DOI: 10.15477/SZE.MMTDI.2017.009

Ipari gyártósorok ütemezésének modelljei és alkalmazásaik ipari jellegű feladatokon

Doktori értekezés

Írta: Hajba Tamás

Témavezető: Dr. Horváth Zoltán
Széchenyi István Egyetem
Matematika és Számítástudomány Tanszék
Győr

2016

Infrastrukturális Rendszerek Modellezése és Fejlesztése
Multidiszciplináris Műszaki Tudományi Doktori Iskola

DOI: 10.15477/SZE.MMTDI.2017.009

Tartalomjegyzék

DOI: 10.15477/SZE.MMTDI.2017.009

DOI: 10.15477/SZE.MMTDI.2017.009

DOI: 10.15477/SZE.MMTDI.2017.009

Táblázatok jegyzéke

DOI: 10.15477/SZE.MMTDI.2017.009

DOI: 10.15477/SZE.MMTDI.2017.009

1. fejezet

Bevezetés

Egy XXI. századi vállalat a működése során szinte minden területen optimalizálási problémával találja magát szemben. A raktározás, szállítás, beruházás, adózás során például a következő kérdések merülhetnek fel:

- milyen időközönként és mennyit rendeljünk egy adott alapanyagból, hogy a gyár szükségletei ki legyenek elégítve és a raktározás költsége minimális legyen;
- milyen útvonalon szállítsuk el a megrendelőhöz az árut, hogy a szállítási költség minimális legyen;
- hová építsük a következő gyárat, hogy a megtérülési idő minimális legyen;
- milyen formában adjunk a dolgozóknak bért, hogy a vállalat által fizetett adó minimális legyen?

Kutatásaim során én egy, az ütemezés területéről származó optimalizációs problémával foglalkoztam, melyben munkák valamely szempont szerinti optimális sorrendjét kell meghatározni egy gyártosoron.

A gyárakban a termelési rendszerek az idők folyamán, részben a vevők igényeinek változása miatt is egyre bonyolultabb termékek megjelenésével jelentősen átalakultak. Amíg egy termék előállítása csak egy (vagy néhány) művelet elvégzését igényelte, addig a temékeket egyetlen gép vagy munkás állította elő. A termelés hatékonyságának növelése érdekében sok esetben több ugyanolyan gépet is beállítottak az üzembe, melyek egymás mellett párhuzamosan dolgoztak. Amint a gyártandó termékek előállításához egyre több műveletet kellett elvégezni, úgy vált ez a módszer egyre inkább használhatatlanná. Sok művelet elvégzését igénylő termék összeszerelésénél több párhuzamos gépet alkalmazva az egyik probléma a következő: mivel mindegyik gép elvégzi az összes műveletet, ezért mindegyik gépnél szükség van az összes alkatrészre, illetve eszközre, melyet az összeszerelés során használnak. Ennek a biztosítása azonban az üzemek mérete miatt nem lehetséges. Így a tömegtermelésben az azonos összeszerelési

DOI: 10.15477/SZE.MMTDI.2017.009

műveleteket igénylő, összetett termékek előállítására megjelentek a gyártósorok. A gyártósorok mentén vannak a munkaállomások, ahol az összeszerelés egy (vagy néhány) műveletét végzik el emberek vagy robotok. A gyártósor mentén tehát az egyes állomások lényegében sorosan vannak kapcsolva; egy adott gép csak akkor végezhet el egy műveletet egy munkadarabon, ha a munkadarabot már megmunkáltuk az adott gépet megelőző gépen. Képzeljük el, hogy egy termék előállítását egy gyárban egyetlen gép végzi. Ekkor ha a gép meghibásodik, akkor a gyárban teljesen leáll a termelés. Ezzel szemben ha egy gyártásoron az egyik gép elromlik, akkor a soron az őt megelőző gépek tovább dolgozhatnak, míg a soron utána jövő gépek is tovább tudnak addig dolgozni, amíg van megmunkálásra váró munkájuk. Gyártósorok használatának másik nagy előnye például az egygépes előállításhoz képest, hogy növelni lehet az időegység alatt előállított termékek számát. Tegyük fel például, hogy van egy munkánk, amely 2 műveletből áll; mindkét művelet elvégzési ideje 1 időegység. Ha egy géppel csináljuk mindkét műveletet, akkor 2 terméket 4 időegység alatt lehet előállítani. Ellenben ha egy két gépes gyártósort használunk, ahol az első gép végzi az első műveletet, a második gép pedig a második műveletet, akkor 3 időegység alatt elő tudunk állítani 2 terméket.

1.1. Motivációk és célok gyártósorok működésének matematikai modellezéséhez

Egyetemi éveim során az ELTE matematikus szakán negyed, illetve ötödévben különféle operációkutatással kapcsolatos tárgyakat hallgattam. Az optimalizálással kapcsolatos érdeklődésem a későbbiekben is megmaradt. A Széchenyi István Egyetem oktatójaként két ipari projektben is részt vettem, melyekben a gyártás területéről származó feladatokat kellett megoldanunk. Az egyik projektben egy műpadlógyártással foglalkozó cég egyik üzemében a termelés optimalizása volt a feladatunk. A munka során azt tapasztaltuk, hogy a napi gyártás menetét egyetlen mérnök határozta meg a korábbi tapasztalatai alapján Excel táblázatok segítségével. A projekt során kifejlesztettünk egy viszonylag egyszerű heurisztikát, melynek a segítségével a gyár jelentős megtakarításokat ért el. Később egy TÁMOP projekt keretén belül a gyártósorok ütemezésével kezdtem foglalkozni. A témakör áttekintésekor kiderült, hogy a nagyvállalatok a termelésütemezési problémák megoldására különböző szimulációs szoftvereket alkalmaznak. Az egyik ilyen gyakran használt szoftver a Technomatics Plant Simulation [5]. Ezekben a szoftverekben általában felépíthető a gyártási folyamatok egy szimulációs modellje. A szoftverek tartalmaznak továbbá valamilyen beépített optimalizáló eljárást (genetikus algoritmus, szimulált hűtés), melyek az optimalizáció során a szimulációs modellt használják a célfüggvény kiértékelésére. Mivel a beépített eljárások heurisztikák, ezért ilyen módon általában nem az optimális megoldást, hanem egy ahhoz közeli ”jó” megoldást kapunk. A kutatásom célja éppen az

DOI: 10.15477/SZE.MMTDI.2017.009

volt, hogy olyan eljárást találjak, mely megadja az optimális megoldást szimuláció használata nélkül. Ehhez kétszintű modellezést alkalmaztam. Először is az ipari termelés sajátosságainak figyelembevételével megadtam a gyártósorok optimalizálásának több új matematikai modelljét. Ezt követően ezen matematikai modelleknek különféle matematikai programozási modelljeit adtam meg, melyeknek az optimális megoldását erre alkalmas szoftverekkel kaphatjuk meg.

Mint a disszertációban majd látni fogjuk, ezzel a módszerrel sikerült két ipari feladat optimális megoldását kevesebb, mint 10 perc alatt meghatározni, tehát az új modellezési eszközök segítségével akár ipari feladatokat is megoldhatunk. Természetesen az ipari feladatok méretéből adódóan nem várhatjuk, hogy ezzel a módszerrel minden feladat optimális megoldását rövid időn belül megkaphatjuk, ugyanakkor az új modellezési eszközök segítséget nyújthatnak más típusú módszerek (például heurisztikák) kiértékelésében.

1.2. A kutatás módszertani ismertetése

A kutatásom során először átnéztem a gyártósorok modellezéséhez kapcsolódó szakirodalmat. Ezután az általam modellezni kívánt gyártósor működését legjobban leíró permutációs flow shop feladatnál áttekintettem a különböző egzakt, illetve heurisztikus módszereket. Megállapítottam, hogy az ipari termelés számos sajátosságát a klasszikus permutációs flow shop feladat nem veszi figyelembe, ezért új feladattípusokat definiáltam. Felhasználva az irodalomból korábban ismert eredményeket, mindegyik új feladattípusnak megadtam többféle vegyes egészértékű lineáris programozási (MILP) modelljét.

A modellek validálásához és összehasonlításához új tesztkészleteket hoztam létre. Az új MILP modelleket többféle szempont szerint is összehasonlítottam. A kis méretű tesztfeladatoknál, melyeknél a modellek segítségével megkaptuk az optimális megoldást, a modelleket összehasonlítottam a futásidők alapján, míg a nagy méretű feladatoknál a modelleket a megoldásuk során kapott célfüggvényérték, illetve alsó korlát alapján vetettem össze. A modelleknek egy tesztkészleten egy adott szempont szerinti összehasonlításához használt egyik fontos mérőszám a modellek átlagos rangja volt. A rangok számításánál holtverseny esetén a következőképpen jártam el: ha egy feladatnál egy adott szempont szerint 2 (vagy 3) modell holtversenyben állt az $i.$ és $(i{+}1)$-edik ($i.$, $(i{+}1)$-edik, $(i{+}2)$-edik helyen), akkor mindkét (mindhárom) modell rangját $(2i+1)/2$-nek ($(3i+3)/3$-nak) vettem. Ezzel a módszerrel minden feladatnál a modellek rangjának összege $k(k+1)/2$ lett, ahol k a modellek számát jelöli.

1.2.1. A futtatási környezet

A vegyes egészértékű lineáris programozási modelleket a tesztelés során a GAMS [86] modellező nyelven írtam le, majd a CPLEX 12.3 [24] segítségével oldottam meg őket. A futtatás

DOI: 10.15477/SZE.MMTDI.2017.009

során a CPLEX alapbeállításait használtam a következők kivételével. Mivel gyakorlati feladatok során az optimális ütemezés megadására korlátozott idő áll csak rendelkezésre, ezért a CPLEX-ben a futásidő hosszát 10 percben maximalizáltam. Ezen kívül, mivel a CPLEX-ben lehetséges a párhuzamos futtatás, ezért a determinisztikus párhuzamos futtatás opciót használtam 4 thread segítségével. A futtatásokat egy Intel Xeon E31225 3.1 GHz-es 4 GB RAM-mal felszerelt számítógépen végeztem el.

1.3. A disszertáció felépítése

A disszertáció a következő felépítést követi. A 2. fejezetben bemutatom a gyártósorok ütemezési problémáit, illetve az őket leíró matematikai modelleket, majd röviden ismertetem a szakirodalomból ismert legfontosabb megoldó módszereket. Az új eredményeimet a disszertáció 3., 4., 5. és 6. fejezete tartalmazza. A 3. fejezet az ismétlődő permutációs flow shop feladattal foglalkozik. A 4. fejezet a speciális ismétlődő permutációs flow shop feladatról szól. Az 5. fejezetben a palettákat és véges puffert tartalmazó ismétlődő permutációs flow shop témakörét taglalom. A 6. fejezetben a lot méretet, palettákat és véges puffert tartalmazó ismétlődő permutációs flow shop feladat MILP modelljeit mutatom be. A 7. fejezetben a disszertáció eredményeit összegzem, míg a 8. fejezet tartalmazza a téziseimet. Az 1. és 2. tézis eredményeit a 3. fejezet, a 3. tézis eredményeit a 4. fejezet, a 4. tézis eredményeit az 5. fejezet tartalmazza.

DOI: 10.15477/SZE.MMTDI.2017.009

2. fejezet

Gyártósorok, ütemezési problémáik és megoldási módszereik

2.1. Gyártósorok ütemezési problémái

Az ipari termelés során egy termék előállításához sok esetben egy munkadarabon/félkész terméken kell egymás után különböző műveletek elvégezni (a műveletek elvégzési sorrendje ismert). Hasonló típusú termékeknél (pl. különböző típusú motoroknál) az elvégzendő műveletek és azok sorrendje is megegyezhet. Ilyen esetekben az egyes műveletek elvégzése történhet egy gyártósor mentén. A gyártósor mellett vannak a munkaállomások, ahol egy vagy több művelet gépi vagy emberi végrehajtása történik. A gyártósorra feltett munkadarabokon mindegyik állomáson elvégzik az előírt műveleteket, majd a munkadarab továbbhalad a gyártósoron a következő állomásra. Miután egy munkadarabot az utolsó állomáson is megmunkáltak, az elkészült termék lekerül a gyártósorról. Ha adott egy gyártósor és megmunkálásra váró munkadaraboknak egy halmaza, akkor a célunk annak a meghatározása, hogy az egyes munkaállomásokon milyen sorrendben munkáljuk meg a munkadarabokat, hogy valamilyen célfüggvény szerint az ütemezés optimális legyen. A következő két részben ezt az ütemezési feladatot megfogalmazzuk egy matematikai feladatként, majd felsorolunk néhány lehetséges célfüggvényt.

2.1.1. A flow shop probléma (FSP)

Ha a gyártósoron egy állomáson a megmunkálás során vagy után a munkadarabot leveszik a sorról és csak akkor rakják vissza amikor a következő állomásra küldik tovább, akkor előfordulhat, hogy egy i munkadarabot egy állomáson előbb kezdünk el megmunkálni, mint egy j munkadarabot, mégis a következő állomáson a j munkadarabot munkáljuk meg előbb, mint az i munkadarabot (”előzés”). Ebben az esetben az ütemezési feladat egy **flow shop**

DOI: 10.15477/SZE.MMTDI.2017.009

problémaként (FSP) fogalmazható meg.
A flow shop probléma: Adott M gép és N munkadarab. Mindegyik munkadarabot meg kell munkálni az összes gépen. A megmunkálások sorrendje minden munka esetén ugyanaz: minden munkát először az első gépen, majd a másodikon, stb. kell megmunkálni. A munkák megmunkálási sorrendje a különböző gépeken eltérő is lehet. A standard FSP a következő feltételeket tartalmazza:

- minden gép folyamatosan rendelkezésre áll a 0 időponttól kezdve;
- minden munkadarab rendelkezésre áll a 0 időpontban;
- egy gépen egyszerre csak egy munkadarab munkálható meg;
- egy munkadarab egyszerre csak egy gépen munkálható meg;
- a mundarabokat megszakítás nélkül kell megmunkálni a gépeken;
- két gép között tetszőleges számú munkadarab várakozhat;
- a munkadaraboknak ismert a gépeken való megmunkálási idejük;
- a gépek közt a munkadarabok szállítási idejét elhanyagoljuk vagy beleszámítjuk a megmunkálási időbe.

Mivel az egyes gépeken a munkadarabok sorrendje eltérhet, ezért az FSP-ben a feladat az, hogy minden gépre mondjuk meg, hogy milyen sorrendben munkáljuk meg rajta a munkadarabokat, és azt is, hogy mikor kezdjük el a munkadarabok megmunkálását az egyes gépeken. Mivel egy adott gépre $N!$ sorrendben rakhatjuk fel a munkadarabokat, és összesen M gép van, ezért az összes lehetséges ütemezések száma $(N!)^M$.

2.1.2. A permutációs flow shop probléma (PFSP)

Ha egy gyártósoron végighaladó munkadarabok sem a megmunkálások alatt, sem a megmunkálások során nem kerülnek le sorról, akkor a munkadarabok sorrendje minden munkaállomáson ugyanaz lesz. Ezen ütemezési feladathoz tartozó speciális FSP-t **permutációs flow shop problémának** (PFSP) nevezzük.
A permutációs flow shop probléma tehát egy olyan flow shop probléma, melyben mindegyik gépen ugyanolyan sorrendben munkáljuk meg a munkadarabokat. Mivel egy PFSP-ben a munkadarabok sorrendje minden gépen ugyanaz, ezért az ütemezéshez elegendő azt megmondani, hogy az első gépre milyen sorrendben rajuk fel a munkadarabokat, illetve azt, hogy az egyes gépeken mikor kezdjük el megmunkálni a munkadarabokat. Míg egy FSP-ben a lehetséges felrakások száma $(N!)^M$, addig egy PFSP-ben a lehetséges felrakások száma „csak” $N!$.

DOI: 10.15477/SZE.MMTDI.2017.009

A PFSP különféle variánsai

Az ipari termelésben sok esetben a 2.1.1. részben felsorolt feltételezések nem teljesülnek. Például előfordulhat, hogy bizonyos munkadarabok a termelés megkezdésekor nem állnak rendelkezésre, mert folyamatosan érkeznek be a gyárba. Ilyenkor azt mondjuk, hogy a j munkadarabnak egy r_j *rendelkezésre állási* ideje van, ami azt jelenti, hogy a j munkadarabot legkorábban az r_j időpontban kezdhetjük megmunkálni az első gépen. A klasszikus PFSP-ben feltesszük, hogy ha egy munkadarab megmunkálását befejeztük egy gépen és a következő gép üres, akkor a munkadarabot egyből megkezdhetjük megmunkálni a következő gépen. A valóságban azonban nem mindig ez a helyzet. Gondoljunk például arra, hogy egy gép befest egy munkadarabot, vagy valamiféle hőkezelést végez rajta. Ekkor a művelet befejezése után ahhoz, hogy a következő gépen elkezdhessük az adott munkadarab megmunkálását, az első esetben a festéknek meg kell száradnia, a második esetben pedig a munkadarabnak le kell hűlnie egy bizonyos hőmérsékletre. Ez azt jelenti, hogy a j munkadarabnak az r gépen való befejezési ideje és az $r + 1$ gépen való kezdési ideje között legalább egy bizonyos időnek el kell teltenie. Ezt az időt *késleltetési időnek* nevezik és l_{rj}-vel jelöljük. A gyártás során gyakran nem csak az a feladatunk, hogy legyártsuk az adott terméket, hanem az is, hogy azt az elkészülte után eljuttassuk a rendeltetési helyére (raktár, másik üzem, stb,). Az ehhez szükséges időt *szállítási időnek* hívjuk és a j munkadarab esetén q_j-vel jelöljük. Ebben az esetben a j munka "végleges elkészülése" alatt azt értjük, hogy a munkadarabot az utolsó gépen való megmunkálása után el is szállítjuk a rendeltetési helyére. Technológiai megfontolások miatt előfordulhat továbbá, hogy bizonyos termékeknek folyamatosan kell végimenniük a gépeken, tehát amint az egyik gépen befejeztük a megmunkálását, a következőn máris el kell kezdenünk őket. Ebben az esetben *várakozásmentes PFSP*-ről beszélünk. Szintén gyakori helyzet, hogy miután befejeztük egy gépen egy munkadarab megmunkálását, a következő munkadarab megmunkálásának kezdete előtt a gépet át kell állítani. Ezt az időt *átszerelési időnek* nevezik. Az egyes gépeken az átszerelési idő függhet a megmunkálás sorrendjétől is, ilyenkor *sorrendfüggő átszerelési időket tartalmazó PFSP*-ről beszélünk. Végül bizonyos esetekben megengedett, hogy egy munkadarabnak *megszakítsuk* a megmunkálását egy gépen, azaz a megmunkálást egy időpontban abbahagyjuk, majd egy későbbi időpontban innen folytatjuk a munkadarab megmunkálását.

2.1.3. Lehetséges célfüggvények

A permutációs flow shop problémákban használt célfüggvények a munkadarabok utolsó gépen való befejezési idejeinek valamilyen függvénye. A leggyakrabban használt célfüggvény az átfutási idő, ami az utolsó gépen az utoljára megmunkált munkadarab befejezési ideje, melyet minimalizálni szeretnénk. Az átfutási idő minimalizálásánál tulajdonképpen arra vagyunk kíváncsiak, hogy mi az a legkorábbi időpont, amire mindegyik munkadarab megmunkálható az

DOI: 10.15477/SZE.MMTDI.2017.009

összes gépen.

Egy másik lehetséges célfüggvény a munkadarabok utolsó gépen való befejezési ideinek összege, melyet szintén minimalizálni szeretnénk.

Előfordulhat, hogy minden munkadarabnak van egy határideje, amire az összes gépen meg kell munkálnunk. Ha a határidőhöz képest később fejezzük be az utolsó gépen a megmunkálást (”csúszás”), akkor kötbért kell fizetnünk (ami a csúszás idejével arányos), míg ha a határidőhöz képest előbb fejezzük be az utolsó gépen a megmunkálást (”sietés”), akkor tárolnunk kell az elkészült terméket, ami szintén pénzbe kerül (ami a sietés idejével arányos). Ennek megfelelően lehetséges célfüggvény a csúszások és sietések súlyozott összege, melyet szintén minimalizálni szeretnénk.

Brucker [13] megmutatta, hogy a flow shop problémának az átfutási idő, mint célfüggvény esetén mindig van olyan optimális megoldása, melyben az első két gépen megegyezik a munkák sorrendje és az utolsó két gépen is megegyezik a munkák sorrendje. Ebből következik, hogy két, illetve három gép esetén az FSP-nek az átfutási idő, mint célfüggvény esetén mindig van olyan optimális megoldása, melyben a munkák sorrendje mindegyik gépen azonos, tehát ilyenkor a keresési térben elegendő a permutációk halmazával foglalkoznunk. Azonban háromnál több gép esetén lehet mutatni olyan FSP-t, melynek az optimális megoldásában a munkák sorrendje nem minden gépen ugyanaz. Például legyen 4 gépünk és két munkánk. Az első munkának az egyes gépeken való megmunkálási ideje legyen rendre $(4,1,1,4)$, míg a második munkának az egyes gépeken való megmunkálási ideje legyen rendre $(1,4,4,1)$. Ekkor mindkét permutációhoz tartozó ütemezésnél az átfutási idő 14 lesz, míg az optimális ütemezésnél az átfutási idő 12. A két permutációhoz tartozó ütemezést és az optimális ütemezést a 2.1. ábra mutatja.

2.2. A PFSP-nek a szakirodalomból ismert megoldási módszerei

Mivel egy PFSP-ben a leggyakrabban az átfutási időt szeretnénk minimalizálni, ezért a dolgozat további részében PFSP alatt azt a feladatot értjük, melyben a cél az átfutási idő minimalizálása.

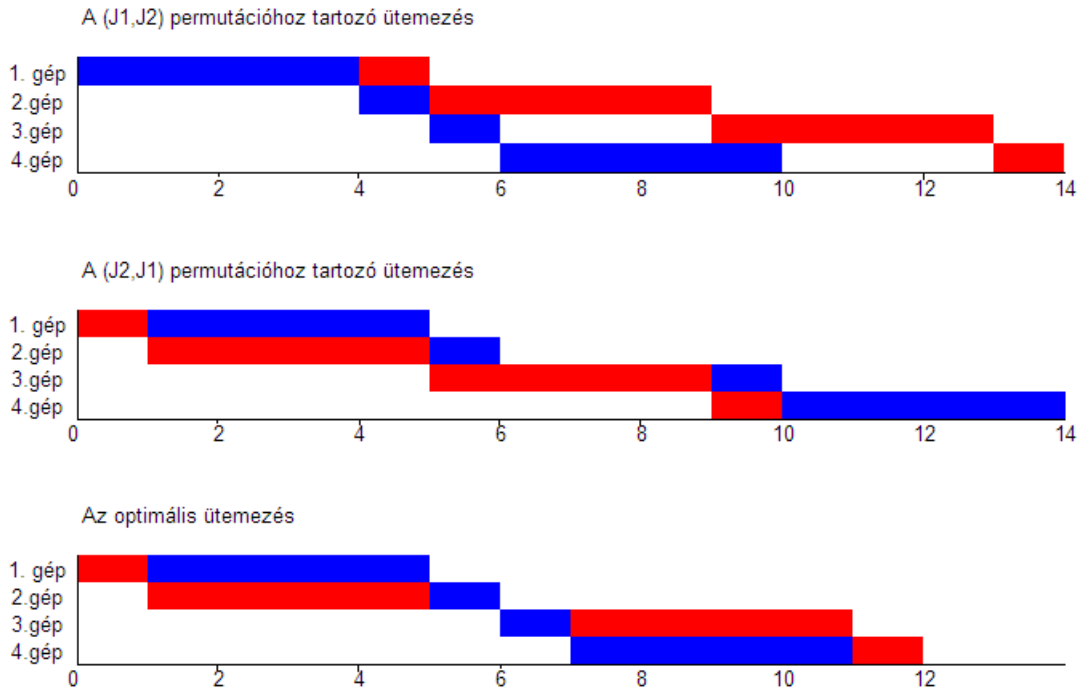
Két gép esetén a PFSP optimális megoldását **Johnson algoritmusával** [45] kaphatjuk meg. Jelölje $P_{1,i}$ illetve $P_{2,i}$ az i munka megmunkálási idejét az első, illetve a második gépen, és jelöljük továbbá $J_1,J_2,\ldots J_N$-nel az egyes munkadarabokat.

Johnson algoritmusa [45]

1. lépés Legyen $k := 1$ és $l := N$. Legyen továbbá $J = (J_1,J_2,\ldots,J_N)$ a nem ütemezett munkák halmaza.

2. lépés Keressük meg a $\{P_{1,i}|J_i \in J\} \bigcup \{P_{2,i}|J_i \in J\}$ halmaz egy minimális elemét, és

DOI: 10.15477/SZE.MMTDI.2017.009



2.1. ábra. Példa arra, hogy egy FSP optimális megoldásában a munkák sorrendje nem egyezik meg az összes gépen

legyen i a hozzá tartozó index. Ha a minimum nem egyértelmű, akkor válasszuk a legkisebb indexet.

3. lépés Ha a 2. lépésben választott i index esetén $P_{1,i} \leq P_{2,i}$ teljesül, akkor a J_i munkát tegyük a lista k. helyére, töröljük J_i-t a J listából, és k értékét növeljük eggyel. Ugorjunk az 5. lépésre.

4. lépés Ha a 2. lépésben választott i index esetén $P_{2,i} < P_{1,i}$ teljesül, akkor a J_i munkát tegyük a lista l. helyére, töröljük J_i-t a J listából, és l értékét csökkentsük eggyel. Ugorjunk az 5. lépésre.

5. lépés Ha a J lista üres, az algoritmus leáll. Ekkor megkaptuk a munkák optimális ütemezési sorrendjét. Ha a J lista nem üres, akkor ugorjunk a 2. lépésre.

Johnson algoritmusa tehát 2 gép esetén megadja a munkák optimális sorrendjét. Az optimális ütemezéshez azonban még azt is meg kell adnunk, hogy mikor kezdjük el megmunkálni az egyes munkadarabokat a gépeken. Az átfutási idő minimalizálása esetén ezt nemcsak kettő, hanem tetszőleges gépszám esetén rekurzív módon egyszerűen meghatározhatjuk. Legyen ugyanis $\pi = (\pi(1), \pi(2), ..., \pi(N))$ a munkák egy tetszőleges sorrendje egy M gépet tartalmazó PFSP-ben. Jelölje P_{ri} az i munka megmunkálási idejét az r gépen, $B_{r,i}$ pedig az i munka kezdési idejét az r gépen. Mivel a célunk az átfutási idő minimalizálása, ezért az adott sorrend-

hez tartozó optimális ütemezés esetén a következő egyszerű észrevételeket tehetjük: létezik olyan optimális ütemezés, melynél

- a permutáció első munkadarabja várakozási idő nélkül halad végig a gépeken, azaz amint befejeztük a sorrend első munkájának a megmunkálását egy gépen, azonnal elkezdhetjük megmunkálni a következő gépen;
- az első gépen nincsenek állóidők, azaz amint befejezzük egy munkadarab megmunkálását az első gépen, a sorrendben utána következő munkadarab megmunkálását azonnal elkezdhetjük az első gépen;
- a sorrend i-edik munkájának az r gépen való kezdési ideje egyenlő a sorrend i. munkájának az $(r-1)$ gépen való befejezési idejének és a sorrend $(i-1)$-edik munkájának az r gépen való befejezési idejének a maximumával.

Ezek alapján egy adott permutáció esetén az optimális ütemezésben a munkák kezdési idejei a következő rekurzió alapján számíthatók ki:

$$B_{1,\pi(1)} = 0 \tag{2.1}$$

$$B_{r,\pi(1)} = B_{r-1,\pi(1)} + P_{r-1,\pi(1)} \qquad (2 \leq r \leq M) \tag{2.2}$$

$$B_{1,\pi(i)} = B_{1,\pi(i-1)} + P_{1,\pi(i-1)} \qquad (1 < i \leq N)$$

$$B_{r,\pi(i)} = \max\left(B_{r,\pi(i-1)} + P_{r,\pi(i-1)}, B_{r-1,\pi(i)} + P_{r-1,\pi(i)}\right) \tag{2.3}$$

$$(2 \leq r \leq M; 2 \leq i \leq N)$$

2 gép esetén a PFSP Johnson algoritmusával $O(n \log n)$ időben megoldható. Ezzel szemben Garey, Johnson és Sethi [31]-ben belátta, hogy ha a gépek száma legalább három, akkor mind az FSP, mind a PFSP már NP-nehéz. A PFSP megoldási módszereit 2 nagy csoportra bonthatjuk. Az első csoportba tartoznak a különböző heurisztikák, melyek nagy méretű feladatok esetén is elfogadható időn belül adnak egy megoldást (a munkák egy permutációját), amely azonban nem feltétlenül optimális. A második csoportba az egzakt megoldó módszerek tartoznak. Ezek optimális megoldást adnak, de a PFSP nehézsége miatt (NP nehéz legalább 3 gép esetén) nagy méretű feladatok esetén nem várható, hogy az optimális megoldást elfogadható időn belül megkapjuk a segítségükkel. A következő részben bemutatjuk a kétféle megközelítésnek az irodalomból ismert legfontosabb módszereit.

2.2.1. Heurisztikák

A PFSP-t megoldó heurisztikák szintén két csoportba sorolhatók. A konstruktív heurisztikák valamilyen szabály alapján ”építik fel” a munkadarabok egy permutációját, míg a metaheurisztikák egy vagy több kiindulási permutáción próbálnak javítani.

DOI: 10.15477/SZE.MMTDI.2017.009

Konstruktív heurisztikák

Johnson [45] cikke után a PFSP megoldására számos konstruktív heurisztika született (lásd például [14, 25, 37, 76, 87]). Empírikus tapasztalatok [29, 88, 111] alapján sokáig **Nawaz, Enscore és Ham** [76]-ban ismertetett NEH heurisztikája számított a legjobb konstruktív heurisztikának. Mint majd a későbbi részben látni fogjuk, számos javító heurisztika használja kiinduló megoldásként a NEH által adott megoldást.

A NEH heurisztika:

1. lépés Minden i munkára számítsuk ki a $T_i = \sum_{r=1}^{M} P_{ri}$ össz-megmunkálási időt.

2. lépés A munkákat rendezzük egy L listába a T_i értékek szerinti csökkenő sorrendben.

3. lépés Az L lista első két eleméből képezhető 2 permutáció közül válasszuk ki azt, amelyikre az átfutási idő a kisebb. Jelölje Π_2 ezt a permutációt. Legyen $i = 3$.

4. lépés Az L lista i. elemét szúrjuk be a Π_{i-1} permutáció összes lehetséges helyére. Az így kapott i darab permutáció közül válasszuk ki azt, amelynél az átfutási idő a legkisebb. Legyen ez a permutáció Π_i.

5. lépés Ha $i = N$, akkor az algoritmus megáll és a Π_N permutációt adja vissza megoldásként. Ha $i < N$ akkor növeljük meg i értékét eggyel és ugorjunk a 4. lépésre.

A NEH heurisztika tehát egy kételemű permutációból indul ki. Minden lépés során a meglévő permutációt egy új elemmel bővíti oly módon, hogy az új elemet a meglévő permutáció összes lehetséges helyére beszúrja és az így létrejövő permutációkból kiválasztja a legjobbat, vagyis csak egyet tart meg közülük.

Chakraborty és Laha [18] eljárása alapvetően abban tér el a NEH heurisztikától, hogy az algoritmusnak van egy k paramétere, és az algoritmus mindig a legjobb k részpermutációt tárolja el. A szerzők tesztjei alapján a változtatás szignifikánsan javított a NEH hatékonyságán.

Chakraborty és Laha algoritmusa: [18]

1. lépés Minden i munkára számítsuk ki a $T_i = \sum_{r=1}^{M} P_{ri}$ össz-megmunkálási időt.

2. lépés A munkák rendezzük egy L listába a T_i értékek szerinti csökkenő sorrendben.

3. lépés A sorrend első négy eleméből készítsük el a 24 darab részpermutációt, és válasszuk ki közülük a legjobb k darabot.

4. lépés Legyen i=5.

5. lépés A sorrend i. elemét szúrjuk be az aktuális k darab részpermutáció minden lehetséges helyére (így kapunk $k \cdot i$ permutációt). Vegyük ezen $k \cdot i$ permutáció közül a k legjobbat, vagyis azokat, melyeknél az átfutási idő a legkisebb.

DOI: 10.15477/SZE.MMTDI.2017.009

6. lépés Ha $i = N$, akkor az algoritmus megáll és az aktuális k darab permutáció közül a legjobbat adja vissza megoldásként. Ha $i < N$ akkor növeljük meg i értékét eggyel és ugorjunk az 5. lépésre.

Kalczynski és Kamburowski [46] 2008-ban a NEH heurisztika módosításával egy új, a NEH-nél jobb eredményeket adó konstruktív heurisztikát mutattak be. Eljárásuk abban különbözik a NEH-től, hogy a NEH 2. lépésében a munkák listába rendezéséhez egy új szabályt használ, illetve a NEH 4. lépésében a legjobb részpermutáció kiválasztásakor holtverseny esetén szintén egy új szabályt használ.

Meta-heurisztikák

A PFSP megoldására használt meta-heurisztikák a kombinatorikus optimalizálás számos módszerét ölelik fel. Ebben a részben áttekintjük a legfontosabb eljárásokat.

Szimulált hűtés

A szimulált hűtés módszer egy feladatfüggetlen eljárás. Az általános módszer kiindul a feladat egy megoldásából, majd véletlenszerűen kiválasztja az aktuális megoldás egy ”szomszédját”. Ha ez jobb az aktuális megoldásnál, akkor ez lesz az új aktuális megoldás. Ha a kapott megoldás nem jobb az aktuális megoldásnál, akkor a kapott megoldástól, az aktuális megoldástól és egy T paramétertől függő valószínűséggel fogadjuk el az új megoldást aktuális megoldásnak. Ezt az eljárást ismételjük addig, amíg valamilyen megállási kritérium nem teljesül. Az algoritmus során a T paraméter értéke (a hőmérséklet értéke) időnként csökken. A hőmérséklet használatának a célja az, hogy - mivel kis valószínűséggel egy rosszabb megoldásra lépünk - elkerüljük azt, hogy bennragadjunk egy lokálisan optimális megoldásban. A [17, 53, 59, 62, 74, 77] cikkekben a szimulált lehűtés módszerét alkalmazták a PFSP különféle változataira.

Iterált mohó algoritmus

Ruiz és Stützle [90] iterált mohó algoritmusa a NEH heurisztika stratégiáját követi. Az eljárásuk a NEH heurisztika által adott permutációból indul ki. Az algoritmus minden lépése egy destrukciós és egy konstrukciós fázisból áll. A destrukciós fázis során az aktuális megoldásból véletlenszerűen kiválasztunk d elemet és ezeket elhagyjuk a permutációból. Így kapunk egy π_D (d elemű) és egy π_R ($n - d$) elemű permutációt. A konstrukciós fázis során a π_D permutáció elemeit a NEH heurisztika szerint egyesével visszatesszük a π_R permutációba. Így kapjuk meg az új permutációt. A hatékonyság növelésének érdekében minden egyes konstrukciós fázis után a kapott új permutációra egy lokális keresést alkalmazunk. A lokális keresés a következőképpen működik:

DOI: 10.15477/SZE.MMTDI.2017.009

lokális keresés(π):

1. lépés Válasszunk véletlenszerűen egy k indexet, és permutáció k. helyén lévő elemet szúrjuk be a π permutáció összes lehetséges, k-tól különböző helyére.

2. lépés Válasszuk ki az így kapott $n-1$ permutáció közül a legjobbat. Legyen ez $\pi^{'}$.

3. lépés Ha a $\pi^{'}$ permutációhoz tartozó átfutási idő kisebb, mint a π-hez tartozó átfutási idő, akkor legyen $\pi := \pi^{'}$.

4. lépés Ha az utolsó n darab, egymást követő 3. lépés során a π permutáció nem változott, akkor a keresés megáll és a π permutációt adja vissza. Ellenkező esetben ugorjunk a 2. lépésre.

Ezek után az iterált mohó algoritmus működése a következőképpen foglalható össze:

1. lépés Legyen π a NEH heurisztika által adott permutáció..

2. lépés Alkalmazzuk π-re a lokális keresést. Legyen π a lokális kereséssel kapott permutáció.

3. lépés Legyen a legjobb megoldás π_b egyenlő π-vel.

4. lépés (Destrukciós fázis) Véletlenszerűen válasszunk ki d elemet és hagyjuk el őket π-ből. Jelölje π_R a létrejött $n-d$ elemű permutációt és π_D a létrejött d elemű permutációt.

5. lépés (Konstrukciós fázis) A π_D permutáció elemeit a NEH heurisztika szerint tegyük vissza egyesével a π_R permutációba. Legyen $\pi^{'}$ az így kapott permutáció.

6. lépés Alkalmazzunk $\pi^{'}$-re egy lokális keresést. A kapott permutáció legyen $\pi^{''}$.

7. lépés Ha $\pi^{''}$ átfutási ideje kisebb, mint π átfutási ideje, akkor legyen $\pi = \pi^{''}$ ($\pi^{''}$ lesz az új aktuális megoldás). Ellenkező esetben $\exp\{-(C_{\max}(\pi^{''}) - C_{\max}(\pi))/T\}$ valószínűséggel legyen $\pi = \pi^{''}$.

8. lépés Ha π átfutási ideje kisebb, mint π_b átfutási ideje, akkor legyen $\pi_b = \pi$.

9. lépés Ha a megállási kritérium teljesül, akkor az algoritmus megáll és a π_b permutációt adja vissza megoldásként. Különben ugorjunk a 4. lépésre.

Az algoritmusban használt T hőmérséklet az adott feladat méretétől és a megmunkálási időktől függ, nevezetesen

$$T = a\frac{\sum_{r=1}^{M}\sum_{i=1}^{N} p_{ri}}{N \cdot M \cdot 10} \tag{2.4}$$

DOI: 10.15477/SZE.MMTDI.2017.009

ahol a egy állítható paraméter ($0 < a < 1$).

Tabu keresést használó heurisztikák

A tabu keresés egy általános eljárás, melyet sikerrel alkalmaztak számos nehéz, kombinatorikus optimalizásási feladat közel optimális megoldásának meghatározására [34, 35]. A tabu keresésnél definiálnunk kell a *lépés* fogalmát. Lépés alatt egy olyan függvényt értünk, mely egy megoldáshoz egy másik megoldást rendel hozzá. Egy adott megoldás *szomszédainak* halmazán azon megoldások halmazát értjük, melyekhez eljuthatunk az adott megoldásból 1 lépés segítségével. A tabu keresés egy kiinduló megoldásból indulva mindig az aktuális megoldáson próbál javítani, úgy, hogy a megoldás szomszédjaiból (vagy azok egy alkalmas részhalmazából) kiválasztja a legjobb megoldást, és az eljárást ezzel a megoldással folytatja tovább. Annak elkerülése érdekében, hogy az eljárás során visszatérjünk egy már korábban vizsgált megoldáshoz, az algoritmus fenntart egy *tabu listát*. Minden egyes alkalommal, amikor végrehajtunk egy lépést, a lépés különféle attribútumait egy bizonyos ideig eltároljuk ebben a listában. A tabu listában található lépések a tiltott lépések. A nevével ellentétben az eljárás során tiltott lépést is végrehajthatunk abban az esetben, ha a lépés valamilyen értelemben ”profitábilis”. A tabu keresés során többféle megállási kritériumot is használhatunk. A leggyakrabban a futásidőre, az iterációk számára, illetve a javulást nem hozó iterációk számára vonatkozó korlátokat szokás használni megállási kritériumként. A [1, 27, 36, 69, 75, 78, 103, 113] cikkekben a tabu keresés heurisztikát alkalmazták PFSP-kre.

Nowicki és Smutnicki tabu keresést használó algoritmusa [78]

A PFSP-ben a megoldások a permutációk. Lépés alatt egy $v = (a,b)$, $1 \leq a,b \leq N$, $a \neq b$ párt értünk, ahol a és b a permutáció két különböző indexe. Egy π permutáción a $v = (a,b)$ lépés végrehajtása azt jelenti, hogy a permutáció $a.$ helyén álló munkát kivesszük az $a.$ helyről és betesszük a $b.$ helyre. Például a $(2,4,5,3,1)$ permutáción a $(4,2)$ lépést végrehajtva a $(2,3,4,5,1)$ permutációhoz jutunk. Mivel az (a,a) lépések nem megengedettek ezért összesen $n \cdot (n-1)$ lépés van. Könnyen látható azonban, hogy az $(a-1,a)$, illetve az $(a,a-1)$ lépéseket egy π permutáción végrehajtva ugyanahhoz a permutációhoz jutunk, ezért egy π permutáció összes szomszédjának a száma $n \cdot (n-1) - (n-1) = (n-1)^2$. Werner [112]-ben belátta, hogy bizonyos típusú (a,b) lépések végrehajtásával nem kaphatunk jobb permutációt. Azért, hogy az eljárás során a szomszédok közti keresésre fordított időt csökkentsék, Nowicki és Smutnicki tovább szűkítette egy adott permutáció szomszédainak a számát. Ezt úgy érték el, hogy tesztek során arra a megállapításra jutottak, hogy csak speciális tulajdonságó (a,b) lépésekkel érdemes foglalkozni. Ezt felhasználva a lehetséges lépések, és így a szomszédok számát sikerült jelentősen csökkenteniük. A szerzők azt is megmutatták, hogy a definiált

lépéseknek megvan az a tulajdonsága, hogy tetszőleges permutációból kiindulva véges sok lépés segítségével eljuthatunk egy optimális megoldáshoz.

A tabu lista: A tabu lista egy $T = (T_1,T_2,\ldots,T_{maxt})$ halmaz, ahol $maxt$ az algoritmus egy paramétere, és minden T_i munkáknak egy (g,j) párját jelöli. Az algoritmus kezdetén $T_i = (0,0)$ minden $1 \leq i \leq maxt$ esetén. Amikor egy (a,b) lépést hajtunk végre az aktuális permutáción, azaz $\pi(a)$ munkát kivesszük az $a.$ helyről és beszúrjuk a $b.$ helyre, akkor egyrészt a tabu lista elemeit eltoljuk balra, azaz $1 \leq j < tmaxt$ esetén $T_j := T_{j+1}$ lesz, továbbá a lista utolsó eleme $T_{maxt} := (\pi(a),\pi(a+1))$ lesz, ha $a < b$, különben pedig $T_{maxt} := (\pi(a-1),\pi(a))$ lesz. Az eltolás hatására a tabu lista korábbi első eleme kikerül a listáról.

Azt mondjuk, hogy egy π permutációra nézve egy (a,b) lépés akkor lesz tiltott, ha $a < b$ esetén a $(\pi(j),\pi(a))$, $j = a+1,\ldots,b$ párok legalább egyike eleme a T tabu listának, míg $a > b$ esetén pedig a $(\pi(a),\pi(j))$, $j = b,\ldots,a-1$ párok közül legalább az egyik a tiltó listában van.

A keresési stratégia: Az algorimus először minden a indexre meghatározza a lehetséges (a,b) lépések (a szerzők által leszűkített) halmazából a legjobb (a,b_1), $a < b_1$ és (a,b_2), $a > b_2$ lépést. Ezáltal kapunk legfeljebb $2(n-1)$ darab lépést. Ezen lépéseket három részre osztjuk: nem tiltott lépések; tiltott, de nyereséges lépesek; tiltott és nem nyereséges lépések. Egy v lépésről akkor mondjuk, hogy nyereséges, ha $C_{\max}(\pi_v) < F(C_{\max}(\pi))$ ahol az F függvényt az algoritmus során folyamatosan módosítjuk. Ha létezik nem tiltott lépés vagy tiltott, de nyereséges lépés, akkor az összes ilyen lépés közül kiválasztjuk a legjobbat, végrehajtjuk a lépést és lépésnek megfelelő párt a tiltó listára tesszük. Ha mindegyik lépés tiltott és nem nyereséges, akkor a $(0,0)$ párt a tiltó listára tesszük (miáltal a legrébbi pár eltűnik a tiltó listáról) és újra próbálkozunk.

Ezek után készen állunk **Nowitzki és Smutnicki algoritmusának** ismertetésére: induljunk ki egy π kezdeti permutációból (ez lehet például a NEH által meghatározott permutáció), a T tiltó lista legyen üres, legyen továbbá az F függvény azonosan végtelen. A fenti keresési stratégiának megfelelően minden iterációban megkeressük a legjobb v lépést, ezt végrehajtva kapjuk a π_v permutációt. A v lépésnek megfelelően módosítva a T tiltó listát kapjuk a T' tiltó listát. Az F függvényt a következők szerint módosítjuk: a $C = C_{\max}(\pi)$ és $C' = C_{\max}(\pi_v)$ jelöléseket használva legyen $F(C) := \min\{F(C),C'\}$ és $F(C') := \min\{F(C'),C\}$. Ha a π_v permutáció jobb az eddig ismert legjobbnál, akkor kicseréljük a legjobb permutációt π_v-re. A következő iterációt a π_v permutációval és a T' tiltó listával kezdjük el. Az algoritmus mindig eltárolja az addigi legjobb permutációt, a hozzá tartozó lehetséges lépésekkel és tiltó listával együtt. Ha egy bizonyos iterációszám alatt az ismert legjobb átfutási idő nem csökken, akkor visszalépünk a legjobb permutációra. A legjobb permutáció szomszédai közül elhagyjuk a rajta végrehajtott lépést, majd előlről kezdjük az eljárást a legjobb permutációval és a neki megfelelő tiltólistával (hideg újraindítás). Az algoritmus akkor áll le, ha vagy egy bizonyos iterációszámot (mely az algoritmus paramétere) túllépünk, vagy többszöri hideg újraindítás

után a lehetséges lépesek halmaza üressé válik.

Genetikus algoritmust használó heurisztikák

A genetikus algoritmusokat sikerrel alkalmazták számos nehéz, kombinatorikai feladat (pl. utazó ügynök probléma) közelítő megoldására. Az algoritmus népszerűsége viszonylagos egyszerűségében rejlik. A genetikus algoritmusok az állat és növényvilágban a fajoknak Darwin által leírt evolúcióját „kódolva" jutnak el az adott probléma egy "jó" megoldásához. A genetikus algoritmus egy populációból indul ki, melynek egyedei a gének (ezek az adott probléma megoldásai), a gének részei pedig a kromoszómák. A populáció fejlődéséhez az algoritmus vagy a populáció két egyedére (2 génre) alkalmaz egy keresztezést (más néven rekombinációt), vagy egyik egyedére alkalmaz egy mutációt (megváltoztaztatja egy gén (azaz egy megoldás) szerkezetét). A létrejövő új egyedekről aztán a kiválasztás során, többnyire egy fittségi függvény segítségével, eldöntjük, hogy melyek lesznek közülük az új populáció egyedei. Genetikus algoritmusokat használtak PFSP-k megoldására például a [19, 21, 75, 89, 116] cikkekben.

Hangya kolónia algoritmusok

Az első, hangya kolónia (ACO) algoritmust Dorigo és Gambardella [26] ismertette. Cikkükben az új eljárást az utazó ügynök feladat egy közel-optimális megoldásának megtalálására használták fel. Azóta az ACO algoritmusokat számos nehéz optimizálási feladatra, köztük a PFSP-re is sikeresen alkalmazták. Az algoritmus azt a módszert próbálja utánozni, ahogy a hangyák kommunikálnak egymással, miközben mondjuk egy kiindulási pontból próbálnak a legrövidebb úton eljutni az élelemig. Útjuk során a hangyák egy bizonyos illatanyagot, úgynevezett feromont hagynak maguk után. A feromon intenzitása az idővel csökken, így ha egy hangya egy rövid utat talál meg a kezdő és a végpont között, akkor ezen út mentén hosszabbb ideig lesz érezhető az általa lerakott illatanyag, következésképpen a többi hangya is nagyobb valószínűséggel választja ezt az utat. Ráadásul minél többen választanak egy adott utat, annál nagyobb lesz a feromon intenzitása az adott úton.

Az utóbbi évtizedekben a PFSP megoldására is számos hangya kolónia algoritmus született [2, 20, 82, 115, 117]. Az algoritmusok egy részében több hangya, egy részében pedig egyetlen hangya szerepel. Az egy hangya használatának az előnye, hogy az algoritmus könnyen implementálható, és minden iterációban a hangya által kapott megoldásra egyszerűen alkalmazhatunk egy javító technikát, például egy lokális keresést. Egyetlen hangya használatának a hátránya viszont, hogy az iterációszám jóval nagyobb lehet, mintha hangya populációt használnánk. Az egyetlen hangyát használó ACO algoritmusokban elsőként meg kell adni a kiindulási feromonok értékét. Ezt követően minden iterációban generálunk egy megoldást, melyen aztán egy lokális kereséssel próbálunk javítani. Ezután a kapott megoldásnak megfe-

lelően módosítjuk a feromonok értékét. Végezetül a megállási kritériumok teljesülése esetén az algoritmus a futása alatt talált legjobb megoldást adja vissza megoldásként. A következőkben részletesesen ismertetjük Rajendran és Ziegler egyik, egy hangyát használó ACO algoritmusát a PFSP-re.

Rajendran és Ziegler [82] PACO algoritmusa a PFSP-re

Az ismertetésre kerülő algorimusban τ_{ik} fogja jelölni annak a feromonnak az inttenzitását, hogy az i munkát a sorrend k. helyére tesszük. A PACO algoritmus röviden a következőképpen foglalható össze:

PACO algoritmus:

1. lépés Vegyük a NEH algoritmus által meghatározott permutációt, majd erre háromszor alkalmazzunk egy munka alapú lokális keresést. A kapott sorrend lesz a kezdeti megoldás.

2. lépés Inicializáljuk a kezdeti paramétereket és a τ_{ik} feromon nyomokat.

3. lépés Konstruáljunk egy új permutációt.

4. lépés Javítsunk rajta egy munka alapú lokális keresés háromszori alkalmazásával

5. lépés Módosítsuk a feromon értékeket.

6. lépés Ha a megállási kritérium teljesül, akkor megoldásként adjuk vissza a talált permutációk közül a legjobbat. Egyébként ugorjunk a 3. lépésre.

Az algoritmus tehát minden egy iterációban, az aktuális feromon értékek, azaz a τ_{ik}-k alapján konstruál egy új permutációt. Nézzük most az egyes lépéseket részletesebben.

A munka alapú lokális keresés a következőképpen működik. Vegyük először az 1-es munkát (tehát nem a permutáció első elemét!), hagyjuk el a permutációból, és szúrjuk be a maradék $n-1$ hely mindegyikére. Az így kapott $n-1$ darab permutáció közül válasszuk ki a legjobbat (azaz a legkisebb $C_{\max}$ értékűt). Ha ez jobb, mint az eredeti permutáció, akkor ez lesz az új permutáció. Ezzel az új permutációval folytassuk az eljárást a $2,3,\ldots,n$ munkákkal.
A kezdeti feromon értékeket a következőképpen állítjuk be: Legyen minden (i,k) párra

$$\tau_{ik} = \begin{cases} 1/Z_{best}, & \text{ha } |\text{az } i \text{ munka indexe a kezdeti sorrendben} - k| + 1 \leq n/4; \\ 1/(2Z_{best}), & \text{ha } n/4 < |\text{az } i \text{ munka indexe a kezdeti sorrendben} - k| + 1 \leq n/2; \\ 1/(4Z_{best}), & \text{egyébként.} \end{cases}$$

Ennek az inicializálásnak az a magyarázata, hogy mivel a kiindulási permutáció már viszonylag jó megoldása a feladatnak, ezért egy i munkát próbaljuk olyan k helyre tenni, azaz azon τ_{ik} feromon értékek leszenek nagyok, melyeknél k közel van i-nek a kezdeti permutációbeli helyéhez.

Új permutáció konstruálása: az üres permutációból kiindulva mindig egy új munkát teszünk az aktuális nem teljes permutáció végére. Ha már az első $k-1$ helyet feltöltöttük, akkor a következőléppen választjuk ki a k. helyre kerülő munkát: minden i még nem ütemezett munkára számítsuk ki a $T_{ik} := \sum_{q=1}^{k} \tau_{ik}$ értéket, generáljunk a $[0,1]$ intervallomon egy u véletlen számot. Ha $u \leq 0.4$, akkor legyen a jelenlegi legjobb sorrend legelső, még nem ütemezett munkája a sorrend k. tagja.
Ha $0.4 < u \leq 0.8$, akkor a jelenleg ismert legjobb sorrendből a még nem rendezett munkák közül az 5 legelőrébb állóból válasszuk ki azt, amelyiknek a T_{ik} értéke maximális, és legyen ez az új sorrend k. eleme.
Ha $0.8 < u$ akkor a jelenleg ismert legjobb sorrendből a még nem rendezett munkák közül az 5 legelőrébb állóból véletlenszerűen választunk egyet a T_{ik} értékekkel arányos valószínűséggel, és ez lesz az új sorrend következő eleme.

A feromon értékeket az alábbiak szerint módosítjuk:
Ha a munkák száma legfeljebb 40, akkor legyen

$$\tau_{ik}^{uj} = \begin{cases} 0{,}75 \cdot \tau_{ik}^{regi} + 1/(diff \cdot Z_{best}), & \text{ha } |h-k| \leq 1; \\ 0{,}75 \cdot \tau_{ik}^{regi} & \text{különben.} \end{cases}$$

Ha munkák száma legalább 41, akkor legyen

$$\tau_{ik}^{uj} = \begin{cases} 0{,}75 \cdot \tau_{ik}^{regi} + 1/(diff \cdot Z_{best}), & \text{ha } |h-k| \leq 2; \\ 0{,}75 \cdot \tau_{ik}^{regi} & \text{különben.} \end{cases}$$

A fenti képletekben $diff$=(|az i munka helyének az indexe a jelenlegi legjobb permutációban$-k|+1)^{1/2}$.
A PACO algoritmus 40-szer generál új permutációt, majd ezek közül kiválasztja a legjobbat. Az így megkapott permutációra azoban még alkalmazunk egy lokális cserén alapuló keresést. Ennek során először vesszük az 1-es munkát (nem a sorrend első munkáját), és felcseréljük a permutációban mindegyik munkával. kiválasztjuk az eredeti és ezen $n-1$ permutáció közül a legjobbat. Ezután az eljárást folytatjuk ezzel a permutációval és a 2-es munkával, majd a 3-assal, és így tovább. A Paco algoritmus az eljárás végén kapott permutációt adja vissza megoldásként.

DOI: 10.15477/SZE.MMTDI.2017.009

Hibrid meta-heurisztikák

Optimalizálási feladatok esetén gyakori módszer az ismert eljárások ”vegyítése”. Az íly módon létrejövő meta-heurisztikáktól általában az várható, hogy együttesen jobb eredményt adnak, mint az alapját képező heurisztikák. Ilyen hibrid metaheurisztikákat alkalmaztak például a PFSP-re a [28, 70, 83, 119].

Raj-részecske optimalizáló heurisztikák

Az utóbbi időben a Kennedy és Eberhart [47] által kifejlesztett **raj-részecske optimalizáló** (angolul particle sworm optimization, PSO) heurisztikát is sikerrel alkalmazták a PFSP-re. A PSO ötlete, csakúgy mint a hangya kolónia heurisztikáé, az állatvilágból ered: egy madárraj mozgását próbálja utánozni, ahogy együttesen megtalálják az élelmet. A heurisztika annyiban rokonságot mutat a genetikus algoritmusokkal is, hogy itt is van egy populációnk (ez a raj), mely a probléma lehetséges megoldásaiból áll (ezek felelnek meg madárraj egyedeinek). A genetikus algoritmusokkal ellentétben azonban a PSO-ban nincsen sem mutáció, sem keresztezés. A madárraj minden egyede minden iterációban meghatározza a repülési irányát, méghozzá egyrészt a saját legjobb megoldásának, másrészt a raj által ismert legjobb globális megoldásának a függvényében, majd minden madár az aktuális megoldásról ”átrepül” egy másik megoldásra. Ezután minden madár frissíti a saját legjobb megoldását, illetve a raj legjobb megoldását is frissítjük. Az eljárást bizonyos megállási feltétel teljesüléséig folytatjuk. A PSO-t alkalmazták a PFSP-re többek között a [56, 57, 105] cikkekben.

2.2.2. Egzakt módszerek

A PFSP optimális megoldásának megtalálására az egyik lehetőség a korlátozás és szétválasztás (B&B) típusú módszerek használata. A B&B módszerek általánosan a következőképpen működnek. Legyen H egy véges halmaz, és tegyük fel, hogy a H halmazon értelmezve van egy valós értékű F függvény. Meg szeretnénk találni H-nak azt az x elemét, melyre $F(x)$ minimális, tehát az F függvényt akarjuk minimalizálni a H halmazon. Bontsuk fel a H halmazt a $H_1, H_2, \ldots H_l$ diszjunkt részhalmazokra, és minden H_i részhalmaz esetén állítsuk elő a H_i-n értelmezett F függvénynek egy L_i alsó korlátját. L_i tehát minden i esetén egy olyan szám, melyre $L_i \leq F(x)$ teljesül minden $x \in H_i$ esetén. Ha (valahonnét) ismerjük a H halmaz egy olyan y elemét, melyre $F(y) < L_i$ valamely i indexre, akkor ez azt jelenti, hogy az F függvény a minimumát biztosan nem veheti fel a H_i halmazon, hiszen a definíció alapján minden $x \in H_i$ elemre $F(y) < L_i \leq F(x)$ teljesül. Vagyis a továbbiakban a H_i részhalmazzal már nem kell foglalkoznunk, elhagyhatjuk a keresési térből. Ha nincs ilyen H_i halmaz, akkor valamilyen szabály alapján bontsuk szét valamelyik H_i halmazt diszjunkt részhalmazokra. Arra, hogy melyik halmazt bontjuk kisebb részekre, több szabály is használható. Az egyik lehetőség például, hogy a legkisebb alsó korláttal rendelkező halmazt bontjuk tovább kisebb részekre. Az algorit-

mus akkor áll meg, ha találunk egy olyan $y \in H$ elemet, melyre az aktuális keresési fa összes leveléhez tartozó H_i halmazokra $F(y) \leq L_i$ tejesül. Azt lehet tehát mondani, hogy a B&B algoritmus mindig a keresési teret pásztázza végig és az alsó korlátok segítségével próbálja a keresési teret egyre szűkíteni.
Egy másik lehetséges módszer a PFSP optimális megoldásának a megtalálására, hogy a PFSP-t megfogalmazzuk mondjuk egy (vegyes) egészértékű lineáris programozási feladatként, és az így felírt matematikai modellt megoldjuk egy alkalmas szoftverrel (pl. CPLEX, GUROBI, SCIP). Egy harmadik lehetőség ha a PFSP-t egy korlátozás programozási feladatként írjuk fel és a felírt modellt megoldjuk egy alkalmas szoftverrel (CPLEX, AIMMS) vagy a korlátozás programozás módszerével. Mindegyik módszerre számos példa található. B&B típusú eljárásokat alkalmaztak a PFSP-re a [3, 7, 12, 15, 22, 43, 51, 52, 61, 66, 67, 81, 102] cikkekben. Vegyes egészértékű lineáris programozási feladat segítségével oldottak meg PFSP-t a [80, 85, 94, 95, 96, 98, 101, 110, 114, 118] cikkekben. A [32, 50, 54, 92, 93] cikkekben a flow shop feladatnál általánosabb job shop feladatot oldották meg korlátozás programozási technikákkal. Beck et al. [6]-ban a korlátozás programozást a lokális kereséssel vegyítő eljárást adtak a job shop feladatra. Malapert et al. [64]-ben a job shop feladatnál általánosabb open shop feladatra adtak egy korlátozás programozási technikát használó eljárást.

Mint korábban említettük, a PFSP már 3 gép esetén is NP-nehéz, ami azt jelenti, hogy egyik módszer esetében sem várhatjuk, hogy a módszer segítségével tetszőleges feladatnak elfogadható időn belül megkapjuk az optimális megoldást. Ennek ellenére mindkét módszert érdemes tanulmányozni, hiszen mind a számítógépek kapacitása és gyorsasága, mind a MILP/CP modelleket megoldó szoftverek hatékonysága rohamosan fejlődik. Például a párhuzamosítás lehetőségét kihasználva [48]-ban és [49]-ben párhuzamos B&B algoritmust adtak a PFSP-re.

B&B módszerek a PFSP-re

A B&B típusú módszerek egyik kritikus pontja az alsó korlát keresésének a technikája. Egy erős alsó korlát megtalálása lehet, hogy sok időt vesz igénybe, viszont a segítségével esetleg a keresési tér nagyobb részétől szabadulhatunk meg, mint egy gyengébb korláttal. A következőkben bemutatjuk, hogyan lehet a PFSP esetén az átfutási időre alsó korlátokat adni.

Alsó korlátok konstruálása a PFSP-re [51]

Tegyük fel, hogy egy PFSP-nél a k. gép kivételével az összes többi gépnél megengedjük, hogy egyszerre több munkát is megmunkáljon. Ekkor ez azt jelenti, hogy $1 < k$ esetén az összes munkát elkezdhetjük az első gépen a 0 időpontban, és egy j munkát a $(k-1)$. gépen $\sum_{r=1}^{k-1} P_{rj}$ időpontra be is fejezzük. Továbbá ha a j munkát a k. gépen C_{kj}-re fejezzük be, akkor a j munkát az utolsó gépen a $C_{kj} + \sum_{r=k+1}^{M} P_{rj}$ időpontban fejezzük be. Ez azt jelenti, hogy a PFSP-nek ez a relaxációja egy olyan 1 gépes FSP-nek felel meg, melyben minden j munkának

van egy r_{kj} rendelkezésre állási ideje (ennél korábban nem lehet elkezdeni a gépen) és egy q_{kj} szállítási ideje (a megmunkálás után ennyi idő szükséges a munkának a rendeltetési helyére szállításához), ahol

$$r_{kj} = \begin{cases} \sum_{r=1}^{k-1} P_{rj} & \text{ha } k > 1 \\ 0 & \text{ha } k = 1, \end{cases} \tag{2.5}$$

és

$$q_{kj} = \begin{cases} \sum_{r=k+1}^{M} P_{rj} & \text{ha } k < M \\ 0 & \text{ha } k = M, \end{cases} \tag{2.6}$$

továbbá a munkákat folyamatosan kell megmunkálni (azaz megszakítás nem engedélyezett). Az ilyen típusú feladatokat szokás $1|r_j,q_j|C_{\max}$ formával jelölni. Ennek a feladatnak az optimális megoldása minden egyes k esetén a PFSP egy alsó korlátja lesz. Sajnos azonban ez a feladat még mindig NP nehéz [60]. Azonban ezt a feladatot tovább relaxálva alsó korlátokat kaphatunk a PFSP optimumára.

Az első lehetőség, hogy az összes rendelkezési álló időt, illetve az öszes szállítási időt is ugyanannyinak vesszük, nevezetesen minden j-re legyen $r_{kj} = \min_{j\in J} r_{kj}$ és $q_{kj} = \min_{j\in J} q_{kj}$. Könnyen látható, hogy ebben az estben a munkák bármely sorrendje optimális megoldást ad, és az optimum ($C_{\max}$) értéke, mely egyben az eredeti PFSP egy alsó korlátja lesz:

$$LB1_k = \min_{j\in J} r_{kj} + \sum_{j\in J} P_{kj} + \min_{j\in J} q_{kj}. \tag{2.7}$$

Mivel így minden k esetén a PFSP egy alsó korlátját kapjuk, ezért az

$$LB1 = \max_{1\le k\le m} LB1_k \tag{2.8}$$

a PFSP egy alsó korlátját adja. Mivel $LB1$ kiszámításához minden gépen meg kell határozni a megmunkálási idők összegét, továbbá ki kell számítani az $r_{k,j}$ és $q_{k,j}$ értékeket, ezért az $LB1$ korlát $O(mn)$ időben kiszámítható. Az $LB1$ korlátot használják a [12, 15, 43, 61, 66, 67]-beli cikkek.

Egy másik lehetőség, ha az $1|r_j,q_j|C_{\max}$ feladatban csak a q_{kj} szállítási időket tesszük egyenlővé a $q_{kj} = \min_{j\in J} q_{k,j}$ módon. Ebben az esetben a feladat optimuma, ami egyúttal a PFSP egy alsó korlátja, az

$$LB2_k = C^k_{\max}(J) + \min_{j\in J} q_{kj} \tag{2.9}$$

képlettel számítható ki, ahol $C^k_{\max}(J)$ a k. gépen definiált $1|r_j|C_{\max}$ feladat optimumát jelöli. Ez az optimum pedig a Jackson szabállyal [44] polinomiális időben kiszámítható. A Jackson szabály szerint az $1|r_j|C_{\max}$ feladat optimális megoldását adja a munkáknak a rendelkezésre állási idő szerinti növekvő sorrendbe való rendezése. Felhasználva, hogy az $LB2_k$ értékek

DOI: 10.15477/SZE.MMTDI.2017.009

minden k esetén a FPSP egy alsó korlátját adják, a PFSP egy alsó korlát kaphatjuk a következő módon:

$$LB2 = \max_{1 \leq k \leq m} LB2_k \tag{2.10}$$

Az $LB2$ alsó korlát kiszámítása során a munkákat minden gép esetén sorba kell rendezni a rendelkezésre álló idejük szerint, ezért az $LB2$ korlát $O(mn \log n)$ időben kiszámítható. Az $LB2$ korlátot használják [3, 7] cikkekben használt B&B algoritmusok.

Egy harmadik lehetőség, ha az $1|r_j,q_j|C_{\max}$ feladatban megengedjük, hogy egy munka megmunkálását meg lehessen szakítani. Ez a feladat a következő módon oldható meg: minden pillanatban munkáljuk meg azt a munkát, amelyiknek a legnagyobb a szállítási ideje [42]. Ha $LB3_k$-val jelöljük a $k.$ gépen ennek a feladatnak az optimumát, akkor a PFSP optimumára ebből az

$$LB3 = \max_{1 \leq k \leq m} LB3_k \tag{2.11}$$

alsó korlátot kaphatjuk. Az $LB3$ alsó korlát $O(mn \log n)$ időben kiszámítható. Az $LB3$ alsó korlátot használja a [15]-beli B&B algoritmus.

Az eddigi alsó korlátoknál erősebb korlátokat kaphatunk, ha nem csak egy, hanem két gépre, mondjuk a $k.$-ra és az $l.$-re követeljük meg, hogy egyszerre csak egy munkát munkálhatnak meg, a többi gépről pedig feltesszük, hogy egyszerre több munkát is megmunkálhatnak. Így egy olyan két gépes feladatot kapunk, melyben minden j munkának van egy r_{kj} rendelkezésre állási ideje, egy q_{lj} szállítási ideje, továbbáegy l_j késleltetési ideje, amit az

$$l_j = \begin{cases} \sum_{r=k+1}^{l-1} P_{rj} & \text{if } k < l-1 \\ 0 & \text{if } k = l-1, \end{cases} \tag{2.12}$$

képlettel definiálhatunk. A késleltetési idő azt jelenti, hogy miután az első gépen befejeztük a j munka megmunkálását, legalább l_j időnek el kell telnie, míg a második gépen elkezdhetjük megmunkálni a j munkát. Ezt a 2 gépes PFSP-t szokás $F2|r_j,l_j,q_j|C_{\max}$ alakkal jelölni.

Ha az $F2|r_j,l_j,q_j|C_{\max}$ feladatban minden munka rendelkezésre álló idejét az $r_{kj} = \min_{j \in J} r_{kj}$ és szállítási idejét a $q_{lj} = \min_{j \in J} q_{lj}$ formulával egyenlővé tesszük, akkor ebben az esetben az $F2|r_j,l_j,q_j|C_{\max}$ feladat optimuma az

$$\min_{j \in J} r_{kj} + C^{kl}_{\max}(J) + \min_{j \in J} q_{lj} \tag{2.13}$$

képlettel határozható meg, ahol $C^{kl}_{\max}(J)$ a k,l gépekre felírt $F2|l_j|C_{\max}$ feladat optimumával egyezik meg. Mint azt Rinooy Kan [84] megmutatta, ez utóbbi feladat optimális ütemezését polinom időben megkapjuk, ha a 2.2. részben ismertetett Johnson algoritmust alkalmazzuk arra kétgépes PFSP-re, melyben egy j munka megmunkálási ideje az első gépen $P_{1j} + l_j$, míg a

második gépen $P_{2j} + l_j$. Ebből azt kapjuk, hogy minden $1 \leq k < l \leq N$ esetén a PFSP egy alsó korlátját szolgáltatja az

$$LB4_{k,l} = \min_{j \in J} r_{kj} + C^{kl}_{\max}(J) + \min_{j \in J} q_{lj} \tag{2.14}$$

formula, ahol $C^{kl}_{\max}(J)$ a k. és l. gépekhez tartozó $F2|l_j,prmu|C_{\max}$ feladat minimális átfutási idejét jelöli. Ebből a PFSP-re a

$$LB4 = \max_{1 \leq k < l \leq M} LB4_{kl} \tag{2.15}$$

alsó korlát adódik.
Az $LB4$ alsó korlát kiszámításához összesen $\binom{M}{2}$ (hiszen ennyi különböző (k,l), $k < l$ pár van) 2 gépes PFSP-re kell a Johnson algoritmust alkalmazni, így az $LB4$ alsó korlát kiszámításának időigénye $O(m^2 n \log n)$. Az $LB4$ alsó korlátot használták az [22, 52, 66, 81, 102]-beli B&B algoritmusok.

Hogy az $LB4$ korlát kiszámításának időigényét csökkentsük, vehetjük csak azokat a párokat, melyeknél $l = k + 1$ (azaz csak egymást követő gépeket nézünk). Így a PFSP-re az

$$LB5 = \max_{1 \leq k < M} \left(\min_{j \in J} r_{kj} + C^{k}_{\max}(J) + \min_{j \in J} q_{lj} \right) \tag{2.16}$$

alsó korlátot kapjuk, ahol $C^{k}_{\max}(J)$ a k. és a $(k+1)$. gépekhez tartozó $F2|l_j|C_{\max}$ relaxált optimum értékét jelöli. Mivel $LB5$ kiszámításához csak $M - 1$-szer kell alkalmazni a Johnson algoritmust egy kétgépes PFSP-re, ezért $LB5$ kiszámításának az időigénye $O(mn \log n)$. A [15, 73] cikkekben a B&B algoritmusok az $LB5$ alsó korlátot használják.
Az alábbiakban ismertetünk egy B&B algoritmust.

Ladhari és Haouari B&B algoritmusa a PFSP-re [51]:
Az algoritmusban a keresési fa gyökere az összes permutáció halmazának felel meg, míg a levelek mindegyike egyetlen permutációt tartalmaz. A keresési fa egy tetszőleges csúcsa permutációk egy olyan halmazának felel meg, melynek az 'eleje' illetve a 'vége' ismert (pl. a (2,4,-,-,3,6) csúcs a (2,4,1,5,3,6) és (2,4,5,1,3,6) permutációkat tartalmazza). A gyökértől haladva úgy jutunk el a levelekig, azaz egy permutációig, hogy mindig egy részpermutációt bővítünk egy új elemmel, mégpedig vagy a részpermutáció elejét, vagy a végét (az algoritmus ráadásul ezt a két lépést felváltva teszi). Pl. a (6,2,5,4,3,1) permutációhoz a (-,-,-,-,–), (6,-,-,-,-,-), (6,-,-,-,-,1), (6,2,-,-,-,1), (6,2,-,-,3,1), (6,2,5,-,3,1), (6,2,5,4,3,1) pontokon keresztül jutunk el. Nézzük most ezt egy kicsit részletesebben. A keresési fa egy tetszőleges csúcsát egy (σ_1,σ_2) párral azonosíthatjuk, ahol σ_1 és σ_2 a munkák két diszjunk részpermutációja. A (σ_1,σ_2)-nek megfelelő halmazba azok a permutációk tartoznak, melyek a σ_1 részpermutációval kezdődnek, a σ_2

részpermutációval végződnek, és a σ_1, illetve σ_2-ben nem szereplő munkák a σ_1 ás σ_2 között helyezkednek el. A keresési fa kiindulási csúcsában $\sigma_1 = \sigma_2 = \emptyset$. A kiindulási legjobb megoldás a NEH heurisztika által adott megoldás lesz. A szétválasztás a következőképpen történik: legyen mondjuk $V = (\sigma_1,\sigma_2)$ a keresési fa egy pontja. Ekkor minden j munkára, mely nem szerepel sem σ_1-ben, sem σ_2-ben (azaz j még nincsen ütemezve), létrehozzuk V-nek egy V_j leszármazottját. Íly módon tehát $n - (n_1 + n_2)$ új csúcsa lesz a keresési fának, ahol n_1 és n_2 a σ_1, illetve σ_2 elemszámát jelöli. Ezután mindegyik új V_j-re meghatározzuk a V_j-ben szereplő permutációk átfutási idejének egy alsó korlátját. Ha valamely V_j esetén ez az alsó korlát nagyobb, mint a jelenleg ismert legjobb megoldás értéke, akkor az optimális megoldás biztosan nem lehet a V_j-nek megfelelő permutációk között, vagyis a V_j csúccsal nem kell többé foglalkoznunk (passzívvá válik). Ellenkező esetben azt mondjuk, hogy V_j aktív lesz. A V_j csúcsok létrehozása következőképpen történik:

Ha a V csúcsnak a keresési fában a gyökértől való távolsága páros, ami ekvivalens azzal, hogy $n_1 + n_2$ páros, akkor legyen $V_j = (\sigma_1 j,\sigma_2)$, azaz a j munkát tegyük a σ_1 részpermutáció végére. Ha a V csúcsnak a keresési fában a gyökértől való távolsága páratlan, ami ekvivalens azzal, hogy $n_1 + n_2$ páratlan, akkor legyen $V_j = (\sigma_1,j\sigma_2)$, azaz a j munkát tegyük a σ_2 részpermutáció elejére.

Az algoritmus a következő módon határozza meg, hogy melyik csúcsot válasszuk szét a következő lépés során. Ha a V csúcs szétválasztásakor létrejövő új V_j csúcsok közt van aktív, akkor ezen aktív V_j csúcsok közül válasszuk ki azt, amelyiknek az alsó korlátja a legkisebb. Ha a V csúcs szétválasztásakor létrejövő új V_j csúcsok mindegyike passzív, akkor válasszuk ki a keresési fában az aktív csúcsok közül a V-hez legközelebbit, és válasszuk szét ezt a csúcsot.

Egy V csúcs alsó korlátját az algoritmus a V csúcsnak a gyökértől való távolságától függő módon határozza. Mivel egy $V = (\sigma_1,\sigma_2)$ cúcsnak a gyökértől való távolsága éppen $n_1 + n_2$, ezért az eljárást a következőképpen írhatjuk le:

- Ha $0 \leq n_1 + n_2 \leq \lfloor \frac{n}{3} \rfloor$, akkor a V-beli permutációk alsó korlátjának meghatározásához használjuk az $LB1$ korlátot.

- Ha $\lfloor \frac{n}{3} \rfloor < n_1 + n_2 \leq \lfloor \frac{2n}{3} \rfloor$, akkor számítsuk ki $LB1(V)$-t. Ha V nem válik passzivvá (azaz $LB1(V)$ értéke kisebb a legjobb ismert átfutási időnél), akkor számítsuk ki $LB4(V)$-t, és legyen ez V alsó korlátja.

- Ha $\lfloor \frac{2n}{3} \rfloor + 1 \leq n_1 + n_2$, akkor először kiszámítjuk $LB1(V)$-t. Ha V nem válik passzivvá, akkor számítsuk ki $LB4(V)$-t. Legyen k^* és l^* az a két gép, melyre $LB4_{k^*,l^*} = \max_{1 \leq k < l \leq M} LB4_{kl}$. Ha az $LB4(V)$ alsó korlát segítségével a V csúcs még mindig nem válik passzivvá, akkor Haourari és Ladhari B&B algoritmusával [41] számítsuk ki a k^*,l^* gépeken értelmezett $F2|r_j,l_j,q_j,prmu|C_{\max}$ feladat egy optimális megoldását. Legyen ennek a megoldásnak a $C_{\max}$ értéke a V csúcs alsó korlátja.

DOI: 10.15477/SZE.MMTDI.2017.009

A V csúcshoz tartozó alsó korlátnak ilyen módon való kiszámításának a következő a magyarázata. Az $LB4$ alsó korlát mindig erősebb az $LB1$ alsó korlátnál, ennek azonban az az ára, hogy az $LB4$ korlát kiszámítása $m \log n$-szer annyi időt igényel, mint az $LB1$ korlát kiszámítása. Ezért azoknál a V csúcsoknál, melyeknél a még nem ütemezett munkák száma nagy, megelégszünk az $LB1$ alsó korlát használatával. Amint a nem ütemezett munkák száma egy bizonyos szint alá csökken, akkor ha az $LB1$ korláttal nem sikerül passzívvá tenni a V csúcsot, akkor kiszámítjuk az erősebb $LB4$ korlátot is. Végezetül, ha a még nem ütemezett munkák száma kicsi és sem az $LB1$ sem az $LB4$ korlát nem teszi a V csúcsot passzivvá, akkor megkeressük a $F2|r_j,l_j,q_j,prmu|C_{\max}$ feladat egy optimális megoldását.

A PFSP MILP modelljei

A 2.1.2. részben definiált klasszikus PFSP-nek számos MILP modellje ismert. Wagner [110] a 3 gépes PFSP egy csak egész változókat tartalmazó MILP modelljét adta meg, melyet később Baker [4] általánosított M gép esetére. Stafford [95, 96] megmutatta, hogy a Wagner modell megoldása nem ad megvalósítható ütemezést, ezért új egyenlőtlenségek hozzáadásával kijavította Wagner modelljét. A későbbiekben Stafford és Tseng [98], Wilson [114] és Stafford, Tseng és Gupta [100, 101] is új MILP modelleket adtak a PFSP-re. Stafford és Tseng [97, 99]-ben a sorrendtől függő átszerelési időket tartalmazó PFSP MILP modelljeit írta fel. Manne [65], illetve Liao és You [58] a PFSP-nél általánosabb job shop ütemezési feladatra mutatott be MILP modellt. Zhu és Heady [118], illetve Ronconi és Birgin [85] a PFSP azon változatára adott MILP modellt, amikor a célfüggvény a sietések és késések összege. Ez utóbbi cikkben azt a feladatot is vizsgálták, amikor az egymást követő gépek között egyáltalán nincsen puffer (ebben az esetben egy munka elvégzése után csak akkor vehetjük le a gépről a munkát ha a következő gép üres).

A PFSP MILP modelljei, annak alapján, ahogy egy permutációt leírnak, két csoportba oszhatók. A Wagner család modelljei úgy adnak meg egy permutációt, hogy a sorrend minden j helyére és i munkájára megmondják, hogy az i munka kerül-e a permutáció j. helyére. Ezzel szemben a Manne család modelljei azt mondják meg minden $(i,j)|\ i < j$ párra, hogy az i munka megelőzi-e a sorrendben a j munkát, vagy sem. A következőkben bemutatjuk a Wagner család, illetve a Manne család néhány modelljét. A modellekben a következő jelöléseket fogjuk használni:

Adott munkáknak egy kezdeti sorrendje. Minden munkát a kezdeti sorrendbeli helyével $(i, 1 \leq i \leq N)$ azonosítunk.

Paraméterek:

M a gépek száma

N a munkák száma

DOI: 10.15477/SZE.MMTDI.2017.009

P_{ri} az i munka a megmunkálási ideje az r gépen

Folytonos változók:

E_{rj} a permutáció j. munkájának a befejezési ideje az r gépen

B_{rj} a permutáció j. munkájának a kezdési ideje az r gépen

C_{rj} a j munka befejezési ideje az r gépen

S_{rj} a j munka kezdési ideje az r gépen

X_{rj} állóidő az r gépen a sorrend j. munkájának kezdése előtt

Y_{rj} a sorrend j. munkájának a várakozási ideje miután befejeződött a megmunkálása az r gépen

$C_{\max}$ az átfutási idő.

Bináris változók:

Z_{ij} 1, ha az i munka a permutáció j. helyére kerül, 0 egyébként

D_{ij} 1, ha az i munka megelőzi a sorrendben a j munkát, 0 egyébként ($i < j$).

A Wagner család tagjai

A Wilson modell [114]

A Wilson modell feltételei biztosítják a következő feltételek teljesülését:

- Minden munka a permutáció pontosan egy helyére kerül.

$$\sum_{j=1}^{N} Z_{ij} = 1 \qquad 1 \leq i \leq N \tag{2.17}$$

- A sorrend mindegyik helyére pontosan egy munka kerül.

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad 1 \leq j \leq N \tag{2.18}$$

- Az első gépen a munkák megmunkálása folytonos, vagyis nincsenek állóidők.

$$B_{1j} + \sum_{i=1}^{N} P_{1i} Z_{ij} = B_{1,j+1} \qquad 1 \leq j \leq N-1 \tag{2.19}$$

- A sorrend első munkáját a 0 időpontban kezdjük el megmunkálni az első gépen.

$$B_{11} = 0 \tag{2.20}$$

DOI: 10.15477/SZE.MMTDI.2017.009

- A sorrend első munkájának egyetlen gép előtt sem kell várakoznia.

$$B_{r1} + \sum_{i=1}^{N} P_{ri} Z_{i1} = B_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{2.21}$$

- Egy munkát csak azután kezdhetünk megmunkálni egy gépen, miután az adott munkát megmunkáltuk az előző gépen.

$$B_{rj} + \sum_{i=1}^{N} P_{ri} Z_{ij} \leq B_{r+1,j} \qquad 1 \leq r \leq M-1;\ 2 \leq j \leq N \tag{2.22}$$

- A sorrend $(j+1)$. munkáját csak azután kezdhetjük megmunkálni egy gépen, hogy a sorrend j. munkáját befejeztük az adott gépen.

$$B_{rj} + \sum_{i=1}^{N} P_{ri} Z_{ij} \leq B_{r,j+1} \qquad 2 \leq r \leq M;\ 2 \leq j \leq N-1 \tag{2.23}$$

- Az átfutási idő a sorrend utolsó munkájának az utolsó gépen való kezdési idejének és megmunkálási idejének az összege.

$$C_{\max} = B_{MN} + \sum_{i=1}^{N} P_{Mi} Z_{iN} \tag{2.24}$$

A PFSP-re adott Wilson modell a következőképpen foglalható össze:
minimalizáljuk (2.24)-et a (2.17) – (2.23) feltételek mellett.

A TS2 modell

A TS2 modell [101] a kezdési idők helyett a befejezési időket használja. A modell egyenlőtlenségei garantálják a következő feltételek teljesülését.

- Minden munka a permutáció pontosan egy helyére kerül.

$$\sum_{j=1}^{N} Z_{ij} = 1 \qquad 1 \leq i \leq N \tag{2.25}$$

- A sorrend mindegyik helyére pontosan egy munka kerül.

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad 1 \leq j \leq N \tag{2.26}$$

- A permutáció $(j+1)$. elemének a befejezési ideje egy gépen nem lehet kisebb, mint a sorrend j. elemének az adott gépen a bejezési idejének és a megmunkálási idejének az összege.

$$E_{rj} + \sum_{i=1}^{N} P_{ri} Z_{i,j+1} \leq E_{r,j+1}, \qquad 1 \leq r \leq M, 1 \leq j \leq N-1 \tag{2.27}$$

- Egy munka befejezési ideje az $(r+1)$. gépen legalább annyi, mint a munka befejezési ideje az r. gépen plusz a munka megmunkálási ideje az $(r+1)$. gépen.

$$E_{rj} + \sum_{i=1}^{N} P_{r+1,i} Z_{ij} \leq E_{r+1,j}, \qquad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{2.28}$$

- A sorrend első elemének a befejezési ideje nem lehet kisebb az első gépen, mint a megmunkálási ideje az első gépen.

$$\sum_{i=1}^{N} P_{1i} Z_{i1} \leq E_{11} \tag{2.29}$$

- Az átfutási idő a sorrend utolsó elemének a befejezési ideje az utolsó gépen.

$$C_{\max} = E_{MN} \tag{2.30}$$

A TS2 modell tehát így foglalható össze :
minimalizáljuk (2.30)-et a (2.25) – (2.29) feltételek mellett.

A WST modell

A WST modellt Stafford és Tseng [98] vezették be Wagner [110] és Stafford [96] korábbi munkáinak a felhasználásával. A modell egyenlőtlenségei biztosítják a következő feltételek teljesülését:

- Minden munka a permutáció pontosan egy helyére kerül.

$$\sum_{j=1}^{N} Z_{ij} = 1 \qquad 1 \leq i \leq N \tag{2.31}$$

- A sorrend mindegyik helyére pontosan egy munka kerül.

$$\sum_{i=1}^{N} Z_{ij} = 1 \qquad 1 \leq j \leq N \tag{2.32}$$

DOI: 10.15477/SZE.MMTDI.2017.009

- A sorrend első munkája várakozás nélkül halad végig a gépeken.

$$Y_{r1} = 0, \qquad 1 \leq r \leq M-1 \tag{2.33}$$

- A sorrend $(j+1)$. elemét addig nem kezdhetjük el az $(r+1)$. gépen, míg be nem fejeztük az r. gépen, illetve amíg a sorrend j. elemét be nem fejeztük az $(r+1)$. gépen.

$$\sum_{i=1}^{N} P_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{N} P_{r+1,i} Z_{ij} + X_{r+1,j+1} + Y_{rj} \tag{2.34}$$
$$1 \leq r \leq M-1;\ 1 \leq j \leq N-1$$

$$X_{r1} + Y_{r1} + \sum_{i=1}^{N} P_{ri} Z_{i1} = X_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{2.35}$$

- Az átfutási idő az utolsó gépen lévő megmunkálási és várakozási idők összegével egyezik meg.

$$C_{\max} = \sum_{i=1}^{N} P_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{2.36}$$

A WST modell tehát így összegezhető:
minimalizáljuk (2.36)-et a (2.31) – (2.35) feltételek mellett.

A Manne család tagjai

A Manne család tagjai a sorrend i. elemének a kezdési, illetve befejezési ideje helyett az i munka befejezési, illetve kezdési idejét használják.

A Manne modell

Stafford és Tseng [97]-ben a Manne [65] által a job shop feladatra adott MILP modellt írták át a PFSP-re. A modellben A egy nagy számot jelöl. A modell feltételei a következőek:

- Az első gépen egyetlen munka sem fejeződhet be korábban az első gépen lévő megmunkálási idejénél.

$$C_{1i} \geq P_{1i}, \qquad (1 \leq i \leq N) \tag{2.37}$$

- Egy munka befejezési ideje az $r+1$ gépen nem lehet kisebb, mint a befejezési ideje az r gépen plusz a megmunkálási ideje az $r+1$ gépen.

$$C_{rj} + P_{r+1,j} \leq C_{r+1,j}, \qquad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{2.38}$$

- Ha $i < j$ esetén az i munka megelőzi a permutációban a j munkát, akkor a j munkát csak akkor kezdhetjük el egy gépen, ha az i munkát már befejeztük az adott gépen. Hasonlóan, ha $i < j$ esetén a j munka megelőzi a permutációban az i munkát, akkor az i munkát csak akkor kezdhetjük el egy gépen, ha a j munkát már befejeztük az adott gépen.

$$C_{ri} - C_{rj} + AD_{ij} \geq P_{ri}, \qquad 1 \leq r \leq M, 1 \leq i < j \leq N \tag{2.39}$$

$$C_{ri} - C_{rj} + AD_{ij} \leq A - P_{rj}, \qquad 1 \leq r \leq M, 1 \leq i < j \leq N \tag{2.40}$$

  Vegyük észre, hogy ha az i munka megelőzi a sorrendben a j munkát, azaz $D_{ij} = 1$, akkor a (2.39) egyenlőtlenség automatikusan teljesül, míg a (2.40) egyenlőtlenség éppen a kívánt feltételt adja meg. Hasonlóan, ha a j munka előzi meg a sorrendben az i munkát, azaz $D_{ij} = 0$, akkor a (2.40) egyenlőtlenség teljesül automatikusan, (2.39) pedig éppen a befejezési időkre vonatkozó feltételt írja le.

- Az átfutási idő nagyobb, mint bármelyik munkának az utolsó gépen való befejezési ideje.

$$C_{\max} \geq C_{Mi}, \qquad 1 \leq i \leq N \tag{2.41}$$

A PFSP-re adott Manne modell tehát a következő alakban írható:
minimalizáljuk (2.41)-et a (2.37) – (2.40) feltételek mellett.

A Liao-You modell

Az eredeti Manne [65] modell (2.39) és (2.40)-nek megfelelő feltételeit Liao és You [58] egy új, nemnegatív q_{rik} változó bevezetésével kapcsolták össze. (A modellben A ismét egy nagy számot jelöl). A modell a következőképpen írható fel.

- Egy munkát addig nem kezdhetünk el egy gépen, amíg be nem fejeztük az előző gépen.

$$S_{rj} + P_{rj} \leq S_{r+1,j}, \qquad 1 \leq r \leq M - 1, 1 \leq j \leq N \tag{2.42}$$

- Ha $i < j$ estén az i munka megelőzi a permutációban a j munkát, akkor a j kezdési ideje egy gépen nem lehet kisebb, mint az i befejezési ideje az adott gépen. Hasonlóan, ha $i < j$ estén a j munka megelőzi a permutációban az i munkát, akkor az i munkát csak akkor kezdhetjük el egy gépen, ha a j munkát már befejeztük az adott gépen.

$$S_{ri} - S_{rj} + AD_{ij} - P_{rj} = q_{rij}, \qquad 1 \leq r \leq M, 1 \leq i < j \leq N \tag{2.43}$$

$$A - P_{ri} - P_{rj} \geq q_{rij}, \qquad 1 \leq r \leq M, 1 \leq i < j \leq N \tag{2.44}$$

Figyeljük meg, hogy ha az i munka megelőzi a j munkát, akkor (2.43) és (2.44) egyenlőtlenségekből éppen a kezdési időkre vonatkozó feltétel következik. Továbbá ha a j munka előzi meg az i munkát, akkor (2.44) automatikusan teljesül, míg a (2.43) egyenlőtlenségből, figyelembe véve, hogy a q_{rij} változó nemnegatív, ismét a kezdési időkre vonatkozó feltételt kapjuk.

- Az átfutási idő nagyobb, mint bármelyik munkának az utolsó gépen való befejezési ideje.

$$C_{\max} \geq S_{Mi} + P_{ri}, \qquad 1 \leq i \leq N \tag{2.45}$$

A PFSP-re adott Liao-You modell így összegezhető:
minimalizáljuk (2.45)-et a (2.42) – (2.44) feltételek mellett.

A Wagner család modelljeiben a bináris változók (Z_{ij}, $1 \leq i,j \leq N$) száma N^2, míg a Manne család modelljeiben a bináris változók ($D_{i,j}$, $1 \leq i \leq j \leq N$) száma $N(N-1)/2$, vagyis a Wagner család modelljei több mint kétszer annyi bináris változót tartalmaznak, mint a Manne család tagjai. Ennek ellenére, numerikus tesztek [100, 101] azt mutatták ki, hogy a Wagner család modelljeivel sokkal gyorsabban oldhatunk meg PFSP-ket, mint a Manne család modelljeivel.

Korlátozás programozás

A korlátozás programozás (constraint programming, CP) egy programozási paradigma, mellyel nagy méretű kombinatorikus problémák modellezhetők és oldhatók meg hatékonyan. A korlátozás programozást számos területen, többek között a számítógépes grafikában, biológiában és ütemezési feladatoknál alkalmazták sikeresen.

Egy **korlátozás-kielégítési probléma** (constrained satisfaction problem, CSP) a következőképpen foglalható össze:

- adott változóknak egy $X = \{x_1,x_2,\ldots,x_n\}$ véges halmaza;
- minden x_i változóhoz adott egy D_i véges halmaz melyből az x_i változó az értékeit felveheti (D_i az x_i értékkészlete);
- adott korlátozásoknak egy véges halmaza, mely azt adja meg, hogy milyen összefüggéseknek kell fennállni a változók között.

A CSP megoldásán azt értjük, hogy minden x_i változóhoz hozzárendeljük a D_i értékkészletének pontosan egy elemét (lekötjük az x_i változót) úgy, hogy az összes korlátozás teljesüljön. A célunk lehet egy megoldás vagy az összes megoldás megtalálása is. A **korlátozás optimalizálási probléma** (constraint optimization problem, COP) abban különbözik a CSP-től,

DOI: 10.15477/SZE.MMTDI.2017.009

hogy adott egy $O: \ D_1 \times D_2 \times \ldots \times D_n \to \mathbb{R}$ célfüggvény melyet minimalizálni (maximalizálni) akarunk. A feladat a vátozók egy olyan lekötésének a megtalálása, melyre teljesül az összes korlátozás és a célfüggvény értéke minimális (maximális).

Egy LP feladatban minden feltétel egy lineáris egyenlőtlenség vagy egyenlőség. Ezzel szemben egy COP-ben (CST-ben) nincs ilyen megkötés; egy korlát tartalmazhat például nemlineáris egyenlőtlenséget (pl. $x_1 \cdot x_2 \leq \min(x_4,x_5)$) sőt még logikai műveleteket is (pl. $x_1 \leq 3$ vagy $x_1 \geq x_2$). Lényegében azt mondhatjuk, hogy egy, az $x_1, x_2, \ldots, x_k$ változókra vonatkozó c korlát nem más, mint a $D_1 \times D_2 \times \ldots \times D_k$ direkt szorzaton értelmezett reláció; a direkt szorzatnak egy $(u_1,u_2, \ldots ,u_k)$ eleme pontosan akkor eleme a relációnak, ha a változóknak az $x_1 = u_1, x_2 = u_2, \ldots, x_n = u_n$ lekötésére teljesül a c korlát.

A korlátozás kielégítési probléma megoldási módszere

A CSP egy korlátja tetszőleges számú változón lehet értelmezve. Ha egy c korlát egyetlen változót tartalmaz, akkor **unáris korlátról** beszélünk. Ha c egy unáris korlát, mely az x_i változót tartalmazza, akkor x_i-nek a D_i értékkészletéből elhagyhatjuk azokat az értékeket, melyekre nem teljesül a c korlát. Ezután már a c korlát a D_i összes elemére teljesülni fog, ezért a c korlátot el is hagyhatjuk a feladatból. Ezért feltehetjük, hogy minden korlát legalább 2 változót tartalmaz. Ha egy korlát pontosan két változót tartalmaz, akkor azt **bináris korlátnak** nevezzük. Azok a CSP-k, melyekben csak bináris korlátok vannak, ábrázolhatók egy olyan gráffal, melynek a pontjai a változóknak felelnek meg, az élei pedig az él két végpontjához tartozó változókon értelmezett bináris korlátot reprezentálják. Belátható, hogy (esetleg új változók bevezetésével) minden CSP átalakítható olyan CSP-vé, melyben csak bináris korlátok vannak.

Tegyük fel, hogy van egy olyan CSP-nk, melyben csak bináris korlátok vannak és legyen c egy bináris korlát, mely az x_i és x_j változók között áll fenn. Az mondjuk, hogy a feladatnak a fenti módon megfeleltetett gráfban a c korlátnak megfelelő (x_i,x_j) él **él-konzisztens**, ha minden $u \in D_i$ értékhez létezik olyan $v \in D_j$ érték, melyre a változóknak az $x_i = u$, $x_j = v$ lekötése kielégíti a c korlátot. Ha egy (x_i,x_j) él nem él-konzisztens, akkor x_i értékkészletében létezik legalább egy olyan u érték, melyhez nem található olyan $v \in D_j$ érték, hogy a változóknak az $x_i = u$, $x_j = v$ lekötése kielégítse a c korlátot. Az ilyen u értékek elhagyhatók D_i-ből, hiszen az x_i változó értéke a feladat egyetlen megoldásában sem vehet fel ilyen értéket. Ezen u értékeknek D_i-ből való törlése után a gráf (x_i,x_j) éle már konzisztens lesz. Ezt az eljárást nevezzük a c korlát **propagálásának**. Ha a gráf minden éle konzisztens, akkor magát a CSP-t él-konzisztensnek nevezzük. Élek egymás utáni propagálásaval minden CSP él konzisztenssé tehető. Az egyik lehetséges módszer a [63]-beli AC3 algoritmus. Az eljárás során először az összes bináris korlátot beletesszük egy Q halmazba. Ezután kiválasztjuk a Q halmaz egy (x_i,x_j) változókat tartalmazó korlátját, propagálunk ezzel az éllel és elhagyjuk Q-ból ezt a korlátot. Ha

DOI: 10.15477/SZE.MMTDI.2017.009

a propagálás során az x_i változó értékkészlete megváltozott, akkor az összes $(x_k,x_i), k \neq i$ élnek megfelelő korlátot hozzávesszük a Q halmazhoz és iteráljuk az eljárást ddig, amíg a Q halmaz üres nem lesz. Ha az eljárás során valamely változó tartománya üres lesz, akkor a feladatnak nincs megoldása. Az eljárás végén - tehát amikor a Q halmaz üres lesz - kapott feladat már él-konzisztens lesz. Egy feladat él-konzisztenssé tételére további eljárások találhatók [8, 9, 11, 71]-ben.
Ha egy feladat él-konzisztenssé tétele során a végső CSP-ben minden D_i tartomány egyelemű, akkor megkaptuk a feladat egy megoldását. Ha a propagálások segítségével él-konzisztenssé tett CSP-ben van olyan változó, melynek értékkészlete legalább két elemből áll, akkor még nem tudjuk a kiindulási feladat egy megoldását leolvasni. Ilyenkor különféle keresési technikákat alkalmazhatunk, melyeket az alábbiakban ismertetünk. Mielőtt ezekre a technikákra rátérnénk, fontos megjegyezni, hogy önmagában abból, hogy egy korlátozás kielégítési feladat él-konzisztens, nem következik, hogy a feladatnak van is megoldása. Egy egyszerű példa erre egy 3 változót tartalmazó CSP, melyben mindegyik változó az $(1,2,3,4,5)$ számok valamelyikét veheti fel, és 3 korlát van melyek azt mondják ki, hogy bármely két változó összegének 5-nek kell lennie. Könnyen látható, hogy bármely korlát él-konzisztens, a feladatnak azonban még sincsen megoldása.

Keresési eljárások: A keresési eljárásokban a keresési fa minden csúcsa tartalmazza az összes változó értékkészletét (azaz a D_i halmazokat); egy változó értékkészlete különböző csúcsokban különböző lehet. Ezen kívül mindegyik csúcs lekötött, illetve le nem kötött változókat is tartalmaz. A lekötött változó azt jelenti, hogy a változóhoz hozzárendeljük az értékkészletének az egyik értékét. A keresés során egy csúcs gyermekeit úgy kapjuk, hogy kiválasztjuk egy még le nem kötött x változóját, melynek az értékkészlete mondjuk az $(u_1,u_2,\ldots,u_k)$ elemekből áll. Ekkor az x változó értékét $u_1,u_2,\ldots,u_k$-ra rögzítve kapjuk az adott csúcs k darab gyerekét. Mikor a keresés során egy változó rögzítésével egy új csúcsra lépünk, mindig valamilyen értelemben konzisztenssé tesszük az új csúcsot. A két leggyakrabban alkalmazott eljárás új csúcs konzisztenssé tételére a [10]-beli **előre ellenőrzés** (forward checking, FC), illetve a [91]-beli **él-konzisztencia megőrzés** (maintaining arc consistency, MAC) technika, melyeket az alábbiakban ismertetünk. Ha az új csúcs konzisztenssé tétele során valamely változó értékkészlete üres lesz, akkor a fa ezen ágát már nem kell tovább vizsgálnunk, hiszen nem kaphatunk belőle megoldást, és visszalépünk egy korábbi csúcsra. Ha az új csúcs konzisztenssé tétele után minden változó értékkészlete egyetlen elemből áll, akkor megkaptuk a feladat egy megoldását. Végezetül ha az új csúcs konzisztenssé tétele után létezik olyan változó, melynek az értékkészlete legalább 2 elemből áll, akkor generáljuk a fentiek szerint ennek az új csúcsnak egy gyerekét és folytassuk tovább az eljárást. Az előre ellenőrzést, illetve élkonzisztencia megőrzést használó keresési eljárások ismertetéséhez feltesszük, hogy a CSP minden korlátja bináris. Ha a keresés során a keresési fa tartalmazza egy csúcsának

DOI: 10.15477/SZE.MMTDI.2017.009

valamely lekötetlen x változójához tartozó összes gyerekét, akkor azt mondjuk, hogy az adott csúcsot már átvizsgáltuk.

Előre ellenőrzést használó keresés:

1. lépés Tegyük először a korlátozás programozási feladatot él-konzisztenssé. A keresési fa gyökér csomópontja az él-konzisztenssé tett feladat D_i értékkészleteit tartalmazza, továbbá minden változó kötetlen. Ha a gyökérben valamely változó értékkészlete üres, akkor a feladatnak nem létezik megoldása. Ha a gyökérben minden változó értékkészlete egyetlen elemből áll, akkor megkaptuk feladat egy megoldását.

2. lépés Válasszuk ki a keresési fának azt a legkésőbb létrehozott P pontját, melyet még nem vizsgáltunk át, tartalmaz le nem kötött változót és melyben minden változó értékkészlete nemüres. Ha ilyen csúcs nem létezik, akkor a feladatnak nincs megoldása.

3. lépés Ha P-nek még nincs gyereke a fában, akkor a még le nem kötött változók közül válasszunk ki egyet (legyen ez mondjuk x) és x értékkészletéből válasszunk ki egy u értéket. Ha P-nek már vannak gyerekei a fában, melyek az x változóhoz tartoznak, akkor x értékkészletéből válasszunk ki egy olyan u értéket, melyre a csúcsnak az $x = u$ lekötéssel létrehozható gyereke még nem szerepel a fában.

4. lépés A 3. lépésben kiválasztott x változóval és u értékkel hozzuk létre a fában az $x = u$ lekötéssel P egy új gyerekét melyet P'-vel jelülünk. (P' tehát eggyel több lekötött változót tartalmaz, mint P.) Vegyük azokat a korlátokat, melyek az x változón kívül még egy P-ben le nem kötött változót tartalmaznak. Jelöljük ezen korlátok halmazát C-vel. A P-beli, le nem kötött változókhoz tartozó értékkészletekből kiindulva él propagálások segítségével tegyük a C halmazt él-konzisztenssé. P'-ben egy le nem kötött változó értékkészlete legyen a változónak a C él-konzisztenssé tétele során kapott értékkészlete.

5. lépés Ha P'-ben minden le nem kötött változó értékkészlete egy elemet tartalmaz, akkor megkaptuk a feladat egy megoldását. Ha valamely változó értékkészlete üres lesz, akkor a fának ezen az ágán nem létezik a feladatnak megoldása. Ilyenkor ugorjunk a 2. lépésre. Ha P'-ben egyik változó értékkészlete sem üres és van olyan változó, melynek értékkészlete legalább 2 elemből áll, akkor ugorjunk a 2. lépésre.

Az ismertetett keresés 4. lépése az előre ellenőrzés eljárás. Ennek a lényege tehát az, hogy amikor a keresési fa egy új csúcsának létrehozásakor egy x változó értékét lekötjük, akkor az összes, a későbbiekben lekötendő változó értékkészletéből kidobjuk azokat az értékeket,

DOI: 10.15477/SZE.MMTDI.2017.009

melyek valamelyik korlát miatt ”ellentmondanak” x ezen lekötésének. Így egy változó lekötésekor a korábban lekötött változókkal már nem kell foglalkoznunk, hiszen egy olyan korlát, mely egy már korábban lekötött változót és az éppen most lekötött változót tartalmazza biztosan teljesülni fog.

Él-konzisztencia megőrzést használó keresés:

1. lépés Tegyük először a korlátozás programozási feladatot él-konzisztenssé. A keresési fa gyökér csomópontja az él-konzisztenssé tett feladat D_i értékkészleteit tartalmazza, továbbá minden változó kötetlen. Ha a gyökérben valamely változó értékkészlete üres, akkor a feladatnak nem létezik megoldása. Ha a gyökérbn minden változó értékkészlete egyetlen elemből áll, akkor megkaptuk feladat egy megoldását.

2. lépés Válasszuk ki a keresési fának azt a legkésőbb létrehozott P pontját, melyet még nem vizsgáltunk át, tartalmaz le nem kötött változót és melyben minden változó értékkészlete nemüres. Ha ilyen csúcs nem létezik, akkor a feladatnak nincs megoldása.

3. lépés Ha P-nek még nincs gyereke a fában, akkor a még le nem kötött változók közül válasszunk ki egyet (legyen ez mondjuk x) és x értékkészletéből válasszunk ki egy u értéket. Ha P-nek már vannak gyerekei a fában, melyek az x változóhoz tartoznak, akkor x értékkészletéből válasszunk ki egy olyan u értéket, melyre a csúcsnak az $x = u$ lekötéssel létrehozható gyereke még nem szerepel a fában.

4. lépés A 3. lépésben kiválasztott x változóval és u értékkel hozzuk létre a fában az $x = u$ lekötéssel P egy új gyerekét melyet P'-vel jelülünk. (P' tehát eggyel több lekötött változót tartalmaz, mint P.) Vegyük azokat a korlátokat, melyeknek mindkét változója a P csúcs még le nem kötött változói közü kerül ki. Jelöljük ezen korlátok halmazát C-vel. A P-beli, le nem kötött változókhoz tarlmazó értékkészletekből kiindulva él propagálások segítségével tegyük a C halmazt él-konzisztenssé. P'-ben egy le nem kötött változó értékkészlete legyen a változónak a C él-konzisztenssé tétele során kapott értékkészlete.

5. lépés Ha P'-ben minden le nem kötött változó értékkészlete egy elemet tartalmaz, akkor megkaptuk a feladat egy megoldását. Ha valamely változó értékkészlete üres lesz, akkor a fának ezen az ágán nem létezik a feladatnak megoldása. Ilyenkor ugorjunk a 2. lépésre. Ha P'-ben egyik változó értékkészlete sem üres és van olyan változó, melynek értékkészlete legalább 2 elemből áll, akkor ugorjunk a 2. lépésre.

Látható, hogy az él-konzisztencia megőrzést használó keresés a 4. lépésben tér el az előre ellenőrzést használó kereséstől. Míg az FC-t használó keresésben a 4. lépésben csak

DOI: 10.15477/SZE.MMTDI.2017.009

azokat a korlátokat vizsgáljuk meg, melyek az újonnan lekötött változó mellett egy később lekötendő változót tartalmaznak, addig a MAC-et használó keresésben az összes olyan korlátot megnézzük, melyek nem tartalmaznak korábban lekötött változókat, így a MAC-et használó keresésben a $4.$ lépésbeli C halmaz több korlátot tartalmaz, mint az FC-t használó keresés $4.$ lépése. Ezért a MAC-et használó keresés $4.$ lépése több időt igényel, mint az FC-t használó keresés $4.$ lépése, cserébe viszont a MAC-et használó keresés jobban tudja redukálni a később lekötendő változók értékkészletét mint az FC-t használó keresés és így kisebb keresési fához vezethet.

Mind az FC-t, mind a MAC-et használó keresésben amikor egy olyan csúcsra ugrunk melynek még nincs gyereke a fában, akkor ki kell választanunk a csúcs egy még le nem kötött x változóját melynek lehetséges lekötéseivel kapjuk meg a csúcs gyerekeit. Arra, hogy a lehetséges változók közül melyiket válasszuk ki, több módszer is létezik. A változók választásának sorrendje lehet **statikus**, amikor is előre meghatározzuk a változók sorrendjét és a változók lekötését ezen sorrend szerint végezzük el, illetve lehet **dinamikus**, amikor is a lekötendő változó függ a keresés aktuális állapotától. A változó dinamikus kiválasztására számos heurisztika létezik. Az egyik gyakran használt heurisztika a **first-fail** eljárás, melynek során mindig azt a változót választjuk ki a még le nem kötött változók közül, melynek az értékkészlete a legkevesebb elemből áll. Ha több változónak is ugyanannyi elemből áll az értékkészlete, akkor egy másik ismert heurisztika azt a változót választja, amelyik a legtöbb korlátban szerepel.

Miután kiválasztottuk az elágazáshoz a következő változót, el kell döntenünk, hogy milyen sorrendben választjuk a változó tartományában szereplő értékeket (milyen sorrendben fogjuk a csúcs gyerekeit meglátogatni). Ennek eldöntésére szintén léteznek statikus, illetve dinamikus heurisztikák. Mehta és Dong [68] három statikus heurisztikát mutatott be a változók értékének választására. A heurisztikák lényege, hogy minden változó minden értékéhez hozzárendelünk egy súlyt, mely a keresés alatt változatlan marad. Amikor a keresés során egy le nem kötött változót lekötünk, akkor az értékeit a súlyok szerinti csökkenő sorrendben rendeljük hozzá a változóhoz. Egy x változó u értékének súlyát úgy kapjuk, hogy minden olyan y változóra, mellyel x szerepel együtt egy korlátban, meghatározzuk, hogy y-nak hány olyan v értéke van, melyre az $x = u$, $y = v$ lekötésekre teljesülnek a korlátok. Az x változó u értékének súlyát ezen számok lapján kapjuk; az egyik heurisztikában ezen számok összege, egy másikban ezen számok szorzata lesz az u súlya. Az értékek rendezésére dinamikus heurisztikákat ismertettek a [30, 33, 109] cikkekben. Frost és Dechter [30]-ban négy dinamikus heurisztikát ismertetett az értékek sorbarendezésének problémájára. Mindegyik heurisztikának az az ötlete, hogy mindig azt az értéket próbáljuk választani, amelyik a legnagyobb eséllyel van benne egy megoldásban. Például az egyik heurisztika mindig azt az értéket választja, mely a még le nem kötött változók értékei közül a lehető legtöbbel kompatibilis.

DOI: 10.15477/SZE.MMTDI.2017.009

A korlátozás optimalizálási feladat megoldása

A korlátozás optimalizálási feladat megoldható korlátozás kielégítési feladatok megoldásának segítségével. Ez történhet például az alábbi módon. Tegyük fel, hogy adott egy COP az O célfüggvénnyel. Vegyünk fel egy új x_0 változót és a feladathoz adjuk hozzá az $x_0 = O(x_1,x_2,\ldots,x_n)$ korlátot. Az x_0 változó értéke tehát a célfüggvény értékével lesz egyenlő. Oldjuk meg először a COP feladat korlátait tartalmazó CSP-t. Amikor a keresés során kapunk egy megoldást, melyben a célfüggvény, azaz x_0 értéke K, akkor a CSP feladathoz adjuk hozzá az $x_0 \leq L - 1$ feltételt és folytassuk tovább a keresését. Az eljárás akkor áll meg amikor a feladatnak nem létezik az összes korlátozást kielégítő megoldása. Ekkor az utoljára talált megoldás az eredeti COP feladat optimális megoldása lesz.

Ha a COP feladat esetén rendelkezésünkre áll az optimum egy alsó korlátja (LB) és felső korlátja (UB), akkor a feladatot egy dichotóm kereséssel is megoldhatjuk. Először adjuk hozzá a korlátozások halmazához az $x_0 \leq (UB + LB)/2$ feltételt és oldjuk meg a CSP-t. Ha a korlátozás kielégítési feladatnak létezik X megoldása, akkor UB értékét írjuk át $O(X)$-re, ha pedig nem létezik megoldása, akkor LB értéke legyen $(UB + LB)/2$ és iteráljuk az eljárást. A keresés az $LB = UB$ esetben áll meg, ilyenkor megtaláltuk az optimális megoldást. Ezt az eljárást használta például Kovács et al. [50] ipari ütemezési feladatok megoldására.

A permutációs flow shop feladat korlátozás programozás modellje

A PFSP MILP modelljeihez képest a CP modellben egy permutáció egyszerűen megadható azzal, hogy minden i indexre megmondjuk, hogy melyik munka kerül a permutáció i. helyére. A CP modell felírása során feltesszük, hogy minden megmunkálási idő egész szám. A modellben a következő változókat fogjuk használni:

Egész értékű változók:

S_{rj} a j munka kezdési ideje az r gépen

x_i j, ha a permutáció i. helyére a j munka kerül.

A PFSP egy CP modellje a következőképpen írható fel:

- A sorrend minden helyére pontosan egy munka kerül.

$$x_i \in \{1,2,\ldots,N\} \qquad 1 \leq i\ N \tag{2.46}$$

$$x_i \neq x_j \qquad 1 \leq i < j \leq N \tag{2.47}$$

- A kezdési idők nemnegatív egész számokat vehetnek fel a $[0,W]$ intervallumból, ahol W az optimális megoldásban a kezdési idők egy felső becslése (W például lehet az összes gépen az összes megmunkálási időnek az összege).

$$S_{rj} \in \{0,1,\ldots,W\} \qquad 1 \leq r \leq M, 1 \leq j \leq N \tag{2.48}$$

- Egy munkát csak azután kezdhetünk megmunkálni egy gépen, hogy az előző gépen befejeztük a megmunkálását.

$$S_{rj} + P_{rj} \leq S_{r+1,j} \qquad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{2.49}$$

- Egy munkát csak azután kezdhetjük el megmunkálni egy gépen, hogy a sorrendben őt közvetlenül megelőző munkát befejeztük az adott gépen.

$$S_{r,x_i} + P_{r,x_i} \leq S_{r,x_{i+1}} \qquad 1 \leq r \leq M, 1 \leq i \leq N-1 \tag{2.50}$$

- Az átfutási idő az utolsó munkának az utolsó gépen való befejezési idejével egyezik meg.

$$\min C_{\max} = S_{M,x_N} + P_{M,x_N} \tag{2.51}$$

A (2.47) feltételhez hasonlóan számos feladat tartalmaz olyan korlátozást, hogy bizonyos változók értékeinek páronként különbözőnek kell lenniük. A hatékonyabb propagáció miatt erre a feltételre a CP rendszerekben bevezettek egy úgynevezett globális korlátot, az $alldifferent$ korlátot. Az $alldifferent$ korláttal való propagálás során a CP rendszerek egy páros gráfban keresnek teljes párosítást (a korlát részletes ismertetése megtalálható [108]-ban). A (2.47) feltétel tehát így írható fel hatékonyabban:

$$alldifferent(x_1,x_2,\ldots,x_N) \tag{2.52}$$

A PFSP egy lehetséges CP modellje így összegezhető:
minimalizáljuk (2.51)-ot a (2.46), (2.52, (2.48) – (2.50) feltételek mellett.

A PFSP fenti CP modelljében a (2.47) globális korlát kivételével az összes korlát bináris. Az S_{rj} változók értékkészlete kezdetben egy $[0,W]$ intervallumba eső egész számokkal egyezik meg. Az intervallum bal oldali végpontja a j munkának a lehetséges legkorábbi kezdési idejét adja meg az j gépen, míg a jobb oldali végpontja a k munkának a lehetséges legkésőbbi kezdési idejét adja meg az r gépen. Az élekkel való propagálás során egy S_{rj} változó esetén az értékkészletéhez tartozó intervallumnak vagy a jobboldali végpontja csökken, vagy a balol-

dali végpontja növekszik. Álljon például az S_{rj} változó értékkészlete az (a_1,b_1), míg az $S_{r+1,j}$ változónak az értékkészlete az (a_2,b_2) intervallum egész számaiból. Ekkor az $(S_{rj},S_{r+1,j})$ élnek megfelelő (2.49) korláttal való propagálás után az $S_{r,j}$ változó értékkészletében
$b_1 := \min(b_1,b_2 - P_{rj})$ lesz.

A különböző ütemezési feladatok korlátozás programozás technikával való megoldása során az **intervallum konzisztencia** tesztek [16, 106] nagyon hatékonynak bizonyultak. Ezen tesztek segítségével a PFSP-ben azt tudjuk megvizsgálni, hogy melyik munka lehet a sorrend első, illetve utolsó eleme. Tegyük fel például, hogy mondjuk egy r gépen a munkák lehetséges legkorábbi kezdési idejei rendre $(a_1,\ a_2,\ldots,a_N)$ és a lehetséges legkésőbbi befejezési idejei $(b_1,\ b_2,\ldots,b_N)$. Ha létezik olyan i munka, hogy minden $j \neq i$ munkára

$$a_j + \sum_{i=1}^{N} P_{ri} > \max_{1 \leq j \leq N}(b_j + P_{rj}) \tag{2.53}$$

egyenlőtlenség teljesül, akkor ebből az következik, hogy a permutáció első elemének az i munkának kell lennie. Az egyenlőtlenség ugyanis éppen azt jelenti, hogy bármely olyan permutáció esetén, melyben nem az i munka van az első helyen, az r gépen az utolsó munka befejezési ideje mindig nagyobb lesz, mint a munkáknak az adott gépen lévő legkésőbbi befejezési idejeinek a maximuma. Hasonló jellegű teszt segítségével szerezhetünk a permutáció utolsó elemére vonatkozó információt.

DOI: 10.15477/SZE.MMTDI.2017.009

3. fejezet

Ismétlődő permutatációs flow shop probléma (R-PFSP)

3.1. Az R-PFSP definíciója

Az iparban a gyártósorok ütemezési feladatai során sok esetben a megmunkálandó munkadarabok nem mind különbözőek. Ilyenkor ráaadásul az összes munkadarabhoz képest a különböző típusú munkadarabok száma általában kevés (például 10 típusú motor mindegyikéből szeretnénk húszat összeszerelni, azaz összesen 200 munkadarabunk van). A kiemelt gyakorlati jelentőségük miatt az ilyen ütemezési feladatoknak megfelelő speciális PFSP feladatokra [38]-ban bevezettük az **ismétlődő permutációs flow shop probléma** (R-PFSP) fogalmát.
Ismétlődő permutációs flow shop problémának nevezzük azon PFSP-ket, melyben vannak azonos típusú munkadarabok. Két munkadarab akkor azonos típusú, ha megmunkálási idejük mindegyik gépen megegyezik. Míg N darab különböző munkadarab esetén a lehetséges permutációk száma $N!$, addig az olyan R-PFSF-ben, melyben T különböző típus van és az egyes típusokból rendre n_1,n_2,$\dots$ n_T darab van ($n_1 + n_2 + \dots n_T = N$), a lehetséges permutációk száma $\frac{N!}{n_1! \cdot n_2! \cdots n_T!}$. Ez azt jelenti, hogy figyelembe véve az ismétlődéseket, a lehetséges permutációk száma, így a keresési tér is jelentősen csökkenthető.

3.2. Az R-PFSP modelljei

A következőkben bemutatjuk az R-PFSP három MILP modelljét, melyeket [38]-ban vezettünk be. A modellekben a következő jelöléseket fogjuk alkalmazni:
Paraméterek:

M a gépek száma

N a munkadarabok száma

T a különböző típusok száma

DOI: 10.15477/SZE.MMTDI.2017.009

n_t a t típusú munkadarabok száma ($1 \leq t \leq T;\ \sum n_t = N$)

P_{ri} az i típusú munkadarab megmunkálási ideje az r gépen

Folytonos változók:

C_{rj} a sorrend j. helyén álló munkadarab megmunkálásának befejezési ideje az r gépen

B_{rj} a sorrend j. helyén álló munkadarab megmunkálásának kezdési ideje az r gépen

X_{rj} állóidő az r gépen a sorrend j. munkadarabjának a megmunkálásának a kezdése előtt

Y_{rj} a sorrend j. munkadarabjának a várakozási ideje a pufferban, miután megmunkálták az az r gépen

$C_{\max}$ átfutási idő.

Bináris változók:

Z_{ij} értéke 1, ha a sorrend j. helyére egy i típusú munka kerül, különben az értéke 0.

3.2.1. Az R-Wilson modell

Az R-Wilson modell [38] Wilsonnak [114] a PFSP-re adott MILP modelljének az R-PFSP-re való módosított változata. Az R-Wilson modell feltételei biztosítják a következő feltételek teljesülését:

- Az i típusú munkadarabok száma a permutációban pontosan n_i.

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \leq i \leq T \tag{3.1}$$

- A sorrend mindegyik helyére pontosan egy típusú munkadarab kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \leq j \leq N \tag{3.2}$$

- Az első gépen nincsen állóidő.

$$B_{1j} + \sum_{i=1}^{T} P_{1i} Z_{ij} = B_{1,j+1} \qquad 1 \leq j \leq N-1 \tag{3.3}$$

- A sorrendben első munkadarabot az első gépen a 0 időpontban kezdjük el megmunkálni.

$$B_{11} = 0 \tag{3.4}$$

- A sorrendben első munkadarabnak egyetlen gép előtt sem kell várakoznia.

$$B_{r1} + \sum_{i=1}^{T} P_{ri} Z_{i1} = B_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{3.5}$$

- Addig nem kezdhetjük meg egy munkadarab megmunkálását egy gépen, míg az előző gépen be nem fejeztük az adott munkadarab megmunkálását.

$$B_{rj} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r+1,j} \qquad 1 \leq r \leq M-1;\ 2 \leq j \leq N \tag{3.6}$$

- Egy munkadarab megmunkálását addig nem kezdhetjük el egy gépen, míg a sorrendben őt közvetlenül megelőző munkadarab megmunkálását be nem fejeztük az adott gépen.

$$B_{rj} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r,j+1} \qquad 2 \leq r \leq M;\ 2 \leq j \leq N-1 \tag{3.7}$$

- Az átfutási időt szeretnénk minimalizálni.

$$\min\ C_{\max} = B_{MN} + \sum_{i=1}^{T} P_{Mi} Z_{iN} \tag{3.8}$$

Az R-PFSP-re adott R-Wilson modell a következőképpen összegezhető:
Minimalizáljuk (3.8)-at, az (3.1) – (3.7) feltételek mellett.

3.2.2. Az R-TS2 modell

A [38]-ban bevezetett R-TS2 modell a [99]-beli MILP modellnek a módosított változata az R-PFSP-re. Az R-TS2 modell egyenlőtlenségei garantálják a következő feltételek teljesülését:

- Az i típusú munkadarabok száma a permutációban pontosan n_i.

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \leq i \leq T \tag{3.9}$$

- A sorrend mindegyik helyére pontosan egy típusú munkadarab kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \leq j \leq N \tag{3.10}$$

- A sorrendben $(j+1)$. munkadarab befejezési ideje egy gépen nem lehet kisebb, mint a sorrend j. munkadarabjának a befejezési ideje ugyanazon a gépen plusz a sorrend $(j+1)$. munkadarabjának a megmunkálási ideje az adott gépen.

$$C_{rj} + \sum_{i=1}^{T} P_{ri} Z_{i,j+1} \leq C_{r,j+1}, \qquad 1 \leq r \leq M, 1 \leq j \leq N-1 \tag{3.11}$$

- Egy munkadarab befejezési ideje egy gépen nem lehet kisebb, mint a munkadarabnak a befejezési ideje a megelőző gépen plusz a munkadarab megmunkálási ideje az adott gépen.

$$C_{rj} + \sum_{i=1}^{T} P_{r+1,i} Z_{ij} \leq C_{r+1,j}, \qquad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{3.12}$$

- A sorrend első munkadarabját az első gépen nem lehet korábban befejezni, mint a sorrend első munkadarabjának a megmunkákási ideje az első gépen.

$$\sum_{i=1}^{T} P_{1i} Z_{i1} \leq C_{11} \tag{3.13}$$

- Az átfutási idő a sorrend utolsó elemének a befejezési ideje az utolsó gépen.

$$\min\ C_{\max} = C_{MN} \tag{3.14}$$

Az R-PFSP-re adott R-TS2 modell a következőképpen összegezhető:
Minimalizáljuk (3.14)-et, az (3.9) – (3.13) feltételek mellett.

3.2.3. Az R-WST modell

A harmadik MILP modell, a [38]-ban bemutatott R-WST modell Wagner [110], illetve Stafford és Tseng [95] modelljeinek a módosítása az R-PFSP feladatra. Az R-Wilson és R-TS2 modellekkel ellentétben az R-WST modell a gépek állóidejét ($Y_{r,j}$) és a munkadarabok várakozási idejét ($X_{r,j}$) használja az egyenlőtlenségekben a munkadarabok kezdési, illetve befejezési ideje helyett. Az egyenlőtlenségek biztosítják a következő feltételek teljesülését:

- Az i típusú munkadarabok száma a permutációban pontosan n_i.

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \leq i \leq T \tag{3.15}$$

- A sorrend mindegyik helyére pontosan egy típusú munkadarab kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \leq j \leq N \tag{3.16}$$

- A sorrend első munkadarabjának a megmunkálása folyamatos, azaz egyetlen gép előtt sem kell várakoznia.

$$Y_{r1} = 0, \qquad 1 \leq r \leq M-1 \tag{3.17}$$

- A sorrend $(j+1)$. tagját addig nem kezdhetjük el egy gépen, míg az előző gépen be nem fejeztük, illetve míg a sorrend j. tagját be nem fejeztük ugyanezen a gépen.

$$\sum_{i=1}^{T} P_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{T} P_{r+1,i} Z_{ij} + X_{r+1,j+1} + Y_{rj} \tag{3.18}$$
$$1 \leq r \leq M-1;\ 1 \leq j \leq N-1$$

  Ahhoz, hogy jobban látható legyen, hogy ez az egyenlőtlenség a fenti két dolgot fejezi ki, képzeljük el a sorrend j. tagjának az r gépen való befejezési illetve a sorrend $(j+1)$. tagjának az $r+1$ gépen a kezdési ideje közti időt. Ezt az időt kétféleképpen is nézhetjük. Egyrészt miután az r gépen befejeződött a j. munka, a gépnek van egy állóideje, mielőtt elkezdjük megmunkálni a sorrend $(j+1)$. tagját (ez éppen $X_{r,j+1}$). Ezután megmunkáljuk az r gépen a sorrend $(j+1)$. tagját, majd a $(j+1)$. munkadarab a pufferban várakozik, hogy elkezdhessük megmunkálni az $(r+1)$ gépen (ez a várakozási idő $Y_{r,j+1}$). Ezen 3 tag összege éppen az egyenlőtlenség bal oldala. Máshogyan is kiszámíthatjuk azonban ezt az időközt: miután az r gépen befejeződött a j. munka, a pufferban várakozik, hogy elkezdhessük megmunkálni az $(r+1)$ gépen (ennek az ideje Y_{rj}). Ezután megmunkáljuk az $(r+1)$ gépen a sorrend j. munkadarabját, majd ezután az $(r+1)$ gépen van egy állóidő a sorrend $(j+1)$. munkadarabjának elkezdéséig (ez az idő éppen $X_{r+1,j+1}$). Ezen három tag összege éppen az egyenlőtlenség jobb oldalát adja.

- Tetszőleges r gépen az első munka előtti állóidő megegyezik a sorrend első munkájának az első $r-1$ gépen való megmunkálási idejének összegével.

$$X_{r1} + Y_{r1} + \sum_{i=1}^{T} P_{ri} Z_{i1} = X_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{3.19}$$

- Az átfutási idő egyenlő az utolsó gépen a megmunkálási idők összegének és az állóidők

DOI: 10.15477/SZE.MMTDI.2017.009

összegének összegével.

$$\min\ C_{\max} = \sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{3.20}$$

Az R-WST modell az alábbi módon foglalható össze:
Minimalizáljuk (3.20)-at, az (3.15) – (3.19) feltételek mellett.

3.3. A PFSP és R-PFSP modelljei komplexitásának összehasonlítása

Lineáris programozásból jól ismert, hogy egy vegyes egészértékű programozási feladatban szereplő egészértékű-, illetve folytonos változók száma, továbbá a feladatban szereplő egyenlőtlenségek száma is fontos szerepet játszik a feladat megoldásának ”nehézségi fokában”. Ebben a részben ezen három mennyiség alapján összehasonlítjuk a klasszikus PFSP-nek az irodalomból korábban ismert 3 modelljét (Wilson, TS2, WST), illetve az R-PFSP-re bevezetett új modelleket. A PFSP és R-PFSP modelljei komplexitását a 3.1 táblázatban foglaltuk össze. A táblázatból jól látható, hogy az alapvető eltérés a klasszikus PFSP modelljei, illetve a nekik

3.1. táblázat. A PFSP és R-PFSP modelljei komplexitása

modell	bináris változók	folytonos változók	egyenlőtlenségek
Wilson	N^2	MN	$2MN - M + N + 1$
WST	N^2	$2MN - N + 1$	$MN + M + N + 1$
TS2	N^2	$MN + 1$	$2MN - M + N + 1$
R-Wilson	NT	MN	$2MN - M + T + 1$
R-WST	NT	$2MN - N + 1$	$MN + M + T + 1$
R-TS2	NT	$MN + 1$	$2MN - M + T + 1$

$N =$ munkák száma, $M =$ gépek száma, $T =$ típusok száma

megfelelő R-PFSP-beli modell között, hogy az R-PFSP-beli modell mindig kevesebb, nevezetesen T/N-szer annyi bináris változót tartalmaz, mint a neki megfelelő klasszikus modell. Ez abból adódik, hogy az ismétlődések figyelembevételénél a sorrend mindegyik helyére (N darab hely) elég volt megadni, hogy milyen típusú munka kerül oda (T lehetőség), míg a klasszikus modelleknél minden helyre meg kell adni, hogy milyen munka (N lehetőség) kerül oda. A folytonos változók száma a klasszikus modellekben és a nekik megfelelő R-változatban megegyezik, míg az R-PFSP-beli modellek a klasszikus változathoz képest kevesebb egyenlőtlenséget tartalmaznak. Ez ismét az ismétlődések figyelembevételéből adódik: az R-PFSP-nél minden típusra fel kell írni, hogy az adott típusból összesen hány munka van (T egyenlőtlenség), míg a

DOI: 10.15477/SZE.MMTDI.2017.009

PFSP-nél minden munkára meg kell adni, hogy pontosan 1 van belőle (N egyenlőtlenség).

Az R-Wilson, R-TS2 és R-WST modelleket egymással öszehasonlítva látható, hogy mindhárom modell ugyanannyi bináris változót tartalmaz. Az R-Wilson modell tartalmazza a legkevesebb folytonos változót, ennél az R-TS2 modell eggyel több folytonos változót tartalmaz. Azt mondhatjuk, hogy az R-WST modell nagyjából kétszer annyi folytonos változót tartalmaz, mint az R-Wilson, illetve az R-TS2 modellek. Ez abból adódik, hogy az R-Wilson, illetve az R-TS2 modellek a munkadaraboknak a gépeken a kezdési, illetve a befejezési idejét használják az egyenlőtlenségekben, míg az R-WST modell a gépeken az egyes munkadarabok előtti állóidőket és a munkadaraboknak a gépek utáni várakozási idejét használja a feltételekben. A legkevesebb egyenlőtlenség az R-WST modellben szerepel. Az R-Wilson és R-TS2 modellekben megegyezik a feltételek száma, és mindkét modell körülbelül kétszer annyi feltételt tartalmaz, mint az R-WST modell.

3.4. Az R-Wilson, R-TS2 és R-WST modellek relaxáltjai optimumának összehasonlítása

Egy MILP modell esetén, melyben a célfüggvénynek a minimumát keressük, ha a modellben szereplő egész értékű változóktól nem követeljük meg az egészértékűséget, akkor a modell egy relaxációját kapjuk. A relaxáció egy LP feladat, melynek az optimuma az eredeti feladat egy alsó korlátját adja . A CPLEX [24] is ezt az alsó korlátot használja kiindulási korlátként, így egy feladat különböző MILP modelljeinek összehasonlításánál lényeges lehet megvizsgálni, hogy melyik relaxáltnak a legkisebb az optimuma.

Az R-Wilson, R-TS2 és R-WST modellek mindegyike a Z_{ij} bináris változókat használja, így mindhárom modell relaxáltjában azt követeljük meg, hogy a Z_{ij} változók mindegyike 0 és 1 közötti valós szám legyen. Megmutatjuk, hogy az R-Wilson, R-TS2 és R-WST modellek relaxáltjainak az optimuma megegyezik.

3.1. Tétel. *Az R-Wilson, R-TS2 és R-WST modellek relaxáltjainak az optimumértéke megegyezik.*

Bizonyítás Először megmutatjuk, hogy az R-TS2 modell relaxáltjának az optimuma legfeljebb annnyi, mint az R-Wilson modell relaxáltjának az optimuma. Vegyük ehhez az R-Wilson modell relaxáltjának egy optimális megoldását; legyenek az optimális megoldásban szereplő változók értékei Z^o_{ij}, $(1 \leq i \leq T, 1 \leq j \leq N)$ és B^o_{rj}, $(1 \leq r \leq M, 1 \leq j \leq M)$. Ezen változókra tehát teljesülnek a (3.1) – (3.7) korlátozó feltételek, és az R-Wilson modell relaxáltjának az optimuma pedig $B^o_{MN} + \sum_{i=1}^{T} P_{Mi} Z^o_{iN}$.
Legyen most $1 \leq r \leq M$ és $1 \leq j \leq N$ esetén $C^o_{rj} = B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij}$. Megmutatjuk, hogy a

Z^o_{ij} és C^o_{rj} értékek az R-TS2 modell relaxáltjának egy megoldását adják, azaz teljesülnek rájuk a (3.9) – (3.13) feltételek.

Mivel az R-Wilson modell (3.1) feltétele megegyezik az R-TS2 modell (3.9) feltételével, ezért a Z^o_{ij} értékekre teljesül a (3.9) feltétel. Hasonlóan, az R-Wilson modell (3.2) feltétele megegyezik az R-TS2 modell (3.10) feltételével, így a Z^o_{ij} értékek kielégítik a (3.10) feltételt. Az R-Wilson modell (3.3) és (3.7) feltételei szerint

$$B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij} \leq B^o_{r,j+1} \qquad 1 \leq r \leq M;\ 1 \leq j \leq N-1$$

Mindkét oldalhoz hozzáadva a $\sum_{i=1}^{T} P_{ri} Z^o_{i,j+1}$ kifejezést és felhasználva hogy $C^o_{rj} = B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij}$, azt kapjuk, hogy

$$C^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{i,j+1} \leq C^o_{r,j+1}, \qquad 1 \leq r \leq M, 1 \leq j \leq N-1$$

azaz a Z^o_{ij}, C^o_{rj} értékekre teljesül az R-TS2 modell (3.11) feltétele. Az R-Wilson modell (3.5) feltétele szerint

$$B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij} \leq B^o_{r+1,j} \qquad 1 \leq r \leq M-1;\ 1 \leq j \leq N.$$

Mindkét oldalhoz hozzáadva a $\sum_{i=1}^{T} P_{r+1,i} Z^o_{ij}$ kifejezést és felhasználva a C^o_{rj} definícióját azt kapjuk, hogy

$$C^o_{rj} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{ij} \leq C^o_{r+1,j}, \qquad 1 \leq r \leq M-1, 1 \leq j \leq N$$

azaz a Z^o_{ij}, C^o_{rj} értékekre teljesül az R-TS2 modell (3.12) feltétele. A (3.4) feltétel szerint $B^o_{11} = 0$, így

$$\sum_{i=1}^{T} P_{1i} Z^o_{i1} \leq B^o_{11} + \sum_{i=1}^{T} P_{1i} Z^o_{i1} = C_{11}$$

vagyis a (3.13) feltétel is teljesül. Tehát a Z^o_{ij}, C^o_{rj} értékekre teljesül az R-TS2 modell összes feltétele. Továbbá a (3.8) és (3.14) alapján az R-TS2 modell relaxáltjának a Z^o_{ij}, C^o_{rj} értékekhez tartozó célfüggvényértéke megegyezik az R-Wilson modell relaxáltjának az optimumával, amiből az következik, hogy az R-TS2 modell relaxáltjának az optimuma legfeljebb annyi, mint az R-Wilson modell relaxáltjának az optimuma.

Most megmutatjuk, hogy az R-WST modell relaxáltjának az optimuma legfeljebb annyi, mint az R-TS2 modell relaxáltjának az optimuma. Legyen ehhez Z^o_{ij}, $(1 \leq i \leq T, 1 \leq j \leq N)$

és C^o_{rj}, $(1 \leq r \leq M, 1 \leq j \leq M)$ az R-TS2 modell olyan optimális megoldása, melyben

$$C^o_{11} = \sum_{i=1}^{T} P_{1i} Z^o_{i1} \quad \text{és} \quad C^o_{r1} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{i1} = C^o_{r+1,1}, \qquad 1 \leq r \leq M-1 \tag{3.21}$$

teljesül (ilyen megoldás létezik). Definiáljuk az X^o_{rj} és $Y^o_{r,j}$ értékeket a következőképpen:

$$\begin{aligned}
X^o_{r1} &= C^o_{r1} - \sum_{i=1}^{T} P_{ri} Z^o_{i1}, \qquad 1 \leq r \leq M \\
X^o_{rj} &= C^o_{rj} - C^o_{r,j-1} - \sum_{i=1}^{T} P_{ri} Z^o_{ij}, \quad 2 \leq j \leq N, 1 \leq r \leq M \\
Y^o_{rj} &= C^o_{r+1,j} - C^o_{rj} - \sum_{i=1}^{T} P_{r+1,i} Z^o_{ij}, \quad 1 \leq j \leq N, 1 \leq r \leq M-1
\end{aligned}$$

Megmutatjuk, hogy a $(Z^o_{ij}, X^o_{rj}, Y^o_{rj})$ értékek az R-WST modell relaxáltjának egy lehetséges megoldását adják. Az R-TS2 modell (3.11), (3.12) feltételei és (3.21) miatt az így definiált X^o_{rj}, Y^o_{rj} változók mindegyike nemnegatív. Mivel az R-WST modell (3.15) és (3.16) feltételei megegyeznek az R-TS2 modell (3.9) és (3.10) feltételeivel, ezért a Z^o_{ij} értékek teljesítik az R-WST modell relaxáltjának a (3.15) és (3.16) feltételeit. A (3.21) miatt

$$Y^o_{r1} = C^o_{r+1,1} - C^o_{r1} - \sum_{i=1}^{T} P_{r+1,i} Z^o_{i1} = 0, \quad 1 \leq r \leq M-1$$

azaz az Y^o_{r1} értékekre teljesül az R-WST modell (3.17) feltétele. Szintén a (3.21)-t felhasználva azt kapjuk, hogy

$$\begin{aligned}
X^o_{r+1,1} &= C^o_{r+1,1} - \sum_{i=1}^{T} P_{r+1,i} Z^o_{i1} = C^o_{r1} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{i1} - \sum_{i=1}^{T} P_{r+1,i} Z^o_{i1} = \\
&= X^o_{r1} + \sum_{i=1}^{T} P_{ri} Z^o_{i1} \quad 1 \leq r \leq M-1
\end{aligned}$$

ami azt jelenti, hogy az X^o_{r1} értékekre teljesül az R-WST modell (3.19) feltétele. Most fejezzük ki a $C^o_{r+1,j+1} - C^o_{rj}$ különbséget kétféleképpen. Egyrészt

$$\begin{aligned}
C^o_{r+1,j+1} - C^o_{rj} &= C^o_{r+1,j+1} - C^o_{r+1,j} + C^o_{r+1,j} - C^o_{rj} = X^o_{r+1,j+1} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{i,j+1} + \\
&+ Y_{rj} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{ij}.
\end{aligned}$$

Másrészt

$$C^o_{r+1,j+1} - C^o_{rj} = C^o_{r+1,j+1} - C^o_{r,j+1} + C^o_{r,j+1} - C^o_{rj} = Y_{r,j+1} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{i,j+1} + \\ + X^o_{r,j+1} + \sum_{i=1}^{T} P_{ri} Z^o_{i,j+1}.$$

A kétféle felírás összehasonlításából azt kapjuk, hogy

$$X^o_{r+1,j+1} + Y^o_{rj} + \sum_{i=1}^{T} P_{r+1,i} Z^o_{ij} = Y^o_{r,j+1} + X^o_{r,j+1} + \sum_{i=1}^{T} P_{ri} Z^o_{i,j+1}.$$

teljesül minden $1 \leq r \leq M-1$, $1 \leq j \leq N-1$ esetén, ami pontosan azt jelenti, hogy a $(Z^o_{ij}, X^o_{rj}, Y^o_{rj})$ értékekre teljesül az R-WST modell (3.18) feltétele. Beláttuk tehát, hogy a $(Z^o_{ij}, X^o_{rj}, Y^o_{rj})$ értékek az R-WST modell relaxáltjának egy lehetséges megoldását adják. Végezetül

$$C^o_{MN} = C_{M1} + \sum_{j=2}^{N} (C^o_{Mj} - C^o_{M,j-1}) = X^o_{M1} + \sum_{i=1}^{T} P_{Mi} Z^o_{i1} + \sum_{j=2}^{N} (X^o_{Mj} + \sum_{i=1}^{T} P_{Mi} Z^o_{ij}) = \\ = \sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X_{Mp}$$

ami azt jelenti, hogy az R-WST modell relaxáltjának a $(Z^o_{i,j}, X^o_{rj}, Y^o_{rj})$ megengedett megoldásához tartozó célfüggvényértéke megegyezik az R-TS2 modell relaxáltjának optimumával. Ezért az R-WST modell relaxáltjának az optimuma legfeljebb annyi, mint az R-TS2 modell relaxáltjának az optimuma.

Most megmutatjuk, hogy az R-Wilson modell relaxáltjának az optimuma legfeljebb annyi, mint az R-WST modell relaxáltjának az optimuma. Legyen ehhez $(Z^o_{i,j}, X^o_{rj}, Y^o_{rj})$ az R-WST modell relaxáltjának egy optimális megoldása. Először megmutatjuk, hogy minden $1 \leq j \leq N$, $2 \leq r \leq M$ esetén fennáll az

$$\sum_{i|i\leq j} X^o_{ri} + \sum_{i|i<j} \sum_{l=1}^{T} P_{rl} Z^o_{li} = \sum_{i|i\leq j} X^o_{r-1,i} + \sum_{i|i\leq j} \sum_{l=1}^{T} P_{r-1,l} Z^o_{li} + Y^o_{r-1,j} \tag{3.22}$$

egyenlőség. A bizonyítást j-re vonatkozó teljes indukcióval végezzük. A $j = 1$ esetben azt kell bizonyítanunk, hogy

$$X^o_{r1} = X^o_{r-1,1} + \sum_{i=1}^{T} P_{r-1,i} Z^o_{i1} + Y^o_{r-1,1}$$

ami viszont éppen az R-WST modell (3.19) feltétele, így valóban teljesül. Tegyük fel ezután, hogy a (3.22) egyenlet telesül valamely $1 \leq j$ esetén. Ekkor

$$\sum_{i|i\leq j+1} X^o_{ri} + \sum_{i|i<j+1} \sum_{l=1}^{T} P_{rl} Z^o_{li} = \sum_{i|i\leq j} X^o_{ri} + \sum_{i|i<j} \sum_{l=1}^{T} P_{rl} Z^o_{li} + X_{r,j+1} + \sum_{l=1}^{T} P_{rl} Z_{lj} =$$

$$= \sum_{i|i\leq j} X^o_{r-1,i} + \sum_{i|i\leq j} \sum_{l=1}^{T} P_{r-1,l} Z^o_{li} + Y^o_{r-1,j} + X_{r,j+1} + \sum_{l=1}^{T} P_{rl} Z_{lj} = \sum_{i|i\leq j} X^o_{r-1,i} +$$

$$+ \sum_{i|i\leq j} \sum_{l=1}^{T} P_{r-1,l} Z^o_{li} + X^o_{r-1,j+1} + Y^o_{r-1,j+1} + \sum_{i=1}^{T} P_{r-1,i} Z_{i,j+1} = \sum_{i|i\leq j+1} X^o_{r-1,i} +$$

$$+ \sum_{i|i\leq j+1} \sum_{l=1}^{T} P_{r-1,l} Z^o_{li} + Y^o_{r-1,j+1}.$$

Ezzel beláttuk, hogy a (3.22) állítás $j+1$ esetén is igaz (a bizonyítás második sorának az első egyenlőségénél az indukciós feltevést, míg a második sor végén lévő egyenlőségnél az R-WST modell (3.18) feltételét használtuk), ezért az állítás minden j-re teljesül.
Definiáljuk ezek után a B_{rj} értékeket a következő módon: legyen

$$B^o_{rj} = \sum_{i|i\leq j} X^o_{ri} + \sum_{i|i<j} \sum_{l=1}^{T} P_{rl} Z^o_{li}$$

Megmutatjuk, hogy a (Z^o_{ij}, B^o_{rj}) értékek kielégítik az R-Wilson modell relaxáltjának a feltételeit a (3.3) és (3.4) feltételek kivételével. Mivel az R-WST és R-Wilson modellek első két feltétele megegyezik, ezért a (Z^o_{ij}) értékekre teljesül az R-Wilson modell relaxáltjában a (3.1) és (3.2) feltétel. Az R-WST modell (3.19) feltételéből és a B^o_{r1} változók definíciójából azonnal következik, hogy a (Z^o_{ij}, B^o_{rj}) értékekre teljesül a R-Wilson modell (3.5) feltétele. A B^o_{rj} értékek definícióját, a (3.22) egyenlőséget és az Y^o_{rj} értékek nemnegativitását felhasználva azt kapjuk, hogy

$$B^o_{r+1,j} = \sum_{i|i\leq j} X^o_{r+1,i} + \sum_{i|i<j} \sum_{l=1}^{T} P_{r+1,l} Z^o_{li} = \sum_{i|i\leq j} X^o_{r,i} + \sum_{i|i\leq j} \sum_{l=1}^{T} P_{rl} Z^o_{li} + Y^o_{rj} =$$

$$= \sum_{i|i\leq j} X^o_{ri} + \sum_{i|i<j} \sum_{l=1}^{T} P_{rl} Z^o_{li} + \sum_{i=1}^{T} P_{ri} Z_{lj} + Y^o_{rj} = B^o_{r,j} + \sum_{i=1}^{T} P_{ri} Z^o_{lj} + Y^o_{rj} \geq$$

$$\geq B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij}$$

azaz teljesül az R-Wilson modell relaxáltjának a (3.5) feltétele. Hasonlóan adódik, hogy

$$B_{r,j+1} = \sum_{i|i\leq j+1} X^o_{ri} + \sum_{i|i<j+1} \sum_{l=1}^{T} P_{rl} Z^o_{li} = \sum_{i|i\leq j} X^o_{ri} + \sum_{i|i<j} \sum_{l=1}^{T} P_{rl} Z^o_{li} + X^o_{r,j+1} +$$
$$+ \sum_{i=1}^{T} P_{ri} Z^o_{ij} = B^o_{rj} + X_{r,j+1} + \sum_{i=1}^{T} P_{ri} Z^o_{ij} \geq B^o_{rj} + \sum_{i=1}^{T} P_{ri} Z^o_{ij}$$

azaz teljesül az R-Wilson modell (3.6) feltétele. Végezetül a definíció alapján

$$\sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X^o_{Mp} = B^o_{MN} + \sum_{i=1}^{T} P_{Mi} Z^o_{iN}$$

azaz az R-Wilson modell relaxáltjában a (Z^o_{ij}, B^o_{rj}) értékekhez tartozó célfüggvény értéke megegyezik az R-WST modell relaxáltjának az optimumával.
A (Z^o_{ij}, B^o_{rj}) értékek tehát kielégítik az R-Wilson modell relaxáltjának feltételeit a (3.3) és (3.4) feltételek kivételével. Csökkentsük ezután a B_{1j} értékeket úgy, hogy a (3.3) és (3.4) feltételek is teljesüljenek. A többi feltétel továbbra is teljesülni fog és a célfüggvény értéke sem csökkenhet. Ebből már következik, hogy az R-Wilson modell relaxáltjának az optimuma legfeljebb akkora, mint az R-WST modell relaxáltjának az optimuma.
Így tehát azt kaptuk, hogy az R-TS2 modell relaxáltjának optimuma legfeljebb annyi, mint az R-Wilson modell relaxáltjának az optimuma; az R-WST modell relaxáltjának optimuma legfeljebb annyi, mint az R-TS2 modell relaxáltjának az optimuma; míg az R-Wilson modell relaxáltjának az optimuma legfeljebb annyi, mint az R-WST modell relaxáltjának az optimuma. Ebből pedig az következik, hogy az R-Wilson, R-TS2 és R-WST modellek relaxáltjainak az optimuma megegyezik.

3.5. Benchmark feladatok

A modellek összehasonlításához kétféle kísérletet végeztünk el. Az 1. kísérletben kis méretű feladatokon a futási idők alapján összehasonlítottuk egymással az R-TS2, R-Wilson és R-WST modelleket, illetve összehasonlítottuk a TS2, Wilson és WST modelleket az R-verziójukkal. Ehhez egy 20 cellából álló tesztkészletet hoztunk létre. Minden cellához 5 tesztfeladat tartozott, az azonos cellában lévő feladatoknál az M és N értékek megegyeztek. Az M értékei a $\{6,7,8,9,10\}$ halmazból, T értékei pedig a $\{6,7,8,9\}$ halmazból kerültek ki. Az ismétlődések száma minden példában 5 volt, azaz a munkák száma $\{30,35,40,45\}$ lehetett.
A második kísérletben az R-TS2, R-Wilson és R-WST modelleket nagy méretű feladatokon hasonlítottuk össze a a legjobb célfüggvényértékek, illetve az alsó korlátok alapján. Ehhez egy 12 cellából álló tesztkészletet hoztunk létre 5 feladattal cellánként. A gépek száma a $\{20,25\}$

DOI: 10.15477/SZE.MMTDI.2017.009

halmazból, a típusok (T) száma a $\{10,15\}$ halmazból míg a munkák száma a $\{120,180,240\}$ halmazból került ki. Minden típusból ugyanannyi munka volt, azaz $n_1 = n_2 \cdots = n_T = N/T$ teljesült. Ezen kívül a 2. kísérletben azt is megvizsgáltuk, hogy a CPLEX 3 lehetséges paraméterének módosítása milyen hatással van a modellek által adott megoldásokra. Mivel a szakirodalomban a PFSP különböző megoldó módszereit a Taillard-féle tesztkészleten [104] szokták összehasonlítani, ezért a megmunkálási időket mind a kis, mind a nagy méretű példák esetén a Taillard tesztkészlethez használt módon generáltam: minden megmunkálási idő egy, az $[1,99]$ intervallumból véletlenszerűen választott szám volt.

3.6. Futási eredmények

3.6.1. Az R-PFSP modelljeinek az összehasonlítása kis méretű feladatokon

Az R-Wilson modell a 100 feladat mindegyikében megtalálta az optimális megoldást a 10 perces időkorlát alatt. Az R-TS2 modell 1 feladatnál nem találta meg 10 perc alatt az optimális megoldást (a kapott legjobb célfüggvényérték 2223, míg az optimum 2215 volt). Az R-WST modell 3 esetben nem adott optimális megoldást; ezekben a példákban a kapott legjobb célfüggvényértékek (az optimumok) $(3212(3181);\ 2240(2215);\ 2369(2868)$ voltak. A modellek futási idejének cellánkénti átlagát, illetve szórását a 3.2. táblázat tartalmazza.
A 20 cellából 15-nél az R-Wilson modellé, 3 cellánál az R-WST modellé, 2 cellánál pedig az R-TS2 modellé volt a legkisebb átlagos futásidő. A futásidők szórása 15 cellánál az R-Wilson modellnél, 3 cellánál az R-TS2 modellnél, 2 cellánál pedig az R-WST modellnél volt a legkisebb. A 100 feladatból 44-nél az R-Wilson modell, 37-nél az R-WST modell és 19 esetben az R-TS2 modell adta meg leggyorsabban az optimális megoldást. A 3 modellt az egyes feladatokon a futásidő szerinti növekvő sorba rendezve a 100 feladaton az R-Wilson modell átlagos rangja $1,76$, az R-WST modell átlagos rangja $2,05$ míg az R-Wilson modell átlagos rangja $2,21$ volt. Mindezek alapján azt mondhatjuk, hogy a kis méretű feladatoknál a futásidők alapján a legjobb az R-Wilson modell, őt a követi az R-WST modell és a harmadik az R-TS2 modell.

Ezután minden cellában a legnehezebb feladatot (vagyis összesen 20 feladatot) megpróbáltuk megoldani a Wilson, TS2 és WST modellekkel is. A futásidőket a 3.11. táblázat tartalmazza. Az R-Wilson modell mind a 20 feladatnál 10 percnél kevesebb idő alatt megadta az optimális megoldást, az R-TS2 és R-WST modellek 1 feladatnál ($M = 10, T = 7$ esetén) nem adtak 10 percen belül optimális megoldást. Ezzel szemben a klasszikus modellek csak a 6 illetve 7 gépes feladatokat tudták megoldani 10 percen belül. 8 gép esetén a Wilson modell 1, az TS2 és WST modellek 3, 9 gép esetén a Wilson és WST modellek 1, a TS2 modell 2, míg 10 gép esetén a Wilson modell 3, a TS2 és WST modellek 4 feladatnál nem adták meg 10 percen

DOI: 10.15477/SZE.MMTDI.2017.009

3.2. táblázat. Az R-PFSP modelljei futási idejének átlaga és szórása a kis méretű feladatoknál (másodpercben)

		Átlag			Szórás		
M	T	R-TS2	R-Wilson	R-WST	R-TS2	R-Wilson	R-WST
6	6	1,71	0,65	0,79	1,89	0,48	0,67
	7	0,66	0,62	0,51	0,38	0,31	0,38
	8	0,88	0,69	0,9	0,56	0,35	0,51
	9	1,08	0,93	0,47	0,87	0,81	0,20
7	6	3,3	2,84	4,08	6,12	5,02	8,11
	7	1,35	0,61	2,55	1,21	0,35	3,94
	8	1,52	1,32	1,97	2,16	1,60	2,17
	9	0,98	0,76	1,02	0,62	0,61	1,09
8	6	3,38	3,04	2,94	3,01	3,82	3,08
	7	4,79	6,40	9,63	8,12	10,51	19,10
	8	8,98	8,98	10,57	11,96	13,84	16,63
	9	235,65	124,78	*	236,83	128,39	*
9	6	9,66	6,00	16,06	17,06	8,79	25,19
	7	2,79	3,02	4,58	2,95	2,59	4,35
	8	10,77	6,62	6,91	13,65	6,44	4,75
	9	8,84	5,03	13,98	8,84	5,43	18,80
10	6	7,62	5,01	13,31	7,77	4,97	14,32
	7	*	213,52	*	*	291,46	*
	8	49,58	26,97	*	88,82	35,57	*
	9	34,06	17,63	71,66	57,07	30,32	128,02

$T =$ típusok száma, $M =$ gépek száma, munkák száma $N = 5T$
*: legalább 1 feladat esetén nem kaptunk optimális megoldást a 10 perces futásidő alatt

belül az optimális megoldást. Abban az esetben, amikor a futásidőn belül nem kaptunk optimális megoldást, a 3.11. táblázatban zárójelben feltüntettük a kapott célfüggvényértéknek az optimumtól való relatív eltérését, azaz a $100 \cdot (C_{\max}^{\text{modell}} - C_{\text{opt}})/C_{\text{opt}}$ értéket, ahol $C_{\max}^{\text{modell}}$ a modell által az adott feladatra kapott legjobb célfüggvényérték, C_{opt} pedig a feladat optimuma. Az R-Wilson modell mind a 20 cellában jóval kevesebb idő alatt adta meg az optimális megoldást (pl. $M = 10, T = 6$ esetén több mint 28-szor gyorsabb volt), mint a Wilson modell (sőt, a Wilson modell 5 feladatnál nem is adott optimális megoldást). Az R-TS2 modell a 20-ból 19 feladatnál jóval gyorsabb (pl. $M = 8, T = 6$ estén több mint 18-szor) volt, mint a TS2 modell (a TS2 modell 8 feladatnál nem is adott optimális megoldást a futásidő alatt); 1 feladatnál

DOI: 10.15477/SZE.MMTDI.2017.009

3.3. táblázat. A TS2, Wilson, WST, R-TS2, R-Wilson és R-WST modellek futásideje (másodpercben), illetve zárójelben a kapott célfüggvényértéknek az optimumtól való relatív eltérése, amikor nem kaptunk optimális megoldást

M	T	R-TS2	R-Wilson	R-WST	TS2	Wilson	WST
6	6	2,13	1,45	1,61	6,32	10,70	38,61
	7	1,15	1,09	1,10	3,62	3,93	2,61
	8	1,83	1,24	1,56	6,80	5,61	3,95
	9	2,58	2,37	0,49	8,01	7,09	5,47
7	6	14,24	11,80	18,59	140,16	137,53	119,99
	7	3,04	1,12	9,45	11,85	12,51	159,08
	8	5,35	4,11	5,11	41,88	47,60	90,75
	9	1,36	1,87	2,78	7,03	10,93	5,93
8	6	7,96	9,62	7,75	147,19	171,56	122,56
	7	19,28	24,95	43,76	*(0,90)	377,85	*(0,09)
	8	29,42	33,43	39,39	*(0)	460,17	*(0,03)
	9	539,68	125,29	522,83	*(0,51)	*(1,05)	*(0,18)
9	6	40,06	21,39	60,02	*(0,45)	*(0,70)	*(1,10)
	7	6,68	5,71	10,00	51,99	46,92	40,73
	8	34,10	17,41	12,29	*(0,23)	82,96	86,46
	9	13,40	8,03	45,57	119,21	57,22	257,87
10	6	15,94	8,97	29,78	*(0)	523,134	*(0)
	7	*(0,36)	592,97	*(1,13)	*(2,03)	*(1,94)	*(0,95)
	8	186,58	87,90	362,56	*(1,55)	*(1,73)	*(1,04)
	9	134,86	71,51	297,71	*(0,09)	*(0,09)	*(0,09)

$T =$ típusok száma, $M =$ gépek száma, munkák száma $N = 5T$
*: nem kaptunk optimális megoldást a 10 perces futásidő alatt

($M = 10$, $T = 7$ esetén) sem az R-TS2 sem a TS2 modell nem adott optimális megoldást; ennél a feladatnál az R-TS2 modellnek az optimumtól való relatív eltérése jóval kisebb volt, mint a TS2 modellé (0,36, illetve 2,03). Az R-WST modell a 20-ból 19 feladatnál jóval (pl. $M = 7$, $T = 7$ esetén több mint 17-szer) gyorsabb volt, mint a WST modell (a WST modell 8 feladatnál nem is adott optimális eredményt). Egy feladatnál ($M = 10$, $T = 7$) sem az R-WST sem a WST modell nem adott optimális eredményt 10 perc alatt; ebben az esetben a WST modellnek az optimumtól való relatív eltérése egy kicsit kisebb volt, mint az R-WST modellé (0,95, illetve 1,13). Továbbá a 20 feladatból 19 esetben az R-TS2, R-Wilson és R-WST modellek leglassabbika is gyorsabban oldotta meg az adott feladatot, mint a TS2, Wilson, WST modellek közül a legjobb. Ezek alapján megállapíthatjuk, hogy az új modellekkel

DOI: 10.15477/SZE.MMTDI.2017.009

(R-Wilson, R-TS2, R-WST) nagyobb feladatok, és gyorsabban oldhatók meg, mint a klasszikus modellekkel (Wilson, TS2, WST).

3.6.2. Az R-PFSP modelljeinek az összehasonlítása nagy méretű feladatokon

A nagy méretű feladatok esetén azért, hogy a futásidő alatt a CPLEX biztosan találjon kezdő megoldást, mindegyik modellnél megadtam a munkák egy kiinduló sorrendjét (azaz a Z_{ij} bináris változók értékét adtam meg, és ezzel a kezdő megoldással indult el a CPLEX.) A kiinduló megoldás a következő volt: a sorrend elejére kerültek az első típusú munkák, őket követték a második típusú munkák és így tovább.

A 60 feladat közül az R-TS2 és R-Wilson modellek két feladatban adtak optimális megoldást, míg az R-WST modell egy feladatnál adott optimális megoldást (a futásidőn belül). A három modell közül az R-TS2 modell 12, az R-Wilson modell 18 míg az R-WST modell 34 esetben esetben adta a legjobb eredményt (azaz a legkisebb felső korlátot). Az R-TS2 modell 33, az R-Wilson modell 31 míg az R-WST modell 22 esetben esetben adta a legjobb (azaz legnagyobb) alsó korlátot.

Minden feladat esetén meghatároztam mindhárom modell relatív eltérését (RELGAP-jét) is az alábbi módon:

$$RELGAP_{\text{modell}} = 100 \cdot \frac{C_{\max}^{\text{modell}} - L_{best}}{C_{\max}^{\text{modell}}}$$

ahol $C_{\max}^{\text{modell}}$ a modell által az adott feladatra kapott eredmény, L_{best} pedig a három modell által kapott 3 alsó korlát maximuma. A RELGAP értékek cellánkénti átlagát, illetve szórását a 3.4 táblázat tartalmazza.

Az R-WST modellnek mind a 12 cella esetén az átlagos RELGAP-je és a RELGAP-ek szórása is kisebb volt, mint az R-Wilson modellnek. A 12 cellából 9-ben az R-WST átlagos RELGAP-je kisebb volt, mint az R-TS2 modellé, továbbá 10 cellánál az R-WST modell RELGAP-jének szórása kisebb volt, mint az R-TS2 modellé. Az R-TS2 modellnek 6 cellában volt kisebb az átlagos RELGAP-je, mint az R-Wilson modellé és az R-TS2 modell RELGAP-jének szórása 8 cellában volt kisebb, mint az R-Wilson modellé. Az eredmények alapján azt mondhatjuk, hogy a nagy méretű feladatokon a CPLEX alapbeállításait használva és triviális kiinduló megoldást megadva a RELGAP-ek alapján az R-WST modell volt a legjobb, míg az R-TS2 és R-Wilson modellek nagyjából egyformán teljesítettek.

Ezután megvizsgáltam, hogy a CPLEX beállításai befolyásolják-e a három modell rangsorát. Három beállítási opciót vizsgáltam:

- egy adott csúcsban hogyan válasszuk ki a szétválasztandó változót (varsel)
- milyen intenzitással alkalmazzunk vágásokat (cuts)

DOI: 10.15477/SZE.MMTDI.2017.009

3.4. táblázat. RELGAP értékek átlaga és szórása

			Átlag			Szórás		
T	M	N	R-TS2	R-Wilson	R-WST	R-TS2	R-Wilson	R-WST
10	20	120	1,41	1,05	0,72	1,17	1,23	0,79
10	20	180	6,93	5,87	2,43	10,61	10,74	1,89
10	20	240	0,62	5,89	1,62	0,68	11,57	2,55
10	25	120	3,26	5,43	4,25	2,15	5,89	4,17
10	25	180	10,81	7,81	3,49	13,46	10,69	4,08
10	25	240	14,62	12,01	6,17	12,09	15,61	3,95
15	20	120	5,50	12,53	6,46	6,16	14,38	5,43
15	20	180	12,89	13,17	5,04	12,32	9,05	2,31
15	20	240	9,54	17,5	3,29	9,21	10,15	3,25
15	25	120	8,60	9,70	5,74	3,68	4,26	2,55
15	25	180	17,78	14,38	8,30	10,47	8,21	6,41
15	25	240	17,40	15,42	4,51	12,06	10,01	5,39

$T =$ típusok száma, $M =$ gépek száma, $N =$ munkák száma, cellánként 5 feladat
Minden típusból ugyanannyi darab van: $n_1 = n_2 = \cdots = n_T = N/T$
10 perces futásidő; CPLEX alapbeállításai

- milyen gyakran alkalmazzuk a csúcs heurisztikát (heurfreq).

Mindhárom opciónál 2 lehetséges beállítást vizsgáltam. A varsel opciónál az egyik beállítás az alapbeállítás volt, amikor is a szétválasztandó változót automatikusan választjuk (varsel=0), míg a másik beállítás az erős vágások használata volt (varsel=3). A cuts opciónál az egyik beállítás szintén az alapbeállítás volt, amikor automatikusan generáljuk a vágásokat (cuts=0) míg a másik esetben agresszívan próbálunk vágásokat generálni (cuts=2). A heurfreq opciónál az egyik beállítás szintén az alapbeállítás volt (heurfreq=0), míg a másik esetben minden huszadik csúcsnál alkalmaztuk a csúcs heurisztikát (heurfreq=20). Így a három opciót figyelembe véve összesen 8-féle beállítás volt. Mindegyik beállítás leírható egy számhármassal, ahol az első szám a varsel opció értékét (0 vagy 3), a második szám a cuts opció értékét (0 vagy 2) a harmadik pedig a heurfreq opció értékét (0 vagy 20) adja meg.

Mind a 8 beállítás esetén megvizsgáltam, hogy az egyes modellek a 60 feladatból hány esetén adták a legjobb eredményt, hány esetben adták a legjobb alsó korlátot, illetve mennyi volt az egyes modellek rangjának az átlaga a legjobb eredmények, illetve az alsó korlátok esetén. A rangszámításnál a következőképpen jártam el: ha egy feladatnál 2 modell került holtversenyben

DOI: 10.15477/SZE.MMTDI.2017.009

az első helyre (2. helyre), akkor mindkét modell rangja $1{,}5$ ($2{,}5$) lett, míg hármas holtverseny esetén mindhárom modell rangja 2 lett (így mindegyik feladatnál a három modell rangjának az összege mindig 6 volt). Az eredményeket a 3.5. és 3.6. táblázatok tartalmazzák.

3.5. táblázat. Az R-PFSP modelljeinek összehasonlítása a felső korlátok alapján a CPLEX különböző beállításai esetén a nagy feladatokon

			Hány esetben adott legjobb célfüggvényértéket			Rangok átlaga a célfüggvény-értékek alapján		
v	c	h	R-TS2	R-Wilson	R-WST	R-TS2	R-Wilson	R-WST
0	0	0	12	18	34	2,18	2,22	1,6
3	0	0	12	11	40	2,16	2,3	1,54
0	2	0	11	10	43	2,25	2,37	1,38
0	0	20	10	18	35	2,3	2,06	1,64
3	2	0	8	12	42	2,34	2,25	1,4
3	0	20	14	16	32	2,27	2,07	1,67
0	2	20	14	14	34	2,17	2,1	1,73
3	2	20	18	14	32	2,17	1,99	1,88

$v =$ varsel opció $\in \{0{,}3\}$, $c =$ cuts opció $\in \{0{,}2\}$, $h =$ heurfreq opció $\in \{0{,}20\}$
Összesen 60 feladat; $M \in \{20{,}25\}$, $N \in \{120{,}180{,}240\}$, $T \in \{10{,}15\}$
10 perc futásidő

A 3.5. táblázatból látható, hogy mind a 8 beállítás esetén az R-WST modell adta a legtöbb feladatnál (mindegyik beállításnál az esetek legalább 53 százalékában) a legkisebb célfüggvényértéket és mindegyik beállításnál az R-WST modell rangja volt a legkisebb, azaz a 3 modell közül a legjobb célfüggvényértékek alapján mindegyik bellításnál az R-WST modell volt a legjobb. A táblázat alapján továbbá azt mondhatjuk, hogy a legjobb célfüggvényérték alapján az R-TS2 és R-Wilson modellek a beállítástól függetlenül nagyjából egyformán teljesítettek. Így összességében az mondható, hogy a különböző beállítások a modellek legjobb célfüggvény szerinti sorrendjét nem befolyásolták. A 3.6. táblázatból az olvasható ki, hogy a modelleknek az alsó korlátok szerinti sorrendjét befolyásolta a beállítás. A 8 beállításból 3 esetben ($(v{,}c{,}h) \in \{(3{,}0{,}0),(0{,}0{,}20),(3{,}2{,}20)\}$) az alsó korlátok szempontjából a 3 modell nagyjából azonos eredményeket adott. 3 beállításnál ($(v{,}c{,}h) \in \{(0{,}0{,}0),(0{,}2{,}0),(3{,}2{,}0)\}$) az R-Wilson és R-TS2 modellek jobb alsó korlátokat adtak, mint az R-WST modell, az R-Wilson és R-TS2 modellek között viszont nem volt különbség. Egy beállításnál $(v{,}c{,}h) = (0{,}2{,}20)$ az R-Wilson modell adta a legjobb alsó korlátokat, míg a két másik modell eredményei lényegesen nem tértek el egymástól. Végezetül a $(v{,}c{,}h) = (3{,}0{,}20)$ beállítás esetén az alsó korlátokat tekintve a WST modell volt a legjobb, őt követte az R-Wilson modell, majd az R-WST modell.

Ezután megvizsgáltam, hogy melyik modell és melyik beállítással adja a legjobb

DOI: 10.15477/SZE.MMTDI.2017.009

3.6. táblázat. Az R-PFSP modelljeinek összehasonlítása az alsó korlátok alapján a CPLEX különböző beállításai esetén a nagy feladatokon

			Hány esetben adott legjobb alsó korlátot			Rangok átlaga az alsó korlátok alapján		
v	c	h	R-TS2	R-Wilson	R-WST	R-TS2	R-Wilson	R-WST
0	0	0	33	31	22	1,83	1,94	2,23
3	0	0	29	34	32	2,05	1,94	2,01
0	2	0	35	40	20	1,85	1,78	2,37
0	0	20	25	26	33	2,06	2,04	1,9
3	2	0	33	36	27	1,9	1,92	2,18
3	0	20	22	29	39	2,24	2,05	1,71
0	2	20	24	36	27	2,05	1,87	2,07
3	2	20	29	30	32	2,01	1,98	2,01

$v =$ varsel opció $\in$ 0,3, $c =$ cuts opció $\in$ 0,2, $h =$ heurfreq opció $\in$ 0,20
Összesen 60 feladat; $M \in \{20,25\}$, $N \in \{120,180,240\}$, $T \in \{10,15\}$
10 perc futásidő

célfüggvényértéket, illetve legjobb alsó korlátot. Így összesen 24 módszert (3 modell x 8 beállítás) hasonlítottam össze. Az egyes módszerek rangjait a 3.7. táblázat tartalmazza.

3.7. táblázat. A 24 módszer (modell x beállítás) összehasonlítása a CPLEX különböző beállításai esetén a nagy feladatokon

			Rangok átlaga a legjobb célfüggvényérték alapján			Rangok átlaga az alsó korlátok alapján		
v	c	h	R-TS2	R-Wilson	R-WST	R-TS2	R-Wilson	R-WST
0	0	0	13,09	13,84	8,54	10,97	11,65	14,48
3	0	0	15,63	15,90	10,31	9,02	8,58	8,04
0	2	0	15,45	16,78	9,93	12,33	11,06	15,16
0	0	20	12,18	11,38	6,97	15,68	15,38	15,58
3	2	0	16,72	16,61	10,23	9,16	9,51	11,31
3	0	20	13,23	12,08	8,61	13,83	13,13	10,83
0	2	20	13	12,08	9,64	15,62	13,13	15,76
3	2	20	13,96	12,76	11,05	13,27	13,15	13,41

$v =$ varsel opció $\in \{0,3\}$, $c =$ cuts opció $\in \{0,2\}$, $h =$ heurfreq opció $\in \{0,20\}$
Összesen 60 feladat; $M \in \{20,25\}$, $N \in \{120,180,240\}$, $T \in \{10,15\}$
10 perc futásidő

DOI: 10.15477/SZE.MMTDI.2017.009

A 3.7. táblázat szerint a célfüggvényértékek alapján a legkisebb rangja az R-WST modellnek a heurfreq$=20$ beállítással volt. Érdemes megjegyezni, hogy az R-WST modellt a 8 közül bármelyik beállítással használva a kapott módszer rangja (a legjobb célfüggvényérték alapján) kisebb mind a 16, az R-TS2, illetve R-Wilson modellt használó módszer rangjánál. Az R-WST-t tartalmazó módszerek további összehasonlításához minden feladatnál az R-WST modell mind a 8 beállításához kiszámítottam az $U^*_{R-WST,b} = 100 \cdot (C^{R-WST,b}_{\max} - U_{\text{best}})/C^{R-WST,b}_{\max}$ értéket, ahol $C^{R-WST,b}_{\max}$ az R-WST modellnek az adott beállítással az adott feladatra adott célfüggvényértéke, míg U_{best} a 24 módszer által adott célfüggvényértékek minimumával egyezik meg. Az $U^*_{R-WST,b}$ értékek cellánkénti átlagát a 3.8. táblázat tartalmazza.

3.8. táblázat. Az R-WST modell különböző beállításaihoz tartozó $U^*_{R-WST,b}$ értékek cellánkénti átlaga

			Beállítások							
			$v=0$ $c=0$ $h=0$	$v=3$ $c=0$ $h=0$	$v=0$ $c=2$ $h=0$	$v=0$ $c=0$ $h=20$	$v=3$ $c=2$ $h=0$	$v=3$ $c=0$ $h=20$	$v=0$ $c=2$ $h=20$	$v=3$ $c=2$ $h=20$
T	M	N								
10	20	120	0,16	0,43	0,29	0,28	0,72	0,55	0,42	0,75
10	20	180	1,47	0,51	0,44	0,84	2,13	1,03	1,29	2,14
10	20	240	1,05	0,79	0,81	0,82	1,85	0,99	0,59	1,24
10	25	120	2,38	2,30	3,11	1,59	0,86	1,52	2,70	1,46
10	25	180	2,62	2,29	2,95	1,79	2,74	1,80	2,08	2,28
10	25	240	3,09	3,85	4,87	1,87	4,57	1,92	2,74	2,57
15	20	120	3,43	4,06	1,88	2,03	0,62	2,51	2,06	2,15
15	20	180	2,76	2,76	1,43	1,22	1,35	1,22	2,09	2,33
15	20	240	1,08	0,94	3,06	1,36	1,33	1,04	2,06	1,18
15	25	120	1,16	3,12	3,86	0,53	4,51	2,32	5,72	5,66
15	25	180	3,64	2,52	6,31	3,46	5,71	2,25	4,05	3,47
15	25	240	1,79	1,80	2,37	1,37	2,35	1,26	0,44	0,44

$T =$ típusok száma, $M =$ gépek száma, $N =$ munkák száma, minden típusból ugyanannyi darab van: $n_1 = n_2 = \cdots = n_T = N/T$, $v =$ varsel opció $\in \{0,3\}$
$c =$ cuts opció $\in \{0,2\}$, $h =$ heurfreq opció $\in \{0,20\}$, cellánként 5 feladat,
10 perc futásidő, $U^*_{R-WST,b} = 100 \cdot (C^{R-WST,b}_{\max} - U_{\text{best}})/C^{R-WST,b}_{\max}$

A 3.8. táblázatból kiolvasható, hogy a 12 cellából (az átlagok alapján) 4 cellánál az R-WST modell a $(v = 0,c = 0,h = 20)$ beállítással volt a legjobb; az R-WST modell a $(v = 3,c = 2,h = 0)$, $(v = 3,c = 0,h = 20)$, $(v = 0,c = 2,h = 20)$ beállítások esetén két cellában adta a legjobb átlagot, míg a további négy beállítás mindegyike egy cellában adta a legjobb átlagot.

DOI: 10.15477/SZE.MMTDI.2017.009

Ezeket az eredményeket és a módszerek rangjait is figyelembe véve azt mondhatjuk, hogy a 24 módszer közül a célfüggvényértékeket figyelembe véve a legjobb módszer az R-WST modell és a heurfreq=20 beállítás alkalmazása volt.

Az alsó korlátoknál a rangok alapján az 5 legjobb módszer az R-TS2 modell a ($v = 2$,$c = 0$,$h = 0$), illetve a ($v = 2$,$c = 2$,$h = 0$) beállítással, az R-Wilson modell a ($v = 2$,$c = 0$,$h = 0$), illetve a ($v = 2$,$c = 2$,$h = 0$) beállítással és az R-WST módszer a ($v = 2$,$c = 0$,$h = 0$) beállítással volt. Ezen 5 módszer esetén mindegyik feladatnál kiszámítottam az $L^*_{\text{modell},b} = 100 \cdot (L_{\text{best}} - L_{\text{modell},b})/L_{\text{best}}$ értékeket, ahol $L_{\text{modell},b}$ az adott feladaton az adott modellnek a b beállítással adott alsó korlátja, míg L_{best} a 24 módszer által a feladatra adott legjobb alsó korlát. Az 5 módszer L^* értékeinek cellánkénti átlagát a 3.9. táblázat tartalmazza.

3.9. táblázat. Az 5 legjobb módszer $L^*_{\text{modell},b}$ értékeinek cellánkénti átlaga

			Beállítások				
			R-TS2 $v=2$ $c=0$ $h=0$	R-TS2 $v=2$ $c=2$ $h=0$	R-Wilson $v=2$ $c=0$ $h=0$	R-Wilson $v=2$ $c=2$ $h=0$	R-WST $v=2$ $c=0$ $h=0$
T	M	N					
10	20	120	0,057	0,080	0,041	0,157	0,071
10	20	180	0,079	0,018	0,017	0,070	0,063
10	20	240	0,060	0,091	0,053	0,079	0,040
10	25	120	0,188	0,141	0,145	0,095	0,124
10	25	180	0,039	0,043	0,013	0,021	0,032
10	25	240	0,069	0,113	0,012	0,082	0,088
15	20	120	0,216	0,208	0,177	0,232	0,009
15	20	180	0,155	0,155	0,089	0,089	0,142
15	20	240	0,023	0,023	0,026	0,013	0,029
15	25	120	0,225	0,122	0,114	0,167	0,196
15	25	180	0,050	0,049	0,044	0,038	0,043
15	25	240	0,020	0,005	0,030	0,032	0,018

$T =$ típusok száma, $M =$ gépek száma, $N =$ munkák száma
minden típusból ugyanannyi darab van: $n_1 = n_2 = \cdots = n_T = N/T$
$v =$ varsel opció $\in \{0,3\}$, $c =$ cuts opció $\in \{0,2\}$, $h =$ heurfreq opció $\in \{0,20\}$, cellánként 5 feladat, 10 perc futásidő
$L^*_{\text{modell},b} = 100 \cdot (L_{\text{best}} - L_{\text{modell},b})/L_{\text{best}}$

A 12 cellából hatnál az R-Wilson modell a ($v = 2$,$c = 0$,$h = 0$) beállítással, négy cellánál az R-Wilson modell a ($v = 2$,$c = 2$,$h = 0$) beállításnál, 2 cellánál az R-WST modell a

DOI: 10.15477/SZE.MMTDI.2017.009

($v = 2$,$c = 0$,$h = 0$) beállításnál és egy cellánál az R-TS2 modell a ($v = 2$,$c = 0$,$h = 0$) beállítással adta a legkisebb átlagos $L^*_{\text{modell},b}$ értéket. Ezen eredmény és a módszerek rangjai alapján azt mondhatjuk, hogy a legjobb alsó korlátot az R-WST modell és a ($v = 2$,$c = 0$,$h = 0$) beállítás segítségével kaphatjuk.

3.6.3. Egy ipari feladat megoldása az R-PFSP modelljeinek segítségével

Az R-Wilson, R-TS2 és R-WST modellek hatékonyságát egy nagy méretű ipari feladaton is kipróbáltuk [38]. A feladat egy autóipari cégtől származott, a gépek száma $M = 57$, a munkák száma $N = 227$, a típusok száma $N = 12$ volt. A legkisebb megmunkálási idő 20 másodperc, a legnagyobb megmunkálási idő 9081 másodperc volt. A CPLEX beállításai determinisztikus párhuzamos futtatás 4 szálon, illetve az alapbeállítások voltak. Az R-Wilson modell segítségével 129,68 másodperc, az R-TS2 modell segítségével 362,20 míg az R-WST modell segítségével 451,38 másodperc alatt megkaptuk az optimális megoldást, vagyis mindhárom modell segítségével néhány perc alatt megkaptuk egy ipari feladat optimális megoldását. A magyarázat, hogy mindegyik modell esetén ilyen gyorsan megkaptuk az optimális megoldást az, hogy a relaxált feladatok optimum értéke, azaz a kiinduló alsó korlát nagyon közel volt a feladat optimumához: a kiinduló alsó korlátnak az optimumtól való relatív eltérése mindhárom modell esetén 0,17 százalék volt.

3.6.4. A futási eredmények értékelése

A teszteredmények alapján a következő megállapításokat tehetjük:

- Az ismétlődéseket tartalmazó permutációs flow shop feladatok az R-Wilson, R-TS2, és R-WST modellekkel jóval gyorsabban oldhatók meg, mint a klasszikus modellekkel.
- A kis méretű feladatokon a futásidő alapján az R-Wilson modell volt a legjobb, megelőzve az R-WST és R-TS2 modelleket.
- A nagy méretű feladatoknál a legjobb célfüggvényértékek alapján (a CPLEX alapbeállításait használva) az R-WST modell volt a legjobb, míg az R-Wilson és R-TS2 modellek között nem volt lényeges eltérés, a legjobb alsó korlátok szerint pedig az R-WST modell volt a legrosszabb, az R-Wilson és R-TS2 modellek között pedig nem volt lényeges eltérés.
- A nagy méretű feladatokon megvizsgáltam, hogy a CPLEX három opciójának (varsel, heurfreq, cuts) változtatása milyen hatással van az R-PFSP 3 új modelljének rangsorára. A tesztek azt mutatták, hogy a 3 modellnek a legjobb célfüggvényértékek szerinti rangsorát a 3 opció változtatása nem befolyásolja, míg a modelleknek a legjobb alsó korlátok szerinti rangsorát a beállítások befolyásolják.

DOI: 10.15477/SZE.MMTDI.2017.009

- A nagy méretű feladatokon a legjobb célfüggvényértéket az R-WST modellnek és a varsel$=0$, heurfreq$=20$, cuts$=0$ beállításnak a segítségével érhettük el.
- A nagy méretű feladatokon a legjobb alsó korlátokat az R-WST modellnek és a varsel$=2$, heurfreq$=0$, cuts$=0$ beállításnak a segítségével érhettük el.
- Mindhárom modell segítségével kevesebb, mint 8 perc alatt sikerült megoldani egy 57 gépet, 227 munkát, 12 típust tartalmazó ipari feladatot.

3.7. Az R-PFSP modelljeinek összehasonlítása a heurisztikákkal

Az R-TS2 modellt [40]-ben összehasonlítottuk a NEH [76] heurisztikával, Nowicki és Schmutnicki tabu kereséses heurisztikájával [78] és Rajendran és Ziegler Paco heurisztikájával [82]. (A heurisztikák implementálását, futtatását Kiss-Tóth Christian végezte.) Ezt az összehasonlítást most kibővítjük, és az R-PFSP mindhárom modelljét összehasonlítjuk a három heurisztikával. Az összehasonlításhoz a [40]-beli tesztkészletet használjuk, melyben a gépek száma $M = 50$ volt, a típusok száma az $\{5,10,20\}$ halmazból, míg a munkák száma a $\{100,200\}$ halmazból került ki. Mindegyik típusból ugyanannyi munka volt, azaz $n_1 = n_2 \cdots = n_T = N/T$ teljesült. Mindegyik (M,N,T) hármashoz 5 tesztpéldát generáltunk, így összesen 30 feladatot tartalmazott a tesztkészlet. A megmunkálási idők a Taillard-féle tesztkészlethez hasonlóan az $[1,99]$ intervallumból véletlenszerűen választott számok voltak. Mind a PACO, mind a TABU keresés heurisztika a NEH által adott megoldást használta kiinduló megoldásként. A PACO heurisztikában a hangya összesen 300-szor keresett új utat.

A 30 példa mindegyikét megpróbáltuk megoldani az R-TS2, R-Wilson és R-WST modellek segítségével, továbbá a NEH, TABU, PACO heurisztikákkal is. Ezen kívül mindegyik feladatnál kiszámítottuk az optimális megoldás egy alsó korlátját is. Alsó korlátnak a 2.2.2. fejezetben ismertetett $LB5$ alsó korlátot használtuk. A MILP modelleknek a CPLEX-szel való megoldása során mindegyik modellnél megadtuk a munkák egy kiinduló sorrendjét (azaz a Z_{ij} bináris változók értékét adtuk meg, és ezzel a kezdő megoldással indult el a CPLEX.) A kiinduló megoldás a következő volt: a sorrend elejére kerültek az első típusú munkák, őket követték a második típusú munkák és így tovább.

A MILP modellek, illetve a heurisztikák RELGAP értékeit a következő módon határoztuk meg:

$$\text{RELGAP}_{\text{mod}} = 100 \cdot \frac{C^{\text{mod}}_{\max} - L_{best}}{C^{\text{mod}}_{\max}}$$

ahol $C^{\text{mod}}_{\max}$ az adott MILP modell, illetve heurisztika által az adott feladatra kapott eredmény, L_{best} pedig az $LB5$ alsó korlát és a három MILP modell megoldása során kapott 3 alsó korlát

DOI: 10.15477/SZE.MMTDI.2017.009

3.10. táblázat. Az R-PFSP MILP modelljeinek, illetve a heurisztikák RELGAP értékeinek átlaga és szórása

1. rész: A RELGAP értékek átlaga

	$N = 100$			$N = 200$		
	$T = 5$	$T = 10$	$T = 20$	$T = 5$	$T = 10$	$T = 20$
R-TS2	8,57	24,41	27,28	11,02	24,03	27,56
R-Wilson	8,57	24,41	27,93	17,27	24,02	27,56
R-WST	5,95	16,69	25,07	3,48	16,28	27,56
NEH	7,22	13,58	12,45	3,54	7,61	8,68
TABU	5,50	8,50	8,34	2,60	3,95	4,69
PACO	3,17	6,58	8,43	0,83	1,68	3,95

2. rész: A RELGAP értékek szórása

	$N = 100$			$N = 200$		
	$T = 5$	$T = 10$	$T = 20$	$T = 5$	$T = 10$	$T = 20$
R-TS2	6,91	3,42	1,39	8,12	7,46	0,75
R-Wilson	6,88	3,42	1,91	2,83	7,47	0,75
R-WST	3,19	3,67	2,57	2,84	10,83	0,75
NEH	2,60	2,78	2,83	1,30	4,03	1,41
TABU	2,69	2,69	2,80	0,97	2,42	1,66
PACO	1,55	2,29	2,87	0,49	0,96	1,29

$N =$ munkák száma; $T =$ típusok száma; $M = 50$
minden típusból ugyanannyi darab van: $n_1 = n_2 = \cdots = n_T = N/T$
a MILP modellek futásánál 10 perc futásidő

maximuma. A RELGAP értékek cellánkénti átlagát a 3.11. táblázat tartalmazza. A táblázat szerint a TABU és PACO heurisztikáknak mindegyik cellában kisebb az átlagos RELGAP értéke, mint az R-TS2, R-Wilson és R-WST modellek átlagos RELGAP értéke. A NEH heurisztikának mindegyik cellában kisebb az átlagos RELGAP-je, mint az R-TS2, illetve R-Wilson modelleké. Az 5 típust tartalmazó feladatoknál $N = 100$ és $N = 200$ esetén is az R-WST modellnek kisebb az átlagos RELGAP-je, mint a NEH heurisztkának. A $T = 10$ és $T = 20$ esetek mindegyikében a NEH-nek kisebb az átlagos RELGAP-je, mint az R-WST modellé. A táblázatból látható továbbá, hogy $T = 10$, illetve $T = 20$ esetén a MILP modellek RELGAP-je sokkal nagyobb, mint a heurisztikáké. A RELGAP értékeken kívül mindegyik MILP modellnél megvizsgáltuk, hogy hány esetben adtak jobb $C_{\max}$ értéket, mint az egyes heurisztikák. Az

R-TS2 modell esetén 4, az R-Wilson modell esetén 2, az R-WST modell esetén 7 feladatnál kaptunk jobb $C_{\max}$ értéket, mint a NEH heurisztikával (összesen 30 feladat volt). Az R-TS2 modell esetén 3, az R-Wilson modell esetén 2, az R-WST modell esetén 5 feladatnál kaptunk jobb $C_{\max}$ értéket, mint a TABU heurisztikával. A PACO heurisztika mindegyik feladatnál jobb $C_{\max}$-ot szolgáltatott a MILP modellek által kapott $C_{\max}$ értékeknél. Fontos megjegyezni, hogy a MILP modellek csak olyan esetben adtak jobb eredményt a NEH, illetve a TABU heurisztikánál, amikor a típusok száma 5 volt. Tehát azon 10 példánál, melyeknél 5 típus volt, az R-WST modellel 7 esetben kisebb $C_{\max}$ értéket kaptunk, mint a NEH heurisztikával és 5 esetben jobb $C_{\max}$ értéket kaptunk, mint a TABU heurisztikával. Ez is azt támasztja alá, hogy az R-PFSP MILP modelljeinek segítségével olyan feladatokat lehet hatékonyan megoldani, melyekben kevés a különböző típusok száma.

A $C_{\max}$ értékek mellett összehasonlítottuk a MILP modelleknek a CPLEX-szel való megoldása során kapott alsó korlátokat az $LB5$ alsó korláttal is. Az eredményeket a 3.11. táblázat tartalmazza.

3.11. táblázat. Az R-PFSP MILP modelljeinek megoldása során kapott alsó korlátok összehasonlítása az $LB5$ korláttal

	Hány esetben kaptunk a modellel		
	jobb alsó korlátot az $LB5$ korlátnál	ugyankkora alsó korlátot mint az $LB5$ korlát	rosszabb alsó korlátot az $LB5$ korlátnál
R-TS2	16	4	10
R-Wilson	14	4	12
R-WST	14	3	13

$M = 50$; $N \in \{100,200\}$; $T \in \{5,10,20\}$; $n_1 = n_2 = \cdots = n_T = N/T$
összesen 30 feladat; 10 perc futásidő

Mindhárom MILP modell több esetben adott jobb, mint ahány esetben rosszabb alsó korlátot az $LB5$ korlátnál (az R-TS2 modell több, mint másfélszer annyi esetben adott jobb korlátot az $LB5$ korlátnál, mint ahány esetben rosszabb korlátott adott az $LB5$ korlátnál). Ez azt mutatja, hogy nagy méretű PFSP-k esetén az irodalomból ismert alsó korlátot szolgáltató módszerekhez képest jobb alsó korlátot kaphatunk az R-TS2, R-Wilson, illetve R-WST modellek segítségével.

DOI: 10.15477/SZE.MMTDI.2017.009

4. fejezet

Lot méretet tartalmazó ismétlődő permutációs flow shop feladat (RL-PFSP)

Bizonyos gyártósoroknál a munkadaraboknak a gyártósorhoz való szállítása egy alkalmas eszközön történik. Egy ilyen eszközön csak rögzített számú, azonos típusú munkadarab vihető. Ha a munkadarabokat ezután a gyártósorhoz való szállítás sorrendjében kell felrakni a sorra, akkor az azt eredményezi, hogy csak olyan sorrendek fordulhatnak elő, melyben minden azonos típusú munkadarabból álló, maximális hosszú blokk osztható egy előre adott számmal. A feladatnak megfelelő matematikai problémára [38]-ban bevezettük a **lot méretet tartalmazó ismétlődő permutációs flow shop probléma** (RL-PFSP) fogalmát.
Lot méretet tartalmazó ismétlődő permutációs flow shop probléma: Legyen adva egy S pozitív egész szám (a lot méret). Egy olyan R-PFSP-t, melyben csak olyan $\pi = (\pi(1), \pi(2), ..., \pi(n))$ permutációk megengedettek, melyekben minden $0 \leq k \leq N/S - 1$ esetén a $(\pi(k \cdot S + 1)$,$\pi(k \cdot S+2), \ldots, \pi((k+1) \cdot S)$ munkák azonos típusúak, lot méretet tartalmazó ismétlődő permutációs flow shop problémának nevezünk.

Egy RL-PFSP-ben tehát a munkadarabokat "S-esével" rakjuk egymás után sorrendbe, ahol egy S-es csoportban azonos típusú munkadarabok állnak. Például ha $S = 2$ és van 2 típusunk és mindegyikből 6 darab, akkor az 112221122211 permutáció nem lesz megengedett, hiszen az $5.$ és $6.$ helyen (a 3. S-es csoportban) különböző típusok szerepelnek.

A lot méret figyelembevételével, illetve alkalmazásával a lehetséges permutációk száma az ismétlődéses permutációk számához képest tovább csökkenthető. Lot méretet tartalmazó feladatokban ugyanis egy permutációnál elegendő megmondani, hogy a sorrendben a $k.$ S-es csoport (lot) milyen típusú munkadarabokat tartalmaz, azaz a permutáció megadásához elegendő a lotokat sorba rendezni. Ezt figyelembe véve azt kapjuk, hogy míg T típus esetén az ismétlődő permutációk száma $\frac{N!}{n_1! \cdot n_2! \cdots n_T!}$, addig ha csak olyan permutációkat engedünk meg, melyben a lot méret S, akkor a lehetséges permutációk száma $\frac{(N/S)!}{(n_1/S)! \cdot (n_2/S)! \cdots (n_T/S)!}$. Pédául ha van 3 típusunk és mindegyikből 6 munkánk, akkor az ismétlődő permutációk száma 17153136, míg S=3 lot

DOI: 10.15477/SZE.MMTDI.2017.009

méret esetén a lehetséges permutációk száma mindössze 90.

4.1. Lot méretet tartalmazó ismétlődő permutációs flow shop feladatok MILP modelljei

Mivel egy RL-PFSP-ben elegendő a lot-ok sorrendjét megadni, ezért az előző fejezetben az R-PFSP-t leíró MILP modellek egyszerűsíthetők. Vezessük be a következő jelöléseket:

S a lot mérete
L_i az i típusú munkákat tartalmazó lotok száma ($1 \leq i \leq T$, $L_i = n_i/S$)
L az összes lot száma ($L = L_1 + L_2 + \ldots L_T$)
Z_{ij} bináris változó; $Z_{ij} = 1$ pontosan akkor, ha a j. lot i típusú munkákat tartalmaz ($1 \leq i \leq T, 1 \leq j \leq L$).

Az ismertetésre kerülő MILP modellek mindegyikében azt használjuk ki, hogy ha a j. lot i típusú munkákat tartalmaz, akkor a munkák sorrendjében a $((j-1)S+1)$-edik, $((j-1)S+2)$-edik, ... , jS-edik munka mindegyikének a típusa i.

4.1.1. Az RL-Wilson modell

Az RL-Wilson [38] modell az R-Wilson modellből származik a lot méret figyelembevételével. A modell egyenlőtlenségei garantálják a következő feltételek teljesülését:

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{4.1}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad 1 \leq j \leq L \tag{4.2}$$

- Az első gépen nincsen állóidő.

$$B_{1,S(j-1)+k} + \sum_{i=1}^{T} P_{1i} Z_{ij} \;=\; B_{1,S(j-1)+k+1} \qquad 1 \leq j \leq L;\ 1 \leq k \leq S;\ jk < N \tag{4.3}$$

- A sorrendben első munkadarabot az első gépen a 0 időpontban kezdjük el megmunkálni.

$$B_{11} = 0 \tag{4.4}$$

- A sorrendben első munkadarabnak egyetlen gép előtt sem kell várakoznia.

$$B_{r1} + \sum_{i=1}^{T} P_{ri} Z_{i1} = B_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{4.5}$$

- Egy munkadarabot csak azután kezdhetünk el megmunkálni egy adott gépen, miután az adott munkadarab megmunkálását befejeztük az előző gépen.

$$B_{r,S(j-1)+k} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r+1,S(j-1)+k} \tag{4.6}$$
$$1 \leq r \leq M-1;\ 1 \leq j \leq L;\ 1 \leq k \leq S;\ 1 < jk$$

- Egy munkadarab megmunkálását csak az után kezdhetjük el egy adott gépen, hogy a sorrendben őt megelőző munkadarab megmunkálását befejeztük az adott gépen.

$$B_{r,S(j-1)+k} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r,S(j-1)+k+1} \tag{4.7}$$
$$2 \leq r \leq M;\ 1 \leq j \leq L;\ 1 \leq k \leq S, 2 \leq jk < N$$

- Az átfutási időt minimalizáljuk.

$$\min\ C_{\max} = B_{MN} + \sum_{i=1}^{T} P_{Mi} Z_{iL} \tag{4.8}$$

Az RL-PFSP-re adott RL-Wilson modell tehát a következőképpen összegezhető:
minimalizáljuk (4.8)-et, a (4.1) – (4.7) feltételek mellett.

4.1.2. Az RL-TS2 modell

Az R-TS2 modellből a lot méret figyelembevételével [38]-ban bevezettük az RL-PFSP feladat RL-TS2 modelljét. A modell egyenlőtlenségei a következő feltételeket írják le:

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{4.9}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad\qquad 1 \leq j \leq L \tag{4.10}$$

- Egy adott gépen egy i munkadarab befejezési ideje nem lehet kisebb, mint az adott gépen a sorrendben őt megelőző munkadarab befejezési idejének és az i munkadarab megmunkálási idejének az összege.

$$\begin{aligned} C_{r,S(j-1)+k} + \sum_{i=1}^{T} P_{ri} Z_{ij} &\leq C_{r,S(j-1)+k+1} \\ &\qquad 1 \leq r \leq M;\ 1 \leq j \leq L;\ 1 \leq k \leq S-1 \end{aligned} \tag{4.11}$$

$$C_{r,jS} + \sum_{i=1}^{T} P_{ri} Z_{i,j+1} \leq C_{r,jS+1}, \qquad 1 \leq r \leq M;\ 1 \leq j < L \tag{4.12}$$

  Látható, hogy az R-TS2 modell (3.11) feltételét itt gyakorlatilag két részre kell szétszedni aszerint, hogy az i munkadarab egy lotnak az utolsó eleme-e (azaz a sorrendben a sorszáma osztható S-sel) vagy sem.

- Egy adott munkadarabot csak azután fejezhetünk be egy adott gépen, miután befejeztük az előző gépen, és megmunkáltuk az adott gépen.

$$\begin{aligned} C_{r,S(j-1)+k} + \sum_{i=1}^{T} P_{r+1,i} Z_{ij} &\leq C_{r+1,S(j-1)+k} \\ &\qquad 1 \leq r \leq M-1;\ 1 \leq j \leq L, 1 \leq k \leq S \end{aligned} \tag{4.13}$$

- Az első gépen az első munkadarabot nem fejezhetjük be előbb, mint amennyi az első munkadarabnak az első gépen a megmunkálási ideje.

$$\sum_{i=1}^{T} P_{1i} Z_{i1} \leq C_{11} \tag{4.14}$$

- Az átfutási idő az utolsó munkadarabnak az utolsó gépen való befejezési idejével egyezik meg.

$$\min\ C_{\max} = C_{MN} \tag{4.15}$$

Az RL-TS2 modell az alábbi módon foglalható össze:
minimalizáljuk (4.15)-et, a (4.9) – (4.14) feltételek mellett.

DOI: 10.15477/SZE.MMTDI.2017.009

4.1.3. Az RL-WST modell

Az RL-WST modellt [38] az R-WST modell módosításából kaphatjuk meg. A modell korlátozó feltételei biztosítják a következő feltételek teljesülését:

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{4.16}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad 1 \leq j \leq L \tag{4.17}$$

- A sorrend első munkadarabja várakozás nélkül halad végig a gépeken.

$$Y_{r1} = 0, \qquad 1 \leq r \leq M-1 \tag{4.18}$$

- A sorrend $(j+1)$. tagját addig nem kezdhetjük el egy gépen, míg az előző gépen be nem fejeztük, illetve míg a sorrend j. tagját be nem fejeztük ugyanezen a gépen.

$$\begin{aligned} \sum_{i=1}^{T} P_{ri} Z_{ij} &= \sum_{i=1}^{T} P_{r+1,i} Z_{ij} - X_{r,S(j-1)+k+1} - Y_{r,S(j-1)+k+1} + \\ &+ X_{r+1,S(j-1)+k+1} + Y_{r,S(j-1)+k} \qquad (4.19) \\ &\qquad 1 \leq r \leq M-1;\ 1 \leq j \leq L;\ 1 \leq k \leq S-1 \end{aligned}$$

$$\begin{aligned} \sum_{i=1}^{T} P_{ri} Z_{i,j+1} &= \sum_{i=1}^{T} P_{r+1,i} Z_{i,j} - X_{r,jS+1} - Y_{r,jS+1} + X_{r+1,Sj+1} + Y_{r,Sj} \qquad (4.20) \\ &\qquad 1 \leq r \leq M-1;\ 1 \leq j \leq L-1 \end{aligned}$$

  A R-WST modellhez képest itt is az az egyik lényeges eltérés, hogy az R-WST modell (3.18) feltételét szét kell szedni két részre ((4.19) és (4.20)), attól függően, hogy az adott munka egy lot utolsó eleme-e vagy sem.

- Az permutáció első tagja várakozás nélkül megy végig a gépeken.

$$X_{r+1,1} = X_{r,1} + Y_{r,1} + \sum_{i=1}^{T} P_{ri} Z_{i1} \qquad 1 \leq r \leq M-1 \tag{4.21}$$

- Az átfutási idő az utolsó gépen lévő megmunkálási és várakozási idők összegével egyezik

DOI: 10.15477/SZE.MMTDI.2017.009

meg.

$$\min\ C_{\max} = \sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{4.22}$$

Az RL-WST modell tehát a következőképpen adható meg:
minimalizáljuk (4.22)-et, a (4.16) – (4.21) feltételek mellett.

4.2. A PFSP és az RL-PFSP modelljei komplexitásának összehasonlítása

A PFSP és RL-PFSP modelljeinek a komplexitását a 4.1 táblázatban foglaltuk össze.

4.1. táblázat. A PFSP és RL-PFSP modelljeinek komplexitása

modell	bináris változók	folytonos változók	egyenlőtlenségek
Wilson	N^2	MN	$2MN - M + N + 1$
WST	N^2	$2MN - N + 1$	$MN + M + N + 1$
TS2	N^2	$MN + 1$	$2MN - M + N + 1$
RL-Wilson	$N \cdot T/S$	MN	$2MN - M - N + N/S + T + 1$
RL-WST	$N \cdot T/S$	$2MN - N + 1$	$MN + M - N + N/S + T + 1$
RL-TS2	$N \cdot T/S$	$MN + 1$	$2MN - M - N + N/S + T + 1$

$N =$ munkák száma, $M =$ gépek száma, $T =$ típusok száma, $S =$ lot méret

Az RL-PFSP modelljeit a PFSP modelljeivel összehasonlítva azt láthatjuk, hogy a legfontosabb eltérés, hogy a lotokat tartalmazó feladat modelljei jóval kevesebb bináris vátozót, nevezetesen $\frac{T}{N \cdot S}$-szer annyit tartalmaznak, mint a PFSP modelljei. Ez abból adódik, hogy egy RL-PFSP-ben figyelembe véve a lotokat a permutáció megadásához elegendő megadnunk a lotok sorrendjét (a munkák sorrendje helyett). A lotok sorrendjének megadásához pedig elegendő megadni minden i-re, hogy az i. lot a sorrendben milyen típusú munkákat tartalmaz. Az RL-Wilson, RL-TS2 és RL-WST modellek esetében ismét csak azt látjuk, hogy mindhárom modell ugyanannyi bináris változót tartalmaz. Az RL-Wilson és RL-TS2 lényegében ugyanannyi, míg az RL-WST nagyságrendileg kétszer annyi folytonos változót tartalmaz. Az egyenlőtlenségek száma az RL-WST modellben a legkevesebb; az RL-Wilson és RL-TS2 modellek ehhez képest körülbelül kétszer annyi egyenlőtlenséget tartalmaznak.

4.3. Tesztkészlet generálása

A RL-Wilson, RL-TS2 és RL-WST modellek összehasonlításához szintén generáltunk egy tesztkészletet, a 3.6. fejezetben leírt módon, azaz a megmunkálási idők az $[1{,}99]$ intervallumból

DOI: 10.15477/SZE.MMTDI.2017.009

véletlenül választott egész számok voltak. A feladatokban a gépek száma $M = 10$, a típusok száma $T = 8$ volt. Az azonos típusú munkából $n_i \in \{4,8,12,16,24\}$ darab volt, a lotok mérete (S) a $2,4$ és 8 értékeket vehette fel. Minden n_i értékhez 5 tesztpéldát generáltunk majd a kapott feladatokat megoldottuk a szóba jöhető S lot méretek mindegyikével (azaz amikor S osztója n_i-nek). Így a tesztkészlet 13 cellát (különböző (n_i,S) párt) és összesen 65 feladatot tartalmazott.

4.4. Futási eredmények

A futásidők cellánkénti átlagát, illetve szórását a 4.3. táblázat tartalmazza.
Az RL-Wilson modell a 65 feladatból 62-nél 10 percen belül megadta az optimális megoldást. Az $n_i = 16$ esetben 1 feladatot sem $S = 2$, sem $S = 4$ lot mérettel nem tudott megoldani 10 perc alatt; a RELGAP értékek $3,62$, illetve $3,25$ voltak. Az $n_i = 24$ esetben 1 feladatot $S = 4$ lot mérettel nem tudott megoldani a modell; a RELGAP érték $1,50$ volt.
Az RL-TS2 modell 65 feladatból 61 esetben adott optimális megoldást 10 perc alatt. Az $n_i = 16$ esetben 1 feladatot sem $S = 2$, sem $S = 4$ lot mérettel nem tudott megoldani 10 perc alatt; a RELGAP értékek $1,95$ és $2,04$ voltak. Az $n_i = 24$ esetben 1 feladatot $S = 4$ és $S = 8$ lot mérettel nem tudott megoldani a modell; a RELGAP értékek $0,76$, illetve $0,51$ voltak.
Az RL-WST modell 59 feladatnál adta meg az optimális megoldást 10 perc alatt. Az $n_i = 16$ esetben 1 feladatot az $S = 2; 4; 8$ lot méretek egyikével sem tudta megoldani a modell; a RELGAP értékek rendre $(2,10;\ 2,10;\ 0,13)$ voltak. Az $n_i = 24$ esetben is 1 feladatot egyik lot mérettel sem tudott megoldani a modell; a RELGAP értékek rendre $(0,21;\ 1,10;\ 1,09)$ voltak.

4.2. táblázat. Az RL-PFSP modelljeinek összehasonlítása a leggyorsabban megoldott feladatok száma és modellek futásidő szerinti rangjának átlaga alapján

Hány esetben volt a leggyorsabb			Rangok átlaga a futásidő alapján		
RL-TS2	RL-Wilson	RL-WST	RL-TS2	RL-Wilson	RL-WST
8	20	35	2,31	1,89	1,78

Összesen 65 feladat; gépek száma $M = 10$, típusok száma $T = 8$, $S \in \{2,4,8\}$
$n_i \in \{4,8,12,16,24\}$, 10 perc futásidő

A 4.2. táblázat alapján a legtöbb feladatot (35-öt) az RL-WST modell oldotta meg a leggyorsabban, megelőzve ezzel az RL-Wilson modellt (20) és az RL-TS2 modellt (8). A futásidő szerinti rangok átlagánál szintén ugyanez a sorrend: a legkisebb rangja az RL-WST modellnek volt, őt követte az RL-Wilson és az RL-TS2 modell. Érdemes ezt az eredményt összevetni a futásidők cellánkénti átlagával, illetve szórásával. A 4.3. táblázatból kiolvasható, hogy annál

DOI: 10.15477/SZE.MMTDI.2017.009

4.3. táblázat. Az RL-PFSP modelljeinek összehasonlítása

1. rész: Átlagos futásidő

S	Modell	$n_i = 4$	$n_i = 8$	$n_i = 12$	$n_i = 16$	$n_i = 24$
	RL-TS2	7,06	28,21	40,81	*	69,49
2	RL-Wilson	6,78	15,41	45,12	*	65,39
	RL-WST	12,69	47,03	44,49	*	*
	RL-TS2	2,05	19,43	66,05	*	*
4	RL-Wilson	1,83	17,11	26,68	*	*
	RL-WST	2,58	21,52	71,93	*	*
	RL-TS2		2,95		98,07	*
8	RL-Wilson		2,7		68,75	95,00
	RL-WST		2,17		*	*

2. rész: A futásidők szórása

S	Modell	$n_i = 4$	$n_i = 8$	$n_i = 12$	$n_i = 16$	$n_i = 24$
	RL-TS2	7,79	39,00	68,29	*	135,05
2	RL-Wilson	6,76	20,82	77,88	*	126,08
	RL-WST	17,66	63,82	97,11	*	*
	RL-TS2	0,91	34,89	141,22	*	*
4	RL-Wilson	0,74	31,26	54,41	*	*
	RL-WST	1,18	39,50	151,65	*	
	RL-TS2		2,29		204,77	*
8	RL-Wilson		1,94		142,39	165,02
	RL-WST		2,17		*	*

3. rész. A $C^S_{\max}$-nek a $C^1_{\max}$-től való átlagos relatív eltérése (%-ban)

S	$n_i = 4$	$n_i = 8$	$n_i = 12$	$n_i = 16$	$n_i = 24$
2	1,18	0,05	0,1	*	*
4	7,23	1,14	0,3	*	*
8		4,34		1,67	0,44

gépek száma $M = 10$, típusok száma $T = 8$, $S =$ lot méret
$n_i =$ a munkák száma az i típusból, $(n_1 = n_2 = \cdots = n_T)$
munkák száma $N = T \cdot n_i$
*: 1 példában nem kaptunk optimális megoldást 10 perc alatt

DOI: 10.15477/SZE.MMTDI.2017.009

a 10 cellánál, melyeknél legalább az egyik modell mind az 5 példára megadta az optimális megoldást 10 perc alatt, 8 cellánál az RL-Wilson modellnek volt a legkisebb átlagos futásideje, míg 2 cellánál az RL-WST-nek volt a legkisebb átlagos futásideje (a szórások átlaga mind a 10 cellában az RL-Wilson modellnél volt a legkisebb). Erre a jelenségre magyarázatot ad, ha csak a 9 legnehezebb példán (melyben mindegyik modell futásideje legalább 20 másodperc volt) hasonlítjuk össze a 3 modellt. Ezeknél a feladatoknál az RL-Wilson modell 7, az RL-WST és RL-TS2 modell 1-1 esetben volt a leggyorsabb, míg 8 esetben az RL-WST, 1 esetben pedig az RL-TS2 modell volt a leglassabb. Vagyis ez azt jelenti, hogy a könnyű feladatokat az RL-WST modell oldotta meg a leggyorsabban, míg a nehezebb feladatokat az RL-Wilson modell oldotta meg a leggyorsabban.

$S = 2$ és $S = 4$ esetén 12 ismétlődésig (azaz 96 munkáig) mindhárom modell megoldotta az összes feladatot, 16 ismétlődésnél viszont már volt olyan feladat, amelyet egyetlen modell sem tudott 10 perc alatt megoldani. Ezt a feladatot $S = 8$ esetén az R-Wilson és R-TS2 modell is megoldotta 10 perc alatt.

Hasonlóan $n_i = 24$ és $S = 4$ esetén egy feladatot egyik modell sem tudta megoldani 10 perc alatt, míg ugyanezt a feladatot az RL-Wilson modell $S = 8$-cal megoldotta 10 perc alatt.

A futásidőkön kívül az optimumértékeket összehasonlítottuk a lotokat nem tartalmazó (azaz $S = 1$) feladat optimumával is. Jelölje $C^S_{\max}$ az S lotméretű feladat optimumát. Ekkor a $C^S_{\max}$-nek a $C^1_{\max}$-től (azaz a lotokat nem tartalmazó feladat optimumától) való relatív eltérését a

$$100 \cdot \frac{C^S_{\max} - C^1_{\max}}{C^1_{\max}}$$

képlettel definiálhatjuk. A 4.3. táblázat harmadik része a cellánkénti relatív eltérések átlagát tartalmazza. Az eredmények alapján arra következtethetünk, hogy ha rögzítjük S értékét és növeljük az ismétlődések (azaz n_i) számát, akkor a $C^S_{\max}$ értékek egyre közelebb kerülnek $C^1_{\max}$-hez. (A tesztfeladatokban például az $n_i = 16$ esetben 5-ből 3 példában az $S = 8$ lot mérethez tartozó optimum értéke megegyezett a lot nélküli feladat optimum értékével.)

4.5. A futási eredmények értékelése

A futási eredményekből az alábbi következtetéseket vonhatjuk le.

- Az RL-PFSP modelljeinek segítségével nagy méretű feladatok is megoldhatók.
- A lot méret növelésével a megoldható feladatok mérete növekszik. Például $S = 8$ esetén az RL-Wilson modell 10 gépes, 192 munkát tartalmazó példákat is meg tudott oldani 10 perc alatt, míg $S = 4$ esetén volt olyan 10 gépes, 96 munkát tartalmazó feladat, melyre az RL-Wilson modell nem adott 10 perc alatt optimális megoldást.

DOI: 10.15477/SZE.MMTDI.2017.009

- Nagy méretű R-PFSP feladatok egy jó közelítő megoldását kaphatjuk, ha a feladatot egy alkalmasan választott lot mérettel oldjuk meg.
- A tesztfeladatok közül az egyszerűbbeket az RL-WST modell, a nehezebbeket pedig az RL-Wilson modell oldotta meg a leggyorsabban.

DOI: 10.15477/SZE.MMTDI.2017.009

5. fejezet

Palettát és véges puffert tartalmazó PFSP

5.1. Palettát és véges puffert tartalmazó ismétlődő permutációs flow shop probléma

A gyártósoron a munkadarabok szállítása az egymást követő állomások közt többféleképpen történhet. Az iparban alkalmazott egyik módszer az, hogy a gyártósor elején mindegyik munkadarabot felrakják egy szállítóeszközre, egy úgynevezett palettára, és a munkadarab ezen a palettán halad végig a soron (egy paletta tehát egy munkadarabot szállít). Amikor a munkadarabot az utolsó munkaállomáson is megmunkálták, akkor a munkadarab lekerül a palettáról és a paletta visszakerül az első munkaállomás elé, ahol rárakhatnak egy új munkadarabot. Mivel a paletták száma korlátozott - többnyire a munkaállomások számánál néhány darabbal van több belőlük - ezért egy munkadarab csak akkor kerülhet fel a gyártósorra, ha van szabad paletta. Ráadásul, a paletták méretét is figyelembe véve az adódik, hogy két munkaállomás között csak véges számú paletta, azaz véges számú munkadarab várakozhat. A gépek közti pufferek számát a következőképpen számíthatjuk ki: ha a paletták hossza l és két egymást követő gép közt a távolság d, akkor ezen két gép között $\lfloor \frac{d}{l} \rfloor - 1$ számú paletta fér el a soron, így ezen két gép között legfeljebb ennyi munka várakozhat, tehát a két gép közötti puffer mérete $\lfloor \frac{d}{l} \rfloor - 1$. Az ilyen típusú ütemezési feladatok matematikai modelljének leírására [39]-ben bevezettük a **palettákat és véges puffert tartalmazó ismétlődő permutációs flow shop probléma** (PB-R-PFSP) fogalmát.

Palettákat és véges puffert tartalmazó ismétlődő permutációs flow shop problémának nevezünk egy PFSP-t, mely tartalmaz azonos típusú munkákat, az egymást követő gépek között a puffer nagysága véges, továbbá a munkadarabok szállítása a gépek között palettákon történik, minden paletta egy munkadarabot szállít és a paletták száma korlátos.

A munkadaraboknak az egyes gépeken való kezdési idejének szempontjából a PB-R-PFSP és az R-PFSP az alábbi két dologban tér el.

DOI: 10.15477/SZE.MMTDI.2017.009

- Az R-PFSP-ben egy munkadarabot azonnal elkezdhetünk megmunkálni az első gépen, mihelyst az első gép szabaddá válik. A PB-R-PFS-ben viszont egy munkadarabot csak akkor kezdhetünk el megmunkálni az első gépen, ha az első gép szabad és van szabad paletta is

- Az R-PFSP-ben a gépek közti puffer nagysága végtelen, így ha egy munkadarab megmunkálásával végeztünk egy gépen, akkor azonnal levehetjük a gépről és berakhatjuk a pufferbe. Ezzel szemben egy PB-R-PFSP-ben egy munkadarabot a megmunkálásának a befejezése után csak akkor vehetünk le a gépről, ha gép után van puffer és a pufferben van szabad hely, vagy a gép után egyáltalán nincs puffer, viszont a következő gép szabad.

Vezessük be a következő jelöléseket:

K a paletták száma

b_r az r. és $(r+1)$. gépek közti puffer nagysága ($1 \leq r \leq M-1$)

Mivel összesen K darab paletta van, ezért az első gépen a sorrend egy tetszőleges, mondjuk a j. munkadarabját addig nem kezdhetjük el megmunkálni, amíg egy paletta üressé nem válik. Mivel K darab paletta van, ezért ez azt jelenti, hogy az első gépen a j. munkadarab kezdési ideje nem lehet kisebb, mint az utolsó gépen a sorrend $(j-K)$-adik munkájának a befejezési ideje.

Az egymást követő gépek közti véges puffer méretből adódóan azt kapjuk, hogy egy tetszőleges munkadarabot, mondjuk a sorrend j. tagját addig nem kezdhetjük el megmunkálni az r. gépen, amíg a sorrend $(j-(b_r+1))$-edik tagját el nem kezdjük megmunkálni az $(r+1)$-edik gépen. Ha ugyanis a sorrend $(j-(b_r+1))$-edik tagját még nem kezdtük el megmunkálni az $(r+1)$-edik gépen, akkor az azt jelenti, hogy a következő két eset valamelyike áll fenn.

- A sorrend $(j-(b_r+1))$-edik tagját még nem munkáltuk meg az r. gépen.

- A sorrend $(j-(b_r+1))$-edik tagját már megmunkáltuk meg az r. gépen.

Az első esetben világos, hogy a j. tag még nem munkálható meg az r. gépen. A második esetben, ha a $(j-(b_r+1))$-edik tag még nem került le az r. gépről, akkor a j. tagot nyilván nem kezdhetjük el megmunkálni az r. gépen. Ha viszont a $(j-(b_r+1))$-edik tag lekerült az r. gépről és a pufferban várakozik, akkor lévén, hogy az r. és $(r+1)$. gép között legfeljebb b_r munka várakozhat, a sorrend $(j-1)$-edik tagja még nem kerülhetett le az r. gépről, ezért a j. tagot nem kezdhetjük el az r. gépen.

Az egyszerűség kedvéért vezessük be a $D_{r,j}$ jelölést, mely a sorrend j. munkájának az r gépen való megmunkálási idejét jelöli. Könnyen látható, hogy

$$D_{r,j} = \sum_{i=1}^{T} P_{ri} Z_{i,j} \tag{5.1}$$

DOI: 10.15477/SZE.MMTDI.2017.009

Ezzel a jelöléssel egy adott permutáció esetén az átfutási idő az alábbi rekurzív módon számítható ki.

$$B_{11} = 0 \tag{5.2}$$

$$B_{r1} = B_{r-1,1} + D_{r-1,1} \qquad (2 \leq r \leq M) \tag{5.3}$$

$$B_{1j} = \max\left(B_{1,j-1} + D_{1,j-1}, B_{2,j-b_r-1}, B_{M,j-K} + D_{M,j-K}\right) \tag{5.4}$$
$$(1 < j \leq N)$$

$$B_{rj} = \max\left(B_{r,j-1} + D_{r,j-1}, B_{r+1,j-b_r-1}, B_{r-1,j} + D_{r-1,j}\right) \tag{5.5}$$
$$(2 \leq r \leq M; 2 \leq j \leq N)$$

$$C_{\max} = B_{MN} + D_{MN} \tag{5.6}$$

Látható, hogy a paletták száma az első gépen való kezdési időket befolyásolja. A fenti képletekben B_{rj}, illetve D_{rj} értéke 0-nak értendő ha, $M < r$ vagy $j < 0$.

5.2. A PB-R-PFSP MILP modelljei

A [39] cikkben megadtuk a PB-R-PFSP három MILP modelljét. A modellek az R-PFSP-re adott R-Wilson, R-TS2 és R-WST modellek módosításából származnak. Mindhárom modellnél azt kell figyelembe venni, hogy egy PB-R-PFSP feladatban az első gépen lehetnek állóidők, továbbá a modelleknek tartalmazniuk kell a paletták számára, illetve a pufferek számára vonatkozó feltételeket.

5.2.1. A PB-R-Wilson modell

A PB-R-Wilson modell tartalmazza az R-Wilson modell (3.1), (3.2), (3.4), (3.5), (3.6) feltételeit.

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \leq i \leq T \tag{5.7}$$

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \leq j \leq N \tag{5.8}$$

$$B_{11} = 0 \tag{5.9}$$

DOI: 10.15477/SZE.MMTDI.2017.009

$$B_{r1} + \sum_{i=1}^{T} P_{ri} Z_{i1} = B_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{5.10}$$

$$B_{rj} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r+1,j} \qquad 1 \leq r \leq M-1;\ 2 \leq j \leq N \tag{5.11}$$

Mivel egy PB-R-PFSP feladatban az első gépen lehetnek állóidők, ezért a PB-R-Wilson modell (3.3) feltétele ”beolvad” a (3.7) feltételbe - egy munkadarabot egy tetszőleges gépen addig nem kezdhetünk el, míg a sorrendben őt közvetlenül megelőző munkadarabot meg nem munkáltuk az adott gépen - amely a következőképpen módosul:

$$B_{rj} + \sum_{i=1}^{T} P_{ri} Z_{ij} \leq B_{r,j+1} \qquad 1 \leq r \leq M;\ 1 \leq j \leq N-1 \tag{5.12}$$

Ezen kívül a PB-R-Wilson modell a következő feltételeket tartalmazza:

- Legfeljebb K darab palettát használhatunk.

$$B_{M,j-K} + \sum_{i=1}^{T} P_{Mi} Z_{i,j-K} \leq B_{1j} \qquad K+1 \leq j \leq N \tag{5.13}$$

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$B_{r+1,j-b_r-1} \leq B_{rj} \qquad 1 \leq r \leq M-1,\ 2+b_r \leq j \leq N \tag{5.14}$$

A célfüggvény megegyezik az R-Wilson modell célfüggvényével:

$$\min\ C_{\max} = B_{MN} + \sum_{i=1}^{T} P_{Mi} Z_{iN} \tag{5.15}$$

A PB-R-Wilson modell tehát a következőképpen összegezhető: minimalizáljuk (5.15)-öt, az (5.7) – (5.14) feltételek mellett.

5.2.2. A PB-R-TS2 modell

Mivel az R-TS2 modellben nem használjuk ki, hogy az első gépen nincsenek állóidők, ezért a PB-R-TS2 modell tartalmazza az R-TS2 modell (3.9), (3.10), (3.11), (3.12) és (3.13) feltételeit.

DOI: 10.15477/SZE.MMTDI.2017.009

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \le i \le T \tag{5.16}$$

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \le j \le N \tag{5.17}$$

$$C_{rj} + \sum_{i=1}^{T} P_{ri} Z_{i,j+1} \le C_{r,j+1}, \qquad 1 \le r \le M, 1 \le j \le N-1 \tag{5.18}$$

$$C_{rj} + \sum_{i=1}^{T} P_{r+1,i} Z_{ij} \le C_{r+1,j}, \qquad 1 \le r \le M-1, 1 \le j \le N \tag{5.19}$$

$$\sum_{i=1}^{T} P_{1i} Z_{i1} \le C_{11} \tag{5.20}$$

Ezekhez az egyenlőtlenségekhez ismét azt a két feltételt kell hozzáadnunk, hogy

- Legfeljebb K darab palettát használhatunk.

$$C_{M,j-K} + \sum_{i=1}^{T} P_{1i} Z_{i,j} \le C_{1j} \qquad K+1 \le j \le N \tag{5.21}$$

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$C_{r+1,j-b_r-1} - \sum_{i=1}^{T} P_{r+1,i} Z_{i,j-b_r-1} + \sum_{i=1}^{T} P_{ri} Z_{i,j} \le C_{rj} \tag{5.22}$$
$$1 \le r \le M-1, 2+b_r \le j \le N$$

A célfüggvény megegyezik az R-TS2 modell célfüggvényével.

$$\min\ C_{\max} = C_{MN} \tag{5.23}$$

A PB-R-TS2 modell a következőképpen összegezhető:
minimalizáljuk (5.23)-at, az (5.16)– (5.22) feltételek mellett.

DOI: 10.15477/SZE.MMTDI.2017.009

5.2.3. A PB-R-WST modell

Mivel az R-PFSP-re adott R-WST modellben nem használtuk ki, hogy az első gépen nincsenek állóidők, ezért a PB-R-WST modell tartalmazza az R-WST modell összes feltételét.

$$\sum_{j=1}^{N} Z_{ij} = n_i, \qquad 1 \le i \le T \tag{5.24}$$

$$\sum_{i=1}^{T} Z_{ij} = 1, \qquad 1 \le j \le N \tag{5.25}$$

$$Y_{r1} = 0, \qquad 1 \le r \le M-1 \tag{5.26}$$

$$\sum_{i=1}^{T} P_{ri} Z_{i,j+1} + X_{r,j+1} + Y_{r,j+1} = \sum_{i=1}^{T} P_{r+1,i} Z_{ij} + X_{r+1,j+1} + Y_{rj} \tag{5.27}$$
$$1 \le r \le M-1;\ 1 \le j \le N-1$$

$$X_{r1} + Y_{r1} + \sum_{i=1}^{T} P_{ri} Z_{i1} = X_{r+1,1} \qquad 1 \le r \le M-1 \tag{5.28}$$

Ezeken kívül két új feltétele van a PB-R-WST modellnek.

- Legfeljebb K darab palettát használhatunk.

$$\sum_{i=1}^{j-K} X_{Mi} + \sum_{i=1}^{j-K} \sum_{k=1}^{T} P_{Mk} Z_{ki} \le \sum_{i=1}^{j} X_{1i} + \sum_{i=1}^{j-1} \sum_{k=1}^{T} P_{1k} Z_{ki} \tag{5.29}$$
$$K + 1 \le j \le N$$

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$\sum_{i=1}^{j-b_r-1} X_{r+1,i} + \sum_{i=1}^{j-b_r-2} \sum_{k=1}^{T} P_{r+1,k} Z_{ki} \le \sum_{i=1}^{j} X_{ri} + \sum_{i=1}^{j-1} \sum_{k=1}^{T} P_{rk} Z_{ki} \tag{5.30}$$
$$1 \le r \le M-1, 2+b_r \le j \le N$$

A PB-R-WST modell célfüggvénye megegyezik az R-WST modell célfüggvényével.

$$\min\ C_{\max} = \sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{5.31}$$

A PB-R-WST modell az alábbi módon foglalható össze:
minimalizáljuk (5.31)-et, az (5.24) – (5.30) feltételek mellett.

DOI: 10.15477/SZE.MMTDI.2017.009

5.2.4. A PB-R-TS3 modell

Az PB-R-TS3 modell felírásához a [100]-ban alkalmazott helyettesítéses módszert alkalmazzuk: a C_{rj} változóknak megfogalmazzuk egy ekvivalens felírását, majd az így kapott formulát behelyettesítjük a PB-R-TS2 modell egyenlőtlenségeibe. A PB-R-TS2 modellhez hasonlóan az új modell, melyet PB-R-TS3-nak fogunk nevezni, is a Z_{ij} változókat használja a permutáció leírásához. Így a PB-R-TS3 modell is tartalmazni fogja a PB-R-TS2 modell (3.9) és (3.10) feltételeit. A továbbiakban a képletek rövidebb írása miatt a $\sum_{i=1}^{T} P_{ri}Z_{ij}$ összeget, mely a sorrend j. tagjának az r gépen való megmunkálási idejét adja meg, D_{rj}-vel fogjuk jelölni.

A sorrend j. elemének az r gépen való befejezési idejét, azaz C_{rj}-t kifejezhetjük az alábbi módon is:

$$C_{rj} = \sum_{i=1}^{j} X_{1i} + \sum_{i=1}^{j-1} D_{1i} + \sum_{q=1}^{r-1} Y_{qj} + \sum_{q=1}^{r} D_{qj} \tag{5.32}$$

ahol a jobb oldal

- első tagja az első gépen a sorrend j. elemének kezdéséig az összes állóidő;
- második tagja az a sorrend első $(j-1)$ elemének a megmunkálási idejének az összege az első gépen;
- harmadik tagja a sorrend j. elemének a teljes várakozási ideje az $1,2,\ldots,(r-1)$ gépek után;
- a negyedik tagja pedig a sorrend j. tagjának az első r gépen való megmunkálási idejének az összege.

Helyettesítsük most be $C_{r,j}$-nek ezt a felírását a PB-R-TS2 modell egyenlőtlenségeibe. Az (5.32) kifejezést a (3.11) egyenlőtlenségbe helyettesítve $r = 1$ esetén azt kapjuk, hogy

$$X_{1j} \geq 0, \quad 2 \leq j \leq N \tag{5.33}$$

Mivel a MILP modellek megoldására alkalmas szoftverekben meg lehet adni nem negatív változókat, ezért a RB-R-TS3 modell nem tartalmazza ezt az egyenlőtlenséget.

Az $r \geq 2$ esetén a helyettesítés után az

$$\begin{aligned} \sum_{q=2}^{r} D_{qj} + \sum_{q=1}^{r-1} Y_{qj} &\leq Y_{1,j-1} + \sum_{q=1}^{r-1} D_{q,j+1} + \sum_{q=1}^{r-1} Y_{q,j+1} \\ 2 &\leq r \leq M, 1 \leq j \leq N-1 \end{aligned} \tag{5.34}$$

egyenlőtlenségekhez jutunk.

DOI: 10.15477/SZE.MMTDI.2017.009

Az (5.32) kifejezést a (3.12) egyenlőtlenségbe helyettesítve azt kapjuk, hogy

$$Y_{rj} \geq 0, \quad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{5.35}$$

Ezeket a nemnegativitási feltételeket, az előző gondolatmenethez hasonlóan, nem vesszük bele a modellbe.

Az (5.32) egyenlőséget a (3.13) egyenlőtlenségbe helyettesítve

$$X_{11} \geq 0, \tag{5.36}$$

adódik, melyet a nem negativitás miatt nem veszünk figyelembe a modellben. Az (5.32) egyenlőséget az (5.21) egyenlőtlenségbe behelyettesítve egyszerűsítés után azt kapjuk, hogy

$$\sum_{r=1}^{M-1} Y_{r,j-K} + \sum_{r=2}^{M} D_{r,j-K} \leq \sum_{i=j-K+1}^{j} X_{1i} + \sum_{i=j-K+1}^{j-1} D_{1i} \quad (K < j) \tag{5.37}$$

Az (5.32) egyenlőségnek az (5.22)-be való helyettesítése, majd az azonos tagok elhagyása után az adódik, hogy

$$\sum_{q=1}^{r} Y_{q,j-b_r} + \sum_{q=2}^{r} D_{q,j-b_r} \leq \sum_{i=j-b_r+1}^{j+1} X_{1i} + \sum_{i=j-b_r+1}^{j} D_{1i} + \sum_{q=1}^{r-1} Y_{q,j+1} + \sum_{q=1}^{r-1} D_{q,j+1} \tag{5.38}$$
$$1 \leq r \leq M-1, b_r < j \leq N-1$$

Végezetül (5.32)-őt a célfüggvénybe, (3.14)-be helyettesítve a célfüggvényre a következőket kapjuk:

$$\min C_{\max} = \sum_{i=1}^{N} X_{1i} + \sum_{i=1}^{N-1} D_{1i} + \sum_{q=1}^{M-1} Y_{qN} + \sum_{q=1}^{M} D_{qN} \tag{5.39}$$

A PB-R-PFSP új, PB-R-TS3 modellje tehát a következőképpen összegezhető: minimalizáljuk (5.39)-et, a (3.9), (3.10), (5.34), (5.37), (5.38) feltételek mellett.

5.2.5. A PB-R-PFSP modelljei komplexitásának összehasonlítása

A PB-R-Wilson, PB-R-TS2, PB-R-WST és PB-R-TS3 modellek komplexitását az 5.1. táblázatban foglaltam össze.

A táblázatból látható, hogy a bináris változók száma mind a 4 modell esetén ugyanannyi. A PB-R-Wilson és PB-R-TS3 modellek tartalmazzák a legkevesebb folytonos változót; a PB-R-TS2 modell ennél eggyel több, a PB-R-WST modell pedig körülbelül kétszer ennyi folytonos változót tartalmaz. A PB-R-TS3 modell tartalmazza a legkevesebb egyenlőtlenséget; a

DOI: 10.15477/SZE.MMTDI.2017.009

5.1. táblázat. A PB-R-PFSP modelljeinek a komplexitása

modell	bináris változók	folytonos változók	feltételek
PB-R-Wilson	NT	MN	$3MN - 2M + T - K - B + 2$
PB-R-WST	NT	$2MN - N + 1$	$2MN + T - K - B + 2$
PB-R-TS2	NT	$MN + 1$	$3MN - 2M + T - K - B + 2$
PB-R-TS3	NT	MN	$2MN + T - K - B - 2M - N + 3$

$N =$ munkák száma, $M =$ gépek száma, $T =$ típusok száma
$K =$ paletták száma, $B = \sum_{r=1}^{M-1} b_r$

PB-R-WST modell nagyságrendileg ugyanennyi, míg a PB-R-Wilson és PB-R-TS2 modellek nagyjából másfélszer ennyi egyenlőtlenséget tartalmaznak. Vagyis a négy modell közül a PB-R-TS3 modell tartalmazza a legkevesebb bináris változót, folytonos változót és egyenlőtlenséget is.

5.3. Tesztkészlet generálása a PB-R-PFSP feladatokhoz

A PB-R-PFSP modelljeinek összehasonlításához háromféle tesztkészletet használtunk. Először létrehoztunk egy kis méretű feladatokat tartalmazó tesztkészletet. A gépek száma $M \in \{7,8\}$ a típusok száma pedig $T \in \{6,7\}$ lehetett. Minden lehetséges (M,T) párhoz 5 tesztpéldát generáltunk. Mindegyik feladatban mindegyik típusból pontosan 5 darab volt ($n_i = 5$). A megmunkálási idők (a Taillard által javasolt módon) az $[1,99]$ intervallumból véletlenszerűen választott számok voltak. Mindegyik feladatot megoldottuk $K = 8,9,10$ palettával és $b_r = 0,1,2$ puffermérettel is. Vagyis összesen $2 \cdot 2 \cdot 3 \cdot 3 \cdot 5 = 180$ kis méretű feladaton teszteltük a négy modellt.

Ezt követően a négy modellt nagy méretű feladatokon is összehasonlítottuk. Ehhez a 3.6.2 fejezet tesztkészletét (megmunkálási idők mátrixait) használtuk fel. Ezeknél a feladatoknál a gépek száma $M \in \{20,25\}$, a munkák száma $N \in \{120,180,240\}$, a típusok száma pedig $T \in \{10,15\}$ lehetett. Minden (M,N,T) hármashoz 5 példát készítettünk. A gépek közt a puffer méret 0 volt, a paletták száma pedig minden feladatnál néggyel volt több, mint a gépek száma ($K = M + 4$).

Végezetül a PB-R-PFSP 4 modelljét egy ipari feladaton is összehasonlítottuk, továbbá ezen ipari feladatból kiindulva egy ipari jellegű tesztkészletet is létrehoztunk. Ez a következőképpen történt. Tegyük fel, hogy az ipari feladatban az $r.$ gépen a $j.$ típus megmunkálási ideje $P_{r,j}$ volt. Egy rögzített r esetén az $r.$ gépen a megmunkálási időket a következőképpen generáltuk le. Először is egymástól függetlenül, véletlenszerűen választottunk $k_r = \sum_{i=1}^{T} P_{r,i}$ darab számot $(A_{r,1}, A_{r,2}, \ldots A_{r,k_r})$ az $[1, \sum_{i=1}^{T} P_{ri}]$ intervallumban. Ezután az általunk generált, ipari jellegű

DOI: 10.15477/SZE.MMTDI.2017.009

feladatban a j típusú munka megmunkálási ideje egyenlő lett azon $A_{r,l}$ véletlen számoknak a számával, melyekre a $\sum_{i=1}^{j-1} P_{ri} < A_{r,l} \leq \sum_{i=1}^{j} P_{ri}$ egyenlőtlenség teljesült. Az egyes típusokból gyártandó mennyiségek számát ugyanilyen módszer szerint generáltuk. Vagyis olyan példákat kaptunk, melyekben az összes munkadarab száma, illetve a megmunkálási idők összege minden gépen megegyezett az iparifeladat-beli értékekkel.

5.4. Futási eredmények

5.4.1. A PB-R-PFSP modelljeinek összehasonlítása a kis méretű feladatokon

A PB-R-PFSP 4 modelljének mindegyikét 180 kis méretű feladaton teszteltük. A futási eredmények alapján a modelleket többféle szempont szerint is összehasonlítottuk.
A modelleket először a futási idők alapján vetettük össze. A 180 feladatból a PB-R-TS3, illetve a PB-R-Wilson modell segítségével 111 esetben kaptuk meg az optimális megoldást a 10 perces időkorlát alatt; a PB-R-WST modell esetén 108, a PB-R-TS2 modell esetén 107 feladatnál kaptunk optimális megoldást. Azoknál a feladatoknál, melyeknél legalább az egyik modellel megkaptuk az optimális megoldást, a legtöbb esetben (56) a PB-R-TS3 modell volt a leggyorsabb; a PB-R-TS2 modell 46, a PB-R-Wilson 11, míg a PB-R-WST modell 4 feladatnál volt a leggyorsabb. Azon feladatoknál, melyeket legalább az egyik modellel meg tudtunk oldani, a futásidők rangja alapján a PB-R-TS3 modell volt a legjobb, megelőzve a PB-R-TS2, PB-R-Wilson és PB-R-WST modelleket. Az eredményeket az 5.2. táblázat tartalmazza.

5.2. táblázat. A PB-R-PFSP modelljeinek összehasonlítása a kis méretű feladatokon a futásidők alapján

Modell	megoldott feladatok száma	legkevesebb idő alatt megoldott feladatok száma	rangok átlaga a futásidő alapján
PB-R-Wilson	111	11	2,91
PB-R-TS2	107	46	1,89
PB-R-WST	108	4	3,27
PB-R-TS3	111	56	1,76

$M \in \{7,8\}$; $T \in \{6,7\}$, $N = 5T$; $K \in \{8,9,10\}$, $b_i = b_2 = \cdots = b_{M-1}) \in \{0,1,2\}$
összesen 180 feladat; 10 perc futásidő

Az egyes modellek futási idejének cellánkénti átlagát, illetve szórását az 5.3. és 5.4. táblázatok tartalmazzák. A 36 cellából mindössze 10 olyan volt, melyeknél mindegyik modellel megkaptuk mind az 5 feladat optimális megoldását. A 10 cellából 9-nél a PB-R-TS2 modellnek, 1 cellánál pedig a PB-R-TS3 modellnek volt a legkisebb az átlagos futásideje.

DOI: 10.15477/SZE.MMTDI.2017.009

5.3. táblázat. A PB-R-PFSP modelljeinél a futási idők átlaga és szórása (másodpercben) a 7 gépes feladatokban

1. rész: Átlagos futásidő

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	10,19	4,98	4,54	*	*	*
2	PB-R-TS2	3,54	3,77	3,24	*	*	*
	PB-R-WST	10,33	9,65	6,90	*	*	*
	PB-R-TS3	10,93	5,23	5,28	*	*	*
	PB-R-Wilson	16,35	17,15	10,80	*	*	*
1	PB-R-TS2	6,40	3,72	4,54	*	*	*
	R-PB-WST	14,13	11,18	9,05	*	*	*
	PB-R-TS3	13,06	6,4	5,89			
	PB-R-Wilson	*	*	*	*	*	*
0	PB-R-TS2	*	*	*	*	*	*
	PB-R-WST	*	*	*	*	*	*
	PB-R-TS3	*	*	*	*	*	*

2. rész: A futásidők szórásának átlaga

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	15,58	9,66	8,66	*	*	*
2	PB-R-TS2	6,47	7,28	6,29	*	*	*
	PB-R-WST	19,98	18,65	13,56	*	*	*
	PB-R-TS3	22,74	10,80	10,78	*	*	*
	PB-R-Wilson	34,59	36,62	22,82	*	*	*
1	PB-R-TS2	12,63	7,33	9,16	*	*	*
	R-PB-WST	29,13	22,41	18,22	*	*	*
	PB-R-TS3	27,83	13,17	12,17			
	PB-R-Wilson	*	*	*	*	*	*
0	PB-R-TS2	*	*	*	*	*	*
	PB-R-WST	*	*	*	*	*	*
	PB-R-TS3	*	*	*	*	*	*

gépek száma $M=7$; cellánként 5 feladat; $T=$ típusok száma; típusonként 5 munka; munkák száma $N=5T$; $K=$ paletták száma, $b=$ pufferek száma az egymást követő gépek közt ($b_i=b_2=\cdots=b_{M-1}$); *: legalább 1 feladat esetén 10 perc alatt nem kaptunk optimális megoldást

DOI: 10.15477/SZE.MMTDI.2017.009

5.4. táblázat. A PB-R-PFSP modelljeinél a futási idők átlaga és szórása (másodpercben) a 8 gépes feladatokban

1. rész: Átlagos futásidő

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	*	63,21	7,00	*	*	*
2	PB-R-TS2	*	51,60	4,06	*	*	*
	PB-R-WST	*	60,52	12,11	*	*	*
	PB-R-TS3	*	33,87	6,65	*	*	*
	PB-R-Wilson	*	12,53	5,65	*	*	*
1	PB-R-TS2	*	5,28	4,56	*	*	*
	R-PB-WST	*	34,49	9,56	*	*	*
	PB-R-TS3	*	10,26	4,80			
	PB-R-Wilson	*	*	*	*	*	*
0	PB-R-TS2	*	*	*	*	*	*
	PB-R-WST	*	*	*	*	*	*
	PB-R-TS3	*	*	*	*	*	*

2. rész: A futásidők szórásának átlaga

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	*	137,56	11,87	*	*	*
2	PB-R-TS2	*	11,69	6,09	*	*	*
	PB-R-WST	*	126,95	18,71	*	*	*
	PB-R-TS3	*	71,65	10,80	*	*	*
	PB-R-Wilson	*	21,62	7,54	*	*	*
1	PB-R-TS2	*	9,23	7,15	*	*	*
	R-PB-WST	*	68,74	13,33	*	*	*
	PB-R-TS3	*	18,71	7,57			
	PB-R-Wilson	*	*	*	*	*	*
0	PB-R-TS2	*	*	*	*	*	*
	PB-R-WST	*	*	*	*	*	*
	PB-R-TS3	*	*	*	*	*	*

gépek száma $M = 7$; cellánként 5 feladat; $T =$ típusok száma; típusonként 5 munka; munkák száma $N = 5T$; $K =$ paletták száma, $b =$ pufferek száma az egymást követő gépek közt ($b_i = b_2 = \cdots = b_{M-1}$); *: legalább 1 feladat esetén 10 perc alatt nem kaptunk optimális megoldást

DOI: 10.15477/SZE.MMTDI.2017.009

A PB-R-TS2 modellnek minden cellában kisebb volt az átlagos futásideje, mint a PB-R-Wilson, illetve PB-R-WST modelleké. A 10 cellából 7-nél a PB-R-TS3 modell átlagos futásideje kisebb volt, mint a PB-R-Wilson modellé és 9 cellánál a PB-R-TS3 modell átlagos futásideje kisebb volt, mint a PB-R-WST modellé. A 10 cellából 6-nál a PB-R-Wilson modell átlagos futásideje kisebb volt, mint a PB-R-WST modellé.

Mind a 10 cellában a PB-R-TS2 modellnek volt a legkisebb az átlagos szórása. 6 cellában a PB-R-TS3 modell átlagos szórása kisebb volt, mint a PB-R-Wilson modellé továbbá 9 cellában a PB-R-TS3 modell átlagos szórása kisebb volt, mint a PB-R-WST modellé. A PB-R-Wilson modell átlagos szórása 6 cellában volt kisebb a PB-R-WST modell átlagos szórásánál.

A fenti eredmények (hány példát oldottak meg a modellek; hány esetben voltak a leggyorsabbak; mennyi volt a futásidő alapján a rangjuk; futásidők cellánkénti átlaga és szórása) alapján azt mondhatjuk, hogy a futásidőt figyelembe véve a kis méretű feladatokat tartalmazó tesztkészleten a PB-R-TS2 modell volt a legjobb, megelőzve a PB-R-TS3, PB-R-Wilson és PB-R-WST modelleket.

A futásidőkön kívül a 4 modellt összehasonlítottuk a legjobb eredmények ($C_{\max}$ értékek) és a modelleknek a CPLEX megoldásával kapott alsó korlátai alapján is. A legjobb eredmények összehasonlításához három szempontot használtunk. Mind a négy modellre megnéztük, hogy a 180 feladatból hány esetén szolgáltatta az adott modell a legkisebb $C_{\max}$ értéket, továbbá kiszámítottuk a modellek $C_{\max}$ értékek szerinti rangjának átlagát. Ezen kívül mind a 4 modellre meghatároztuk, hogy mennyi a modell által adott $C_{\max}$ értékek relatív eltérése a 4 modell által kapott $C_{\max}$ értékek minimumától, azaz meghatároztuk az $U^*_{\text{modell}} = 100 \cdot (C^{\text{modell}}_{\max} - U_{\text{best}})/C^{\text{modell}}_{\max}$ értékeket, ahol $C^{\text{modell}}_{\max}$ az adott modell által a feladatra kapott megoldás célfüggvényértéke, U_{best} pedig a 4 modell által a feladatra adott 4 megoldás célfüggvényértékeinek a minimuma, majd mindegyik modellre kiszámítottuk ezen értékek átlagát. Az eredményeket az 5.5. táblázat tartalmazza. Az eredmények alapján látható, hogy mindhárom szempont alapján a PB-R-Wilson modell volt a legjobb, a PB-R-TS2 modell volt a második, a PB-R-TS3 modell volt a harmadik és a PB-R-WST modell volt az utolsó.

A modelleknek az alsó korlátok alapján való összehasonlításához szintén három szempontot vizsgáltunk. Megnéztük, hogy az adott modellt a CPLEX-szel megoldva hány feladatnál kaptuk a legnagyobbb alsó korlátot, illetve meghatároztuk a modellek rangjának átlagát a megoldásuk során kapott alsó korlátok alapján. Ezen kívül megnéztük, hogy mennyi a modellek megoldása során kapott alsó korlátnak a legjobb alsó korláttól való relatív eltérése, és meghatároztuk ezen értékek átlagát, azaz minden modellre kiszámítottuk az $L^*_{\text{modell}} = 100 * (L_{\text{best}} - L_{\text{modell}})/L_{\text{best}}$ értékek átlagát, ahol L_{modell} a feladatnak az adott modellnek a CPLEX-szel való megoldása során kapott alsó korlát, míg L_{best} ezen 4 alsó korlát közül a legnagyobbnak az értéke. Az eredményeket az 5.6. táblázatban foglaltam össze.

5.5. táblázat. A PB-R-PFSP modelljeinek összehasonlítása kis méretű feladatokon a $C_{\max}$ értékek alapján

Modell	Hány feladatnál adta a legkisebb $C_{\max}$ értéket	Rangok átlaga a $C_{\max}$ értékek alapján	Az U^*_{modell} értékek átlaga
PB-R-Wilson	134	2,41	0,089
PB-R-TS2	131	2,46	0,094
PB-R-WST	121	2,61	0,123
PB-R-TS3	125	2,53	0,106

$M \in \{7,8\}$; $T \in \{6,7\}$, $N = 5T$; $b_1 = b_2 = \cdots = b_{M-1} \in \{0,1,2\}$
$K \in \{8,9,10\}$, összesen 180 feladat; 10 perc futásidő
$U^*_{\text{modell}} = 100 * (C^{\text{modell}}_{\max} - U_{\text{best}})/C^{\text{modell}}_{\max}$

5.6. táblázat. A PB-R-PFSP modelljeinek összehasonlítása kis méretű feladatokon az alsó korlátok alapján

Modell	Hány feladatnál adta a legnagyobb alsó korlátot	Rangok átlaga az alsó korlátok alapján	Az L^*_{modell} értékek átlaga
PB-R-Wilson	149	2,40	0,042
PB-R-TS2	137	2,49	0,109
PB-R-WST	131	2,73	0,217
PB-R-TS3	153	2,37	0,100

$M \in \{7,8\}$; $T \in \{6,7\}$, $N = 5T$; $b_1 = b_2 = \cdots = b_{M-1} \in \{0,1,2\}$
$K \in \{8,9,10\}$, összesen 180 feladat; 10 perc futásidő
$L^*_{\text{modell}} = 100 * (L_{\text{best}} - L_{\text{modell}})/L_{\text{best}}$

Az eredményekből látható, hogy az alsó korlátok alapján a PB-R-Wilson és PB-R-TS3 modellek mindegyike mindhárom szempont szerint jobb volt a PB-R-TS2 és PB-R-WST modelleknél. A PB-R-TS2 modell mindegyik szempontban jobb volt a PB-R-WST modellnél. Az első két szempont szerint a PB-R-TS3 modell picivel jobb a PB-R-Wilson modellnél, viszont az L^* értékek átlaga a PB-R-Wilson modellnél jóval kisebb volt, mint a PB-R-TS3 modellnél, így azt mondhatjuk, hogy az alsó korlátok alapján a legjobban a PB-R-Wilson modell teljesített, megelőzve a PB-R-TS3, PB-R-TS2 és PB-R-WST modelleket.

Végezetül mind a 4 modellnél megvizsgáltuk a RELGAP értékek cellánkénti átlagát, ahol

$$\text{RELGAP}_{\text{modell}} = 100 \cdot \frac{C^{\text{modell}}_{\max} - L_{\text{best}}}{C^{\text{modell}}_{\max}}.$$

Az eredményeket az 5.7. és 5.8. táblázatok tartalmazzák. A táblázatokból látható, hogy

DOI: 10.15477/SZE.MMTDI.2017.009

5.7. táblázat. A PB-R-PFSP modellek RELGAP értékeinek cellánkénti átlaga és szórása a 7 gépes feladatokban

1. rész: A RELGAP értékek átlaga

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	0	0	0	1,24	0,57	0,29
2	PB-R-TS2	0	0	0	1,37	0,56	0,15
	PB-R-WST	0	0	0	1,17	0,68	0,07
	PB-R-TS3	0	0	0	1,26	0,67	0,12
	PB-R-Wilson	0	0	0	1,50	0,74	0,61
1	PB-R-TS2	0	0	0	1,55	0,83	0,60
	R-PB-WST	0	0	0	1,62	0,87	0,65
	PB-R-TS3	0	0	0	1,66	0,85	0,66
	PB-R-Wilson	2,17	3,07	3,16	7,13	7,07	7,04
0	PB-R-TS2	2,19	2,95	3,25	7,30	7,09	7,22
	PB-R-WST	2,11	2,72	3,33	7,31	7,39	7,29
	PB-R-TS3	2,24	2,75	3,40	7,28	7,20	6,93

2. rész: A RELGAP értékek szórása

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	0	0	0	2,35	1,27	0,65
2	PB-R-TS2	0	0	0	2,41	1,25	0,33
	PB-R-WST	0	0	0	2,42	1,53	0,16
	PB-R-TS3	0	0	0	2,39	1,51	0,27
	PB-R-Wilson	0	0	0	2,97	1,67	1,36
1	PB-R-TS2	0	0	0	2,81	1,81	1,33
	R-PB-WST	0	0	0	2,84	1,92	1,46
	PB-R-TS3	0	0	0	2,91	1,88	1,48
	PB-R-Wilson	3,08	3,07	3,17	5,56	5,55	5,27
0	PB-R-TS2	3,09	2,95	3,25	5,41	5,37	5,48
	PB-R-WST	3,04	2,72	3,33	5,88	5,46	5,29
	PB-R-TS3	3,26	2,76	3,40	5,50	5,67	5,22

gépek száma $M = 7$; cellánként 5 feladat; $T =$ típusok száma; típusonként 5 munka; munkák száma $N = 5T$; $K =$ paletták száma, $b =$ pufferek száma az egymást követő gépek közt ($b_i = b_2 = \cdots = b_{M-1}$);

DOI: 10.15477/SZE.MMTDI.2017.009

5.8. táblázat. A PB-R-PFSP modellek RELGAP értékeinek cellánkénti átlaga és szórása a 8 gépes feladatokban

1. rész: A RELGAP értékek átlaga

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	0,19	0	0	1,96	0,86	0,52
2	PB-R-TS2	0,14	0	0	2,27	0,92	0,42
	PB-R-WST	0,18	0	0	2,10	0,89	0,46
	PB-R-TS3	0,18	0	0	2,16	1,05	0,47
	PB-R-Wilson	0,17	0	0	2,34	1,04	0,62
1	PB-R-TS2	0,20	0	0	2,41	0,88	0,57
	R-PB-WST	0,27	0	0	2,17	1,13	0,67
	PB-R-TS3	0,21	0	0	2,26	1,11	0,60
	PB-R-Wilson	2,44	2,53	2,55	7,78	7,86	7,77
0	PB-R-TS2	2,38	2,50	2,48	7,65	7,62	7,77
	PB-R-WST	2,48	2,49	3,52	7,71	7,91	7,93
	PB-R-TS3	2,43	2,52	2,49	7,89	7,78	7,61

2. rész: A RELGAP értékek szórása

		$T=6$			$T=7$		
b	Modell	$K=8$	$K=9$	$K=10$	$K=8$	$K=9$	$K=10$
	PB-R-Wilson	0,43	0	0	3,01	1,96	1,16
2	PB-R-TS2	0,31	0	0	3,30	2,07	0,94
	PB-R-WST	0,41	0	0	3,56	1,99	1,02
	PB-R-TS3	0,41	0	0	3,33	2,33	1,06
	PB-R-Wilson	0,37	0	0	3,33	2,25	1,38
1	PB-R-TS2	0,45	0	0	3,60	1,96	1,28
	R-PB-WST	0,62	0	0	3,16	2,37	1,50
	PB-R-TS3	0,47	0	0	3,21	2,45	1,34
	PB-R-Wilson	3,01	3,08	3,06	4,28	4,37	4,24
0	PB-R-TS2	2,90	3,05	2,95	3,96	4,06	4,24
	PB-R-WST	2,94	2,98	3,09	4,22	4,39	4,28
	PB-R-TS3	3,03	3,07	3,04	4,29	4,05	4,04

gépek száma $M=8$; cellánként 5 feladat; $T=$ típusok száma; típusonként 5 munka; munkák száma $N=5T$; $K=$ paletták száma, $b=$ pufferek száma az egymást követő gépek közt ($b_i=b_2=\cdots=b_{M-1}$);

rögzített M, T és K esetén a puffer méretének (azaz b) csökkentése esetén mindegyik modellnél növekedett az átlagas RELGAP érték, továbbá a puffer méretének 1-ről 0-ra való csökkentése esetén az átlagos RELGAP érték sokkal nagyobb mértékben változott meg, mint a puffer méret 2-ről 1-re való csökkentése esetén. Ezért azt mondhatjuk, hogy a modellek számára a 0 puffer méretes feladatok sokkal nehezebbnek bizonyultak, mint az 1, illetve 2 puffer méretes feladatok. Hasonlóan rögzített M, T és b esetén $b \equiv 1$, $b \equiv 2$-re a paletták számának csökkentése során mindegyik modellnél növekedett az átlagos RELGAP érték, míg $b \equiv 0$ esetén ez a tendencia nem volt egyértelműen megfigyelhető.

A paletták száma változtatásának a hatásai

Megvizsgáltuk, hogy a paletták számának 1-gyel való változtatása hogyan változtatja meg az optimum értékét, illetve az egyes modellek futási idejét.

A paletták számának 8-ról 9-re való növelése: A 60 feladatból 32 olyan volt, melynél az optimum értékét 8, illetve 9 paletta esetén is ki tudtuk számítani valamelyik modell segítségével. Ebben a 32 példában a paletták számának 9-re való emelésekor az optimum értéke egyik esetben sem változott.

A PB-R-Wilson modell esetén 21 feladatnál sem 8, sem 9 palettával nem kaptuk meg az optimális megoldást 10 perc alatt. Összesen 7 olyan feladat volt, melyet a PB-R-Wilson modell $K = 9$ esetén meg tudott oldani, míg $K = 8$ esetén nem. 20 feladatnál a PB-R-Wilson modell gyorsabban oldotta meg a feladatot kilenc palettával, mint 8 palettával, míg 9 feladatnál 9 palettánál több volt a futásidő, mint 8 palettánál. 3 feladatnál a PB-R-Wilson modell futásideje 8, illetve 9 paletta esetén (két tizedesjegyre kerekítve) megegyezett.

Összesen 22 olyan feladat akadt, melyet a PB-R-TS2 modellel sem 8, sem 9 paletta esetén nem tudtunk megoldani. 8 olyan feladat volt, melyet a PB-R-TS2 modell kilenc palettával megoldott, nyolc palettával viszont nem tudott megoldani. 19 feladat esetén a paletták számát 8-ról 9-re növelve a PB-R-TS2 modell futásideje csökkent, míg 10 feladat esetén a futásidő nőtt. 1 feladatnál PB-R-TS2 modell futásideje nem változott a paletták számának 8-ról 9-re való emelése során.

A PB-R-WST modell esetén 22 feladatot sem nyolc, sem kilenc palettával nem tudtunk megoldani. 8 olyan feladat akadt, melyet a PB-R-WST modell nyolc palettával nem, kilenccel viszont meg tudott oldani. 1 feladat esetén a PB-R-WST modell segítségével 8 palettával meg tudtuk oldani a feladatot, míg 9 palettával nem. A paletták számát 8-ról 9-re növelve a PB-R-WST modell futásideje 28 feladatnál csökkent, míg 9 feladatnál növekedett.

A PB-R-TS3 modell esetén 22 feladatot sem nyolc, sem kilenc palettával nem tudtunk megoldani. A paletták számának 8-ról 9-re való emelésekor a PB-R-TS3 modell futásideje 29 esetben csökkent, míg 9 esetben nőtt.

DOI: 10.15477/SZE.MMTDI.2017.009

A paletták számának 9-ről 10-re való növelése: A 60 feladatból 38 esetben kaptuk meg az optimális megoldást valamelyik modell segítségével 9, illetve 10 paletta esetén is. Ebben a 38 példában a paletták számának 10-re való emelésekor az optimum értéke 36 esetben változatlan maradt, míg 2 példánál csökkent.

A PB-R-Wilson modell 20 feladatot sem 9, sem 10 palettával nem tudott megoldani 10 perc alatt. Egy olyan feladat volt, melyet a PB-R-Wilson modell kilenc palettával nem, 10 palettával viszont meg tudott oldani. A paletták számának 9-ről 10-re való emelésekor a PB-R-Wilson modell esetén 28 feladatnál csökkent, míg 11 feladatnál növekedett a futásidő.

A PB-R-TS2 modell esetén 21 feladatot sem 9, sem 10 palettával nem tudtunk megoldani, míg egy példát 10 palettával igen, 9-cel viszont nem tudott megoldani a modell. 24 esetben csökkent, 14 esetben pedig nőtt a PB-R-Wilson modell futásideje a paletták számának 9-ről 10-re való emelésekor.

A PB-R-WST modell esetén 21 feladatot sem 9, sem 10 palettával nem tudtunk megoldani az időkorlát alatt. Két olyan példa akadt, melyet a PB-R-WST modell 9 paletta esetén nem oldott meg, 10 paletta esetén viszont igen. A paletták számának 9-ről 10-re való növelésekor a PB-R-WST modell esetén 26 feladatnál csökkent, 10 feladatnál növekedett, 1 feladatnál pedig változatlan maradt a futásidő.

A PB-R-TS3 modell esetén 19 feladatot sem 9, sem 10 palettával nem tudtunk megoldani. Három feladatot 10 paletta esetén igen, 9 paletta esetén viszont nem tudtunk megoldani a PB-R-TS3 modell segítségével. A paletták számának 9-ről 10-re való növelésekor 23 esetben csökkent, 13 esetben nőtt, 2 esetben pedig változatlan maradt a PB-R-TS3 modell futásideje.
Az eredmények azt mutatják, hogy a paletták számának növelése az optimum értékére nem volt nagy hatással, viszont a modellek futtatásakor az optimális megoldás megtalálásának idejét csökkentette.

A gépek közötti pufferek száma változtatásának a hatásai

A pufferek számának 0-ról 1-re való növelése: A $b_r \equiv 0$ típusú 60 feladatból 15-ben kaptuk meg az optimum értékét $b_r \equiv 0$ és $b_r \equiv 1$ esetén is valamelyik modell segítségével. Ennél a 15 példánál az optimum értéke 9 esetben csökkent, 6 esetben pedig változatlan maradt, amikor a gépek közti pufferek száma 0-ról 1-re növekedett.

A 60 feladatból a PB-R-Wilson modell 11 esetben sem $b_r \equiv 0$, sem $b_r \equiv 1$ esetén nem adta meg az optimális megoldást. 37 olyan feladat volt, melyet a PB-R-Wilson modell $b_r \equiv 0$ esetén nem tudott megoldani, míg $b_r \equiv 1$ esetén megadta az optimális megoldást. A maradék 12 példa - amikor mind $b_r \equiv 0$ és $b_r \equiv 1$ esetén is megkaptuk az optimális megoldást - mindegyikében a futásidő a $b_r \equiv 1$ esetén volt kisebb.

A PB-R-TS2 modell esetén a 60-ból 12 olyan példa volt, melyet a modell sem $b_r \equiv 0$, sem $b_r \equiv 1$ esetén nem tudott megoldani. 38 feladat esetén a PB-R-TS2 modell $b_r \equiv 0$ esetén

megadta az optimális megoldást, míg $b_r \equiv 1$ esetén nem. A maradék 10 példa -amikor $b_r \equiv 0$ és $b_r \equiv 1$ esetén is megkaptuk az optimális megoldást- mindegyikében a pufferek számának növelésével a futásidő csökkent.

A PB-R-WST modell a 60 feladatból 12 példát sem $b_r \equiv 0$, sem $b_r \equiv 1$ esetén nem tudott megoldani. 38 olyan feladat volt, melyet a PB-R-WST modell $b_r \equiv 0$ esetén nem tudott megoldani, míg $b_r \equiv 1$ esetén meg tudta oldani. A maradék 10 példa mindegyikében a pufferek számának növelésekor a futásidő csökkent.

A PB-R-TS3 modellel 12 feladatot sem $b_r \equiv 0$, sem $b_r \equiv 1$ esetén nem tudtunk megoldani. 34 olyan feladat volt, melyet a modell $b_r \equiv 0$ esetén nem, míg $b_r \equiv 1$ esetén meg tudott oldani. Annál a 14 feladatnál, melyet a PB-R-TS3 modell $b_r \equiv 0$ és $b_r \equiv 1$ esetén is meg tudott oldani, a futásidő mindegyik esetben $b_r \equiv 1$ esetén volt a kisebb.

A pufferek számának 1-ről 2-re való növelése: A 60 feladatból 50 esetben $b_i \equiv 1$ és $b_i \equiv 2$ esetén is meg tudták határozni az optimum értékét a PB-R modellek az időkorláton belül. A gépek közti pufferek számának 1-ról 2-re való növelése következtében az 50-ből 8 példában csökkent az optimum értéke, míg 42 esetben változatlan maradt.

A PB-R-Wilson modell 10 feladatot sem, $b_i \equiv 1$ sem $b_i \equiv 2$ esetén nem tudott megoldani. 2 olyan feladat volt, melyet a PB-R-Wilson modell $b_i \equiv 2$ esetén meg tudott oldani, míg $b_i \equiv 1$ estén nem. A maradék 48 feladatot a PB-R-Wilson modell 1, illetve 2 pufferszámmal is meg tudta oldani. Ezen feladatoknál a pufferek számának 1-ről 2-re való növelésekor 33 esetben csökkent, 15 esetben nőtt, 1 esetben pedig változatlan maradt a futásidő.

A PB-R-TS2 modell 7 példa esetén sem 1, sem 2 puffer mérettel nem adta meg az optimális megoldást. 2 olyan feladat volt, melyet a PB-R-TS2 modell $b_i \equiv 2$ esetén igen, $b_i \equiv 1$ esetén viszont nem tudott megoldani. Egy feladat volt, melyet a PB-R-TS2 modell $b_i \equiv 1$ meg tudta oldani míg $b_i \equiv 2$ esetén már nem tudta megoldani. 50 feladatnál mind 1, mind 2 puffer méret esetén megtalálta a modell az optimális megoldást. Ezeknél 26 esetben csökkent, 20 esetben pedig növekedett, 1 esetben pedig nem változott a futásidő a pufferek száma növelésének hatására.

A PB-R-WST modell 10 feladatot sem 1, sem 2 puffer mérettel nem tudott megoldani. Két feladat esetén a PB-R-WST modell 2 puffer mérettel megadta az optimális megoldást, míg 1 puffer mérettel nem adott optimális megoldást. 48 feladatot mindkét puffer mérettel meg tudta oldani a modell. Ezeknél a feladatoknál $b_i \equiv 2$ esetén 32 esetben kevesebb, 14 esetben több, 2 esetben pedig ugyanannyi volt a futásideje a PB-R-WST modellnek, mint $b_i \equiv 1$ esetén.

A PB-R-TS3 modell esetén 11 feladatnál sem 1 sem 2 puffer mérettel nem kaptuk meg az optimális megoldást. Egy feladat volt, melyet a PB-R-TS3 modellel 2 puffer méretnél meg tudtunk oldani, míg 1 puffer méretnél a modell segítségével nem kaptuk meg az optimális megoldást. Azon 48 feladat esetén, melyeknél a PB-R-TS3 modell 1, illetve 2 pufferméret esetén is megadta az optimális megoldást, 26 esetben csökkent, 20 esetben nőtt, míg 2 esetben változatlan

DOI: 10.15477/SZE.MMTDI.2017.009

maradt a futásidő a puffer méret 1-ről 2-re való nöbelésekor.

A megfigyelések azt mutatják, hogy a gépek között a pufferek számának 0-ról 1-re való növelése mind az optimum értékét, mind a futásidőt jelentősen befolyásolja, míg a pufferek számának 1-ről 2-re való növelése az optimumokra nem gyakorolt olyan nagy hatást, de a futásidőt szintén csökkentette.

5.4.2. A PB-R-PFSP modelljeinek összehasonlítása nagy méretű feladatokon

A PB-R-PFSP 4 modelljét egy 60 példából álló, nagy méretű feladatokat tartalmazó tesztkészleten is összehasonlítottuk. A gépek száma 20, illetve 25, a típusok száma 10, illetve 15, a munkák száma pedig $120, 180$, illetve 240 volt a példákban. Minden lehetséges (M,N,T) hármashoz 5 feladatot generáltunk; a megmunkálási idők mátrixa a 3.6.2. fejezet tesztkészletében használttal egyezett meg. A puffer mérete az egymást követő gépek között 0 volt, a paletták száma pedig mindig 4-gyel volt nagyobb, mint a gépek száma. A 4 modellt a legjobb célfüggvényérték, illetve az alsó korlátok alapján hasonlítottuk össze.
A legjobb eredmények összehasonlításához az előző fejezetben használt három szempontot vettük figyelembe. Mind a négy modellre megnéztük, hogy a 60 feladatból hány esetén szolgáltatta az adott modell a legkisebb $C_{\max}$ értéket, továbbá kiszámítottuk a modellek $C_{\max}$ értékek szerinti rangjának átlagát. Ezen kívül mind a 4 modellre meghatároztuk, hogy mennyi a modell által adott $C_{\max}$ értékek átlagos relatív eltérése a 4 modell által kapott $C_{\max}$ értékek minimumától, azaz meghatároztuk a 92. oldalon definiált U^*_{modell} értékek átlagát. Az eredményeket az 5.9. táblázat tartalmazza. Az eredmények alapján látható, hogy a PB-R-TS3 modell mindhárom szempont szerint sokkal jobban teljesített a másik három modellnél, míg a PB-R-WST modell mindhárom szempont szerint sokkal rosszabbul teljesített a másik három modellnél. A három szempontot figyelembe véve a PB-R-Wilson és PB-R-TS2 modellek hasonlóan teljesítettek. Összességében tehát azt mondhatjuk, hogy a $C_{\max}$ értékekeket figyelembe véve a PB-R-TS3 modell volt a legjobb, a második helyen a PB-R-Wilson és PB-R-TS2 modellek végeztek, míg a PB-R-WST modell volt a negyedik.

A modelleknek az alsó korlátok alapján való összehasonlításához szintén három szempontot vizsgáltunk. Megnéztük, hogy az adott modellt a CPLEX-szel megoldva hány feladatnál kaptuk a legnagyobbb alsó korlátot, illetve meghatároztuk a modellek rangjának átlagát a megoldásuk során kapott alsó korlátok alapján és a 92 oldalon definiált L^*_{modell} értékek átlagát. Az eredményeket az 5.10. táblázat tartalmazza.

Az 5.10. táblázatból látható, hogy az első két szempont (hány esetben kaptuk a legjobb alsó korlátot; a rangok átlaga az alsó korlátok alapján) alapján a PB-R-TS3 modell volt a legjobb, megelőzve a PB-R-Wilson, PB-R-TS2 és PB-R-WST modelleket. Az U^* értékek átlaga alapján

DOI: 10.15477/SZE.MMTDI.2017.009

5.9. táblázat. A PB-R-PFSP modelljeinek összehasonlítása nagy méretű feladatokon a $C_{\max}$ értékek alapján

Modell	Hány feladatnál adta a legkisebb $C_{\max}$ értéket	Rangok átlaga a $C_{\max}$ értékek alapján	Az U^*_{modell} értékek átlaga
PB-R-Wilson	11	2,67	3,95
PB-R-TS2	10	2,57	3,74
PB-R-WST	3	3,27	5,56
PB-R-TS3	42	1,49	0,42

$M \in \{20{,}25\}$; $T \in \{10{,}15\}$; $N \in \{120{,}180{,}240\}$; $b_1 = b_2 = \cdots = b_{M-1} = 0$
$K = M + 4$; összesen 60 feladat; 10 perc futásidő
$U^*_{\text{modell}} = 100 * (C^{\text{modell}}_{\max} - U_{\text{best}})/C^{\text{modell}}_{\max}$

5.10. táblázat. A PB-R-PFSP modelljeinek összehasonlítása nagy méretű feladatokon az alsó korlátok alapján

Modell	Hány feladatnál adta a legnagyobb alsó korlátot	Rangok átlaga az alsó korlátok alapján	Az L^*_{modell} értékek átlaga
PB-R-Wilson	21	1,99	0,080
PB-R-TS2	20	2,21	0,043
PB-R-WST	2	3,9	*
PB-R-TS3	33	1,89	0,084

$M \in \{20{,}25\}$; $T \in \{10{,}15\}$; $N \in \{120{,}180{,}240\}$; $b_1 = b_2 = \cdots = b_{M-1} = 0$
$K = M + 4$; összesen 60 feladat; 10 perc futásidő
$L^*_{\text{modell}} = 100 * (L_{\text{best}} - L_{\text{modell}})/L_{\text{best}}$
*: 32 esetben nem kaptunk alsó korlátot 10 perc alatt

viszont a PB-R-TS2 modell volt a legjobb, a PB-R-TS3 és PB-R-Wilson modellek azonosan teljesítettek, a legrosszabb pedig a PB-R-WST modell volt. Ezek alapján azt mondhatjuk, hogy az alsó korlátok alapján a PB-R-TS3, PB-R-TS2 és PB-R-Wilson modellek nagyjából egyformán teljesítettek, míg a PB-R-WST modell sokkal rosszabb volt náluk. A PB-R-WST modellnél fontos megjegyezni, hogy a 60 feladatból 32 esetben a CPLEX nem tudta megoldani a relaxált feladatot, így alsó korlátot sem kaptunk 10 perc alatt. Mindegyik feladat esetén igaz volt, hogy a PB-R-WST modell esetén a CPLEX jóval lassabban oldotta meg a relaxált feladatot, mint a 3 másik modell esetén, sőt a PB-R-WST modell esetén a 10 perces futásidőnek nagy része arra ment el, hogy a CPLEX a relaxált feladatot megoldja. Ez egyúttal arra is magyarázatot szolgáltat, hogy miért a PB-R-WST modellel kaptuk legrosszabb $C_{\max}$ értékeket. A relaxált feladatok megoldásához szükséges futásidők átlagát az 5.11. táblázat tartalmazza.

Végezetül megvizsgáltuk az egyes modellek RELGAP értékeinek cellánkénti átlagát és

DOI: 10.15477/SZE.MMTDI.2017.009

5.11. táblázat. A PB-R-PFSP modellek futásideje (másodpercben) a nagy méretű feladatok relaxáltjain

		$M = 20$		$M = 25$	
N	Modell	$T = 10$	$T = 15$	$T = 10$	$T = 15$
	PB-R-Wilson	2,85	3,76	4,11	4,90
120	PB-R-TS2	2,59	3,77	5,58	6,67
	PB-R-WST	131,38	170,98	256,53	291,81
	PB-R-TS3	2,71	3,20	5,83	8,09
	PB-R-Wilson	7,69	8,94	7,89	10,72
180	PB-R-TS2	5,68	7,35	8,11	11,97
	R-PB-WST	406,83	*	*	*
	PB-R-TS3	10,35	8,04	10,95	15,11
	PB-R-Wilson	8,16	23,21	18,17	15,93
240	PB-R-TS2	7,39	9,96	19,17	14,94
	PB-R-WST	*	*	*	*
	PB-R-TS3	9,98	13,06	27,77	17,38

$M =$; gépek száma; $N =$ munkák száma; $T =$ típusok száma; típusonként 5 munka; paletták száma $K = M + 4$; cellánként 5 feladat; $b_i = b_2 = \cdots = b_{M-1} = 0$; 10 perc futásidő
*: 10 perc alatt nem sikerült a relaxált feladatot megoldani

szórását is. Az eredményeket az 5.12. táblázatban foglaltuk össze. A táblázatból látható, hogy a 12 cellából 11-nél a PB-R-TS3 modellnek volt a legkisebb átlagos RELGAP értéke, míg 1 cellánál a PB-R-TS2 modellnek volt a legkisebb az átlagos RELGAP értéke. 10 cellánál a PB-R-WST modellnek, 2 cellánál a PB-R-Wilson modellnek, 1 cellánál a PB-R-TS2 modellnek volt a legnagyobb az átlagos RELGAP értéke (1 cellánál holtverseny volt). A PB-R-TS2 modellnek 7 cellában volt kisebb az átlagos RELGAP-je, mint a PB-R-Wilson modellnek. Ezek az eredmények is azt a korábbi megállapításunkat támasztják alá, hogy a $C_{\max}$ értékek alapján a PB-R-TS3 modell volt a legjobb, a PB-R-WST modell volt a legrosszabb, a PB-R-TS2 és PB-R-Wilson modellek pedig nagyjából hasonlóan teljesítettek.

5.4.3. A PB-R-PFSP modelljeinek összehasonlítása ipari és ipari jellegű feladatokon

A PB-R-PFSP modelljeinek összehasonlításához egy ipari feladatunk is volt. Ebből a feladatból kiindulva az 5.3 fejezetben leírtak szerint további ipari jellegű tesztfeladatokat generáltunk. Összesen kilenc különböző méretű feladattípust hoztunk létre, $N \in \{100,110,120\}$

DOI: 10.15477/SZE.MMTDI.2017.009

5.12. táblázat. A PB-R-PFSP modellek RELGAP értékeinek cellánkénti átlaga és szórása a nagy méretű feladatokon

1. rész: A RELGAP értékek átlaga

		$M = 20$			$M = 25$		
T	Modell	$N = 120$	$N = 180$	$N = 240$	$N = 120$	$N = 180$	$N = 240$
10	PB-R-Wilson	21,93	23,93	22,00	21,75	23,21	27,26
	PB-R-TS2	21,36	24,87	22,55	20,43	24,11	26,39
	PB-R-WST	24,66	26,20	26,68	22,65	24,40	28,58
	PB-R-TS3	22,38	22,93	19,20	19,83	21,07	23,15
15	PB-R-Wilson	25,97	32,31	25,96	24,83	29,09	25,39
	PB-R-TS2	25,62	31,85	26,93	26,73	26,33	24,37
	R-PB-WST	26,21	31,35	27,93	24,67	29,09	27,04
	PB-R-TS3	24,10	27,49	22,17	22,89	23,14	22,66

2. rész: A RELGAP értékek szórása

		$M = 20$			$M = 25$		
T	Modell	$N = 120$	$N = 180$	$K = 240$	$N = 120$	$N = 180$	$N = 240$
10	PB-R-Wilson	3,47	3,37	7,17	2,20	6,61	6,01
	PB-R-TS2	4,51	3,11	6,96	4,71	5,35	7,17
	PB-R-WST	3,53	3,44	3,23	1,61	3,62	5,14
	PB-R-TS3	4,49	1,54	4,04	2,39	4,19	4,79
15	PB-R-Wilson	4,33	2,57	6,23	2,40	3,72	4,18
	PB-R-TS2	4,49	2,40	4,73	2,33	6,86	4,61
	R-PB-WST	4,31	3,01	3,92	1,37	3,72	4,36
	PB-R-TS3	4,69	1,46	7,20	2,87	4,15	4,47

$M =$ gépek száma; $N =$ munkák száma; $T =$ típusok száma; típusonként ugyanannyi munka; $K = M + 4$; $b_1 = b_2 = \cdots = b_{M-1} = 0$; 10 perc futásidő

és $T \in \{5,6,7\}$ paraméterekkel. Mindegyik (N,T) párhoz 5 tesztfeladat tartozott. A gépek száma mindegyik példában 48, míg a paletták száma mindegyik példában 53 volt. A különböző típusokból eltérő számú munkák voltak, a gépek közti puffer méret is gépenként változott. A futási idők cellánkénti átlagát és szórását az 5.13. táblázatok tartalmazzák.

A modellek sorrendje az átlagos futási idő alapján a 9 cella mindegyikében megegyezett: mindegyik cellában a PB-R-TS3 modellnél volt a legkisebb az átlagos futási idő, továbbá a PB-R-TS3 modellt mindegyik cellában a PB-R-TS2, majd a PB-R-Wilson, végül a PB-R-WST

DOI: 10.15477/SZE.MMTDI.2017.009

5.13. táblázat. A PB-R-PFSP modellek futásidejének átlaga és szórása (másodpercben) az ipari jellegű feladatokon

1. rész: Átlagos futásidő

N	T	PB-R-TS3	PB-R-TS2	PB-R-Wilson	PB-R-WST
100	5	33,6	101,8	192.6	318,1
	6	24,3	127,0	168,9	*
	7	27,5	201,3	303,6	*
110	5	54,7	207,1	335,6	*
	6	12,2	103,7	236,7	*
	7	33,7	109,6	311,6	*
120	5	55,8	209,1	*	*
	6	36,2	190,3	292,75	*
	7	33,5	87,8	366,4	*

2. rész: A futásidők szórása

100	5	27,1	78,6	135,1	201,45
	6	17,4	99,7	108,2	*
	7	18,1	98,7	252,2	*
110	5	15,7	82,4	146,1	*
	6	4,7	77,9	116,0	*
	7	16,0	43,1	88,9	*
120	5	22,7	131,9	*	*
	6	28,0	151,3	191,3	*
	7	22,5	17,3	83,0	*

$N =$ munkák száma; $T =$ típusok száma, gépek száma $M = 48$ paletták száma $K = 53$; különböző típusokból különböző számú munka; gépek közt változó puffer méret; *: legalább egy feladatra 10 perc alatt alatt nem kaptunk optimális megoldást

modellek követték.

A futásidők szórása alapján 8 cellánál a PB-R-TS3 modell volt a legjobb, megelőzve a PB-R-TS2, PB-R-Wilson és PB-R-WST modelleket. A szórások alapján egy cellában PB-R-TS2, PB-R-TS3, PB-R-Wilson és PB-R-WST volt a modellek sorrendje. A futásidők cellánkénti átlagán és szórásán kívül megvizsgáltuk még, hogy az egyes modellek segítségével hány feladatot tudtunk megoldani, hány esetben kaptuk az adott modellel a legjobb megoldást, illetve azt is, hogy mennyi volt az egyes modellek rangjának az átlaga a futásidő alapján. Az eredményeket

5.14. táblázat. A PB-R-PFSP modelljeinek összehasonlítása ipari jellegű feladatokon a futásidő alapján

Modell	Hány feladatnál adta meg az optimális megoldást	Hány feladatnál volt a leggyorsabb alapján	Rangok átlaga a futásidő alapján
PB-R-Wilson	44	1	3,1
PB-R-TS2	45	1	2,67
PB-R-WST	23	1	3,54
PB-R-TS3	45	42	1,09

$N \in \{120,180,240\}$; $T \in \{10,15\}$; $M \in \{20,25\}$; $K = 53$; 10 perc futásidő különböző típusokból különböző számú munka; gépek közt változó puffer méret

az 5.14. táblázat tartalmazza. A PB-R-TS3 és PB-R-TS2 modelleknél mind a 45 esetben, a PB-R-Wilson modellnél 44 esetben, míg a PB-R-WST modellnél 22 esetben kaptuk meg az optimális megoldást kevesebb, mint 10 perc alatt. A 45 példából 42 esetben a PB-R-TS3 modellel kaptuk meg a leggyorsabban az optimális megoldást. A rangok alapján a PB-R-TS3 modell volt a legjobb, megelőzve a PB-R-TS2, PB-R-Wilson és PB-R-WST modelleket. Az összes példát tekintve a PB-R-TS3 modellnél a leghosszabb futási idő 82,6 másodperc volt, míg ugyanez az érték a PB-R-TS2 modellnél 399,7 másodperc volt. A PB-R-Wilson és PB-R-WST modell segítségével nem tudtuk az összes feladatot megoldani 10 percen belül.

A teszteredmények tehát azt mutatják, hogy az ipari jellegű feladatok megoldásában a PB-R-TS3 modell volt a leghatékonyabb, a második legjobbnak a PB-R-TS2 modell bizonyult, a harmadik a PB-R-Wilson modell volt, míg a legrosszabb a PB-R-WST modell volt. Az eredmények alapján továbbá azt mondhatjuk, hogy a PB-R-TS3 modell segítségével akár nagy méretű, ipari feladatok is hatékonyan megoldhatók.

5.5. A futási eredmények értékelése

A futási eredményekből az alábbi következtetések vonhatók le.

- A kis méretű feladatokat tartalmazó tesztkészleten való futtatások azt mutatták, hogy
  1. a paletták számának növelése az optimum értékére nincs nagy hatással, viszont a modellek futásidejét kis mértékben csökkenti;
  2. a pufferek számának növelése jelentősen csökkenti a modellek futásidejét;
  3. a 0 puffert tartalmazó feladatok a megoldhatóság szempontjából sokkal nehezebbek az 1, illetve 2 puffert tartalmazó feladatoknál;
  4. a futásidők alapján a 4 modell sorrendje PB-R-TS2, PB-R-TS3, PB-R-Wilson, PB-R-WST volt;

DOI: 10.15477/SZE.MMTDI.2017.009

5. a legjobb célfüggvényértékek alapján a modellek sorrendje PB-R-Wilson, PB-R-TS2, PB-R-TS3, PB-R-WST volt;
6. az alsó korlátok alapján PB-R-Wilson, PB-R-TS3, PB-R-TS2, PB-R-WST volt a modellek sorrendje.

- A nagy méretű feladatokat tartalmazó tesztkészleten való futtásokból megállapítható, hogy

  1. a relaxált feladatot a PB-R-WST modell oldja meg a leglassabban;
  2. a legjobb célfüggvényértékek alapján a PB-R-TS3 modell volt a legjobb, a második a PB-R-TS2 és PB-R-Wilson modell volt, míg a legrosszabb a PB-R-WST modell volt;
  3. az alsó korlátok alapján a PB-R-WST modell volt a legrosszabb, míg a másik három modell nagyjából egyformán teljesített.

- Az ipari jellegű feladatokat a PB-R-TS3 modell jóval hatékonyabban oldotta meg, mint a PB-R-TS2, PB-R-Wilson, PB-R-WST modellek.

DOI: 10.15477/SZE.MMTDI.2017.009

6. fejezet

Palettát, véges puffert és lot méretet tartalmazó R-PFSP

A 3. fejezetben bevezettük az ismétlődő munkákat tartalmazó permutációs flow shop probléma (R-PFSP) fogalmát. A 4. fejezetben bevezetett lot méretet tartalmazó ismétlődő permutációs flow shop feladat (RL-PFSP) az R-PFSP egy általánosítása, hiszen egy (lot méretet nem tartalmazó) R-PFSP tekinthető egy $L = 1$ lot méretet tartalmazó RL-PFSP-nek. Az 5. fejezetben bevezetett palettát és véges puffert tartalmazó PFSP (PB-R-PFSP) szintén az R-PFSP egy általánosítása, mert egy palettákat nem tartalmazó R-PFSP tekinthető egy olyan PB-R-PFSP-nek, melyben a paletták száma megegyezik a munkák számával, míg egy puffer méreteket nem tartalmazó PFSP tekinthető egy olyan PB-R-PFSP-nek, melyben a puffer mérete a szomszédos gépek között a munkák számával egyezik meg (hiszen ekkor két gép között akármennyi munka várakozhat). Ebben a fejezetben megmutatjuk, hogy ez a négy speciális tulajdonság - az ismétlődő munkák, lot méret, paletták, véges puffer - együtt is modellezhető.

Palettát, véges puffert és lot méretet tartalmazó ismétlődő permutációs flow shop feladatnak (PB-RL-PFSP) nevezünk egy olyan PFSP-t, melyben

- a munkák között vannak azonos típusúak;
- az azonos típusú munkákat tartalmazó, maximális hosszú sorozatok hosszának osztathónak kell lennie a lot mérettel;
- minden munkát egy paletta szállít a gépek között és a paletták száma véges;
- az egymást követő gépek között a puffer méret véges.

A következőkben az RL-PFSP, illetve a PB-R-PFSP modellek segítségével megadjuk a PB-RL-PFSP MILP modelljeit.

DOI: 10.15477/SZE.MMTDI.2017.009

6.1. A PB-RL-PFSP MILP modelljei

A MILP modellekben a következő jelöléseket használjuk:

Paraméterek:

M a gépek száma

N a munkadarabok száma

T a különböző típusok száma

n_t a t típusú munkadarabok száma ($1 \leq t \leq T$; $\sum n_t = N$)

P_{ri} az i típusú munkadarab megmunkálási ideje az r gépen

S a lot mérete

L_i az i típusú munkákat tartalmazó lotok száma ($1 \leq i \leq T$, $L_i = n_i/S$)

L az összes lot száma ($L = L_1 + L_2 + \ldots L_T$)

K a paletták száma

b_r az r. és $(r+1)$. gépek közti puffer nagysága ($1 \leq r \leq M-1$;)

Folytonos változók:

C_{rj} a sorrend j. helyén álló munkadarab megmunkálásának befejezési ideje az r gépen

B_{rj} a sorrend j. helyén álló munkadarab megmunkálásának kezdési ideje az r gépen

X_{rj} állóidő az r gépen a sorrend j. munkadarabjának a megmunkálásának a kezdése előtt

Y_{rj} a sorrend j. munkadarabjának a várakozási ideje a pufferban, miután megmunkálták az az r gépen

$C_{\max}$ átfutási idő.

Bináris változók:

Z_{ij} bináris változó; $Z_{ij} = 1$ pontosan akkor, ha a j. lot i típusú munkákat tartalmaz ($1 \leq i \leq T, 1 \leq j \leq L$).

A PB-RL-PFSP MILP modelljeiben egy permutációt úgy adunk meg, hogy a lotok sorrendjében minden lotra előírjuk, hogy milyen típusú munkát tartalmaz. Mivel a lot méret L, ezért a munkák sorrendjének j. helyén álló munka a $\lceil j/L \rceil$-edik lotnak a tagja, ahol $\lceil\rceil$ a felső egészrészt jelöli. Ezért a munkák sorrendjében j. tagnak az r gépen való megmunkálási ideje, amit D_{rj}-vel jelölünk, a

$$D_{rj} = \sum_{i=1}^{T} P_{ri} Z_{i,\lceil j/L \rceil} \tag{6.1}$$

képlettel számítható ki. D_{rj} tehát nem egy változó, hanem a MILP modellekben a $\sum_{i=1}^{T} P_{ri} Z_{i,\lceil j/L \rceil}$ összeg helyett használt egyszerűbb jelölés. A PB-RL-PFSP alábbi MILP mo-

DOI: 10.15477/SZE.MMTDI.2017.009

delljeit az RL-PFSP, illetve PB-R-PFSP 4., illetve 5. fejezetben leírt MILP modelljeinek az ”összerakásából” kapjuk.

6.1.1. A PB-RL-Wilson modell

A PB-RL-Wilson modell a következő feltételeket tartalmazza:

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{6.2}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad 1 \leq j \leq L \tag{6.3}$$

- A sorrend első munkája várakozás nélkül halad végig a gépeken.

$$B_{r1} + D_{r1} = B_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{6.4}$$

- Egy munkát addig nem kezdhetünk megmunkálni egy gépen, amíg be nem fejeztük az előző gépen.

$$B_{rj} + D_{rj} \leq B_{r+1,j} \qquad 1 \leq r \leq M-1;\ 2 \leq j \leq N \tag{6.5}$$

- Egy munkadarabot egy tetszőleges gépen addig nem kezdhetünk el, míg a sorrendben őt közvetlenül megelőző munkadarabot meg nem munkáltuk az adott gépen.

$$B_{rj} + D_{rj} \leq B_{r,j+1} \qquad 1 \leq r \leq M;\ 1 \leq j \leq N-1 \tag{6.6}$$

- Legfeljebb K darab palettát használhatunk.

$$B_{M,j-K} + D_{M,j-K} \leq B_{1j} \qquad K+1 \leq j \leq N \tag{6.7}$$

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$B_{r+1,j-b_r-1} \leq B_{rj} \qquad 1 \leq r \leq M-1,\ 2+b_r \leq j \leq N \tag{6.8}$$

DOI: 10.15477/SZE.MMTDI.2017.009

- Az átfutási idő a sorrend utolsó munkájának az utolsó gépen való befejezési ideje.

$$\min\ C_{\max} = B_{MN} + D_{MN} \tag{6.9}$$

A PB-RL-Wilson modell tehát a következőképpen összegezhető: minimalizáljuk (6.9)-et, a (6.2) – (6.8) feltételek mellett.

6.1.2. A PB-RL-TS2 modell

A PB-R-TS2 modell a következő feltételeket tartalmazza.

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad\qquad 1 \leq i \leq T \tag{6.10}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad\qquad 1 \leq j \leq L \tag{6.11}$$

- A sorrendben $(j+1)$. munkadarab befejezési ideje egy gépen nem lehet kisebb, mint a sorrend j. munkadarabjának a befejezési ideje ugyanazon a gépen plusz a sorrend $(j+1)$. munkadarabjának a megmunkálási ideje az adott gépen.

$$C_{rj} + D_{r,j+1} \leq C_{r,j+1}, \qquad 1 \leq r \leq M, 1 \leq j \leq N-1 \tag{6.12}$$

- Egy munkadarab befejezési ideje egy gépen nem lehet kisebb, mint a munkadarabnak a befejezési ideje a megelőző gépen plusz a munkadarab megmunkálási ideje az adott gépen.

$$C_{rj} + D_{r+1,j} \leq C_{r+1,j}, \qquad 1 \leq r \leq M-1, 1 \leq j \leq N \tag{6.13}$$

- A sorrend első munkáját az első gépen nem fejezhetjük be korábban az első gépen való megmunkálási idejénél.

$$D_{11} \leq C_{11} \tag{6.14}$$

- Legfeljebb K darab palettát használhatunk.

$$C_{M,j-K} + D_{1j} \leq C_{1j} \qquad K+1 \leq j \leq N \tag{6.15}$$

DOI: 10.15477/SZE.MMTDI.2017.009

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$\begin{aligned} C_{r+1,j-b_r-1} - D_{r+1,j-b_r-1} + D_{rj} &\leq C_{rj} \\ 1 &\leq r \leq M-1, 2+b_r \leq j \leq N \end{aligned} \tag{6.16}$$

- Az átfutási idő a sorrend utolsó munkájának az utolsó gépen való befejezési ideje.

$$\min\ C_{\max} = C_{MN} \tag{6.17}$$

A PB-RL-TS2 modell így a következőképpen összegezhető:
minimalizáljuk (6.17)-et, a (6.10) – (6.16) feltételek mellett.

6.1.3. A PB-RL-WST modell

A PB-RL-WST modell a következő egyenlőtlenségeket tartalmazza.

- Az i típusú munkadarabot tartalmazó lotok száma pontosan L_i.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{6.18}$$

- A lotok sorrendjének mindegyik helyére pontosan egy típusú lot kerül.

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad 1 \leq j \leq L \tag{6.19}$$

- A sorrend első munkadarabjának a megmunkálása folyamatos, azaz egyetlen gép előtt sem kell várakoznia.

$$Y_{r1} = 0, \qquad 1 \leq r \leq M-1 \tag{6.20}$$

- A sorrend $(j+1)$. tagját addig nem kezdhetjük el egy gépen, míg az előző gépen be nem fejeztük, illetve míg a sorrend j. tagját be nem fejeztük ugyanezen a gépen.

$$\begin{aligned} D_{r,j+1} + X_{r,j+1} + Y_{r,j+1} &= D_{r+1,j} + X_{r+1,j+1} + Y_{rj} \\ & 1 \leq r \leq M-1;\ 1 \leq j \leq N-1 \end{aligned} \tag{6.21}$$

- Tetszőleges r gépen az első munka előtti állóidő megegyezik a sorrend első munkájának az első $r-1$ gépen való megmunkálási idejének összegével.

$$X_{r1} + Y_{r1} + D_{r1} = X_{r+1,1} \qquad 1 \leq r \leq M-1 \tag{6.22}$$

- Legfeljebb K darab palettát használhatunk.

$$\begin{aligned} \sum_{i=1}^{j-K} X_{Mi} + \sum_{i=1}^{j-K} D_{Mi} &\leq \sum_{i=1}^{j} X_{1i} + \sum_{i=1}^{j-1} D_{1i} \\ K &+ 1 \leq j \leq N \end{aligned} \tag{6.23}$$

- Az r és $r+1$ gépek között legfeljebb b_r munka várakozhat.

$$\begin{aligned} \sum_{i=1}^{j-b_r-1} X_{r+1,i} + \sum_{i=1}^{j-b_r-2} D_{r+1,i} &\leq \sum_{i=1}^{j} X_{ri} + \sum_{i=1}^{j-1} D_{ri} \\ 1 &\leq r \leq M-1, 2+b_r \leq j \leq N \end{aligned} \tag{6.24}$$

- Az átfutási idő az utolsó gépen lévő állóidők és megmunkálási idők összege.

$$\min\ C_{\max} = \sum_{i=1}^{T} n_i \cdot P_{Mi} + \sum_{p=1}^{N} X_{Mp} \tag{6.25}$$

A PB-RL-WST modell az alábbi módon foglalható össze:
minimalizáljuk (6.25)-öt, a (6.18) – (6.24) feltételek mellett.

6.1.4. A PB-RL-TS3 modell

A PB-RL-TS3 modell a lotokra vonatkozó két feltételen kívül a PB-R-TS3 modell feltételeit tartalmazza.

$$\sum_{j=1}^{L} Z_{ij} = L_i \qquad 1 \leq i \leq T \tag{6.26}$$

$$\sum_{i=1}^{T} Z_{ij} = 1 \qquad 1 \leq j \leq L \tag{6.27}$$

$$\begin{aligned} \sum_{q=2}^{r} D_{qj} + \sum_{q=1}^{r-1} Y_{qj} &\leq Y_{1,j-1} + \sum_{q=1}^{r-1} D_{q,j+1} + \sum_{q=1}^{r-1} Y_{q,j+1} \\ 2 &\leq r \leq M, 1 \leq j \leq N-1 \end{aligned} \tag{6.28}$$

DOI: 10.15477/SZE.MMTDI.2017.009

$$\sum_{r=1}^{M-1} Y_{r,j-K} + \sum_{r=2}^{M} D_{r,j-K} \leq \sum_{i=j-K+1}^{j} X_{1i} + \sum_{i=j-K+1}^{j-1} D_{1i} \quad (K < j) \tag{6.29}$$

$$\sum_{q=1}^{r} Y_{q,j-b_r} + \sum_{q=2}^{r} D_{q,j-b_r} \leq \sum_{i=j-b_r+1}^{j+1} X_{1i} + \sum_{i=j-b_r+1}^{j} D_{1i} + \sum_{q=1}^{r-1} Y_{q,j+1} + \sum_{q=1}^{r-1} D_{q,j+1} \tag{6.30}$$

$$1 \leq r \leq M-1, b_r < j \leq N-1$$

$$\min C_{\max} = \sum_{i=1}^{N} X_{1i} + \sum_{i=1}^{N-1} D_{1i} + \sum_{q=1}^{M-1} Y_{qN} + \sum_{q=1}^{M} D_{qN} \tag{6.31}$$

A PB-RL-PFSP új, PB-R-TS3 modellje tehát a következőképpen összegezhető: minimalizáljuk (6.31)-et, a (6.26) – (6.30) feltételek mellett.

DOI: 10.15477/SZE.MMTDI.2017.009

7. fejezet

Összegzés és kitekintés

Gyártósorok ütemezésének klasszikus matematikai modellje a permutációs flow shop feladat. A téma feldolgozása során áttekintettem a PFSP különféle heurisztikus, illetve egzakt megoldó módszereit. Az egzakt módszerek egyik típusában a PFSP-t egy vegyes egészértékű lineáris programozási feladatként (MILP) modellezzük, melyet aztán egy alkalmas szoftverrel megoldunk.

Kutatásaim során megállapítottam, hogy az ipari feladatoknak számos olyan sajátossága van, melyet a klasszikus PFSP nem vesz figyelembe. Az egyik ilyen tulajdonság, hogy a valós életben az ütemezendő munkák nem mind különbőzőek (a különböző típusok száma a munkák számához képest általában kicsi). Az ismétlődő munkákat figyelembe véve bevezettem az ismétlődéses permutációs flow shop feladat fogalmát (R-PFSP) és az irodalomban korábban ismert MILP modellek módosításával az R-PFSP-re új MILP modelleket adtam. A modellek összehasonlításához egy tesztkészletet generáltam. A MILP modelleket több szempont alapján is összehasonlítottam, továbbá megvizsgáltam, hogy a modelleknek a különböző szempontok szerinti sorrendjét a CPLEX beállításai hogyan befolyásolják.

Az ipari termelésben a munkák lehetséges sorrendjét számos feltétel befolyásolhatja. Például ha a munkákat egy speciális eszközön szállítják a sor elejére, akkor az egymást követő, azonos típusú munkákból álló blokkok méretének mindig oszthatónak kell lennie egy adott számmal (a lot mérettel). Ezen feltételt figyelembe véve bevezettem a lot méretet tartalmazó ismétlődéses permutációs flow shop feladat fogalmát (RL-PFSP) és megadtam a feladat 3 MILP modelljét. A modelleket tesztfeladatokon futtatva megállapítottam, hogy lotok alkalmazásával növelhető a MILP modellekkel megoldható feladatok mérete (munkák, illetve gépek száma), továbbá ha a munkák száma a lot mérethez képest nagy, akkor a lot méretet tartalmazó feladat optimális megoldása a lot méret nélküli feladat egy jó közelítő megoldását adja.

A munkáknak gyártósoron való szállítása történhet paletták segítségével. Ezen fizikai feltételt modellezve bevezettem a palettát és véges puffert tartalmazó PFSP (PB-R-PFSP) fogalmát. Az R-PFSP modelljeit általánosítottam a PB-R-PFSP-re, majd egy helyettesítési tech-

DOI: 10.15477/SZE.MMTDI.2017.009

nika segítségével a PB-R-PFSP egy negyedik modelljét (PB-R-TS3) kaptam meg. Ipari jellegű jellegű feladatokon tesztelve a 4 modellt megmutattam, hogy a PB-R-TS3 modell mindhárom modellnél gyorsabb.

A későbbiekben további feltételekkel szeretném bővíteni a MILP modelleket, hogy azok a valós ipari feladatok minél pontosabb modelljét adják. A modellek által kapott eredmény pontosságát az iparban használt termelésirányítási szoftverek segítségével szeretném megvizsgálni. A továbblépés egy másik lehetősegeként megvizsgálnám, hogy más típusú ütemezési feladatoknál (például a job shop feladatoknál) az ipari sajátosságokat figyelembe véve javíthatók-e az irodalomból jól ismert modellek.

DOI: 10.15477/SZE.MMTDI.2017.009

8. fejezet

Tézisek

1. Tézis

(a) Gyártósorok működésének modellezésére az ismétlődő munkák figyelembevételével bevezettem az ismétlődő permutációs flow shop probléma (R-PFSP) fogalmát. Az R-PFSP-nek megadtam 3 új vegyes egészértékű lineáris programozási (MILP) modelljét (R-Wilson, R-TS2, R-WST).

(b) Megmutattam, hogy az új R-TS2, R-Wilson és R-WST modellek relaxáltjainak az optimuma megegyezik.

(c) Az R-Wilson, R-TS2 és R-WST modellek összehasonlításához két tesztkészletet is generáltam. A modellek teszteléséből megállapítottam, hogy az új R-Wilson, R-TS2 és R-WST modellekkel gyorsabban oldhatunk meg R-PFSP-ket, mint a szakirodalomból korábban ismert modellekkel. Kimutattam, hogy a kis méretű feladatokon a futásidő alapján az R-Wilson modell volt a legjobb, megelőzve az R-WST és R-TS2 modelleket.

(d) A nagy méretű feladatoknál kimutattam, hogy a legjobb célfüggvényértékek alapján (a CPLEX alapbeállításait használva) az R-WST modell volt a legjobb, míg az R-Wilson és R-TS2 modellek között nem volt lényeges eltérés, a legjobb alsó korlátok szerint pedig az R-WST modell volt a legrosszabb, az R-Wilson és R-TS2 modellek között pedig nem volt lényeges eltérés.

(e) Mindhárom modell segítségével kevesebb, mint 8 perc alatt sikerült megoldani egy 57 gépet, 227 munkát, 12 típust tartalmazó ipari feladatot.

(f) Az R-PFSP MILP modelljeit összehasonlítottam a NEH, PACO és TABU keresés heurisztikákkal. Megállapítottam, hogy a kevés típust (5) tartalmazó 10 feladat esetén az R-WST modell 7 esetben adott jobb megoldást, mint a NEH, és 5 esetben adott jobb

DOI: 10.15477/SZE.MMTDI.2017.009

megoldást, mint a TABU keresés. Mindhárom MILP modell esetén a modellt a CPLEX-szel megoldva a feladatok közel felénél jobb alsó korlátot kaptunk, mint az irodalomból ismert $LB5$ korlát.

2. Tézis

(a) A nagy méretű feladatokon megvizsgáltam, hogy a CPLEX három opciójának (varsel, heurfreq, cuts) változtatása milyen hatással van az R-PFSP 3 új modelljének rangsorára. Kimutattam, hogy a 3 modellnek a legjobb célfüggvényértékek szerinti rangsorát a 3 opció változtatása nem befolyásolja, míg a modelleknek a legjobb alsó korlátok szerinti rangsorát a beállítások befolyásolják.

(b) A nagy méretű feladatokon kimutattam, hogy a legjobb célfüggvényértéket az R-WST modellnek és a varsel$= 0$, heurfreq$= 20$, cuts$= 0$ beállításnak a segítségével érhetjük el.

(c) A nagy méretű feladatokon kimutattam, hogy a legjobb alsó korlátokat az R-WST modellnek és a varsel$= 2$, heurfreq$= 0$, cuts$= 0$ beállításnak a segítségével érhetjük el.

3. Tézis

(a) Az iparban az egyes munkákat sok esetben egy úgynevezett lotban szállítják a gyártósor elejére, ami a munkák sorrendjére ad speciális feltételt. A lotok modellezésére bevezettem a lot méretet tartalmazó ismétlődő permutációs flow shop probléma (RL-PFSP) fogalmát. Az RL-PFSP-nek megadtam 3 új MILP modelljét (RL-Wilson, RL-TS2, RL-WST).

(b) A tesztpéldákon való futtatás alapján megállapítottam, hogy

i. az egyszerűbb feladatokat az RL-WST, míg a nehezebbeket az RL-Wilson modell oldotta meg a leggyorsabban;

ii. a lot méret növelésével a (10 percen belül) megoldható feladatok mérete növekszik;

iii. nagyméretű R-PFSP-k egy jó közelítő megoldását kaphatjuk a feladatot egy alkalmasan választott lot mérettel megoldva.

4. Tézis

(a) A gyártósoron az egyes munkákat sok helyen paletták szállítják, melyeknek a száma korlátozott. Az ebből adódó fizikai feltételeket figyelembe véve bevezettem a palettát és véges puffert tartalmazó ismétlődő permutációs flow shop probléma (PB-R-PFSP) fogalmát. Az R-PFSP modelljeinek általánosításával a PB-R-PFSP-nek felírtam 3 MILP modelljét (PB-R-Wilson, PB-R-TS2, PB-R-WST).

DOI: 10.15477/SZE.MMTDI.2017.009

(b) Egy helyettesítési technika segítségével a PB-R-TS2 modellből előállítottam egy negyedik modellt, a PB-R-TS3 modellt. Megállapítottam, hogy a 4 modell közül a PB-R-TS3 modell tartalmazza a legkevesebb bináris változót, folytonos változót, illetve egyenlőtlenséget.

(c) Tesztfeladatokat generáltam a PB-R-PFSP-hez. A kis méretű feladatokat tartalmazó tesztkészleten való futtatások során megállapítottam, hogy

i. a paletták számának növelése az optimum értékére nincs nagy hatással, viszont a modellek futásidejét kis mértékben csökkenti;

ii a pufferek számának növelése jelentősen csökkenti a modellek futásidejét;

iii. a 0 puffert tartalmazó feladatok a megoldhatóság szempontjából sokkal nehezebbek az 1, illetve 2 puffert tartalmazó feladatoknál;

iv. a futásidők alapján a 4 modell sorrendje PB-R-TS2, PB-R-TS3, PB-R-Wilson, PB-R-WST volt;

v. a legjobb célfüggvényértékek alapján a modellek sorrendje PB-R-Wilson, PB-R-TS2, PB-R-TS3, PB-R-WST volt;

vi az alsó korlátok alapján PB-R-Wilson, PB-R-TS3, PB-R-TS2, PB-R-WST volt a modellek sorrendje

(b) A nagy méretű feladatokat tartalmazó tesztkészleten való futtásokból megállapítottam, hogy

i. a relaxált feladatot a PB-R-WST modell oldja meg a leglassabban;

ii. a legjobb célfüggvényértékek alapján a PB-R-TS3 modell volt a legjobb, a második a PB-R-TS2 és PB-R-Wilson modell volt, míg a legrosszabb a PB-R-WST modell volt;

iii. az alsó korlátok alapján a PB-R-WST modell volt a legrosszabb, míg a másik három modell nagyjából egyformán teljesített.

(d) Ipari jellegű tesztkészletet generáltam a 4 modell összehasonlításához. Az eredmények alapján megállapítottam, hogy a PB-R-TS3 modellel jóval hatékonyabban oldhatók meg ipari-jellegű PB-R-PFSP-k, mint a PB-R-TS2, PB-R-Wilson, PB-R-WST modellekkel.

DOI: 10.15477/SZE.MMTDI.2017.009

Összefoglaló

Az ipari vállalatok egy fontos feladata a gyártósorok optimális ütemezése. A feladat klasszikus matematikai modellje a permutációs flow shop feladat. A téma feldolgozása során áttekintettem a PFSP különféle heurisztikus, illetve egzakt megoldó módszereit. A célom olyan eljárás megadása volt, mellyel meghatározhatjuk ipari feladatok egzakt optimumát. Az általam választott módszer a probléma vegyes egészértékű lineáris programozási feladatként (MILP) való megfogalmazásán alapul. A kapott MILP modelleket egy erre alkalmas szoftver segítségével megoldva előállíthatjuk az optimális ütemezést.

A klasszikus PFSP nem tartalmazza az ipari termelés minden sajátosságát. Az ütemezési probléma modellezése során három jellegzetes ipari sajátosságot építettem be a modelljeimbe. Az egyik ilyen tulajdonság, hogy az ipari feladatoknál sok esetben az ütemezendő munkák nem mind különbözőek. Ezt figyelembe véve bevezettem az ismétlődéses permutációs flow shop (R-PFSP) fogalmát. Egy másik, a valós életből származó feltétel, hogy az egymást követő azonos típusú munkákból álló blokkok méretére, például logisztikai megfontolások miatt, különféle előírások lehetnek. Ezen feltételt figyelembe véve bevezettem a lot méretet tartalmazó R-PFSP (RL-PFSP) fogalmát. Végezetül modelleztem azt is, hogy a gyártósoron a munkák szállítása történhet palettákon. Így jutottam el a palettákat és véges puffert tartalmazó R-PFSP-hez.

Mindegyik új feladatosztálynak többféle MILP modelljét is megadtam. Az új MILP modelleket az általam generált tesztkészleteken többféle szempont alapján is összehasonlítottam.

Mivel egy legalább 3 gépet tartalmazó PFSP NP-nehéz feladat, ezért egzakt algoritmusokkal nagy méretű feladatok nem oldhatók meg gyorsan. Ennek ellenére, a számítógépek, illetve a MILP-eket megoldó szoftverek fejlődésével az várható, hogy az általam választott módszerrel, azaz MILP modellek segítségével egyre nagyobb méretű feladatokat oldhatunk meg. A disszertációm numerikus futtatásai éppen azt bizonyították, hogy a modellezés során figyelembe véve az ipari termelés sajátosságait, a felírt MILP modellek akár ipari feladatok megoldására is alkalmasak.

DOI: 10.15477/SZE.MMTDI.2017.009

Summary

Optimization of production lines is an important task of industrial companies. The classical mathematical model of the problem is the permutation flow shop problem (PFSP). During the work-up of this field I rewieved the heuristic and exact algorithms to solve PFSPs. My goal was to give such a method that can give optimal solution of industrial problems. The method which I chose is based on the modeling of the problem as a mixed integer linear programming problem (MILP). Solving the MILP models with an appropriate software we can get the optimal solution of the problem. The classical PFSP does not take into consideration special properties of the industrial production. During the modeling of the scheduling problem I built 3 such properties arising from industry in my models.

One such property is that in many cases in real-world problems the jobs that have to be scheduled are not all different. Taking into consideration this property I introduced the permutation with repetition flow shop problem (R-PFSP). Another property, that due to logistics reasons we may have restrictions on the length of blocks containing jobs with the same type. Taking into consideration this property I introduced the permutation with repetition flow shop with lot size problem (RL-PFSP). Finally I also modeled the situation when the jobs are carried on palettes along the line. Hence I arrived at permutation with repetition with palettes and finite buffer size flow shop problem (PB-R-PFSP) this way.

I gave several MILP models for all the new problem classes. I also generated new problem instances for the new problem classes and compared the new MILP models based on different aspects on them.

Since the PFSP with at least 3 machines is NP-hard we cannot solve large problems with exact algorithms quickly. However, with the development of computers and softwares which can solve MILPs one can expect that applying the chosen method, namely solving the problem with the use of MILP models, larger and larger problems become solvable. The numeric tests of my dissertation showed that the new MILP models containing speacial features of industry may be suitable to solve real-world problems.

DOI: 10.15477/SZE.MMTDI.2017.009

Irodalomjegyzék

[1] Adenso-Diaz, B. A. *Restricted neighborhood in the tabu search for the flowshop problem.* European Journal of Operational Research, 62(1) (1992), 27-37

[2] Ahmadizar, F. F., Barzinpour, F., Arkat, J. *Solving permutation flow shop sequencing using ant colony optimization.* IEEE International Conference on Industrial Engineering and Engineering Management (2007) 753-757

[3] Ashour., S. *A branch-and-bound algorithm for flow shop scheduling problems.* AIIE Transactions 2(2) (1970) 172-176

[4] Baker, K. R. *Introduction to Sequencing and Scheduling.* Wiley New York (1974)

[5] Bangsow, S. *Tecnomatix Plant Simulation.* Springer, 2015

[6] Beck, J. C., Feng, T. K., Watson, J. P. *Combining constraint programming and local search for job-shop scheduling.* Informs Journal on Computing 23(1) (2011) 1-14

[7] Bestwick, P. F., Hastings, N. A. J. *A new bound for machine scheduling.* Operational Research Quaterly 27(2) (1976) 479-487

[8] Bessière, C. *Arc-sonsistency and arc-consistency again.* Artificial Intelligence 65 (1994) 179-190

[9] Bessière, C., Freuder, E., Régin, J.-C. *Using constraint metaknowledge to reduce arc consistency computation.* Artific Artificial Intelligence 107 (1999) 125-118

[10] Bessière, C., Mesegeur, P., Freuder, E. C., Larossa, J. *On forward checking for non-binary constraint satisfaction.* Principles and Practice of Constraint Programming, CP'99, Alexandria, VA, (1999) Springer, LNCS 1713, 24-27

[11] Bessière, C., Régin, J.-C. *Refining the basic constraint propagation algorithm.* 17th Intern. Joint Conf. Artificial Intelligence, IJCAI'01, Seattle 1 (2001) 309-315

[12] Brown, A. P. G., Lomnicki, Z. *Some application of the "branch-and-bound" algorithm to the machine sequencing problem.* Journal of the Operational Research Society 17(2) (1966) 173-186

[13] Brucker, P. *Scheduling Algoritms.* Springer, Berlin (2007)

[14] Campbell, H. G., Dudek, R. A., Smith, M. L. *A heuristic algorithm for the n job, m machine sequencing problem.* Management science 16(10) (1970) 630-637

[15] Carlier, J., Rebai, I. *Two branch and bound algorithms for the permutation flow shop problem.* European Journal of Operational Research 90(2) (1996) 238-251

[16] Carlier, P., Pinson, E. *A practical use of Jackson's pre-emptive schedule for solving the job-shop problem.* Annals of Operations Research 26 (1990) 269-287

[17] Chakravarthy, K., Rajendran, C. *A heuristic for scheduling in a flow shop with the bicriteria of makespan and maximum tardiness minimization.* Production Planning and Control 10(7) (1999) 707-714

[18] Chakraborty, U.K., Laha, D. *An improved heuristic for permutation flowshop scheduling.* Int. J. Information and Communication Technology, 1(1) (2007) 89-97.

[19] Chang, P. C., Huang, W. H., Wu, J. L., Cheng, T. C. E. *A block mining and re-combination enhanced genetic algorithm for the permutation flowshop scheduling problem.* International Journal of Production Economics 141(1) (2013) 45-55

[20] Chen, R. M., Lo, S. T., Wu., Lin, T. H. *An effective ant colony optimization-based algorithm for flow shop scheduling.* IEEE Conference on Soft Computing in Industrial Applications, Japan (2008) 101-106

[21] Chen, Y. M., Chen., M. C., Chang, P. C., Chen, S. H. *Extended artificial chromosomes genetic algorithm for permutation flow-shop scheduling problems.* Computers & Industrial Engineering 62(2) (2012) 536-545

[22] Cheng, J., Kise, H., Matsumoto, H. *A branch-and-bound algorithm with fuzzy inference for a permutation flowshop scheduling problem.* European Journal of Operational Research 96(3) (1997) 578-590

[23] Companys, R., *Note on the blocking flowshop problem.* Working paper (2009) http://upcommons.upc.edu/e-prints/handle/2117/420

[24] CPLEX, IBM ILOG *V12. 1: User's Manual for CPLEX.* International Business Machines Corporation (2009)

[25] Dannenbring, D. G. *An evaluation of flow shop sequencing heuristics.* Management Science 23(11) (1977) 1174-1182

[26] Dorigo, M., Gambardella, L. M. *Ant colony systems: a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on Evolutionary Computation 1(1) (1997) 53-66

[27] Eksioglu, B., Eksioglu, S. D., Jain, P. *A tabu search algorithm for the flowshop scheduling problem with changing neighborhoods.* Computer & Industrial Engineering 54(1) (2008) 1-11

[28] Filho, G. R., Nagano, M. S., Lorena, L. A. N., *Evolutionary clustering search for flowtime minimization in permutation flow shop.* Lecture Notes in Computer Science 4771 (2007) 69-81

[29] Framinan, J. M., Gupta, J. N. D., Leisten, R. *A rewiev and classification of heuristics for permutation flow-shop scheduling with makespan objective.* Journal of the Operational Research Society 55(12) (2004) 1243-1255

[30] Frost, D., Dechter, R. *Look-ahead value ordering for constraint satisfaction problems.* Proceedings IJCAI95 (1995) 572-578

[31] Garey, M. R., Johnson, D. S., Sethi, R. *The complexity of flowshop and jobshop scheduling.* Mathematics of Operations Research 1(2) (1976) 117-129

[32] Garrido, A., Salido, M. A., Barber, F., López, M. *Heuristic methods for solving job-shop scheduling problems.* In Proc. ECAI-2000 Workshop on New Results in Planning and Design (PUK2000) (2000) 44-49

[33] Gent, I. P., MacIntyre, E., Prosse, P., Smith, B. M., Walsh, T. *An empirical study of dynamic variable ordering heuristics for the constraint satisfaction problem.* Proceedings of CP'96 (1996) 179-193

[34] Glover, T. *Tabu Search Part I.* ORSA Journal of Computing, 1(3) (1989) 190-206

[35] Glover, T. *Tabu Search Part II.* ORSA Journal of Computing, 2(1) (1990) 4-32

[36] Grabowski, J., Wodecki, M. *A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion.* Computers & Operations Research 31(11) (2004) 1891-1909

[37] Gupta, J. N. D. *Heuristic algorithms for multistage flow shop problem.* AIIE Transactions 4 (1972) 11-18

[38] Hajba, T., Horváth, Z. *New effective MILP models for PFSPs arising from Real Applications.* Central European Journal of Operations Research 21(4) (2013) 729-744

[39] Hajba, T., Horváth, Z. *MILP models for the optimization of real production lines.* Central European Journal of Operations Research 23(4) (2015) 899-912

[40] Hajba T., Horváth, Z., Kiss-Tóth, C., Jósvai, J. *Production Line Optimization with Model Based Methods.* Proceedings of the KoMSO Challenge Workshop: Math for the Digital Factory (2014) Ed. L. Ghezzi, D. Hömberg, Ch. Landry. Springer, 2016. (to appear)

[41] Haouari, M., Ladhari, T. *Minimising maximum lateness in a two-machine flowshop.* Journal of the Operational Research Society 51(9) (2000) 1100-1106

[42] Horn, W. A. *Some simple scheduling algorithms.* Naval Research Logistics Quaterly 21(1) (1974) 177-185

[43] Ignall, E., Schrage, L. E. *Application of the branch-and-bound technique to some flow shop problems.* Operations Research 13(3) (1965) 400-412

[44] Jackson, J. R. *Scheduling a production line to minimize maximum tardiness.* Research Report 43, Management Science Research Project, University of Carolina, Los Angeles, (1955)

[45] Johnson, S. M. *Optimal two- and three stage production schedules with setup times included.* Naval Research Logistics Quaterly 1(1) (1954) 61-68

[46] Kalczynski, P. J., Kamburowski, J. *An improved NEH heuristic to minimize makespan in permutation flow shops.* Computers & Operations Research 35(9) (2008) 3001-3008

[47] Kennedy, J., Eberhart, R. *Particle Swarm Optimization.* Proceedings of IEEE International Conference on Neural Networks (1995) 1942-1948

[48] Kouki, S., Jemni, M., Ladhari, T. *Deployment of Solving Permutation Flow Shop Scheduling Problem on the Grid.* Grid and Distributed Computing, Control and Automation 121 (2010) 95-104

[49] Kouki, S., Jemni, M., Ladhari, T. *Solving the Permutation Flow Shop Problem with Makespan Criterion using Grids.* Grid and Distributed Computing, Control and Automation 4(2) June, (2011) 53-64

[50] Kovács, A., Vánccza, J., Márkus, A. *Structural exploration of constraint-based scheduling problems.* Proc. of the 37th CIRP International Seminar on Manufacturing Systems (2004) 433-439

[51] Ladhari, T., Haouri, M., *A computational study of the permutation flow shop problem based on a tight lower bound.* Computers & Operations Research 32(7) (2005) 1831-1847

[52] Lageweg, B. J., Lenstra, J. K., Rinnooy Kan, A. H. G. *A general bounding scheme for the permutation flow-shop problem.* Operations Research 26(1) (1978) 53-67

[53] Laha, D., Chakraborty, U. K. *An efficient hybrid heuristic for makespan minimization in permutation flow shop scheduling.* International Journal of Advanced Manufacturing Technology 44(5) (2008) 559-569

[54] Larossa, J., Meseguer, P. *Generic CSP techiques for the job shop problem.* In International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems Springer Berlin Heidelberg (1998) 46-55

[55] Lenstra, J. K., Rinooy Kan, A. H. G., Brucker, P. *Complexity of machine scheduling problems.* Annals of Discrete Mathematics 1 (1977), 343-362

[56] Li, B. B., Wang, L., Liu, B. *An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling.* IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans 38(4) (2008) 818-831

[57] Lian, Z., Gu, X., Jiao, B. *A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan.* Chaos, Solitons & Fractals 35(5) (2008) 851-861

[58] Liao, C. L., You, C. T *An improved formulation for the job-shop scheduling problem.* Journal of Operational Research Society 43(11) (1992) 1047-1054

[59] Lin, S. W., Ying, K. C., Lu, C. C., Gupta, J. N. D. *Applying multi-start simulated annealing to schedule a flow line manufacturing cell with sequence dependent family setup times.* International Journal of Production Economics 130(2) (2011) 246-254

[60] Lenstra, J. K., Rinooy Kan, A. H. G., Brucker, P. *Complexity of machine scheduling problems.* Annals of Discrete Mathematics 1 (1977), 343-362

[61] Lomnicki, Z. *A "branch-and-bound" algorithm algorithm for the exact solution of the three-machine scheduling problem.* Journal of the Operational Research Society 16(1) (1965) 89-100

[62] Low, C., Yeh, J. Y., Huang, K. I. *A robust simulated annealing heuristic for flow shop scheduling problems.* International Journal of Advanced Manufacturing Technology 23(9) (2004) 762-767

[63] Mackworth, A. *Consistency in networks of relations.* Artificial Intelligence 8 (1977) 99-118

[64] Malapert, A., Cambazard, H., Guéret, C., Jussien, N., Langevin, A., Rousseau, L.M. *An optimal constraint programming approach to the open-shop problem.* INFORMS Journal on Computing 24(2) (2012) 228-244

[65] Manne, A. S. *On the job-shop scheduling problem.* Operational Research 8(2) (1960) 219-223

[66] McMahon, G. B., Burton, P. G. *Flow-shop scheduling with the branch-and-bound method.* Operations Research 15(3) (1967) 473-481

[67] McMahon, G. B. *A study of algorithms for industrial scheduling problems.* PhD thesis, University of New South Wales, Kensington (1971)

[68] Mehta, D., van Dongen, M. R. C. *Static Value Ordering Heuristicsfor Constraint Satisfaction Problems.* Proceedings of the Second International Workshop on Constraint Propagation And Implementation (2005) 65-79

[69] Moccellin, J. V., *A new heuristic method for the permutation flow shop scheduling problem.* The Journal of the Operational Research Society, 46(7) (1995) 883-886

[70] Moccellin, J. V., dos Dantos, M. O. *An adaptive hybrid methaheuristic for permutation flowshop scheduling.* Control and Cybernetics 29(3) (2000) 761-771

[71] Mohr, R., Henderson, T. C. *Arc and path consitency revisited.* Artifical Intelligence 28 (1986) 225-233

[72] Morten, T. E., Pentico, D. W. *Heuristic Scheduling Systems.* Wiley, New York (1993)

[73] Nabeshima, I. *On the bound of makespans and its application in m machine scheduling problem.* Journal of the Operations Research Society of Japan 9(3-4) (1967) 98-136

[74] Naderi, B., Zandieh, M., Balagh, A. K. G., Roshanaei, V. *An improved simulated annealing for hybrid flow shops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness.* Expert Systems with Application, 36(6) (2009) 9625-9633

[75] Nagano, M. S., Ruiz, R., Lorena, L. A. N., *A constructive genetic algorithm for permutation flowshop scheduling.* Computers and Industrial Engineering 55(1) (2008) 195-207

[76] Nawaz, M., Enscore, E. E., Ham, I. *A heuristic algoritm for the m-machine, n-job flowshop sequencing problem.* 11(1) OMEGA (1983)

[77] Nearchou, A. C. *A novel metaheuristic approach for the flow shop scheduling problem.* Engineering Applications of Artifical Intelligence 17(3) (2004) 289-300

[78] Nowicki, E., Smutnicki, C.: *A fast tabu search algorithm for the permutation flow-shop problem.* European Journal of Operational Research, 91(1) (1996), 160-175

[79] Nowicki, E. *The permutation flow shop with buffers: A tabu search approach.* European Journal of Operational Research 116(1) (1999), 205-219.

[80] Pan, C. H. *A study of integer programming formulations for scheduling problems.* International Journal of System Science 28(1) (1997) 33-41

[81] Potts, C. N. *The job-machine scheduling problem.* PhD thesis, University of Birmingham (1974)

[82] Rajendran, C., Ziegler, H. *Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs.* European J. of Operational Research, 155(2) (2004) 426-438

[83] Ravetti, M. G., Nakamura, F. G., Meneses, C. N., Resende, M. G. C., Mateus, G. R., & Pardalos, P. M. *Hybrid heuristics for the permutation flow shop problem.* AT&T Labs Research Technical Report

[84] Rinooy Kan, A. H. G. *Machine scheduling problems: classification, complexity and computations.* Nijhoff: The Hague; (1976)

[85] Ronconi, D. P., Birgin, E. G. *Mixed-integer Programming Models for Flowshop Scheduling Problems Minimizing the Total Earliness and Tardiness.* Just-in-Time Systems. Optimization and Applications. 60 (2011) 91-105

[86] Rosenthal, E. *GAMS- A user's guide.* GAMS Development Corporation (2008)

[87] Röck, H., Schmidt, G. *Machine Aggregation Heuristics in Shop Scheduling.* Methods of Operations Research 45 (1983) 303-314

[88] Ruiz, R., Maroto, C. *A comprehensive review and evaluation of permutation flowshop heuristics.* European Journal of Operational Research 43 (2005) 479-494

[89] Ruiz, R., Maroto, C., Alcaraz, J. *Two new robust genetic algorithms for the flowshop scheduling problem.* Omega 34(5) (2006) 461-476

[90] Ruiz, R., Stützle, T., *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem.* European Journal of Operational Research 177(3) (2007) 2033-2049

[91] Sabin, D., Freuder, E. C. *Understanding and improving MAC algorithm.* Principles and Practice of Constraint Programming, CP'97, Linz, VA, (1997) Springer, LNCS 1330, 167-181

[92] Sadeh, N., Sycara, K., Yiong, Y. *Backtracking techniques for the job shop scheduling constraint satisfaction problem.* Artificial Intelligence 76 (1995) 455-480

[93] Sadeh, N., Fox, M. S. *Variable and value ordering heuristics for the job shop constraint satisfaction problem.* Artificial Intelligence 86 (1996) 1-41

[94] Seda, M. *Mathematical models of flow shop and job shop scheduling problems.* Proc. of the World Academy of Science, Engineering and Technology 31 (2007) 122-127

[95] Stafford, E. F. *A mixed-integer linear programming model for the classical flowshop sequencing problem.* Research report, University of South Carolina (1983)

[96] Stafford, E. F. *On the development of a mixed-integer linear programming model for the flowshop sequencing problem.* Journal of Operational Research Society 39(12) (1988) 1163-1174

[97] Stafford, E. F., Tseng, F. T. *On the Srikar-Ghosh MILP model for the $N \times M$ SDST flowshop problem.* International Journal of Production Research 28(10) (1990) 1817-1830

[98] Stafford, E. F., Tseng, F. T. *Two models for a family of flowshop sequencing problems.* European Journal of Operational Research 142(2) (2002) 282-293

[99] Stafford, E. F., Tseng, F. T. *New MILP models for the permutation flowshop problem.* Journal of the Operational Research Society. 59(10) (2008) 1373-1386

[100] Stafford E. F., Tseng F. T., Gupta N. D. *An empirical analysis of integer programming formulations for the permutation flowshop.* Omega 32(4) (2004) 285–293

[101] Stafford E. F., Tseng F. T., Gupta N. D. *Comparative evaluation of the MILP flowshop models.* Journal of the Operational Research Society 56(1) (2005) 88-101

[102] Szwarc, W. *Mathematical aspects of the 3×n job-shop sequencing problem.* Naval Research Logistics Quaterly 21(1) (1974) 145-153

[103] Taillard, E.: *Some efficient heuristic methods for flow-shop sequencing.* European Journal of Operational Research, 47(1) (1990), 65-74

[104] Taillard, E. *Benchmarks for basic scheduling problems.* European Journal of Operational Research 64(2) (1993) 278–285

[105] Tasgetiren, F. M., Liang, Y. C., Sevkli, M., Gencyilmaz, G. *A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem.* European Journal of Operational Research 177(3) (2007) 1930-1947

[106] Torres, P., Lopez, P. *On not-first/not-last conditions in disjunctive scheduling.* European Journal of Operational Research 127 (2000) 332-343

[107] Tseng, F. T., Stafford, E. F. *Two MILP models for the $N \times M$ SDST flowshop sequencing problem.* International Journal of Production Research 39(8) (2001) 1777-1809

[108] van Hoeve, W. J. *The alldifferent constriant: A survey.* Sixth Annual Workshop of the ERCIM Working Group on Constraints. (2001)

[109] Vernooy, M., Havens, W. S. *An examination of probabilistic value-ordering heuristics.* Proceedings of IJCAI-99 (1999) 340-352

[110] Wagner, H. M. *An integer linear-programming model for machine scheduling.* Naval Research Logistics Quaterly 6(2) (1959) 131-140

[111] Watson, J. P., Barbulescu, L., Whitley, L. D., Howe, A. E. *Contrasting structured and random permutation flow-shop scheduling problems: search-space topology and algorithm performance.* INFORMS Journal on Computing, 14(2) (2002) 98-123

[112] Werner, F. *On the heuristic solution of the permutation flow shop problem by path algorithms.* Computers & Operations Research 20(7) (1993) 707-722

[113] Widmer, M., Hertz, A.: *A new heuristic method for the flow shop sequencing problem.* European Journal of Operational Research, 41(2) (1989), 186-193

[114] Wilson, J. M. *Alternative formulations of a flow-shop scheduling problem.* Journal of Operational Research Society 40(4) (1989) 395-399

[115] Yagmahan, B., Yenisey, M. *Ant colony optimization for multi-objective flow shop scheduling problem.* Computers & Industrial Engineering 54(3) (2008) 411-420

[116] Yamada, T., Reeves, C., *Permutation flowshop scheduling by genetic local search.* 2^{nd} IEE/IEEE Int. Conf. on Genetic Algorithms in Engineering Systems (Galesia '97), 232-238, Glasgow, UK.

[117] Ying, K. C., Liao, C. J., *An ant-colony system for permutation flowshop sequencing.* Computers & Operations Research 31(5) (2004) 791-801

[118] Zhu, Z., Heady, R. B. *Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach.* Computers & Industrial Engineering 38(2) (2000) 297-305

[119] Zobolas, G. I., Tarantilis, C. D., Ioannou, G. *Minimizing makespan in permutation flow shop scheduling problems using a hybrid methaheuristic algorithm.* Computers & Operations Research 36(4) (2009) 1249-1267