Ütemezések speciális rugalmas gyártórendszereken

Diplomamunka

Írta: Korbács Kitti

Alkalmazott matematikus szak

Témavezető:

Kovács Gergely, főiskolai docens
Operációkutatási Tanszék
Eötvös Loránd Tudományegyetem, Természettudományi Kar

Eötvös Loránd Tudományegyetem
Természettudományi Kar
2009

Tartalomjegyzék

1. Bevezetés

Ahogy elavulttá vált a kézzel gyártott egyedi termék készítése, idővel ugyanúgy korszerűsítésre szorult a modern manufakturális termelés is monoton ismétlődő folyamatai miatt. Az éleződő piaci verseny új igényeket támasztott a termelők számára. A minőség és mennyiség mellett fontossá vált egy bizonyos kereten belül az egyéni igények kielégítése is.

Ebben segít a számítógép által vezérelt rendszer, a rugalmas gyártó rendszer (FMS, Flexible Manufacturing Systems), amely minden szempontból rugalmas. A sorozatgyártás, valamint az egyedi termelés előnyeit egyszerre hordozza magában. Változó méretű, alakú, tömegű termékek; változó műveleti idők; kis- és középsorozatnagyságok jellemzik a gyártósort, amelyek a rendszer rugalmasságát mutatják. Ez a rugalmasság automatizálással érhető el, melyet számítógépek segítségével irányítanak. Az ember szerepe már csak kis területen figyelhető meg, hiszen a tervezésre, illetve a gépek felügyeletére korlátozódik. Sok helyen éjszaka se állnak le a gépek, hanem olyan egyszerűbb programon dolgoznak, amelyekben kicsi a legyártandó munkadarab választéka. Mindezek ellenére az FMS nem valósítja meg az ember nélküli gyárat.

Összefoglalva tehát, a rugalmas gyártó rendszer egy olyan számjegyvezérlésű gépekből álló rendszer, ahol a gyártási folyamatokat, az anyagmozgatást (beleértve a raktározást, a munkadarabok gépre való felrakását, illetve onnan való levételét is) és a szerszámcserét számítógép vezérli. Az FMS kis- és középsorozatok gyártását szolgálja. Ez azt jelenti, hogy az alkatrészek száma, melyet különböző rendszerek állítanak elő, 6 és 100 között mozog.

Mindezeket természetesen minimalizált termelési idő mellett. A rugalmas gyártó rendszerek bonyolult gépsorai ütemezés alkalmazását kívánják meg.

Ez az ütemezés történhet egyrészt a gyártás során egy előre meghatározott egyszerű, ennek köszönhetően gyors eljárás segítségével. Az ilyen heurisztikus szabályokat nevezzük on-line módszernek. Másrészt a termelést megelőzően több tényezőt figyelembe véve felépített matematikai modellel. Az ilyen úgynevezett off-line módszeren alapuló, akár fizikai szimulációkat is tartalmazó optimalizált modellek alkalmazásával tovább növelhető a hatásfok. Az optimalizálás egyik módja már adott gyártórendszer esetén a munkadarabok adagolásának a gépközpontok szerinti ütemezése.

Az anyagmozgató rendszerekben az ember, a gép, a szállítószalag, a számítógép együttműködnek. A kézi anyagmozgató rendszereknél az embernek fizikai munkát kell végeznie, míg a gépi anyagmozgató rendszereknél az ember feladata a gépek működtetése, vezérlése. Ha a rendszer teljesen automatizált, akkor már az ember közreműködésére sincs szükség. A szállítógépek működésük szerint lehetnek folyamatosan, illetve a szakaszosan működő anyagmozgató gépek.

A szakaszosan működő anyagszállító gépek alkalmazása során az anyagszállítás megszakításokkal történik. A szállítóeszköz a munkaciklusok során egyes központoktól (anyagfeladási központ) meghatározott mennyiségű munkadarabot szállít a termelési folyamat következő pontjára (anyagleadási központ). A fel-, illetve lerakodás közben a gép áll. Amikor a gép egyik helyről a másikra szállította az anyagokat, üresen tér vissza a feladási helyre. Ezek az anyagmozgató gépek nem helyhez kötöttek. Néhány példa ilyen szállító eszközökre: daruk, szállító-,vontató-, emelő-, felrakó-targoncák, kanalas rakodógépek.

A folyamatosan működő anyagszállító rendszerek mindig azonos irányba szállítják a munkadarabot, illetve akkor is működnek, ha az anyagok felrakása közben szünet van. A munkadarabok mozgatása közben ezek a szállító rendszerek egyhelyben maradnak, csak egyes részeik mozognak együtt a termékekkel. Ezek a rendszerek helyhez, illetve munkadarabhoz kötöttek, mivel speciális kialakításúak, vagyis ezek a rendszerek nem általános felhasználásra készültek. Példák ilyen rendszerekre: hevederes szállítószalagok, csuklótagos szállítószalagok, egy- és kétpályás függőkonvejorok, elevátorok (serleges, karos, tálcás stb.)

Ebben a dolgozatban egy körpályás futószalag/szállítószalag által kiszolgált rendszert vizsgálok, melynek száz férőhelye van. A futószalagon raklap szállítja a munkadarabokat, melyek addig köröznek a körpályán, míg minden feladat el nem készül az egyes darabokon. A szállítószalag szakaszos mozgást végez, tehát a szalag felváltva mozog, illetve áll. Ez nem azt jelenti, hogy míg a futószalag mozdulatlan, nem történik semmi, hanem épp ebben az időben zajlik a munkadarabok fel, illetve levétele a szállítószalagról, valamint a munkadarabok gépközpontba való belépése szintén ekkor történik. Egy speciális gyártórendszert fogok bemutatni, melynek 3 gépközpontja van. Ezekben a gépközpontokban különböző feladatokat végeznek a gépek, melyeknek feldolgozási ideje különböző is lehet, de én egy egyszerűsített esetet ismertetek, melyben minden gépközpont munkaideje megegyezik. A körpályára addig kerülnek fel a munkadarabok, míg a teljes szalag meg nem telik, és az új

munkadarab akkor kerül a rendszerbe, ha az egyik termék készen távozott a szalagról, így egy hely felszabadult.

A továbbiakban az előbb bemutatott rugalmas gyártórendszer ütemezésével foglalkozom, a befejezési idő minimalizálásának módjait elemezve.

A második fejezetben a gyártórendszerrel kapcsolatos fogalmakkal ismerkedünk meg.

A harmadik fejezet a tanulmányozott rendszer leírását tartalmazza, majd a negyedik fejezetben az adott rendszeren alkalmazott ütemezéseket mutatja be, és az általuk adott eredményeket hasonlítja össze.

Az ötödik fejezetben bemutatunk egy új, módosított rendszert, amelyhez program készült, és a program által adott eredményeket vizsgáljuk.

Majd legvégül az új rendszerben különböző eseteket vizsgálva hasonlítjuk össze az eredeti-, illetve új rendszer által adott befejezési időket, és állítások formájában összefoglaljuk a kapott eredményeket.

2. Fogalmak és jelölések

A gyártórendszer tanulmányozása során a problémát úgy definiálom, mint egy ütemezést, amelyben n munkadarabot kell m gépen feldolgozni úgy, hogy a teljes ráfordított időt, vagyis a befejezési időt minimalizáljuk.

- *Befejezési idő (FT)*: az az idő, amely az első munkadarab futószalagra való belépésétől az utolsó munkadarab futószalagról való távozásáig eltelik.
- *Munkadarab*: a munkadarabok feldolgozása feladatokból áll, és minden egyes feladatnak saját feladatideje van.
- *Álmunka*: egy raklap méretű, vagyis l hosszúságú szünetet jelent a futószalagon.
- *Betöltési idő (lt)*: ennyi idő kell egy munkadarabnak ahhoz, hogy a gépközpontba belépjen, ha ott feldolgozásra van szüksége.
- *Szerelési idő (at)*: a gépközpontban a feldolgozáshoz szükséges idő.
- *Ürítési idő (ut)*: az az idő, amennyi idő alatt a kész munkadarab a gépközpontból eltávozik.
- *Folyamatidő*: a betöltési, a szerelési és az ürítési idő együttesen.
- *MSEE (Machine center with Separate Entrance and Exit)*: gépközpont különálló be és kijárattal.
- *NPE (Non Priority Exit)*: elsőbbségmentes kijárat, a munkának várnia kell a kijáratnál, míg a futószalagon üres hely nem áll a rendelkezésére.

3. A rendszer leírása

A feladat, mellyel foglalkozom, egy $3MSEE/NPE$ esetet vizsgál, mely azt jelenti, hogy 3 gépközponttal rendelkező elsőbbségmentes kijáratot használ. Az egyik gépközponttól a másikig a futószalag raklapon szállítja a munkadarabokat. A termelés elején a futószalag (fő futószalag) üres, és tömören egymás után töltődik fel maximum 100 raklappal, melyek körbe-körbe haladnak a futószalagon.
A futószalag mozgása két részből áll:

- t_1 *periódusidő* - amikor l távolságot halad a munkadarab,
- t_2 *periódusidő* - amíg áll a szalag.

A gépközpontban a feladatok folyamatideje szintén t_1 idő alatt zajlik. Míg a szalag mozdulatlan, a raklap vagy belép a gépközpontba, vagy kilép a gépközpontból, vagy visszakerül a futószalagra. Ezeket természetesen párhuzamosan egymás mellett is tudja végezni.

Ha a raklapnak szüksége van feldolgozásra, akkor belép a gépközpontba és egy $l \times l$-es méretű hely kimarad.

A futószalag sebessége s, amit úgy határozunk meg, hogy a futószalagon l út megtételéhez szükséges idő ellentettje legyen, tehát

$$s = \frac{1}{t_1 + t_2}.$$

A későbbiekben $s = 1$ és $s = 2$ esettel fogunk foglalkozni. $s = 1$ esetén minden munkadarab feldolgozásra kerül, míg $s = 2$ esetén nem tud minden munkadarab lekerülni a gépközpontba, hiszen a futószalag sebessége gyors, és az egymást követő ugyanolyan típusú munkák nem kerülnek feldolgozásra, vagyis maradnak kezeletlen munkák egy kör befejezése után.

Három gépközpont esetén tizenöt féle munkadarabot különböztetünk meg:

$$J_a, J_b, J_c, J_{ab}, J_{ac}, J_{ba}, J_{bc}, J_{ca}, J_{cb}, J_{abc}, J_{acb}, J_{bac}, J_{bca}, J_{cab}, J_{cba}.$$

A munkadarabok száma az alábbiak miatt alakul így:

A J_a, J_b, J_c munkadarabok mindegyikén csak egy feladat elvégzésére van szükség, ami rendre az a, b, illetve a c.

A J_{ac} valamint a J_{ca} munkadarabot azért különböztetem meg egymástól, mert a J_{ac} munkadarabon az első gépközpont elvégzi az a, aztán harmadik gépközpont pedig a c feladatot, tehát egy körön belül távozhat a kész munkadarab, míg a J_{ca} munkadarab befejezéséhez két körre is szükség van, hiszen a gépközpontok sorrendje adott, így az első körben a c munkafolyamat végezhető csak el, és kell egy következő kör, hogy a munkadarab készen hagyhassa el a futószalagot.

Hasonlóan magyarázható a megkülönböztetés a $J_{abc}, J_{acb}, J_{bac}, J_{bca}, J_{cab}, J_{cba}$ esetekben. A J_{abc} egy kör alatt elkészíthető munkadarab, aminek munkafolyamatai rendre az a, b, c. A $J_{acb}, J_{bac}, J_{bca}, J_{cab}$ munkadarabok pedig két kört igényelnek, természetesen más-más munkafázist különböző sorrendben kell elvégezni rajtuk, ami a megkülönböztetést indokolja. J_{cba} munkadarab három kör alatt készül csak el, hiszen első körben csak a c, második körben a b, illetve a harmadik körben az a feladat végezhető el.

Minden feladatnak saját feldolgozási ideje van, azonban az egyszerűség kedvéért ez mindegyiknél egységnek tekinthető. Ez az idő három részből áll:

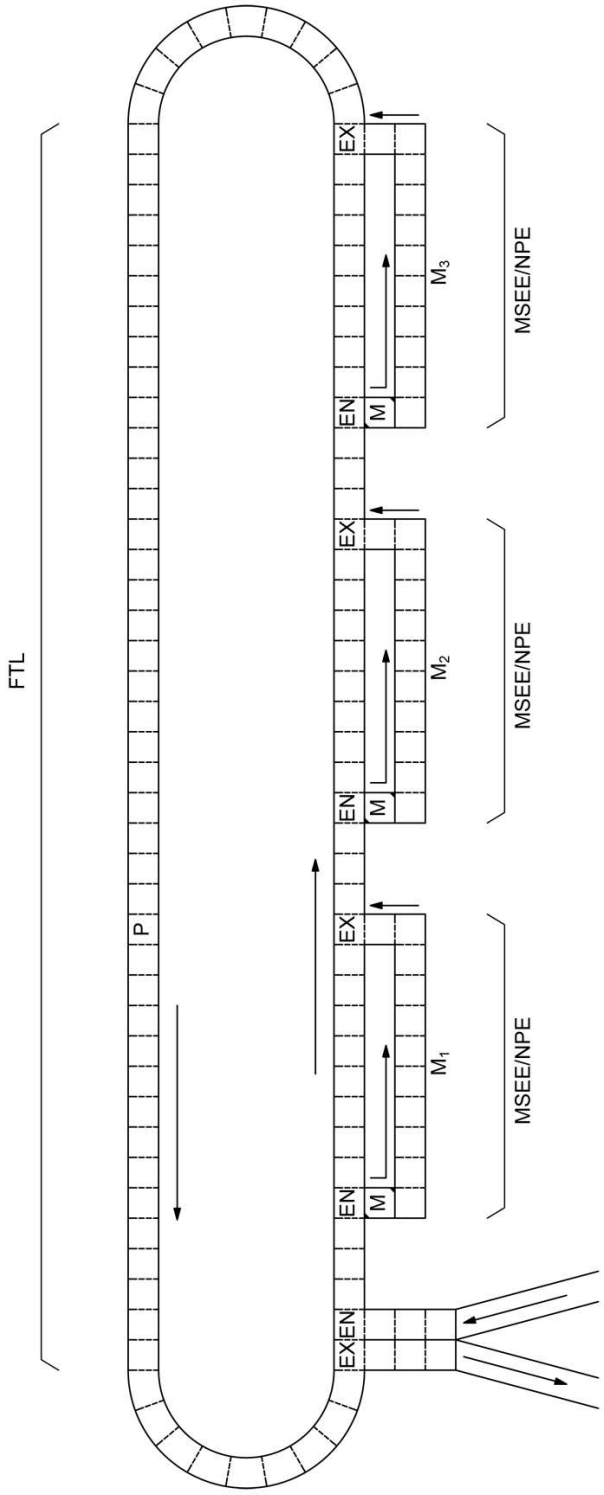
- *betöltési idő (lt)*,
- *szerelési idő (at)*,
- *ürítési idő (ut)*.

Ezen részidők összegét tekintjük egynek, azaz

$$lt + at + ut = 1.$$

Ha egy munkadarab elkészült, akkor vissza kell térnie a fő futószalagra, de ezt csak akkor teheti, ha van szabad hely számára, vagyis van egy $l \times l$-es méretű hely ahová kikerülhet.

A következő szakaszban azt az esetet fogom megvizsgálni, amikor egy munkadarab bekerül egy gépközpontba, és a feldolgozás után saját helyére vissza tud kerülni. Ez azt jelenti, hogy a mellék futószalag sebessége gyorsabb, mint a fő futószalagé.



1. ábra. A rendszer modellje

4. Ütemezési problémák

Ütemezni szeretnénk a $3MSEE/NPE$ gyártó rendszeren a fenti tizenöt féle típusú munkadarabból n darabot. Célunk olyan ütemezési módszer keresése, ami minimalizálja a befejezési időt azzal, hogy a szállítószalagon kialakuló üres helyek számát csökkenti a munkadarabok ismeretében egy előre meghatározott algoritmus segítségével. Kétféle algoritmust fogok bemutatni, melyek lényege az, hogy azok a munkadarabok, melyeknek több körre van szükségük az elkészülésig, ne maradjanak az utolsó körökre, kizárva annak lehetőségét, hogy a legvégén néhány munkadarab miatt működjön a szalag szinte teljesen üresen. Az ütemezések fontosságát a befejezési idő csökkenésével jellemezhetjük, tehát ha random, illetve a két általam bemutatott algoritmusok szerint ütemezem egy előre megadott rendszerben a munkadarabokat, akkor befejezési idő csökkenést érhetek el, vagyis a rendszer termelő képessége javul.

Először egy olyan algoritmust fogok bemutatni, amely csak a munkadarabok feldolgozásához szükséges körök száma szerint rendezi sorba a munkadarabokat. Ennek neve $GCYMINF$ algoritmus.

Ezután a munka illesztő algoritmussal foglalkozom, amely hasonlóan a $GCYMINF$ algoritmushoz, felhasználja a munkadarabok elvégzéséhez szükséges körök száma szerinti rendezést, azzal a többlettel, hogy a munkadarabokat egymáshoz illeszti, vagyis az összeillő munkadarabokat felváltva küldi a rendszerbe. Ez az algoritmus $s = 2$ esetén még nagyobb befejezési idő javulást eredményez, mint a $GCYMINF$ algoritmus.

4.1. $GCYMINF$ algoritmus

A munkadarabokat csoportokba soroljuk aszerint, hogy hány körre van szükségük a futószalagon ahhoz, hogy minden munkafolyamatot elvégezve készen távozzanak a kijáraton át.

A három gépközpontból álló gyártórendszer $M1$, $M2$, $M3$ központokból áll, melyek egymás után sorban helyezkednek el. Az $M1$-es gépközpontban az a munkafolyamat, az $M2$-esben a b munkafolyamat, az $M3$-asban pedig a c munkafolyamat kerül feldolgozásra. A továbbiakban minden munkadarabra meghatározom a feldolgozásához szükséges körök számát.

Például tekintsük a J_{cba} esetet, ahol első körben a c, aztán a második körben a b és a harmadik körben az a munkafolyamat is elvégzésre kerül, így a J_{cba} munkadarab elkészítéséhez három körre van szükségünk, $CY_{min}(J_{cba}) = 3$.

A munkadarabok csoportosítva az elvégzésükhöz szükséges körök száma szerint:

- *Három kör alatt elvégezhető munkadarabok:*
  $J_{cba} = J_c + J_b + J_a$, így $CY_{min}(J_{cba}) = 3$
- *Két kör alatt elvégezhető munkadarabok:*
  $J_{ba} = J_b + J_a$, így $CY_{min}(J_{ba}) = 2$
  $J_{ca} = J_c + J_a$, így $CY_{min}(J_{ca}) = 2$
  $J_{cb} = J_c + J_b$, így $CY_{min}(J_{cb}) = 2$
  $J_{acb} = J_{ac} + J_b$, így $CY_{min}(J_{acb}) = 2$
  $J_{bac} = J_b + J_{ac}$, így $CY_{min}(J_{bac}) = 2$
  $J_{bca} = J_{bc} + J_a$, így $CY_{min}(J_{bca}) = 2$
  $J_{cab} = J_c + J_{ab}$, így $CY_{min}(J_{cab}) = 2$
- *Egy kör alatt elvégezhető munkadarabok:*
  J_a, $CY_{min}(J_a) = 1$
  J_b, $CY_{min}(J_b) = 1$
  J_c, $CY_{min}(J_c) = 1$
  J_{ab}, $CY_{min}(J_{ab}) = 1$
  J_{ac}, $CY_{min}(J_{ac}) = 1$
  J_{bc}, $CY_{min}(J_{bc}) = 1$
  J_{abc}, $CY_{min}(J_{abc}) = 1$

$GCYMINF$ **algoritmus:**
Az n darab különböző típusú munkadarabbal úgy töltjük fel a rendszert, hogy a munkadarabok a CY_{min} nem növekvő sorrendjében kerülnek futószalagra, vagyis a legnagyobb CY_{min}-nel rendelkező munkadarabbal kezdve.

A $GCYMINF$ algoritmus ütemezésének hatékonyságát az 1. táblázatban mutatom be, ahol egy random sorrend, illetve a $GCYMINF$ algoritmus eredményei láthatóak.

4.2. A $GCYMINF$ algoritmus optimalitása

4.2.1. Állítás:

A $GCYMINF$ algoritmus $s = 1$ esetén optimális.

Bizonyítás:

Legyen λ egy olyan ütemezés, amelyet a $GCYMINF$ nem tud generálni, ezért létezik legalább két munkadarab, i és j, amire $CY^j_{min}<CY^i_{min}$ és λ-ban j megelőzi i-t.

Azt szeretnénk belátni, hogy ha λ-ban a munkadarabok sorrendjét úgy változtatjuk, hogy a legnagyobb CY_{min}-nel rendelkező munkák a cserék során minél előrébb kerülnek, akkor egy λ^* $GCYMINF$ ütemezést kapunk, és a kapott FT^* érték kisebb vagy egyenlő lesz, mint a λ-hoz tartozó FT érték.

Jelölések:

n_w − a raklapok maximális száma, amelyek egyszerre a futószalagon lehetnek,
n − az ütemezett munkák száma,
m − a gépközpontok száma,
n_m − azoknak a munkáknak a száma, amelyeknek az elkészüléshez m körre van szüksége, vagyis a $CY_{min} = m$,
n_{m-1}, n_{m-2}, n_{m-3}, ... , n_1, ahol $n = n_1 + \ldots + n_m$.

Induljunk ki a λ ütemezésből. Addig kell cserélgetni a munkadarabokat, míg a legnagyobb CY_{min}-nel rendelkező darabok az elejére nem kerülnek. Ez m gépközpont estén azt jelenti, hogy az m, majd az $m-1$, ..., 1 kört igénylő munkadarabok álljanak egymás után.

Ha megcserélünk két munkadarabot, amelyeknél az i munkadarab megelőzi a k munkadarabot, és igaz rájuk a következő $CY^i_{min}<CY^k_{min}$, akkor abban az esetben, ha k-t követi még tőle is több kört igénylő munkadarab, akkor a befejezési idő változatlan marad, viszont ha k a legutolsó legnagyobb CY_{min}-nel rendelkező munkadarab, akkor a csere befejezési idő csökkenést eredményezhet.

Azt, hogy hány munkadarabnak, és hogyan kell elhelyezkednie ahhoz, hogy befolyásolja egy csere a befejezési időt, a következő sorokban mutatom meg.

Ha a legutolsó, legnagyobb CY_{min}-nel rendelkező munkadarabnak p körre van szüksége a feldolgozáshoz, akkor a befejezési időt a csere abban az esetben befolyásolja, ha

100, vagy annál kevesebb $p-1$ körös,
200, vagy annál kevesebb $p-2$ körös,
$\vdots$
$p \cdot 100$, vagy annál kevesebb 1 körös munkadarab követi.

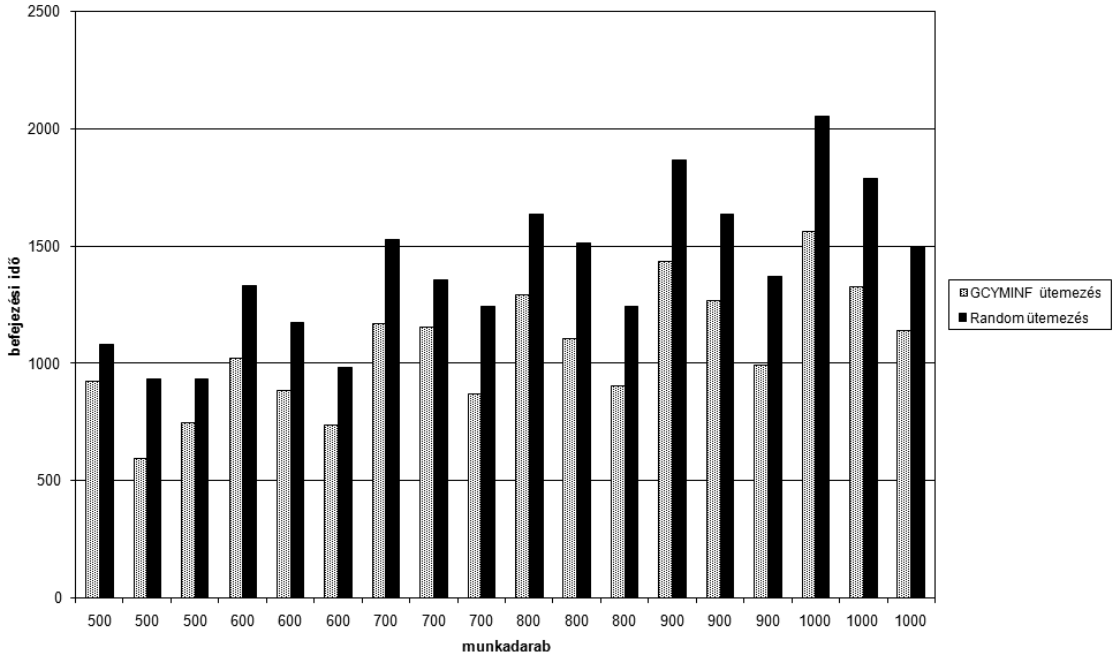
Vagyis az i és a j felcserélése egy λ ütemezésben, ahol $CY^i_{min}<CY^j_{min}$ teljesül, befejezési idő csökkenést vagy változatlan befejezési időt eredményez. A cserék folyamán a λ^* ütemezéshez jutunk.

munkák száma	befejezési idők (egységidőben) random	GCYMINF
100	392	347
100	329	303
100	411	339
200	582	418
200	489	409
200	476	411
300	672	545
300	733	562
300	639	572
400	872	698
400	883	727
400	787	708
500	1082	919
500	932	592
500	934	742
600	1330	1018
600	1176	884
600	980	734
700	1528	1167
700	1356	1154
700	1243	865
800	1634	1290
800	1511	1104
800	1244	903
900	1868	1430
900	1634	1267
900	1370	992
1000	2054	1560
1000	1788	1322
1000	1497	1136

1. táblázat. Befejezési idők

Az 1. táblázat eredményeit megfigyelve észrevehetjük, hogy a $GCYMINF$ algoritmus alkalmazásával átlagban 10-20% munkaidő csökkenést érhetünk el a randomhoz képest. Ez azzal magyarázható, hogy a random sorrendben futószalagra került munkadaraboknál előfordulhat, hogy olyan munkadarab marad a feldolgozás végére, melynek elvégzéséhez akár három futószalagkörre is szüksége van, így elképzelhető, hogy a futószalag teljesen kihasználatlanul, szinte üresen köröz néhány munkadarab miatt. Ezzel szemben a $GCYMINF$ algoritmus kiküszöböli ezt a esetet, hiszen a végére az egy kör alatt feldolgozható darabokat hagyja, így a rendszer teljesen feltöltve teszi meg az utolsó kört is.

Az 1. táblázat eredményeit egy diagrammal szemléltetjük, melyen szépen látszanak a random, illetve a $GCYMINF$ ütemezés által adott befejezési idők közti különbségek, melyet az 2. ábrán figyelhetünk meg.



2. ábra. Az 1. táblázat eredményeinek összehasonlítása

Vegyünk egy példát, amikor a szállítószalagra véletlenszerűen kerülnek fel a munkadarabok. Nézzünk egy egyszerű esetet 50 darab egykörös, 50 darab kétkörös, illetve 50 darab háromkörös munkadarabbal. A legrosszabb eset, amikor a kettő, az egy, majd a három kört igénylő munkadarabok kerülnek ütemezésre. Ekkor a rendszerbe kerül 50 két, illetve az 50 egy körös munkadarab, amelyekből a második futószalagkörre 50 darab, már csak egy folyamat megoldását igénylő munkadarab, illetve 50 darab üres hely marad. Az üres helyeket feltöltjük a megmaradt 50 darab háromkörös darabbal. A harmadik körre már a két munkafolyamatot igénylő munkadarabok kiesnek, de a háromkörös munkadaraboknak még két körre lesz szükségük, így a futószalag a harmadik, illetve a negyedik körben 50-50 üres hellyel fog körözni, tehát 100-zal nő a befejezési idő. Ez a 300 egységidejű befejezési időhöz képest 33%-os romlás. Természetesen ez a legrosszabb eset volt, amely egy random szalagra kerülésnél ritkán fordul elő, de az átlagos 10-20%-os javulás általánosnak mondható.

4.2.2. Állítás:

A $GCYMINF$ algoritmus $s = 2$ esetén nem optimális.

Bizonyítás:

Elegendő egy olyan esetet találni, amikor az algoritmus nem optimális, ezért egy olyan példát mutatok erre az esetre, amely jobb befejezési időt ad, mint a $GCYMINF$ algoritmus $m = 3$ esetén.

A munkadarabok álljanak a J_{cba}, J_a, J_b és J_c típusúakból. Legyen: $n_w = 100$, $s = 2$, $m = 3$, $N_{cba} = n_w/2$, $N_a = n_w/2$, $N_b = n_w/2$, $N_c = n_w/2$. A jelölések a korábban megadott módon értendők.

Először ezeket a munkadarabokat a $GCYMINF$ algoritmus szerint ütemezem, amely áttekinthető formában az 3. ábrán látható. Mivel a feladatok folyamatideje egyenlő 1-gyel, így $s = 2$ esetén nem minden munkadarab kerül feldolgozásra, vagyis maradnak olyan munkadarabok, amelyek kezeletlenek maradnak egy kör után. Az ilyen kezeletlenül maradt munkadarabokra azt mondjuk, hogy elvesztek a futószalagon.

Először az 50 darab három körös J_{cba} munkadarab kerül a rendszerbe, amelyet 50 darab egykörös, vagyis például J_a követ, a $GCYMINF$ algoritmusnak megfelelően. Az első körben az 50 darab J_{cba} típusú munkadarabból a sebesség növelése miatt csak minden második került feldolgozásra, vagyis a második körre az 50 J_{cba}-ból 25 J_{cba} maradt és 25 J_{ba} lett. A munkadarabok feldolgozása így folytatódik addig, amíg a teljes rendelkezésre álló darabszám el nem készül.
Ebben az esetben a befejezési idő, vagyis az

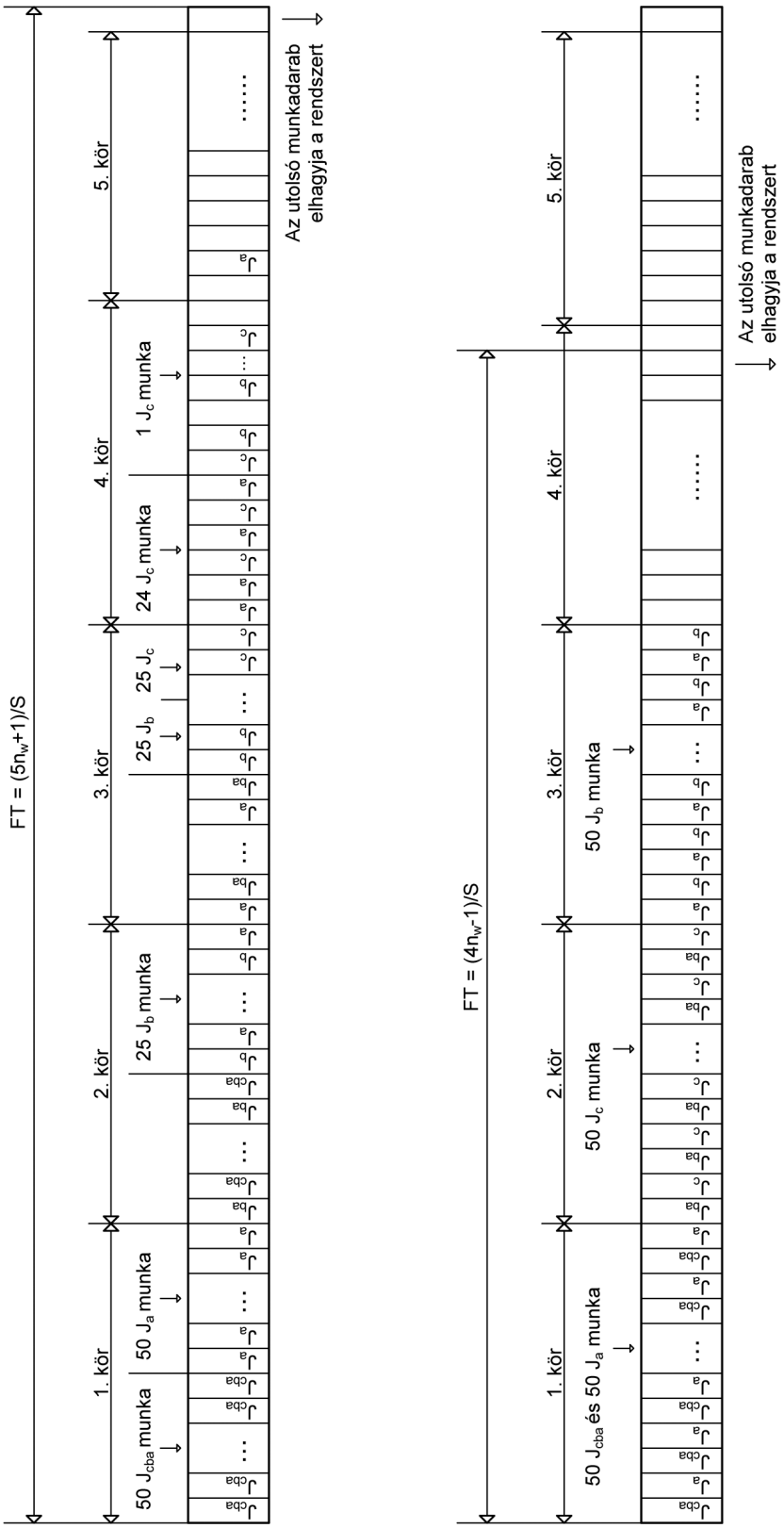
$$FT = \frac{5n_w + 1}{s}.$$

Ennek az ütemezésnek egy javított formájában

$$FT = \frac{4n_w - 1}{s},$$

melyet szintén az 3. ábra mutat. A javulást azzal lehetett elérni, hogy a munkadarabok olyan módon lettek sorbarendezve, hogy azok egyike se vesszen el a futószalagon. Ez azt jelenti, hogy csak minden második munkadarab J_{cba} típusú, és így kerülhet az üres helyekre olyan munkadarab melyek feldolgozása történhet egy körön belül, vagyis olyan munkadarab, amely összeillik vele.

Tehát a jobb megoldást adó ütemezés a munka illesztő algoritmussal jött létre, amelyet a következő részben leírt módon kell alkalmazni.



3. ábra. GCYMINF ütemezés, illetve javítottja $s = 2$ esetén

4.3. Munka illesztő algoritmus

Illeszkedés: Két munkadarab akkor illeszkedik egymáshoz tökéletesen, ha nincs közös műveletük a futószalagon ugyanabban a körben.

Tekintsünk egy példát három gépközpont esetén. A J_{cab} munkadarab illeszkedik a J_{bca} munkadarabhoz, mivel a J_{cab} munkadarabot az első futószalagkörben az $M3$-as gépközpontban kell feldolgozni, és az $M1$ és $M2$ gépközpontban a második futószalagkörben. Míg a J_{bca} munkadarabnak az első körben az $M2$-es gépközpontban, a második körben az $M3$-as gépközpontban, illetve a harmadik körben az $M1$-es gépközpontban történik a feldolgozása.

A munka illesztő algoritmus:

- *Első futószalagkör:*
  - A legnagyobb CY_{min}-nel rendelkező munkadarabot egy hozzá illeszkedő másik munkadarabbal felváltva ütemezzük, a CY_{min} nem növekvő sorrendjében.
  - Ha már nincs több összeillő munkadarab, akkor álmunkákat illesztünk be.
- *Az elsőt követő futószalagkörök:*
  - Ha a bejárathoz egy üres hely érkezik, akkor azt a munkadarabot töltjük be a rendszerbe, amelyik mind az előző, mind a következő munkadarabhoz illeszkedik. Előnyben részesítve azt a munkadarabot, amelyiknek a CY_{min}-je a legnagyobb.
  - Ha nincs illeszkedő munkadarab, akkor egy álmunkát küldünk a futószalagra, vagyis a hely üres marad.
  - Ezt a folyamatot addig folytatjuk, míg az összes munkadarab ütemezésre nem kerül.

A munka illesztő algoritmus bemutatása egy példán keresztül:

A példát $s = 2$ és $m = 2$ esetén vizsgálom.
Két gépközpont esetén a következő munkadarabok lehetségesek:J_a, J_b, J_{ab}, J_{ba}.

A munka illesztési táblázat két gépközpont esetén:

	J_{ba}	J_b	J_a	J_{ab}
J_{ba}	0	0	1	0
J_b	0	0	1	0
J_a	1	1	0	0
J_{ab}	0	0	0	0

A táblázat értékei a következőt jelentik:

A J_{ba} munkadarabhoz illeszkedő munkadarab csak a J_a típusú, mivel csak ezzel a munkabarab típussal nincs közös művelete ugyanabban a futószalag körben, így a táblázatban ide 1-es került.

A J_b munkadarabhoz, amelynek egyetlen körében a b műveletet végezzük el, nem illik természetesen önmaga, valamint a J_{ba} sem, mivel első körben ennél is a b feladat kerül feldolgozásra. Nem illik hozzá a J_{ab} sem, hiszen az első körében az a és a b feladat is feldolgozásra kerül, így a J_b munkadarabhoz hasonlóan az $M2$-es gépközpontba mindkettő bekerül még az első körben, tehát ezekhez 0-t írunk. Összeillik viszont a J_a típussal, mivel teljesen más feladat elvégzésére van szükség a két munkadarabon, így ezekhez 1-es kerül a táblázatban.

A J_a azzal a munkadarabbal illik össze, amelyeknél az első körben nincs a folyamat feldolgozás, tehát ezek a J_{ba} és a J_b.

A J_{ab} típusú munkához pedig olyan típusú munkadarab illene, amelyiknél se a, se b folyamat feldolgozása nincs az első körben, de ilyen munkadarab természetesen nincs ezekből a típusokból.

A munkadarab szükséglet vektor:

$$\mathbf{V} = \begin{pmatrix} N_{ba} \\ N_b \\ N_a \\ N_{ab} \end{pmatrix},$$

ahol N_{ba}, N_a, N_b, N_{ab} rendre a J_{ba}, J_a, J_b, J_{ab} típusú, feldolgozásra váró munkadarabok száma.
Például legyenek N_{ba}, N_b, N_a, N_{ab}, illetve n_w értékei rendre a következők: 50, 100, 75, 50, illetve 100, és szintén $s = 2$ esetet tekintjük. Ekkor a munkaszükségleti vektor a következő:

$$\mathbf{V} = \begin{pmatrix} 50 \\ 100 \\ 75 \\ 50 \end{pmatrix}.$$

Kezdetben, mikor a szállítószalag üres, a legnagyobb CY_{min}-nel rendelkező és a hozzá illeszkedő munkadarabbal felváltva töltjük fel a rendszert. Ez ebben a példában azt jelenti, hogy 50 darab J_{ba}-t 49 darab J_a-val párosítunk, mivel a J_a az egyetlen munkadarabtípus, amely illik a J_{ba} típushoz. Az utolsó helyet pedig egy álmunkával töltjük fel, mivel a második kör során az első körben szereplő J_{ba}-kból J_a lesz, és így nem illenének össze. Eszerint

$$\mathbf{V}' = \begin{pmatrix} 0 \\ 100 \\ 26 \\ 50 \end{pmatrix},$$

ahol a V' a V munka szükségleti vektorból az első kör megtétele után maradt, még feldolgozásra váró munkadarabok számát mutatja.

A második körben a felszabadult üres helyekre a bent maradt munkadarabokhoz, melyek J_a típusúvá váltak, 50 darab J_b munkadarabra van szükségünk. (A megmaradt munkadarabokból csak ez az egyetlen típus, amellyel összeilleszthető.) A futószalag teljes tartalma a második körben elhagyja a rendszert, így a harmadik körben ismét üres futószalaggal indulunk. Így

$$\mathbf{V}'' = \begin{pmatrix} 0 \\ 50 \\ 26 \\ 50 \end{pmatrix},$$

ahol a V'' a második kör megtétele után megmaradt, még a rendszerbe be sem került munkadarabok száma a munka szükségleti vektorban meghatározott sorrendben.

A harmadik körre már csak a 26 darab J_b típusú van párban 26 J_a típusú munkadarabbal, így a fennmaradó 24 J_b munkadarabot 24 darab álmunkával párosítjuk. Ekkor

$$\mathbf{V}''' = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 50 \end{pmatrix},$$

ahol a V''' a harmadik kör után maradt, még feldolgozásra váró munkadarabok számát jelenti a V-nek megfelelő sorrendben. Végül az utolsó körben 50 darab J_{ab} munkadarab 50 darab álmunkával kerül ütemezésre. Az FT értéke ekkor

$$\frac{5n_w - 1}{s},$$

vagyis a befejezési idő 499.

A munka illesztési táblázat két gépközpont esetén nem volt bonyolult, hiszen 4 összeillő munkapárt tartalmaz. Azonban ez nem mondható el három gépközpont esetén, ahol a 15 munkadarab típus van, és 1 munkadarabhoz akár 10 másik munkadarab is illik. Ezt mutatja be a következő táblázat.

A munka illesztési táblázat három gépközpont esetén:

	J_{cba}	J_{cab}	J_{bca}	J_{bac}	J_{acb}	J_{cb}	J_{ca}	J_{ba}	J_{ab}	J_{ac}	J_{bc}	J_a	J_b	J_c	J_{abc}
J_{cba}	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0
J_{cab}	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
J_{bca}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
J_{bac}	1	0	0	0	1	1	0	0	0	1	0	1	0	1	0
J_{acb}	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0
J_{cb}	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0
J_{ca}	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
J_{ba}	1	0	0	0	1	1	0	0	0	1	0	1	0	1	0
J_{ab}	1	1	0	0	0	1	1	0	0	0	0	0	0	1	0
J_{ac}	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0
J_{bc}	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
J_a	1	1	1	1	0	1	1	1	0	0	1	0	1	1	0
J_b	1	1	0	0	1	1	1	0	0	1	0	1	0	1	0
J_c	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0
J_{abc}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

5. Egy módosított rendszer

Az előzőekben bemutatott rendszerben kisebb módosítást végeztem. Célom egy életszerűbb, kevésbé idealizált rendszer bemutatása, elemzése és összehasonlítása a fentiekkel. Az új rendszer legfőbb eltérése az eddigiekhez képest, hogy a munkadarabok a gépközpontokban történő feldolgozás után nem tudnak azonnal visszakerülni a saját helyükre. Tehát a továbbiakban is egy száz férőhelyes, körpályás futószalag által kiszolgált rendszert vizsgálok, amely különbsége az eddigiekhez képest a gépközpontoknál figyelhető meg.

A futószalagot kezdetben üres, ezért szorosan egymás után munkadarabokat rakunk a futószalagra, míg az utolsó hely is feltöltötté nem válik. A futószalag mozgása szintén szakaszos, tehát két részből áll:

- t_1 *periódusidő* - amikor l távolságot halad a munkadarab,
- t_2 *periódusidő* - amíg áll a szalag.

A gépközpontban a feladatok folyamatideje szintén t_1 idő. Míg a szalag mozdulatlan, a raklap vagy belép a gépközpontba, vagy kilép a gépközpontból, vagy visszakerül a futószalagra. Tehát a fő futószalag, valamint a gépközpontot tartalmazó mellék futószalag sebessége megegyezik.

Ez azt eredményezi, hogy míg a főszalagon n lépést kell megtennie egy munkadarabnak, addig a mellékszalagon $n+2$ lépésre van szüksége, hogy ahhoz a helyhez érjen, ahol a fő-, illetve mellékszalag találkoznak. Abban az esetben, ha egy munkadarab a gépközpont bejáratához kerül, és szüksége van feldolgozásra, akkor a szalag mozgásának t_2-es szakaszában, vagyis míg mozdulatlan a rendszer be is kerül a munkagéphez, és a következő, t_1 időben történik a feldolgozása.

A kész munkadarab a t_2-es periódusidőben elhagyja a gépközpontot, és a mellék futószalagra kerül, ahol ugyanazzal a sebességgel halad, mint ahogyan a fő futószalagon tette. Mire a körpályás futószalaghoz ér, az általa elhagyott $l \times l$-es méretű hely már 2 hellyel megelőzte őt, így nem tud a saját helyére visszamenni. Azonban nem csak hogy a saját helyére nem tud menni, addig míg nincs üres hely, a főszalagra sem tud visszakerülni, vagyis vár, míg üres hely nem jön, és akkor t_2 alatt visszamegy és folytatja az útját.

A szalag többi része ugyanúgy működik, vagyis ha a rendszer kijáratához ér egy

munkadarab, és minden feladat végrahajtásra került rajta, elhagyja a rendszert, egyébként pedig köröz tovább, míg a munkafolyamatok elvégzésre nem kerülnek.
Az előző részekben összehasonlítottuk a random, a $GCYMINF$ algoritmus, valamint a munka illesztő algoritmus által adott eredményeket. Ezek számolása mechanikusan is megtörténhet, hiszen azáltal, hogy a feldolgozott munkadarab a saját helyére kerül vissza a fő futószalagon, követhetővé válik a munkadarabok sorrendje, illetve a fennmaradó feladatokat lejegyezve figyelemmel tudjuk kísérni a munkadarabokat elkészülésükig.

A módosított esetben, amikor egy munkadarab egy gépközpontokból történő távozása után szeretne visszakerülni a fő futószalagra, és ezt nem teheti meg úgy, hogy a saját helyére megy, hiszen az a hely már előrébb került, akkor sorrendbeli változások történhetnek. Ez azt jelenti, hogy leghamarabb az eredeti helyéhez képest két hellyel később kerülhet ismét a szalagra, de ez az üres hely hiányában akár sokkal több is lehet.

Ennek a módosított rendszernek az eredményeit szeretném bemutatni a következőkben, ahol végig $s = 1$ esettel foglalkozom.

6. Program

A módosított rendszer tanulmányozása már nem olyan egyszerű, hiszen minél több munkadarabbal dolgozunk, annál hamarabb belebonyolódunk a munkadarabok sorrendjének követésébe, valamint egy idő után ez a megfigyelés teljesen lehetetlenné is válik.

Ezért egy programot készítettem, amely lehetővé teszi a munkadarabok random, illetve a $GCYMINF$ algoritmus által adott ütemezésnek a befejezési idő szerinti összehasonlítását a módosított rendszerben. A program egy beolvasási résszel kezdődik, ahol meg van adva, hogy hány munkadarab ütemezését szeretnénk, majd egy, a programban megírt rész a lehetséges munkadarab típusokból előre megadott darabot összeállít random módon, ahol minden munkadarabnak ugyanannyi esélye van bekerülni, valamint a belőlük alkotott sorrend is véletlen. Ezeket a már meglévő munkákat a program egy random, valamint a $CYMINF$ algoritmusnak megfelelően ütemezi, és eredményképpen a kétféle ütemezés befejezési ideje jelenik meg egy .exe fájlban.

A 2. táblázatban mutatok néhány adatot, amelyek a program futtatásának eredményei.

munkák száma	befejezési idők (egységidőben) random	GCYMINF
100	389	308
100	372	305
100	401	311
200	557	404
200	495	418
200	482	400
300	685	537
300	722	559
300	652	547
400	841	695
400	876	722
400	794	712
500	1007	859
500	962	823
500	991	855
600	1121	985
600	1169	1027
600	1079	975
700	1275	1147
700	1323	1166
700	1245	1124
800	1448	1294
800	1373	1276
800	1415	1262
900	1570	1425
900	1619	1470
900	1556	1436
1000	1799	1634
1000	1644	1579
1000	1717	1565

2. táblázat. Befejezési idők

7. Eredmények a módosított rendszeren

A következőekben meglepő eredményeket mutatok az új rendszer futási idejének növekedésével kapcsolatban (az eredeti rendszerhez képest). Ebben a módosított rendszerben is használom a $GCYMINF$ algoritmust, és megmutatom, hogy míg az eredeti rendszernél a $GCYMINF$ algoritmust használva a körök száma szerinti rendezést nem befolyásolta az, hogy az azonos körigényű munkatípusok milyen sorrendben kerültek feldolgozásra, addig az új módosított rendszerre ez nem igaz.

Vegyük először azt az esetet, amikor csak egy feladatot tartalmazó munkadarabjaink vannak. Meg akarjuk nézni, hogy mennyiben befolyásolja a befejezési időt az, hogy egy munkadarab csak két hellyel később tud visszakerülni a főszalagra.

Az egy feladatos munkadarab típusok a következőek: J_a, J_b, J_c. Ebben az esetben 6 féle sorrend lehetséges. Most külön-külön vizsgálom ezeket a sorrendeket, és a befejezési idők növekedését az eredetihez képest.

7.1. Egy feladatos munkadarabok

7.1.1. Egyetlen típus

7.1.1.1. Állítás:

Ha a rendszerben n darab, azonos típusú, egyetlen feladat megoldását igénylő munkadarabokat ütemezünk, abban az esetben a befejezési idő 2-vel nő az eredeti befejezési időhöz képest.

Bizonyítás:

Csak annyi történik, hogy minden munkadarab 2-vel hátrébb, vagyis a 3., ..., $n+2$. helyre kerül.

7.1.2. J_a, J_b, J_c sorrend

7.1.2.1. Állítás:

A munkadarabok J_a, J_b, J_c sorrendje esetén a befejezési idő 6-tal nő az eredeti rendszer befejezési idejéhez képest.

Bizonyítás:

A munkadarabok feldolgozás előtt így sorakoznak:

$$J_a, \ldots, J_a | J_b, \ldots, J_b | J_c, \ldots, J_c ||,$$

és a számuk rendre n, k, m. Az első gépközponthoz érve az n darab J_a típusú munkadarab bemegy a gépközpontba, és saját helyéhez képest 2 hellyel később megpróbál visszakerülni a főfutószalagra, ha az lehetséges. Ennek az lesz a következménye, hogy az első két hely üres marad, majd elkezdenek visszatérni a feldolgozott munkadarabok, de ez csak $n-2$ munkadarab számára lehetséges, mivel a J_b munkadarabok következnek. Tehát itt 2 munkadarab a gépközpontban ragadt.

$$*, *, J_a, \ldots, J_a | J_b, \ldots, J_b | J_c, \ldots, J_c ||$$

A munkák tovább haladva elérnek a b feladatot feldolgozó gépközponthoz, és az $n-2$ darab J_a elhaladása után a k darab J_b munkadarab lemegy a gépközpontba, így ismét 2 helyet üresen hagyva $k-2$ tud csak visszatérni a főfutószalagra.

$$*, *, J_a, \ldots, J_a | *, *, J_b, \ldots, J_b | J_c, \ldots, J_c ||$$

Ezeket a munkadarabokat közvetlenül követi az m darab J_c munkadarab, amelyek az előzőekhez hasonlóan 2 darab a feldolgozás után nem tud visszatérni, hiszen mire ki szeretne jönni, már a végére beállt az első gépközpontban ragadt 2 darab J_a, majd a második központhoz érve a 2 darab J_b, és csak ezek után tud visszatérni a bennmaradt 2 J_c munkadarab.

$$*, *, J_a, \ldots, J_a | *, *, J_b, \ldots, J_b | *, *, J_c, \ldots, J_c || J_a, J_a, J_b, J_b, J_c, J_c$$

Vagyis azt az eredményt kaptuk, hogy a J_a, J_b, J_c sorrend esetén a befejezési idő 6-tal nőtt meg.

7.1.3. J_c, J_b, J_a sorrend

7.1.3.1. Állítás:

A munkadarabok J_c, J_b, J_a sorrendje esetén a befejezési idő 2-vel nő az eredeti rendszer befejezési idejéhez képest.

Bizonyítás:

A munkadarabok sorrendje a következő:

$$J_c, \ldots, J_c | J_b, \ldots, J_b | J_a, \ldots, J_a ||,$$

melyekből rendre m, k, n darab van. A legelső munkadarab J_c típusú, és ennek feldolgozása a harmadik gépközpontban történik meg, így a feldolgozás után az első 2 munkadarab helye üresen marad, és visszakerül az első $m-2$ J_c munkadarab a $3., \ldots, m.$ helyre.

Azonban az $m+1.$, valamint az $m+2.$ hely már üres ekkorra, mivel a J_b típusú munkadarabok feldolgozása a második gépközpontban már megvolt, így ezek a helyek üresek és a k darab J_b munkadarabból az első $k-2$ az $m+3., \ldots, m+k.$ helyre került, tehát elfoglalhatja az utolsó két J_c munkadarab az $m+1.$, valamint az $m+2.$ helyet. A megmaradt $k-1.$, valamint a $k.$ munkadarab hasonlóan az előzőhöz ki tud menni az $m+k+1.$ és az $m+k+2.$ helyre, hiszen a J_a munkák addigra már feldolgozásra kerültek az első gépközpontban, így azok az $m+k+3., \ldots, m+k+n+2.$ helyre kerültek. Ez a következőképpen néz ki:

$$*, *, J_c, \ldots, J_c | J_c, J_c, J_b, \ldots, J_b | J_b, J_b, J_a, \ldots, J_a || J_a, J_a$$

Ebben a sorrendben a befejezési idő az eredeti rendszer eredményéhez képest csak 2-vel több.

7.1.4. További sorrendek

7.1.4.1. Állítás:

A J_a, J_b, J_c típusú munkák másik négy sorrendjében a befejezési idő növekedés 4 az eredeti rendszerhez képest.

Bizonyítás:

Ez azért van így, mivel az előbb bemutatott legjobb befejezési időt a J_c, J_b, J_a sorrend esetén érjük el, és a többi sorrendben ezekhez képest a feladatok sorrendjében eltérés van. Vagyis a feladatok a legjobb esetben c, b, a sorrendűek, míg a b, c, a sorrendben a b megelőzi a c-t, de a többi feladat viszonya megfelelő marad, vagyis a b megelőzi az a-t. A b, a, c esetén b elhelyezkedése a-hoz képest szintén jó, azonban a c a b-hez képest rossz helyen szerepel. Hasonlóan igaz ez az a, c, b, illetve a c, a, b esetén, ahol szintén a és b cseréjével az optimális sorrendhez jutunk.

Tehát ezekben az esetekben 4-gyel nőtt a befejezési idő az eredetihez képest.

Következmény:

Tehát J_a, J_b, J_c munkatípus különböző sorrendjének ütemezése az új rendszerben $2 \leq x \leq 6$ befejezési idő növekedést okoz.

7.1.5. Egy feladatos munkák n gép esetén

7.1.5.1. Állítás:

n gépközpont esetén, k féle (ahol $k \leq n$) egy feladatos munkadarab befejezési ideje az eredetihez képest $2 \leq x \leq 2 \cdot k$-val fog megnőni.

Bizonyítás:
A legjobb befejezési időt a $J_n, J_{n-1}, \ldots, J_2, J_1$ sorrend esetén kapjuk. Ekkor hasonlóan a 3 gépközponthoz csak 2-vel fog megnőni a befejezési idő, mivel mire visszatérnek az i. gépközpontból a munkadarabok, az őket követő munkadarabok az $i-1$. gépközponban feldolgozásra kerültek, így a J_i-k utolsó 2 munkadarabja az első 2 J_{i-1} helyére tud kerülni, vagyis mindig minden munkadarabnak van helye, mire elkészülve a főfutószalaghoz ér. Tehát összesen 2-vel nő a befejezési idő.

A legrosszabb befejezési időt pedig a $J_1, J_2, \ldots, J_{n-1}, J_n$ esetén kapjuk, amikor az i. gépközpontban a J_i-s munkadarabokból az utolsó kettő nem tud visszatérni a feldolgozás után a futószalagra, mivel a J_{i+1}-es munkadarabok egyből utána következnek, és a feldolgozásuk is csak későbbi gépközpontban lesz, így a legvégén az utolsó 2 bennragadt munkadarab minden típusból a legutolsó J_n után tud visszakerülni, vagyis n típus esetén $2{\cdot}n$-nel nőtt a befejezési idő.

Az összes többi sorrend befejezési idejének növekedése e két eset eredménye között található, hasonlóan átgondolva, mint 3 gépközpont esetén.

7.2. Két feladatos munkadarabok

7.2.1. Állítás:
Ha n darab azonos típusú, két feladatos munkadarabunk van, amelyek csak 1 kört igényelnek a feldolgozásukhoz, akkor a befejezési idő 4-gyel nő az eredeti rendszer befejezési idejéhez képest.

Bizonyítás:
Az n munkadarab az első gépközpontba való feldolgozás után a $3., \ldots, n+2.$ helyre kerül, majd a következő gépközponthoz érve ismét 2-vel későbbi helyre tudnak visszatérni, vagyis az $5., \ldots, n+4.$ helyekre kerülnek.

Tehát a befejezési idő ebben az esetben 4-gyel nőtt meg.

7.3. n feladatos munkadarabok

7.3.1. Állítás:
Ha csak 1 kört igénylő, azonos típusú, n feladatos munkadarabok kerülnek ütemezésre, akkor $2{\cdot}n$-nel nő a befejezési idő.

Bizonyítás:
Minden egyes gépközpontnál 2-vel hátrébb kerül minden munkadarab, így az n feladat megoldása során $2 \cdot n$-nel nő a befejezési idő.

7.4. Vegyes

Talán egy egészen meglepő eredményt kapunk, ha a J_a, J_b, valamint a J_{ab} típusú munkadarabokat ütemezünk. Ha az azonos típusú munkadarabokat egymás után téve, csak a típusok sorrendjén változtatunk, akkor mind a 6 esetben a befejezési idő 4-gyel nő az új rendszerben.

Az eddig bemutatottakhoz hasonlóan, rövid jelölésekkel fogom a munkadarabok feldolgozás utáni helyét szemléltetni.

7.4.1. J_a, J_b, J_{ab} sorrend

7.4.1.1. Állítás:
J_a, J_b, J_{ab} sorrend esetén a befejezési idő 4-gyel nő az eredti rendszer befejezési idejéhez képest.

Bizonyítás:
A J_a, J_b, J_{ab} munkadarabokból legyen rendre n, k, m. Ekkor következőképpen néznek ki:

$$J_a, \ldots, J_a | J_b, \ldots, J_b | J_{ab}, \ldots, J_{ab} | |$$

Az első gépközponthoz jutva a J_a-k és a J_{ab}-k is bemennek, vagyis az új sorrend:

$$*, *, J_a, \ldots, J_a | J_b, \ldots, J_b | J_a, J_a, J_{ab}, \ldots, J_{ab} | | J_{ab}, J_{ab}$$

Később a második gépközponthoz érve a J_b-k és a J_{ab}-k is 2 hellyel később térnek vissza, de az $n+k+3.$, valamint az $n+k+4.$ helyre már vissza tud menni a $k-1.$ és a $k.$ J_b típusú munkadarab, így amikor a második feladat is feldolgozásra került, akkor a kialakult új sorrend a következő:

$$*, *, J_a, \ldots, J_a | *, *, J_b, \ldots, J_b | J_a, J_a, J_b, J_b, J_{ab}, \ldots, J_{ab} | | J_{ab}, J_{ab}, J_{ab}, J_{ab}$$

Tehát 4-gyel nőtt a befejezési idő.

A továbbiakban a többi esetet csak a sorrendek segítségével mutatom be.

7.4.2. J_b, J_a, J_{ab} sorrend

Kezdetben:

$$J_b, \ldots, J_b | J_a, \ldots, J_a | J_{ab}, \ldots, J_{ab} ||$$

Az első gépközpont után:

$$J_b, \ldots, J_b | *, *, J_a, \ldots, J_a | J_a, J_a, J_{ab}, \ldots, J_{ab} || J_{ab}, J_{ab}$$

Majd a második gépközpont után:

$$*, *, J_b, \ldots, J_b | J_b, J_b, J_a, \ldots, J_a | J_a, J_a, *, *, J_{ab}, \ldots, J_{ab} || J_{ab}, J_{ab}, J_{ab}, J_{ab}$$

7.4.3. J_{ab}, J_b, J_a sorrend

A kezdeti sorrend:

$$J_{ab}, \ldots, J_{ab} | J_b, \ldots, J_b | J_a, \ldots, J_a ||$$

Az első gépközpont után:

$$*, *, J_{ab}, \ldots, J_{ab} | J_b, \ldots, J_b | J_{ab}, J_{ab}, J_a, \ldots, J_a || J_a, J_a$$

Majd a második gépközpont után:

$$*, *, *, *, J_{ab}, \ldots, J_{ab} | J_{ab}, J_{ab}, J_b, \ldots, J_b | J_b, J_b, J_a, \ldots, J_a || J_a, J_a, J_{ab}, J_{ab}$$

A fennmaradó három sorrend az eddigiekhez hasonlóan vizsgálható, és ugyanazt az eredményt adják, mint az előzőek, vagyis J_a, J_b, J_{ab} esetén, a munkadarabok bármely sorrendje, melyben az azonos típusúak egymás után következnek, mindig 4-gyel növeli a befejezési időt.

Következmény:
J_a, J_b ésJ_{ab} típusú munkadarabok esetén, ha az azonos munkatípusokat egymás után tesszük, akkor a befejezési idő minden esetben 4-gyel lesz több a módosított rendszerben.

7.5. J_a, J_b, J_{ab}, J_{ba} típusú munkák

A munkák legyenek J_a, J_b, J_{ab}, J_{ba} típusúak, amelyeknek száma rendre n, m, k és p. Tegyünk egy olyan megszorítást, hogy az n, m, k és p összege legyen kisebb egyenlő mint 96. Ezt azért tesszük meg, hogy csak a feldolgozáshoz két kört igénylő J_{ba} típusú munkák maradjanak a második körre. Ezen négy típus esetén a $GCYMINF$ algoritmust alkalmazva, miszerint a feldolgozáshoz szükséges körök száma szerint rendezzük a munkákat, J_{ba} típusú munkák az elejére kerülnek, majd a megmaradt három típusnak hat féle sorrendje lehetséges. Ezekben az esetekben vizsgáljuk azt, hogy a munkák különböző sorrendjében a befejezési idő mennyivel változik az új rendszerben a régihez képest.

Vegyük a különböző sorrendeket, és azt kapjuk, hogy minden esetben 4 hellyel mennek hátrébb a munkadarabok az első körben, így a kevesebb mint 96 munkadarab esetén még az első körben feldolgozásra kerül az összes egykörös munkadarab, és a második körre csak az n darab J_{ba} típusú munkák maradnak, vagyis csak azt kell figyelembe venni, hogy a J_{ba}-k a teljes feldolgozás után hány hellyel kerülnek hátrébb. Mivel a b, illetve az a feladat feldolgozása során 2-2-vel csúszik hátrébb minden munkadarab, ezért a teljes folyamat során 4-gyel nő a befejezési idő. Ez azonban nem igaz abban az esetben, amikor több mint 96 munkadarabunk van, mivel az első körben a munkadarabok 4 hellyel hátrébb csúsznak, és ha nekünk még van olyan munkadarabunk, amelyik nem került fel a futószalagra, de a J_{ba}-k már a második körüket kezdik el, akkor felboríthatják a sorrendet, és akkor nem csak a J_{ba}-kat kell vizsgálni.

Következmény:
J_a, J_b, J_{ab}, J_{ba} típusú munkák esetén, ha számuk kevesebb mint 96, akkor a befejezési időt csak a J_{ba} típusúak befolyásolják, és ezek pedig pontosan 4-gyel növelik a befejezési időt az eredeti rendszerhez képest.

Most pedig egy példát szeretnék mutatni arra az esetre, amikor több mint 96 munkadarabunk van, és nem 4-gyel nő a befejezési idő a módosított rendszerben.

Példa:

Vegyük a J_{ba}, J_b, J_a, J_{ab} munkákat ebben a sorrendben, és számuk legyen rendre p, m, n, illetve k darab, és legyen $p+m+n<96$, valamint $p+m+n+k>98$.
Ekkor a munkák sorrendje a következő az első kör után:

$$*, *, J_{ba}, \ldots, J_{ba}|J_{ba}, J_{ba}, J_b, \ldots, J_b|J_b, J_b, J_a, \ldots, J_a|J_a, J_a, *, *, J_{ab}, \ldots, J_{ab}||,$$

ahol a $||$ előtti J_{ab}-k száma $96-(p+m+n)$. A J_{ba}-k mielőtt elérik a 100. helyet a futószalagon, az előttük kialakult 2 üres helyre belép két J_{ab}. Majd a J_{ba}-k elkezdik a második körüket, így a maradék $k-2-(96-(p+m+n))$ J_{ab} csak utánuk tud a futószalagra kerülni, vagyis a második kör a következőképpen néz ki az első gépközpontban történt feladatmegoldások után:

$$*, *, J_{ba}, \ldots, J_{ba}|J_{ba}, J_{ba}, J_{ab}, \ldots, J_{ab},$$

mivel a J_{ba}-kon már csak az a feladat elvégzése volt hátra, és a J_{ab}-kon pedig elvégzésre került az a feladat.

A második gépközponthoz érve a J_{ab}-k ismét két hellyel hátrébb csúsznak, vagyis a következőképpen néznek ki:

$$*, *, J_{ba}, \ldots, J_{ba}|J_{ba}, J_{ba}, *, *, J_{ab}, \ldots, J_{ab}.$$

Tehát összesen 6-tal nőtt a befejezési idő, mivel az első körben csak a J_{ab}-k előtti helyek maradtak üresek, vagyis a $p+m+n+3$. és a $p+m+n+4$. hely, mivel az 1. és a 2. helyre mikor a bejárathoz került üresen, akkor a $k-(96-(p+m+n))$-ből kettő futószalagra került, és a második körben pedig 4 üres hely keletkezett.

Vagyis mutattunk egy példát, amikor J_{ba}, J_b, J_a, J_{ab} típusok esetén, mikor a munkadarabok száma több mint 98, akkor a befejezési idő növekedés az eredeti rendszerhez képest 6, míg kevesebb mint 96 munkadarab esetén ugyanezekkel a típusokkal a befejezési idő növekedés a munkatípusok bármilyen sorrendjében mindig 4.

8. Összegzés

A dolgozat célja egy speciális rugalmas gyártó rendszer megismerése, valamint a rendszeren végzett különböző ütemezések bemutatása, illetve azok befejezési ideje szerinti összehasonlítás volt.

Egy korábbi tanulmányból [2] elindulva, megismerkedtünk egy speciális rugalmas gyártó rendszerrel, melyen különféle algoritmusok alkalmazásával kapott feldolgozási idők összehasonlítását végeztük el. Kétféle ütemezést mutattunk be, és $s = 1$ esetén a cikkben lévő bizonyítást kijavítva, továbbgondolva a $GCYMINF$ algoritmus optimalitását bizonyítottuk be. A következőkben tárgyaltuk a munka illesztő algoritmust, és egy példát mutattunk arra, hogy $s = 2$ esetén mennyivel kedvezőbb befejezési idő érhető el, mint az $s = 1$ esetén optimális $GCYMINF$ algoritmus alkalmazásával.

A dolgozat második felében pedig változtattunk az eredeti rendszeren, hogy egy életszerűbb, kevésbé idealizált problémát modellezzünk és elemezzük azt. Ehhez egy program készült, amely ezen a módosított rendszeren hasonlítja össze a random, illetve a $GCYMINF$ algoritmus által adott eredményeket. Végül általánosabban vizsgáltuk a módosított rendszert, és bizonyos feltételeket téve mondtunk ki állításokat az eredeti, valamint az új rendszer befejezési idejével kapcsolatban, és mutattunk bizonyítást ezekre.

Összefoglalva tehát, egy már meglévő rendszeren egy jelentős módosítást végezve, a $GCYMINF$ ütemezést használva, egészen meglepő eredményeket kaptunk, melyek azt mutatják, hogy a változtatás szinte alig volt hatással a befejezési időre. Az új feltételek mellett, amelyek valóságosabbá teszik a vizsgált rendszert, elhanyagolható a feldolgozási idő növekedése.

Hivatkozások

[1] Vizvári Béla, Bevezetés a termelésirányítás matematikai módszereibe, egyetemi jegyzet,*ELTE Budapest* (1994): 133-168.

[2] Andrew Kusiak, Flexible manufacturing systems : Methods and studies, *North-Holland, Amsterdam* (1986): 173-189.

[3] C. Sriskandarajah1, S. P. Sethi2 and P. Ladet, A Scheduling methods for a class of flexible manufacturing systems, *Springer, Netherlands* (1989).

[4] C. Sriskandarajah, P. Ladet, R. Germain, Scheduling methods for a manufacturing system, *Flexible Manufacturing Systems: Methods and Studies* (1986).

[5] Elizeth G. Araujo, Gordon A. Dakin, Manfred Huber, and Roderic A. Grupen, Hierarchical Scheduling of Robotic Assembly Operations in a Flexible Manufacturing System, *International Journal of Flexible Automation and Integrated Manufacturing* (1995): 301-316.