Egylépéses gyártási feladatok költség optimális ütemezése időzített automatával

Vida Judit, gazdaságinformatikus Bsc.
Témavezető: Dr. Hegyháti Máté, tudományos főmunkatárs
Széchenyi István Egyetem

2018. március 10.

Tartalomjegyzék

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 hajtanak 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) problématípusban 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 adottnak 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 időfelosztásossal ellentétben kettőt.
Y_{ij} értéke 1, ha i feladatot j berendezés elvégzi
$X_{iji'}$ 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 modellezik.
Létezik ütemezési döntések nélküli S-gráf is, ezt recept gráfnak hívják.
A nyilak, amelyek a csomópontokat kötik össze, a függőségeket reprezentálják a következő esetekben:

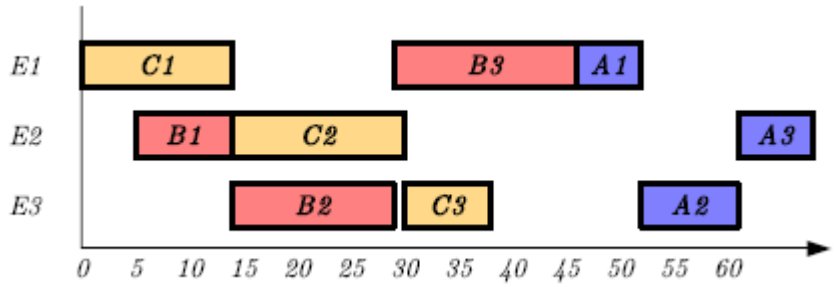
2.2.3. Petri háló és automaták

A Petri hálót és az automatát is gyakran alkalmazzák diszkrét esemény rendszerű modellezéshez,? és hogy kötegelt 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, összességégben 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. Kopanos2009
Az ütemezési feladatok megoldását Gantt diagramon ábrázolják. A példa há-



2.1. ábra. Példa egylépéses ütemezési feladat ábrázolására Gantt diagramon

rom berendezést jelöl az y tengelyen (E1, E2, E3), az x tengelyen pedig az eltelt idő látható percben megadva. A diagram azt mutatja be, hogy egyes berendezések mikor végezték el a feladatokat, amelyek A-val, B-vel és C-vel vannak jelölve. A Gantt diagramról leolvasható, hogy melyik munka mikor kezdődött, és mikor fejeződött be, ebből adódóan pedig megállapítható, hogy milyen hosszú ideig tartott.

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.

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

ahol
K az állapotok halmaza
Σ az események véges halmaza
δ: K x $\Sigma \rightarrow$ K részleges átmeneti függvény
s a kezdőállapot
F az elfogadó állapotok
Az időzített automaták kiegészülnek órákkal, amelyek stopperóraként 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 x ϕ(C) x Σ x C x 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.

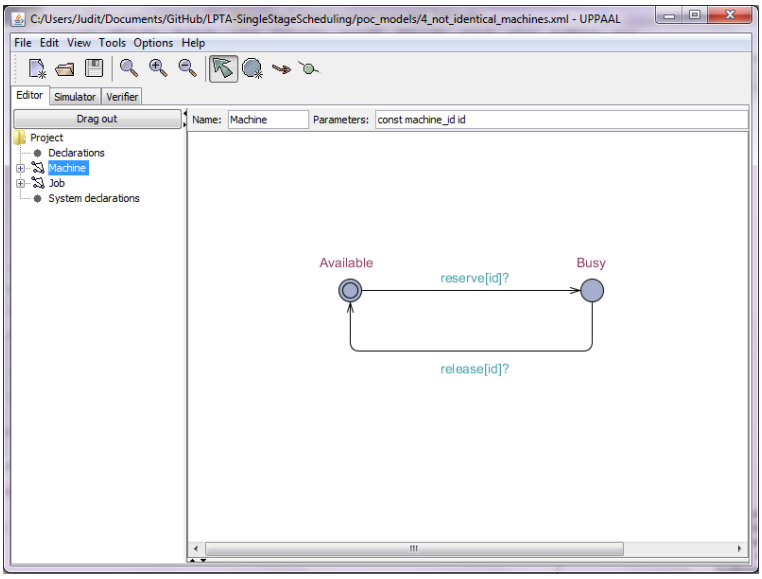
Példának vesszük az (a,1) (b,3) (a,4) (b,6) (b,10) időzített szót.
A fenti felírás azt mutatja, hogy melyik időpillanatban történik az esemény, amiből kiszámíthatjuk, hogy a betűk mennyi késleltetéssel követik egymást.

$$\text{d1}\rightarrow \mathbf{a} \rightarrow \text{d2}\rightarrow \mathbf{b} \rightarrow \text{d1}\rightarrow \mathbf{a} \rightarrow \text{d2}\rightarrow \mathbf{b} \rightarrow \text{b4}\rightarrow \mathbf{b}$$

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.

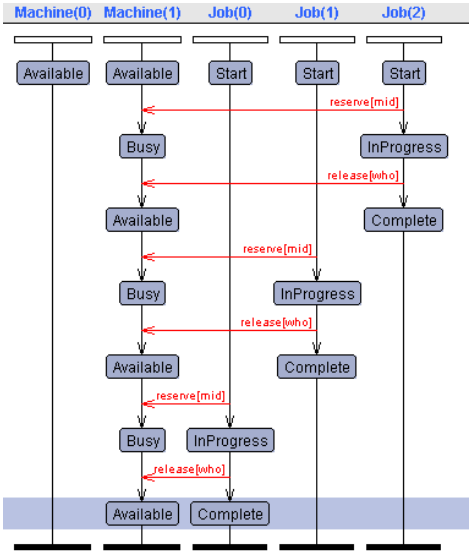
3.2. UPPAAL Cora

A választott irodalmi példa időzített automatákkal való ütemezését az UPPAAL Cora nevű szoftver segítségével modelleztük. A szoftver alkalmas az automaták 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 valamely egyéb korlátozásból adódó várakozás miatt. Ekkor az állapot beállításaiban 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 frissíthetőek 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 egészként 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



3.2. ábra. A folyamat szekvenciadagramja

megadott korlát mellé.
A főmenüben leginkább az általános lehetőségeket találjuk, a Tools és az Options pontokban találhatóak egyéb beállítások az automaták ütemezésével kapcsolatban. Leellenőrizhető a Declarations és a template-ek definiálása során megadott adatok és korlátozások szintaktikai helyessége, az Options pedig az ütemezési beállításokat tartalmazza.
Átállítható például, hogy ütemezés során mélységi, szélességi vagy egyéb keresést alkalmazzon, valamint hasznos funkció, hogy ha a Diagnostic Trace értékét some-ra állítjuk, akkor abban az esetben, ha a Query-ben definiált lekérdezés teljesül, a megoldást be tudja tölteni a szimulátorba. A szimulátorban grafikusan, diagramon és változónként is elemezhető az eredmény.
Az UPPAAL Cora-hoz Java Runtime Enviroment szükséges, mivel a felhasználói interfész Java nyelven van implementálva.

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 elégít ki a korlátozásokat, így infeasble 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

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.

5.1. Első modell

5.1.1. Deklaráció

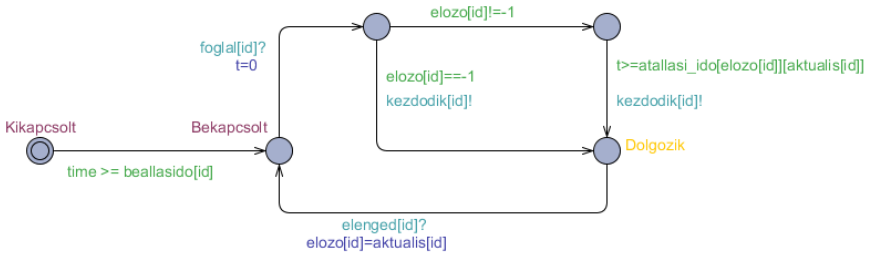
A template-ek meghatározásához a deklarációban adtuk meg a szükséges paramétereket, illetve függvényeket, amelyek szükségesek az automaták ütemezésének futtatásához. Meghatároztuk a gépek és a feladatok számát, ami az irodalmi példa szerint négy gépet és huszonöt feladatot jelent, majd ezeknek kiosztottunk egy saját azonosítót. 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.
A modell három csatornát tartalmaz, amelyek a *foglal, elenged* és *kezdődik* elnevezésűeket, ezek mindegyike annak a gépnek a sorszámát kapja meg paraméterben, amelyiken a feladat végrehajtódik.
Deklaráltuk a globális órát, amely az ütemezés kezdetétől méri az összes időt, amíg az utolsó lépést el nem végzi, ezen kívül a *feladat* és a *gép* template rendelkezik külön órával. A feladat sablon a munkaidővel és a határidővel hangolja össze a saját óráját, a gép pedig a saját beállási idejét méri vele.

5.1.2. Template-ek

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.

Gép template

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 segítségébel. 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 munkaidő 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

Feladat template

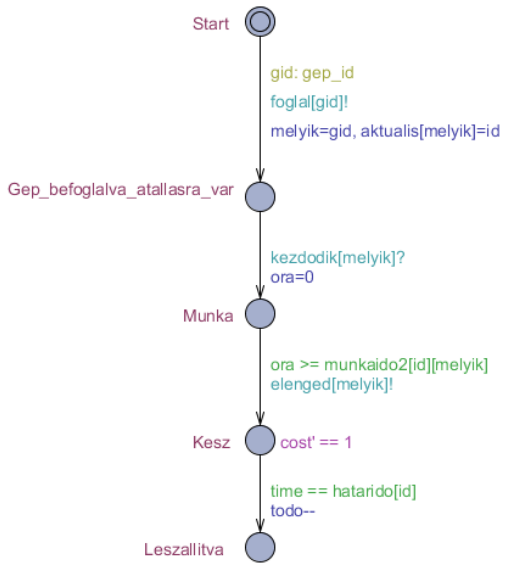
A másik template a *Feladatok*at mutatja be.
Ez a modell öt állapotot tartalmaz, ebben az esetben a feladat a *Start* stádiumból indul, ekkor a munka arra vár, hogy befoglalhasson egy gépet. Amikor ez megtörténik, átlép a *Gép befoglalásra vár* elnevezésű állapotba, előtte viszont a *foglal* csatorna 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, hiszem a saját magához tartozó munkaidőt kell felszámolni a gépen, és a gépnek is tudnia kell, hogy a korábban említett beállási időt meg tudja állapítani. Ezen kívül lekorlátozzuk, hogy ne olyan gépet foglaljon le, amely nem tudja a munkát végrehajtani, ezt a munkaidők megadásánál kiugróan magas számmal különböztettük meg, egységesen 100000-rel.
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.

5.1.3. Lekérdezések

A lekérdezések segítségével adjuk meg, hogy milyen kitételt szeretnénk leellenőriztetni az ütemezés során, melyre pozitív vagy negatív válasz érkezik. A jelenlegi modellben a feltétel, amit megvizsgáltattunk 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 feladat 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.

$$E<>\ todo==0$$

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



5.2. ábra. A gép automatájának modellje

5.2. Továbbfejlesztett modellek

5.2.1. Modell 2

A modell második változatában a *gép* template harmadik állapota az előzőhöz képest *committed* minősítést kapott, tehát ebben az állapotban nem történhet késleltetés, mindenképpen tovább kell haladnia a következő állapotba, ami a *dolgozik* állapot, de előtte megvizsgálja, hogy volt-e előző feladat. Ha volt, akkor a beállási időt kell kivárnia, ezután mehet át a *dolgozik* státuszba.
A másik módosítás a *feladat* template *Start* állapotában történt. Itt hozzáadtunk egy korlátozást, amely megakadályozza, hogy ha kevesebb idő áll rendelkezésre a határidőig, mint az aktuális feladat munkaideje, akkor eleve nem foglalja be a gépeket. Ezzel olyan lefutási lehetőségeket zárunk ki, amelyek nem vezetnének megoldáshoz, viszont növelik a modell méretét, ezzel pedig megnövekszik az ütemezés futtatásának ideje.

5.2.2. Modell 3

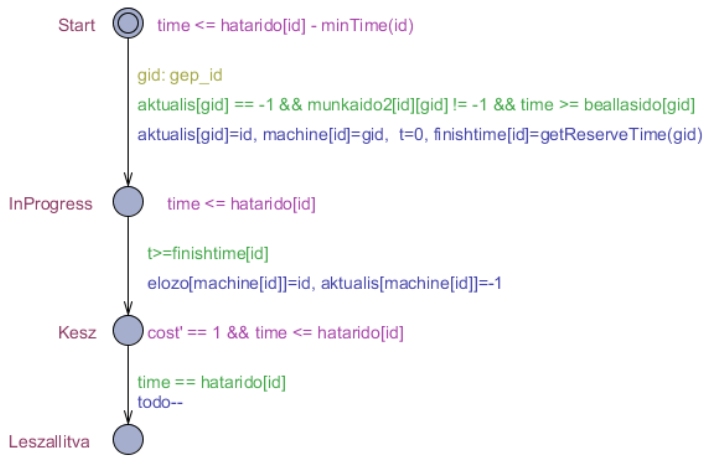
A *gép* sablon *committed* és *dolgozik* állapota között van egy átmeneti állapot, ahova akkor jut a gép, ha már nem az első feladaton dolgozik, ezután vizsgálja meg, hogy milyen hosszú beállási időre van szükség az előző feladatot figyelembe véve. A Modell 3-ban korlátozzuk azt az időt, amíg a gép ebben az állapotban tartózkodhat, ami ezután nem lehet több, mint az előző és az aktuális munka

alapján megállapított átállási idő.
Továbbá a *dolgozik* állapotban sem tartózkodhat tovább, mint a feladathoz szükséges munkaidő, ezzel pedig megelőzhető, hogy a gép *dolgozik* állapotban maradjon, ha a feladat a *munka* státuszból nem tud átmenni a *kész* állapotba, így nem is engedné el a gépen, amit korábban befoglalt. Ennek segítségével a berendezés csak addig lehet *dolgozik* állapotban, amíg le nem telik a tevékenységhez szükséges idő, utána mindenképp fel kell szabadítani a gépet.
A korlátozás segítségével nem maradhat elfoglalt a gép egy olyan munkával, ami meghatározott időn belül nem tud befejeződni.
A *feladat* template-be még három korlátozás került, a *gép befoglalásra vár* státusz után akkor foglalhat be gépet, ha több idő áll rendelkezésre, mint amennyivel nem tudna befejeződni a saját határidejéig. *Munka* állapotól *kész állapotba* akkor válthat át, ha nem haladta meg a határidőt, ugyanígy akkor léphet tovább a *kész* állapotból, ha a határidőig még van idő. A modell 3-ban a *gép* template átállási idővel kapcsolatos korlátozása változott, itt megengedjük az egyenlőség mellett azt is, hogy az óra magasabb értéket mutasson. A *feladat* sablonban hasonló módon a *munká*ból a *kész* állapotba akkor is átválthat, ha a saját órája értéke nem feltétlenül egyenlő a feladat munkaidejével, de nagyobb is lehet.

5.2.3. Modell 5

Az ötödik modell egy template-ből áll, ez pedig a feladatok automatája. Itt egy modellben kezeljük a gépekre vonatkozó adatokat, ez az automata négy állapotból áll, amelyek a *Start, InProgress, Kész* és *Leszállítva*. Az állapotok a modell 3-hoz hasonlóan korlátozottak.

A *Start* státuszból akkor kerül át *InProgress* állapotba, ha megfelel néhány



5.3. ábra. A modell 5

korlátozásnak. A munkaidő értéke nem lehet -1, ez azt jelenti, hogy azon a gépen nem lehet elvégezni az adott feladatot. A beállási időnek el kell telnie, ezért az óra állása nem lehet kisebb, mint ez az érték. Ha ezek a feltételek teljesülnek, elmentjük a munka azonosítóját és a gép azonosítóját, lenullázzuk az órát. A *getReserveTime* függvényt a deklaráció tartalmazza, ezzel a függvénnyel megvizsgáljuk, hogy azon a gépen az aktuális feladat lesz-e az első, és ha nem, akkor az átállási idő és a munkaidő összegét adja vissza, különben nullát. Ezt a számot a *finishtime* nevű változóban tároljuk el.
A következő két állapot közti átmenet akkor történhet meg, ha letelt a finishtimeban megállapított idő, ezután pedig a modell visszaállítja a az *aktuális* változó értékét, az *előző*t pedig frissíti az abban a pillanatban aktuális feladat azonosítójára, mivel a következő lefutás során ez már az előző munkának fog megfelelni.
A *kész* és *leszállítva* státuszok közti átmenet a korábbi modellekhez hasonlóan alakul, az óra meg kell, hogy egyezzen a határidővel, a még elvégzésre váró feladatok számát pedig csökkentjük.

6. fejezet

Teszteredmények, összehasonlítás

A tesztelés parancssori környezetben került futtatásra, ahol azt vizsgáltuk, hogy az egyes modellek hány feladatig találnak megoldást megállapított időn belül, valamint találnak-e megoldást, amennyiben az időkorláton belül végigfut az ütemezés. Ezen kívül megvizsgáltuk a költségeket minden modellre, hogy ebben a tekintetben van-e köztük különbség.

6.1. Az irodalmi példa

A dolgozat során Kopanos[1] példáját adaptáltuk.
A 6.1 táblázat mutatja a P-vel jelölt feladatok feldolgozási idejét egyes u-val jelölt gépeken, de egy munkát nem minden gép tud elvégezni. A feladathoz tartozik határidő, a beállási idő pedig a gépek kezdeti, bekapcsoláshoz szükséges idejét mutatja.
A következő 6.1 táblázatban az egyes feladatok közti átállási idő látható, tehát ennyi időre van szüksége a gépeknek egyik feladatról a másikra való beállításához.

Feladatok	Munkaidő (nap)				Határidő
	u1	u2	u3	u4	
P1	1.538	.	.	1.194	15
P2	1.500	.	.	0.789	30
P3	1.607	.	.	0.818	22
P4	.	.	1.564	2.143	25
P5	.	.	0.736	1.017	20
P6	5.263	.	.	3.200	30
P7	4.865	.	3.025	3.214	21
P8	.	.	1.500	1.440	26
P9	.	.	1.869	2.459	30
P10	.	1.282	.	.	29
P11	.	3.750	.	3.000	30
P12	.	6.796	7.000	5.600	21
P13	11.25	.	.	6.716	30
P14	2.632	.	.	1.527	25
P15	5.000	.	.	2.985	24
P16	1.250	.	.	0.783	30
P17	4.474	.	.	3.036	30
P18	.	1.492	.	.	30
P19	.	3.130	.	2.687	13
P20	2.424	.	1.074	1.600	19
P21	7.317	.	3.614	.	30
P22	.	.	0.864	.	20
P23	.	.	3.624	.	12
P24	.	.	2.667	4.000	30
P25	5.952	.	3.448	4.902	17
Beállási idő	0.180	0.175	0.000	0.237	

6.1. táblázat. A feladatok munkaideje

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20
P1	.	0.3	0.8	1.5	0.6	0.5	2.0	1.1	0.0	.	0.5	1.0	0.2	0.8	0.7	0.5	1.8	.	2.5	0.3
P2	0.2	.	1.3	0.9	2.5	0.2	0.8	2.5	0.4	.	0.6	2.5	0.5	0.2	0.6	0.0	1.1	.	0.8	2.5
P3	0.5	0.9	.	0.5	0.7	0.4	1.5	0.4	0.9	.	0.2	1.5	0.8	0.7	0.0	2.0	0.6	.	0.5	1.3
P4	1.1	0.7	0.2	.	0.8	2.0	0.9	0.0	1.3	.	1.5	1.0	1.8	0.6	1.3	0.6	1.5	.	1.0	0.5
P5	0.5	1.0	0.0	1.3	.	0.5	2.0	1.3	0.9	.	0.4	0.3	2.0	1.0	2.0	0.7	0.2	.	0.3	0.9
P6	0.2	0.0	1.3	1.0	1.0	.	0.7	1.3	0.8	.	0.7	0.6	0.5	0.7	0.5	2.0	0.9	.	1.1	0.5
P7	0.9	0.5	1.1	0.0	1.4	0.6	.	4.0	0.5	.	0.5	0.8	0.3	0.4	1.1	0.5	1.5	.	0.9	1.5
P8	1.5	2.0	0.4	1.3	0.5	0.9	0.7	.	0.9	.	0.4	1.8	0.6	1.5	0.6	0.5	0.7	.	0.9	1.1
P9	2.5	0.6	0.5	0.8	0.6	1.8	0.6	0.2	.	.	2.0	1.5	2.0	0.6	0.9	1.3	1.8	.	0.7	0.8
P10	.	.	.	.	.	.	.	.	.	.	1.0	1.3	.	.	.	.	.	0.0	0.8	.
P11	0.8	1.0	1.3	0.8	1.1	0.4	2.5	0.9	2.0	0.0	.	0.8	1.0	2.5	1.5	0.6	0.8	2.5	1.3	0.6
P12	0.2	0.7	0.6	0.3	0.9	0.3	0.5	0.2	0.4	0.4	0.2	.	2.0	1.1	0.9	0.2	2.0	.	0.6	0.5
P13	0.9	0.8	1.3	1.1	1.3	0.6	0.4	1.5	0.5	.	0.4	1.8	.	0.0	1.8	0.8	0.6	.	2.5	1.0
P14	1.8	1.5	2.0	1.5	0.4	2.5	0.5	0.5	1.1	.	0.6	1.5	0.8	.	0.5	0.5	0.0	.	1.1	1.5
P15	1.5	0.9	1.3	0.9	0.6	0.1	0.2	1.1	0.3	.	1.3	0.5	0.4	0.6	.	1.3	1.0	.	1.3	1.0
P16	1.3	2.0	1.5	0.5	0.4	0.9	1.8	0.6	0.7	.	1.5	2.0	0.6	0.4	0.8	.	0.9	.	0.5	0.2
P17	0.7	0.7	0.9	0.8	1.4	0.6	0.8	1.0	0.6	.	0.9	0.4	0.5	0.9	2.0	1.3	.	.	0.7	1.1
P18	.	.	.	.	.	.	.	.	.	0.0	0.8	1.3	.	.	.	.	.	.	1.3	.
P19	0.6	0.5	1.1	0.5	0.4	1.4	0.9	0.4	0.6	0.4	2.5	0.0	0.7	0.7	0.5	1.3	0.7	0.2	.	2.0
P20	0.7	0.5	2.0	1.4	0.0	1.1	0.5	0.6	1.4	2.0	0.4	0.9	2.0	0.8	0.7	0.3	0.5	.	0.8	.

6.2. táblázat. Beállítási idők P feladatok között

6.2. A tesztelés módja és környezete

A tesztesetek futtatását egy Acer Aspire V3-571 laptopon futtattam, amelyen Windows 7 Professional operációs rendszer fut. A processzor Intel Core i3 3120M 2,50 Hz típusú, a gép 4 GB RAM-mal rendelkezik.
Az UPPAAL Cora szoftver, amelyen az automatákat modelleztük a 2006-os 4.0.2 verzió.
Minden tesztesetet háromszor futtattunk le, hogy még pontosabb eredményt kapjunk az ütemezés elvégzésének gyorsaságáról. Ezeket addig végeztük, amíg az egyes esetek futásának ideje el nem érte az egy perces időkorlátot. Amennyiben minden alkalommal sikeresen megkaptuk az eredményt, a három érték átlagát tekintettük érvényesnek, később ezt vettük figyelembe.

6.3. Első teszteset

Munkaszám	Modell 1	Modell 2	Modell 3	Modell 3v	Modell 5	Költség
1	0,073	0,076	0,06	0,056	0,053	0
2	0,086	0,083	0,056	0,06	0,06	0
3	0,116	0,011	0,073	0,073	0,07	0
4	0,23	0,43	0,18	0,23	0,083	0
5	0,69	*4,48*	1,94	1,63	0,12	0
6	13,3	-	16,33	13,8	0,15	0
7	>1 min		>1 min	>1 min	0,92	0
8					3,41	0
9					6,33	0
10					>1 min	0

6.3. táblázat. Első teszteset

A 6.3 táblázat mutatja be azokat az eredményeket, amelyeket a modellek lefuttatása után kaptunk három érték átlagából. A mennyiségek másodpercben vannak megadva, és az egyes változatokat addig futtattuk, amíg egy percen belül végigért az ütemezéssel.
A költségek jellemzően nem változtak, egy kivétellel mindig 0 volt az eredmény. Az egy kivétel a Modell 2 esetében jelentkezett, ahol 5 munkára 12,388 költséggel talált megoldást.
Amint az eredmények alapján megállapítható, nem minden esetben jelenti azt a modell továbbfejlesztése, hogy több munkát tudunk vele egy perc alatt elvégeztetni, vagy hogy a kisebb mennyiségű munkákat gyorsabban végzi el. A Modell 1 és a Modell 3v ugyanúgy 6 munkát képes elvégezni a meghatározott időkorláton belül, a Modell 3v néhány esetben lassabbnak is bizonyult.
A Modell 3 és a 3v annyiban különbözik egymástól, hogy a korábban Modell 3-nál a meghatározott frissítések közül at átállási idő, a határidő és munkaidő időbeli korlátozásánál szereplő engedmény csak a *3v* modellben szerepel. A két változattal elért teszteredmények alapján nagyobb számú munkára a 3v modell

ad gyorsabban eredményt, viszont egyformán hat feladattal történő ütemezés fér bele egy percbe.
A Modell 2 öt feladatra jelentősen hosszabb idő alatt talált megoldást, hatra pedig már egyáltalán nem. Itt azt az eredményt kaptuk, hogy nincs feasible eredmény. Kisebb számú munkára hasonló időeredmények jöttek ki, mint a Modell 1-nél, így megállapíthatjuk, hogy a Modell 2 korlátozásai nem adtak hozzá a teljesítményéhez.
A Modell 5 ért el kiemelkedően jobb teljesítményt, itt a kilenc feladattal történő ütemezés befejeződött valamivel több, mint hat másodperc alatt, viszont tíz munkával már nem fért bele egy percbe.
Összességében megállapítható, hogy a fokozatosan hozzátett korlátozások nem javították számottevően a számítási teljesítményt, sőt néhány esetben azzal, hogy az automata szerkezete bonyolultabbá vált, több erőforrást vett igénybe az ütemezés, így nem javultak az eredmények, néhány helyen inkább pár másodpercet romlottak. A korlátozások beépítésének célja az lett volna, hogy a keresési fának olyan ágait eleve kizárjuk vele, amelyek előrelátható módon nem vezettek volna jó megoldáshoz, deadlock-ba ért volna az ütemezés. A korlátozások másik célja az volt, hogy ha ugyanabba az állapotba több átmenettel el lehet jutni, de különböző sorrendben, akkor a modell ne járja végig az össze útvonalat, ezeket egyszer elég megvizsgálni. Tovább korlátozható a modell az állapotok *committed* és *urgent* beállításaival, amik nem engednek késleltetést a megjelölt állapotban.
A Modell 5 azért lehetett számottevően gyorsabb, mert egy automata az alapja, és nem kell csatornák segítségével paralell composition-t létrehozni.

6.4. Kapcsolók

A parancssori futtatás során lehetőség van különböző kapcsolók megadására.
Az UPPAAL Cora parancssori következő parancssori paramétereit használtuk: *E, C, n, o, S*, ezek segítségével futtattuk le a második teszteset során a modelleket, hogy megvizsgáljuk, a jól megválasztott kapcsolókkal lehet-e időt spórolni. A gyorsabb lefutás szükséges ahhoz, hogy minél több feladattal elvégezhető legyen az ütemezés.
A kapcsolók funkciói:

- **-E**: Nem írja ki a részletes eredményt a kimenetre, csak a főbb eredményeket, jelen esetben azt, hogy talált-e feasible megoldást és a költségeket.
- **-C**: Lecsökkenti a memóriafelhasználást, főleg olyan modelleknél, ami több órát is tartalmaz. Gyorsítási céllal használható.
- **n**: Kiválasztja az extrapolációs operátort, ebben az esetben lehet automatikus, exrapoláció nélküli, különbség alapú vagy location alapú.
- **o**: Kiválasztja a keresés irányát, amely lehet szélességi, mélységi, optimális, random mélységi, random optimális és heurisztikus.

- **S**: Optimalizálja a helyigényt. Három fokozata van, kikapcsolt, alapértelmezett és a leghatékonyabb.

Az ütemezés során minden modellt lefuttattunk minden kapcsolókombinációval, ezután megvizsgáltuk, hogy melyik modell melyik kombinációval adja az optimális megoldást. Az eredményeket az alábbi táblázat tartalmazza.
A 6.4 táblázatban látható, hogy melyik kapcsolókkal futott le leggyorsabban

Munkaszám	Modell 1 -E-n1-o3-S1	Modell 2 -C-E-n3-o3-S0	Modell 3 -C-E-n0-o3-S2	Modell 3v -C-E-n2-o3-S0	Modell 5 -C-E-n0-o2-S0
1	0,073	0,076	0,073	0,073	0,07
2	0,083	0,08	0,08	0,083	0,076
3	0,1	0,14	0,1	0,093	0,076
4	0,29	0,36	0,21	0,15	0,086
5	0,89	3,63	0,92	0,59	0,11
6	14,37	$>$ 1 min	4,01	4,26	0,16
7	$>$ 1 min		24,61	25,24	0,35
8			$>$ 1 min	$>$ 1 min	1,7
9					2,42
10					30,14
11					$>$ 1 min

6.4. táblázat. A modellek futtatása kapcsolókkal

az adott modell ütemezése. A költség mindenhol 0 maradt, kivéve a Modell 2-ben, ahol a három és négy munkával történő lefutás 8,194 költséggel történt meg, míg az öt feladatos modellben ez az érték 14,388.
A kapcsolókkal való kiegészítés után is az ötödik modell bizonyult a leggyorsabbnak, ami így már 10 feladat ütemezésére képes egy percen belül a korábbi 9-hez képest. Változás történt a Modell 3-ban és a Modell 3v-ben is, ezek 6 feladat helyett 7-et tudnak ütemezni egy percen belül.

7. fejezet

Összefoglalás és jöbőbeli 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

Jelolesek