végre. Megkülönböztethetjük a precedens típust, amely hasonló a többcélúhoz, de nem feltétlenül lineárisan hajtódnak végre a feladatok, és a general network típust, ahol a feladatokat az inputjaik és outputjaik alapján adják meg.

A single stage problémáknak a megoldásához néhány paraméternek adott-nak kell lennie, például a gépek számának, valamint annak, hogy ezek a gépek azonosak-e. Két gép akkor tekinthető azonosnak, ha ugyanazokat a munkákat képesek elvégezni ugyanannyi idő alatt. Szükség van továbbá a feladatok számára és típusára.

Az egylépéses ütemezési problémákat további alosztályokra lehet bontani, a gépek az alábbiakban felsorolt típusúak lehetnek.

- **1 - Single Machine:** Egy gép elérhető, amelyen minden feladatot végre lehet hajtani. A termékeknek (feladatoknak) különböző feldolgozási ideje van.
- **Pm - Identical paralell machines**(Azonos párhuzamos gépek): m számú azonos gép áll rendelkezésre, amelyeken bármelyik egylépéses feladat végrehajtható.
- **Qm - Paralell machines with different speed** (Párhuzamos gépek különböző sebességgel): Hasonló az előző pontban említett típushoz, de minden gépnek meghatározott sebessége van.
- **Rm - Unrelated machines in paralell**(Párhuzamos független gépek): Hasonló a Single stage típushoz, de a feladatok elvégzési ideje egy-egy gépen inputként meghatározott.

2.2. Megoldó módszerek

Vannak olyan módszerek, amelyekkel már a legtöbb problémaosztály szempontjából foglalkoztak, így egylépéses gyártásütemezési feladatokkal is. Az alábbi néhány eljárás a legnépszerűbb megoldók közé tartozik.

2.2.1. MILP modellek

A MILP modellek, tehát a vegyes-egész lineáris programozási modellek a legelterjedtebb megoldó módszerek közé tartoznak, és több altípusuk létezik.

Time discretization based - Időfelosztásos módszerek:

Az időfelosztásos modellek előnye, hogy széles skálán mozog a megoldható problémák típusa. A módszer alapján időpontokat és időréseket különböztetünk meg, A MILP modellek időpontos megadást használnak gyakrabban. Az időbeosztásos típusban a feladatokhoz bináris változókat rendelnek aszerint, hogy a feladat az adott időpontban elvégzésre kerül, vagy nem. Amikor a feladat abban az időpontban megvalósul, akkor 1 lesz a bináris változó értéke, ha nem, akkor 0. Így annyi bináris változóra lesz szükség, ahány időpontot meghatároztunk. Lehetőleg minél kisebb számú időpont felvételével kell megtalálni az optimális megoldást a modell bonyolultságának csökkentése érdekében.

x_{tij} , ahol t időpont, i a feladat, j a berendezés
 X értéke akkor 1, ha t időpillanatban j berendezés elvégzi i feladatot.

Precedencia alapú modellek:

Szintén bináris változókat használ az ütemezéshez, de az időfelosztással ellentétben kettőt.

Y_{ij} értéke 1, ha i feladatot j berendezés elvégzi

$X_{ij'}$ értéke 1, ha j berendezés elvégzi i és i' feladatokat úgy, hogy i -t előbb, mint i' -t.

Az x változó segítségével megállapítható i és i' feladatok egymáshoz való viszonya a gyártási sorrendben.

Két altípust különböztetnek meg az alapján, hogy az időpontokat az optimalizálás előtt meghatározzák, ezek a Fix időpontos időfelosztásos módszerek, vagy pedig csak a feladat közben kerül meghatározásra az időpontok száma. Utóbbiakat Variable time model-eknek hívják, és a lényegük, hogy minél kevesebb bináris változóra legyen szükség. A modellekben folyamatos változókat használnak, amik meghatározzák mindegyik időponthoz tartozó feladatokat.

2.2.2. S-gráf

Az első gráf alapú optimalizációra fejlesztett módszer a gyártásütemezés témakörében, amely nemcsak vizuálisan szemlélteti a folyamatot, de egyben egy matematikai modell is. Irányított gráfokból áll, amelynek a csomópontjai tevékenységek és az azokhoz vezető lépések, amelyeket az élek kötnek össze őket. Ezen kívül tartalmaznak ütemezési éleket, amelyek a meghozott ütemezési döntéseket modellezzik.

Létezik ütemezési döntések nélküli S-gráf is, ezt recept gráfnak hívják.

A nyilak, amelyek a csomópontokat kötik össze, a függőségeket reprezentálják a következő esetekben:

2.2.3. Petri háló és automaták

A Petri hálót és az automatát is gyakran alkalmazzák diszkrét esemény rendszerű modellezéshez, és hogy köteget feladatok elvégzéséhez is alkalmas legyen, ki kellett egészíteni időzítéssel ezeket a módszereket. Hatékonyak, mivel jól szem-

léltetik a modellt, könnyen szimulálható a felépítése és a vezérlés, és a megfelelő felépítés mellett elkerülhetőek a hibák. Bár több előnyük is van a korábban említett népszerű megoldó módszerekhez képest, összességében hatékonyságuk még elmarad a MILP és S-gráf alapú megoldó módszerekétől, valamint optimalizálásra nem alkalmasak, csak a feladat modellezésére.

Az időzített Petri háló alapja, hogy az átviteli jel késleltetés (delay) alapján jön létre. Többen is foglalkoztak a témával, Ghaeli foglalkozott a kötegelt folyamatok ütemezésével ilyen módon, Soares pedig megpróbálta kiterjeszteni a modellt, és valós idejű ütemezést mutatott be kötegelt rendszerekre Petri háló segítségével.

2.3. Az egylépéses ütemezés ábrázolása

Az egylépéses ütemezési problémákat általában táblázat segítségével adják meg, ahol a sorokban tüntetik fel a munkákat, az oszlopokban pedig a rendelkezésre álló berendezéseket, a táblázatbeli metszéspontjaik ábrázolják az egyes munkák megfelelő berendezéseken való elvégzésének munkaidejét. Kopanos részletesen foglalkozott egylépéses ütemezéssel.[1]

Termékek	Feldolgozási idő (óra)			Határidő
	u1	u2	u3	
P1	2,5	1,75	-	15
P2	-	0,6	2,84	20
P3	1	1,23	2	18
P4	0,57	0,98	-	16
P5	1,65	2,3	3	21
Beállítás	0,1	0,05	0,2	

2.1. ábra. Példa egylépéses ütemezési feladat megadására.

A fenti ábra példa egy ütemezési feladat táblázattal való megadására, ahol a feldolgozási idő mutatja a P-vel jelölt termékek egyes gépeken való munkaidejét, a gépek u-val jelöltek. Néhány feladatot nem tud minden gép végrehajtani, ezek nem rendelkeznek feldolgozási idővel az adott gépen.

Minden feladathoz fel van tüntetve a hozzá tartozó határidő, valamint a gépek beállási ideje, amely a bekapcsolás után szükséges időt mutatja, amíg a gép nem tudja elkezdni a munkát.

3. fejezet

Időzített automaták

3.1. Időzített automaták

Egy automata eseményekből és állapotokból áll, az időzített automata pedig kiegészül órákkal, amelyek mérik a globális időt, vagy egy konkrét automata idejét.

Az időzített automata a determinisztikus automaták csoportjába tartozik, ahol a determinisztikus automatákat az alábbi képlettel adják meg.

$$M = (K, \Sigma, \delta, s, F)$$

ahol

K az állapotok halmaza

Σ az események véges halmaza

δ : $K \times \Sigma \rightarrow K$ részleges átmeneti függvény

s a kezdőállapot

F az elfogadó állapotok

Az időzített automaták kiegészülnek órákkal, amelyek stopperóráként működnek, tehát a globálisan deklarált óra az automata teljes lefutási idejét méri.

Az időzített automata képlete:

$$(K, \Sigma, C, Tra, Inv, s)$$

ahol

K az állapotok halmaza

Σ az események halmaza

C az órák halmaza

Tra $K \times \phi(C) \times \Sigma \times C \times K$ időzített transitions

Inv $K \rightarrow \phi(C)$ state invariants -

s a kezdőállapot

Az időzítést egy egyszerű szó levezetésével tudjuk bemutatni, ahol az események meghatározott időben történnek.

Példának vesszük az (a,1) (b,3) (a,4) (b,6) (b,10) időzített szót.

A fenti felírás azt mutatja, hogy melyik időpillanatban történik az esemény, amiből kiszámíthatjuk, hogy a betűk mennyi késleltetéssel követik egymást.

$$d1 \rightarrow a \rightarrow d2 \rightarrow b \rightarrow d1 \rightarrow a \rightarrow d2 \rightarrow b \rightarrow b4 \rightarrow b$$

ahol d a késleltetést (delay-t) mutatja.

Az első a 1 delay eltelte után kezdődhet el, és mivel b 3 delay után következik, a -t követően 2 delay-t kell várnia, a következő betűk pedig ennek alapján ugyanezt a szabályt követik.

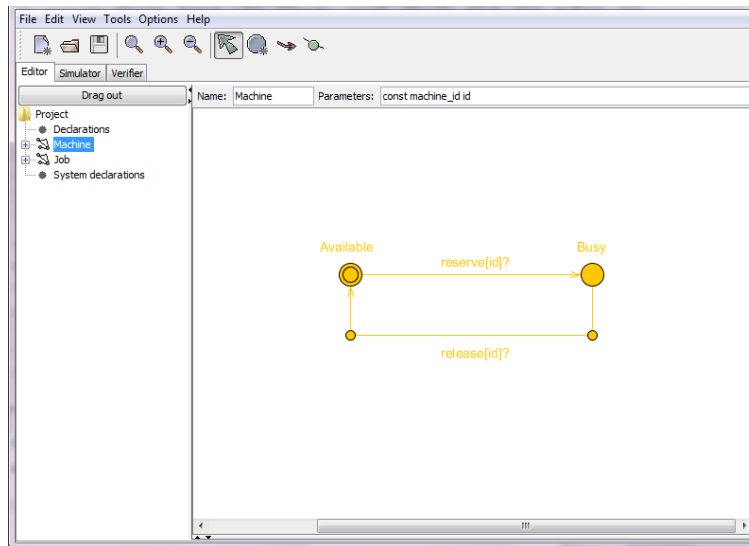
3.2. UPPAAL Cora

A választott irodalmi példa időzített automatákkal való ütemezését az UPPAAL Cora nevű szoftver segítségével modelleztük. A szoftver alkalmas az automata modellezésére, ütemezésére és optimalizálására meghatározott paraméterek alapján.

Az UPPAAL segítségével sablonokat (template-eket) hozhatunk létre, ahol minden egyes template egy különböző automata modellezésére szolgál. A sablon állapotokat és éleket tartalmaz, ahol az élek az állapotátmenetet szimbolizálják. Az állapotokhoz megadhatunk nevet, illetve valamilyen korlátozást, ezen kívül beállíthatjuk az állapotát, ami initial, urgent vagy committed lehet. Az initial az automata kezdőállapotaként jelöli meg a kijelölt státuszt, a committed még inkább korlátozó, mint az urgent, tehát nem késleltetheti a következő átmenetet. Ugyanitt beállítható az is, ha az adott állapotban növekszik a költség valamilyen egyéb korlátozásból adódó várakozás miatt. Ekkor az állapot beállításában az Invariant pontban adható meg az idő előrehaladásával számított költség mértéke.

Az irányított élek mutatják, hogy melyik állapotból melyikbe van lehetőség átkerülni, ezen kívül pedig egyéb beállítások is megadhatóak. Minden élre megadható Select, Guard, Sync és Update információ. A Select-ben nemdeterminisztikus választásra van lehetőség, a Guard korlátozást állít be, aminek teljesülnie kell, hogy a feladat a következő állapotba kerüljön. A Sync lehetőséggel különböző automaták állapotátmeneteit lehet összehangolni, ehhez csatorna létrehozására van szükség. Az Update segítségével frissíthetők a változók értékei és az órák. Az automata template-ek az Editor menüpont alatt helyezkednek el, és itt található még a Declarations menüpont is. A Declarations pontban az egész rendszerre vonatkozó változókat és értékeket lehet megadni, valamint függvényeket létrehozni. Az értékeket egészsként kell meghatározni, mert a szoftver nem számol lebegőpontos alakban. Definiálhatunk órákat (clock), illetve csatornákat (chan), ezek az automaták közötti szinkronizálást segítik elő.

Minden template-hez definiálhatunk lokális változókat és paramétereket, ahol a paraméterek segítségével például meg tudjuk különböztetni a példányokat, ha a modell példányosítva van. Ekkor hozzárendelünk egy ID-t, amely sorszámot ad az automatáknak. A lokális változók között gyakran definiálunk órát, ha



3.1. ábra. Az UPPAAL Cora Editor ablaka

például egy elvégzendő feladatról van szó, külön mérhessük a munkaidejét, amit ilyenkor a modell elején le is kell nullázni.

Az Editor lapon található még a System Declarations pont, ami az automaták konkrét példányosítását végzi.

Az Editor mellett két fontos menüpont található. Ha nem vétettünk szintaktikai hibát, a szimulátor betölti a létrehozott automaták összes példányát, mellette pedig megjeleníti a változókat, amelyek először a kiinduló állapotban vannak, ahogy az automaták is. Itt lehetőség van egy megoldást lefuttatni, ekkor szinte biztos, hogy nem az optimális eredményt kapjuk. A lépéseket saját magunk is kiválaszthatjuk, a végeredményt pedig mindkét esetben vissza lehet játszani, vagy elmenteni egy külön fájlba.

A szimulátor közben bemutatja, hogyan változtak az értékek, hogy az automaták melyik állapotukban vannak, valamint egy másik ábrán szekvencia diagramon láthatjuk a szinkronizáció lépéseit, és az automaták állapotátmenetét.

A 4.2 ábrán látható egy példa egy minta szekvenciadiagramot árázol, ahol két gépen három feladat kerül felosztásra, és bemutatja a szinkronizációs csatornák működését az egyik megoldásban.

A harmadik menüpont a Verifier, itt a Query-ben meg lehet adni lekérdezéseket, amelyet az UPPAAL lefuttat, majd kiírja az eredményt, amely két fajta lehet. Ha a beírt korlát alapján talált megoldást, akkor megkapjuk a *Property is satisfied* üzenetet zölddel kiírva a Status pont alatt. Az Overview-nál is megjelenik a megadott Query, mellette pedig egy zöld jel. Ellenkező esetben a *Property is not satisfied* üzenetet kapjuk, az Overview pedig piros jelet tesz a Query-ben megadott korlát mellé.

4. fejezet

Problémadefiníció

A probléma egy egylépéses szakaszos eljárás ütemezéséhez kapcsolódik, ahol minden termék egy termelési lépés alatt készül el, ezeket a hívjuk munkáknak. A munkákat bármelyik gép (unit) elvégezheti, de egy munka csak egy géphez rendelhető hozzá. Ugyanígy egy gép egyszerre csak egy feladaton dolgozhat, és ha már egy munkát elkezdett, azt egy másik nem előzheti meg.

Adott a munkák és a gépek száma, valamint a gépeknek van egy meghatározott üzembe állási ideje, ez mindegyik berendezés egyedi tulajdonsága. Két munka elvégzése között felszámolunk átállási időt, amíg a gép testre szabja saját beállításait a következő feladathoz. A gyakorlatban ez tisztítást, újra beállítást és egyéb karbantartást jelent. Az átállási időt kétféleképpen lehet megadni, az egyik típus a szekvenciafüggő, amikor a feladatok sorrendje határozza meg az értéket. Mennyiségét az szabja meg, hogy az előző és az utána következő munka között mennyi időre van szüksége a berendezésnek. A másik megadási mód a szekvenciafüggetlen típus, amikor csak a berendezéstől függ az átállási idő. Mindkét modellt be lehet állítani úgy, hogy minkét típus függjön a géptől és a feladattól is. Itt...

A munkákat minden gép különböző idő alatt tudja elvégezni, de olyan eset is lehet, amikor egy gép nem tudja elvégezni az adott munkát. Minden feladat rendelkezik határidővel, amit nem léphet át, miközben várakoznia kell, ha a határidő előtt elkészül. Az ütemezés célja, hogy minimalizáljuk a várakozás költségeit, emellett viszont előfordulhat, hogy a megoldás nem elégíti ki a korlátozásokat, így infeasible lesz. Ha van feasible megoldás, szeretnénk lehetőleg az összes munkát elvégezni határidőre, valamint a modellt kiegészíteni korlátozásokkal úgy, hogy minél kevesebb idő alatt elvégezze az ütemezést, és minél optimálisabb eredményt adjon.

A feladatban munkákat és gépeket különböztetünk meg, amelyeket később P-vel és U-val jelölünk, a product és unit szakirodalomban használt megnevezések után.

5. fejezet

Egylépéses feladat LPTA modellje

5.1. Leírás

Az LPTA modell két template-ből, a rendszer deklarációkból és a változó deklarációkból áll. A template-ek egy-egy automatát írnak le, valamint saját változókkal is rendelkeznek, ezeket később részletesen be fogjuk mutatni. Ebben a példában a két sablon a gépeket és a feladatokat modellezi.

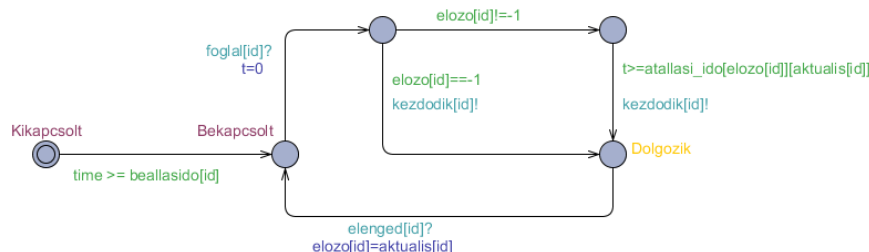
A következő fejezetekben részletesen bemutatjuk a két template-et, a lekérdezéseket és a definiált paramétereket.

5.2. Template-ek és deklaráció

A modellben két különböző template-tel dolgozunk, ezek a Gép és Feladatok elnevezésű sémák. Mivel az ütemezést három különböző változatra bontottuk, itt az első modell kerül bemutatásra, amely nem tartalmaz korlátozásokat az ütemezési folyamat gyorsításának szempontjából.

Az Gép elnevezésű template azt a minta gépet modellezi, amelyen a feladatokat el kell végezni. Három fő állapota van, ezek a kikapcsolt, bekapcsolt és a dolgozik. Kikapcsolt állapotból indul, és akkor kerül át bekapcsoltba, ha letelt az adott gépnek szükséges beállási idő, ezután kezdhet el dolgozni. Ha a berendezés már bekapcsolt állapotban van, egy feladat befoglalhatja, ezt a *foglal* csatorna szinkronizálja, és eltárolja, hogy melyik gép foglalt. Mielőtt *Dolgozik* állapotba jutna, meg kell vizsgálnia, hogy az azon a gépen ez lesz-e az első elvégzett munka, mivel két munka elvégzése között átállási időt számol fel. Ha előtte nem dolgozott, akkor átállási idő nincs, a berendezés átvált *Dolgozik* állapotba a *kezdődik* csatorna segítségével. Ellenben ha egy másik munka már megelőzte a jelenlegi munkát, az eltárolt aktuális és előző feladat azonosítója alapján ki kell

várnia a hozzájuk tartozó átállási időt, és csak azután kezdheti meg a munkát. A *foglal* csatorna közben szinkronizálja a gépet a feladat modelljével, a munka-idő letelte után pedig mindkét automata az *elenged* csatorna segítségével jut a következő állapotba, ami a gép esetében a bekapcsolt állapot, ekkor vár az újabb feladat érkezéséig. Itt váltja át az előző munka azonosítóját, hiszen az elvégzett munka a következő lefutás során már az előző munkának fog számítani.



5.1. ábra. A gép automatájának modellje

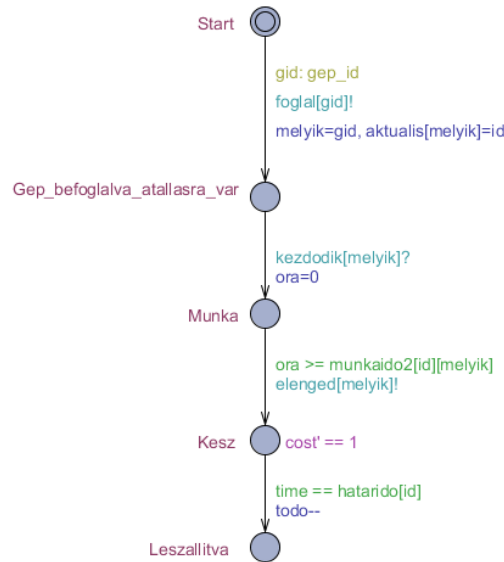
A másik template a *Feladatokat* mutatja be.

Ez a modell öt állapotot tartalmaz, ebben az esetben a feladat a *Start* stádiumból indul, ekkor a munka arra vár, hogy befoglalhasson egy gépet. Amikor ez megtörténik, átlép a *Gép befoglalásra vár* elnevezésű állapotba, előtte viszont a *foglal* csatorna segítségével szinkronizálja a saját státuszát a befoglalt gép státuszával, valamint elmenti, hogy aktuálisan hányas azonosító számmal rendelkező munka kerül elvégzésre ami fontos a munkaidő szempontjából, hiszen a saját magához tartozó munkaidőt kell felszámolni a gépen, és a gépnek is tudnia kell, hogy a korábban említett beállási időt meg tudja állapítani.

A *Gép befoglalásra vár* állapotban arra vár, hogy a befoglalt gépet beállítsák, ekkor kezdődhet el a feladat elvégzése. Minden feladathoz tartozik egy lokális óra, amelyen a munkaidejét méri, ezt a feladat kezdete előtt lenullázzuk, hogy a példányosítás során a következő feladat munkaideje ismét nullától legyen számítva. A gép beállítása után addig tartózkodik a *Dolgozik* állapotban, míg a munkaideje le nem telik, ekkor újra szinkronizálja a saját és a gép státuszát, hogy a gép felszabaduljon, a feladat pedig átkerülhessen *kész* állapotba.

A *kész* állapot azonban még nem azt jelenti, hogy a munka elkészült, hiszen a cél az, hogy pontosan a határidő leteltekor legyen leszállítva, ezért minél hamarabb készült el, annál többet kell várakoznia. Ebben a státuszban számoljuk fel a várakozás költségeit, amelyek az idő függvényében lineárisan növekednek.

Amikor elérkezik a meghatározott határidő, a munka leszállításra kerül, ez a végső *leszállítva* állapot.



5.2. ábra. A gép automatájának modellje

A template-ek meghatározása után a deklarációban adtuk meg a szükséges paramétereket, illetve függvényeket, amelyek szükségesek az automaták ütemezésének futtatásához. Meghatároztuk a gépek és a feladatok számát, ami az irodalmi példa szerint négy gépet és húsz feladatot jelent, majd ezeknek ki lett osztva egy saját azonosító. Szintén a globális deklarációban definiáltuk a munkaidőket gépek szerint, hiszen minden gép más-más idő alatt tud elvégezni egy feladatot. Ugyanígy megadtuk a határidőket, átállási időt, a beállási időt és a csatornákat is. Deklaráltuk a globális órát, amely a template-ektől függetlenül méri a totális időt, valamint a *minTime* függvényt, ami azonban az első modellben még nem szerepel, a későbbi modellekben korlátozási céllal jelenik meg.

5.3. Lekérdezések

A lekérdezések segítségével adjuk meg, hogy milyen kitélt szeretnénk leellenőriztetni az ütemezés során, melyre pozitív vagy negatív válasz érkezik. A jelenlegi modellben a feltétel, amit megvizsgáltunk az volt, hogy létezik-e olyan megoldási lehetőség, ahol minden feladat kész lesz határidőre.

Hogy a lekérdezés szerkezetét leegyszerűsítsük, evezettünk a deklarációban egy változót, amelynek értéke minden esetben megegyezik a munkaszámmal. Amikor a feladatot a *kész* állapotból *leszállítva* állapotba kerül, ezt az értéket mindig

egyel csökkentjük, így optimális esetben, tehát ha minden feladat elkészül, nulla értéket vesz fel. Így a lekérdezés az alábbi módon kerül megadásra.

$$E<> \text{ todo}==0$$

Ez a forma egyszerűbben megadható, és kiváltja a plusz változó nélküli hosszú megadást, ahol egyesével kellene ellenőrizni, hogy az adott feladat elkészült-e.

6. fejezet

Teszteredmények, összehasonlítás

A modellt először a szükséges korlátozásokon kívül egyéb korlátozásokkal nem egészítettük ki, amikkel csökkenteni lehetett volna az ütemezés futási idejét. A Query-ben megadott cél az volt, hogy olyan megoldást találjon, amikor az összes feladatot sikeresen elvégzi, tehát leszállításra kerülnek határidőre.

Mivel az modellt 10 percen belül nem tudta lefuttatni az UPPAAL, az irodalmi példában szereplő 20 feladat helyett először 1 feladattal került tesztelésre az automata futási ideje, majd egyesével növeltük a feladatok számát addig, amíg a modell képes volt tíz percen belül megoldást találni rá. Az eredményeket az alábbi táblázat tartalmazza.

Munkátszáma	Futási idő	Munkátszáma	Futási idő
1 munka	0,08 sec	11 munka	-
2 munka	0,11 sec	12 munka	-
3 munka	0,36 sec	13 munka	-
4 munka	1,99 sec	14 munka	-
5 munka	36,5 sec	15 munka	-
6 munka	10+ min	16 munka	-
7 munka	-	17 munka	-
8 munka	-	18 munka	-
9 munka	-	19 munka	-
10 munka	-	20 munka	-

6.1. ábra. Futási eredmények korlátozás nélkül.

Az eredmények alapján megállapítható, hogy korlátozás nélkül a 20 munka közül csak 5 munkával futott le az ütemezés tíz percen belül, a hatodik munka hozzáadása után tíz perc alatt nem kaptunk eredményt. Emiatt a további bővítést már nem végeztük el, beláthatóan egyiknek sem lett volna eredménye

belátható időn belül.

7. fejezet

Összefoglalás és jövőbeli tervek

Irodalomjegyzék

- [1] G M Kopanos, J M Lainez, and L Puigjaner. An Efficient Mixed-Integer Linear Programming Scheduling Framework for Addressing Sequence-Dependent Setup Issues in Batch Plants. *Industrial & Engineering Chemistry Research*, 48(13):6346–6357, 2009.

A. függelék

Jelölések