

TDK DOLGOZAT

Kozma Ildikó

Miskolci Egyetem
Gépészmérnöki és Informatikai Kar
Alkalmazott Informatikai Tanszék

Szerző: Kozma Ildikó, MSc II. évfolyam
Témavezető: Dr. Kulcsár Gyula, egyetemi docens

A dolgozat címe:
Rugalmas Flow Shop ütemezési feladatok
modellezése és számítógépes megoldása



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

**Rugalmas Flow Shop ütemezési feladatok modellezése
és számítógépes megoldása**

Kozma Ildikó

I. éves mérnök informatikus (MSc) hallgató

Témavezető:

Dr. Kulcsár Gyula, egyetemi docens
ME Alkalmazott Informatikai Tanszék

Miskolc, 2011

TARTALOMJEGYZÉK

BEVEZETÉS.....	1
1. ÜTEMEZÉSI FELADATOK	2
1.1. TERMELÉSÜTEMEZÉS	2
1.2. DISZKRÉT TERMELÉSI FOLYAMATOK	3
1.3. ÜTEMEZÉSI FELADATOK OSZTÁLYOZÁSA.....	5
1.4. ÜTEMEZÉSI FELADATOK MEGOLDÁSI MÓDSZEREI	9
2. A FLOW SHOP ÜTEMEZÉSI FELADAT	11
2.1. AZ ALAP FLOW SHOP ÜTEMEZÉSI FELADAT	11
2.2. A RUGALMAS FLOW SHOP ÜTEMEZÉSI FELADAT	12
2.3. A FLOW SHOP FELADAT MEGOLDÁSI MÓDSZEREI	13
2.3.1. JOHNSON ALGORITMUS.....	13
2.3.2. HEURISZTIKUS MEGOLDÁSOK.....	15
2.3.3. TELJES LESZÁMLÁLÁS	17
2.3.4. MESTERSÉGES INTELLIGENCIA MÓDSZEREK	18
2.4. SZIMULÁCIÓ	23
3. RUGALMAS FLOW SHOP ÜTEMEZÉSI FELADAT SZÁMÍTÓGÉPES MEGOLDÁSA.....	30
3.1. A SZÁMÍTÓGÉPES ALKALMAZÁS	31
3.1.1. AZ ÜTEMEZŐ SZOFTVER FELÉPÍTÉSE.....	31
3.1.2. AZ ÜTEMEZŐ SZOFTVER BEMUTATÁSA.....	34
3.2. TESZTFELADATOK KIÉRTÉKELÉSE.....	44
3.3. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK	47
ÖSSZEFOGLALÁS	49
IRODALOMJEGYZÉK	50

BEVEZETÉS

Napjaink kulcsfontosságú kérdése az optimalizálás, az optimumra való törekvés. Az optimalizálás nem más, mint adott körülmények között megadott feltételek figyelembevételével egy (vagy több) kitűzött cél elérésére való törekvés (ahol egy optimális megoldás a cél(ok) szempontjából mindig valamilyen szélsőértéket fejez ki: legnagyobb, legkisebb). Ennek tudatában felismerhetjük, hogy életünk szinte minden területén optimalizálunk: szeretnénk minél rövidebb idő alatt elvégezni egy feladatot, minél rövidebb úton eljutni valahova, minél jobban beosztani az időnket, stb.

Ha egy kicsit elvonatkoztatunk a hétköznapiaktól, és egy termelővállalatot tekintünk, akkor felismerhetjük, hogy annak működését is több cél együttese határozza meg. Ilyen célok lehetnek például a profit növelése, a szállítókészség növelése, a selejtszint csökkentése, a raktárkészlet csökkentése, az erőforrás-kihasználtság növelése, további hatékonysági mutatók javítása, stb.

A célok együttese – a korlátfeltételekkel együtt – egy nagyon összetett optimalizálási problémát határoz meg, amelynek megoldásához a kiinduló probléma lebontásaként több optimalizálási feladat megoldása válik szükségessé. Ilyen feladat többek között a termelés ütemezése is. Diszkrét termelési folyamatok esetén termelésütemezés alatt a munkadarabokon végzett műveletek és a szükséges erőforrások egymáshoz rendelését, továbbá a műveletek sorrendjének és időbeli lefutásának pontos meghatározását értjük.

Az alkalmazási területtől függően az ütemezési feladatok sokfélék és igen változatosak lehetnek. Dolgozatomban az ütemezési feladatot, mint diszkrét termelési folyamattal kapcsolatos kombinatorikus optimalizálási problémát közelítem meg és a termelésütemezés egy jellegzetes modelljén keresztül mutatom be. Ismertetem a Flow Shop és a Flexible Flow Shop ütemezési feladatosztályokat és azok néhány tipikus megoldási módszerét. Ezt követően bemutatom az intervallum-korlátos erőforrások kezelését támogató kiterjesztett modelletemet, és a megoldására kifejlesztett számítógépi alkalmazásomat. Tesztfeladatokon végzett futási eredményekre alapozva értékelem a javasolt megoldási módszerek hatékonyságát és a paraméterek változtatásának hatását. Végezetül, ismertetem a továbbfejlesztési elképzeléseimet és a gyakorlati alkalmazás lehetőségeit.

1. ÜTEMEZÉSI FELADATOK

Az élet számos területén találkozunk ütemezési problémákkal. Ilyen problémák például az összetett projektek csoportmunkával történő megvalósításai, oktatási intézmények teremkiosztásainak és órarendjeinek összeállítása, vasúti menetrendek meghatározása, logisztikai szállítási feladatok megoldása, stb. Az alapvető döntési feladat közös jellemzőit kiemelve elmondható, hogy minden esetben adott (rendszerint korlátozottan rendelkezésre álló) erőforrásokat rendelünk össze az azokat használókkal úgy, hogy a megoldás eleget tegyen bizonyos előre meghatározott szempontoknak. Éppen ez az a közös jellemvonás, amely által az ismertetésre kerülő ütemezési modellek megoldására javasolt koncepció és kapcsolódó rugalmas módszerek nem csak termelésütemezési feladatokra alkalmazhatóak, hanem más jellegű - többek között a példaként felhozott - problémák esetében is eredményesen adaptálhatóak.

1.1. TERMELÉSÜTEMEZÉS

A termelésütemezés fogalmának megismeréséhez célszerű magából a termelés fogalmából kiindulni, amelyet C. West Churchman a következő módon határozott meg:

„Termelésen a gazdasági javak tervszerű és sokszorozott előállítását értjük.”

Ezt követően értelmezhető a termelésirányítás fogalma, amely az előbbi meghatározásban szereplő tervszerűséget hivatott megvalósítani:

A termelésirányítás tágabb értelemben a termeléssel kapcsolatos döntési feladatok megfogalmazásával és megoldásával foglalkozik, figyelembe véve a termelés főbb tényezőit (eszközök, tárgy, feltételek és célok) és ellenőrizve a döntések végrehajtását. ^[1]

A termelésirányításon szűkebb értelemben termelési feladatok meghatározását és végrehajtásuk megszervezését értjük. ^[1] Az időhorizontok nagysága szerint a feladat csoportosítása átlagos bonyolultságú termékek esetén tipikusan a következő módon alakul kis- és közép-sorozatgyártás esetén:

- termeléstervezés (1-3 hónap),
- termelésütemezés (5-10 nap),
- termelésprogramozás (8-24 óra).

Az időhorizontok relatív nagysága a termékek bonyolultságától és egyéb tulajdonságaitól, a gyártandó darabszámtól és a gyártás rendszerétől függ. ^{[1] [5]}

A gyártórendszerek műhelyszintű irányítása a termelési főterv (MPP: Master Production Plan) megfelelően lebontott részeit a termelésstervező és ütemező modultól (PPS: Production Planning and Scheduling) kapja meg gyártási rendelések formájában. Egy gyártási rendelés tartalmazza a legyártandó munkadarab azonosítóját, megrendelt darabszámát, a legyártás legkorábbi kezdeti időpontját és végső befejezési határidejét, a rendelés prioritását, a művelettervekre és egyéb információs állományokra mutató hivatkozásokat és egyéb speciális jellemzőket. A termelési terv alapján a termelésütemező modul elkészíti az üzem gyártási ütemtervét, amely a korlátozó feltételeket kielégíti és az aktuális termelési cél (vagy célok kompromisszumos) optimumát biztosítja. Az ilyen módon meghatározott gyártási ütemterv alapján elkészíthető a termelési finomprogram, amely a műveletek gépenkénti sorrendjén túlmenően a tervezett tevékenységek indítási és befejezési időpontját is tartalmazza.

1.2. DISZKRÉT TERMELÉSI FOLYAMATOK

A termelésirányítás szűkebb értelemben vett megadásánál a feladatok csoportosításának magyarázatakor diszkrét termelési folyamatot feltételeztem. Azonban az ipari termelési folyamatok két alapvető csoportba sorolhatóak:

- folytonos termelési folyamatok: a folyamat-elemek folyamatosak és folytonosak; főleg a vegyiparban jellemző (pl. műtrágyagyárak, cukorgyárak) de az energiaipari üzemek is ide tartoznak.
- diszkrét termelési folyamatok: a folyamat-elemek térben és időben diszkrét természetűek; a technológia önálló egységeket, alkatrészeket, szerelvényeket és szerelt termékeket állít elő.

A két alaptípus kombinált változatai a hibrid termelési folyamatok, melyek bizonyos részei a folytonos más részei a diszkrét folyamatok sajátosságaival jellemezhetők (pl. gyógyszergyártás, palackozott italok gyártása, stb.).

A termelésirányító rendszerek alapfeladata az alkalmazott technológia elemi folyamatainak irányított végrehajtása. Ahhoz, hogy ezt a feladat megfelelően lássa el a termelésirányító rendszer, elengedhetetlen a termelési folyamat megismerése és jellemzőinek meghatározása.

A diszkrét és folytonos technológiák fő jellemzőit az **1. táblázat** foglalja össze:

	DISZKRÉT TECHNOLÓGIÁK	FOLYTONOS TECHNOLÓGIÁK
folyamat jellege	térben és időben diszkrét	térben és időben folytonos
munkaszervezés	8 órás műszak, munkahelyek	24 órás termelés, folyamatok
alapanyag	darabos, szilárd	ömlesztett, folyékony
folyamatelem	művelet (operáció)	eljárás (procedúra)
megszakítás	viszonylag könnyű, egyszerű	nehéz, veszélyes és költséges
átállás	rövid előkészületet igényel	hosszú és fáradtságos tevékenységet igényel
készletek	a termékek logisztikája bonyolult	a folyamat logisztikája rendszerint egyszerű
sorozatnagyság	rendelésfüggő, könnyen változtatható	nem értelmezhető, de létezik minimális gyártható mennyiség
műveletek sorrendje	nagyon sokféle, sok változatban	nagyon szigorúan kötött, kevés változatban
késztermék	általában alkatrészekből összeszerelt inhomogén termék	egységcsomagokba, vagy tartályokba adagolt homogén termék
automatizálás	az irányító rendszer hierarchikusan szervezett vezérlőkből áll	az irányító rendszer osztott, kaszkád szabályzókból áll

1. táblázat: Diszkrét és folytonos folyamatok összehasonlítása ^[1]

Az ipari technológiák igen eltérőek lehetnek, és ezek az eltérések különböző eszközöket és módszereket igényelnek. Éppen ezért a termelésirányító rendszer szerkezete és működése jelentős mértékben függ attól, hogy milyen jellegű termelési folyamatot irányít/felügyel.

A továbbiakban a termelés ütemezését, mint diszkrét termelési folyamatokon értelmezett feladatot vizsgálom. Ehhez célszerű megadni a diszkrét gyártási folyamatok alapelemét képező művelet (vagy más szóval operáció) fogalmát:

A műveletek meghatározott munkadarabon, meghatározott gépeken, meghatározott sorrendben, meghatározott szerszámok használatával, meghatározott technológiai paraméterek ismeretében elvégezhető eljárások.

1.3. ÜTEMEZÉSI FELADATOK OSZTÁLYOZÁSA

Az ütemezési feladatok osztályozására a szakirodalomban többféle szempontrendszert is használnak a kutatók. Az általánosan elfogadott szempontrendszer szimbólumait felhasználva célszerű egy, a kitűzött feladatnak megfelelő formális modellt alkotni. Ez a modell később a feladat jellemző tulajdonságainak tömör és egyértelmű leírására szolgál. A megfelelő feladatosztályba történő besorolással a probléma átláthatóbbá és könnyebben kezelhetővé tehető.

A szakirodalom többféle osztályozási rendszert használ. Ezek közül az egyik legelterjedtebb az $\alpha / \beta / \gamma$ háromelemes formalizmus, amelyet Graham és társai javasoltak először (1979).^[11] Azóta ez a jelölés-rendszer számos szimbólummal bővült, azonban az alap elképzelés változatlan maradt. Egy ütemezési feladat rövid formális leírása során három alapvető kérdéskört kell megválaszolni, melyek a következők: α az erőforrás-környezetre vonatkozó információkat, β a munkák jellemzőit illetve a rájuk vonatkozó feltételeket, korlátozásokat, és végül γ az ütemezés célfüggvényét (vagy célfüggvényeit) megfogalmazó jelölésrendszer általános szimbóluma. A továbbiakban az α , β és γ szimbólumok által felvehető jelölésekre mutatok példákat (a teljesség igénye nélkül).^{[3] [5]}

α – az erőforrás-környezetre vonatkozó jellemzők szimbólumai és azok jelentése:

Ez a mező tartalmazza az erőforrások számára és jellemzőire, erőforrásokon történő áthaladásra, operációk sorrendjére vonatkozó információkat. Ezeket általában két szimbólum jelzi: $\alpha = [\alpha_1, \alpha_2]$.

- Az α_1 szimbólummal a következő modellek jelölhetők ki:
 - $\circ$ (üres): olyan egy gépes modell, amelyben minden munka esetében egyetlen operációt kell végrehajtani.

- *P*: párhuzamos gépes modell, amelyben minden munkához egyetlen operáció tartozik, és különböző munkákon több egyenértékű gép dolgozhat párhuzamosan.
 - *Q*: párhuzamos gépes modell, amelyben minden munkához egyetlen operáció tartozik, és különböző munkákon több, eltérő sebességű gép dolgozhat párhuzamosan (a gépek sebessége minden munkára azonos).
 - *R*: párhuzamos gépes modell, amelyben minden munkához egyetlen operáció tartozik, és különböző munkákon több, eltérő sebességű gép dolgozhat párhuzamosan (a gépek sebessége munkánként eltérő lehet).
 - *G (General Shop)*: általános többoperációs modell, amelyben minden munka több operációból áll, az operációk száma munkánként eltérő lehet, több gép áll rendelkezésre, és az egyes operációt egy adott gép végezheti el.
 - *J (Job Shop)*: a *G-modell* speciális esete, amikor a munkák operációinak végrehajtási sorrendje kötött.
 - *F (Flow Shop)*: a *J-modell* speciális esete, amikor a munkák operációinak száma és végrehajtási sorrendje minden munka esetében megegyezik, és az adott számú operációt rendre egy adott gépen lehet végrehajtani.
 - *O (Open Shop)*: nyitott műhely modell, a *G-modell* speciális esete, amikor a munkák operációinak végrehajtási sorrendjére nincs megkötés, de számuk minden munka esetében megegyezik, és az adott számú operációt rendre egy adott gépen lehet végrehajtani.
 - *X (Mixed Shop)*: vegyes műhely modell, a *G-modell* speciális esete, amelyben a munkák egy részére a *J*-, más részére az *O-modell* előírásai érvényesek.
- α_2 a gépek számára vonatkozik. Ha a mező üres (nem tartalmaz szimbólumot), akkor az ütemezési feladat olyan rendszerre vonatkozik, amelyben a gépek száma tetszőleges. Ha „*m*” jelölés szerepel a mezőben, akkor az ütemezési feladat többgépes rendszeren értelmezett, amelyben a gépek száma tetszőleges, de választás után rögzített. Az üres és „*m*” jelölések helyett megadható egy pozitív egész szám is, amely a rendszerben jelen lévő gépek számát adja meg.

A szám, valamint a *P*, *Q*, *R* szimbólumokat szokás cella szintű információknak is nevezni. A *G*, *J*, *F*, *O*, *X* szimbólumok megnevezésére pedig a műhelyszintű modell elnevezés használatos. A műhelyszintű jellemzőket ki lehet egészíteni a *Flexible* (rugalmas) és az *Extended* (kiterjesztett) jelzővel, ezekben az esetekben pontosan meg

kell fogalmazni az új jellemzők mögöttes tartalmát. A rugalmas jelző (*Flexible*) arra utal, hogy míg az alapmodellben egy adott művelet (vagy operáció) végrehajtására egyetlen gép alkalmas csupán, addig a rugalmas modellben a műveletet a gépek egy meghatározott halmazába tartozó bármelyik gép elvégezheti. Ezáltal az alapmodell gépválasztási feladattal bővül ki. A kiterjesztett (*Extended*) jelző arra utal, hogy az alapmodellt egyszerre több új jellemzővel bővítették ki. ^[4]

β – a munkákra vonatkozó korlátozások és egyéb végrehajtási jellemzők szimbólumai és azok jelentése:

Ebben a mezőben kerülnek megfogalmazásra a munkák kezdési- és határidejére, sorrendi és végrehajtási módjára vonatkozó információk. Ez a mező is lehet többértékű, ha egyszerre több előírás is érvényben van, de lehet üres is, ha nem vonatkozik külön előírás a vizsgált feladatra.

- *permt*: a munkák sorrendje a gépek között nem változhat meg, vagyis minden gépen a munkák végrehajtási sorrendje ugyanaz.
- *pmtn*: a munka végrehajtása megszakítható, majd később folytatható.
- *prec*: a munkák végrehajtásának sorrendjére szigorú előírás vonatkozik.
- r_i : az i -edik munka (első operációjának) legkorábbi indítására vonatkozó előírás.
- d_i : az i -edik munka (utolsó operációjának) befejezésére vonatkozó előírás (határidő).
- p_{ij} : az i -edik munka j -edik gépre vonatkoztatott operációinak műveleti ideje.
- $s_{j,ik}$: az j -edik gépen az i -edik munkáról k -adik munkára történő átállási ideje (setup time).
- A_i : az i -edik munka gépekhez rendelésére vonatkozó korlátozás, amely előírja a műveletvégzésre alkalmas gépek halmazát.
- *chains*: a munkák részhalmazára vonatkozó sorrendi előírás, amelynek gráfja az élek egy olyan láncolatát írja le, amelyben minden csúcsba legfeljebb egy él vezet és minden csúcsból legfeljebb egy él indul ki.
- *tree*: a munkák részhalmazára vonatkozó sorrendi előírása, amely egy gyökeres fa-gráfnak feleltethető meg; két típusa van:
 - *in-tree*: a csúcspontokból kiinduló élek száma legfeljebb egy.
 - *out-tree*: a csúcspontokba vezető élek száma legfeljebb egy.

- *sp-graph*: a munkák sorrendjének egy olyan előírása, amelynek gráfja egy soros-párhuzamos gráf, ami részgráfokból párhuzamos vagy soros kapcsolással állítható össze.
- *batch*: a sorozat a munkák egy adott halmaza, amelyek végrehajtása valamely gépen összekapcsolva (gépátállítás nélkül) megy végbe; két típusa van:
 - *s-batch*: a munkák egymás után, szekvenciálisan hajtódnak végre.
 - *p-batch*: a munkák az adott gépen párhuzamosan hajtódnak végre.

γ – az ütemezés célfüggvényének szimbólumai és azok jelentése:

Ebben a mezőben kerül megfogalmazásra az ütemezési feladat megoldásával kapcsolatos elvárás, amely az ütemezési hatékonyság kiértékelésének alapja.

Az ütemezési célfüggvény egy olyan számítási eljárás, amely segítségével numerikus formában kifejezhető egy adott ütemterv adott szempont szerinti minősége.

Célfüggvényként a leggyakrabban vizsgált jellemzők a következők:

- C_i : az i -edik munka befejezési időpontja.
- L_i : az i -edik munka késése, amely a befejezési időpont és a határidő különbsége:

$$L_i = C_i - d_i$$
- T_i : az i -edik munka csúszása, amely az i -edik munka pozitív értékű késéséből és a 0 maximumából adódik: $T_i = \max(0, L_i)$
- E_i : az i -edik munka sietése, amely az i -edik munka negatív értékű késéséből és a 0 maximumából adódik: $E_i = \max(0, -L_i)$
- U_i : egységnyi büntetés, amelynek értéke 0, ha a befejezési idő a határidőnél nem nagyobb, egyébként 1
- F_i : az i -edik munka átfutási ideje, amely a befejezési időpont és a tényleges indítási időpont különbségéből számítható: $F_i = C_i - R_i$

A példaként feltüntetett jellemzők bármelyikét G_i helyett beírva a következő képletekbe megkaphatjuk a legfontosabb célfüggvényeket, amelyek lehetnek maximum (γ_1), összeg (γ_2) vagy átlag (γ_3) értékekre vonatkozóak. Ha a célfüggvényben a munkákat nem egyforma mértékben szeretnénk figyelembe venni, akkor azt az egyes munkákhoz prioritásértéket rendelve biztosíthatjuk. Ez tulajdonképpen a célfüggvényben szereplő jellemző egy w_i szorzóval történő kiegészítését jelenti:

$$\gamma_1 = \max(w_i G_i) \quad \gamma_2 = \sum_i w_i G_i \quad \gamma_3 = \frac{\sum_i w_i G_i}{n} .$$

Az ütemezési modellek ismertetett osztályozása is szemlélteti az ütemezési feladatok sokféleségét. A kutatók a vizsgált ütemezési feladatok jellemzőinek leírására használt szimbólumrendszert folyamatosan bővítik, ezáltal tovább szélesítve a modellek rendszerét.

1.4. ÜTEMEZÉSI FELADATOK MEGOLDÁSI MÓDSZEREI

Az ütemezési feladatok feltételekkel korlátozott, sokparaméteres kombinatorikus optimalizálási problémákhoz vezetnek. Az ütemezési feladatok többsége az NP-hard feladatosztályba tartozik. Mivel a különböző ütemezési feladatok nagyon ritkán oldhatóak meg ugyanazokkal a módszerekkel, ezért érdemes az egyes modelleket és megoldási lehetőségeiket külön-külön vizsgálni.

Egy kiválasztott ütemezési modellt tekintve kevés olyan eljárás létezik, amely előállítja a feladat egzakt (optimális) megoldását. Ezek a módszerek általában nem polinomiális futási idejűek, tehát nagyobb feladatok esetén nem célszerű az alkalmazásuk (pl. teljes leszámlálás módszere, szétválasztás és korlátozás módszere, stb.). Speciális esetekben, leegyszerűsített problémákra léteznek polinomiális futási idejű algoritmusok, amelyek optimális megoldásra vezetnek (pl. Johnson algoritmus). Ezek a módszerek általában nem rugalmasak, vagyis módosítás nélkül nem képesek az alapul vett ütemezési modelltől eltérő kibővített feladatok kezelésére.

Nagyobb méretű, de még viszonylag egyszerű feladatok megoldásakor gyakran alkalmazunk heurisztikus módszereket. A megoldási lehetőségek e csoportja általában egyszerű szabályok alapján állítja elő az ütemezési feladat megoldását (pl. SPT-elv - a legrövidebb műveleti idejű munkát előre, stb.). Az ilyen szabály alapú módszerekkel csupán közelítő, kvázi-optimális megoldás adható, viszont az nagyon gyorsan előállítható. Az alapul vett ütemezési modelltől eltérő kibővített feladatok esetén általában ezen módszerek alkalmazása sem célszerű, mivel a kibővített ütemezési feladatok és az alap feladatok megoldása és a megoldáshoz tartozó célfüggvény értéke jelentős mértékben különbözhet egymástól.

A heurisztikus módszerek másik gyakran alkalmazott csoportját képezik a felépítő jellegű heurisztikák. Ezek összetettebb problémák esetén lépésről-lépésre haladva, a korábbi döntések következményét figyelembe véve alakítják ki a viszonylag jó megoldást. Ezek a megoldások szintén erősen feladat-specifikusak. Ha a feladat jellemzőit

változatlanul megtartjuk, de más célfüggvényt használunk minősítő kritériumként, akkor a megoldó algoritmus már nem eredményez „jó közelítő” megoldást. Egyéb jellemzők változtatása esetén nem alkalmazhatóak ezek a módszerek.

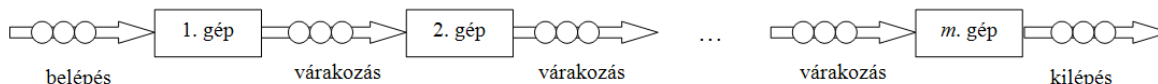
Összegezve megfogalmazható, hogy erős modellfüggésük miatt az egzakt és a heurisztikus módszerek csak szűk problémakör esetén alkalmazhatóak: az ütemezési feladat méretének növelésével vagy az ütemezési modell kibővítésével egyre kevésbé képesek a feladat hatékony megoldására. Éppen ezért ilyen NP-hard problémák esetében célszerű bevetni az optimum-közeli megoldások megtalálását hatékonyan támogató mesterséges intelligencia módszereket, amelyek a keresés során talált legjobb megoldás iteratív javításával igyekeznek minél kedvezőbb eredményt elérni. Az iteratív keresés miatt ezek a módszerek is időigényesek, azonban megfelelően megválasztott végrehajtási paraméterekkel szabályozható a futási idejük. A mesterséges intelligencia módszerek legnagyobb előnye az egzakt és heurisztikus megoldásokkal szemben, hogy tetszőleges ütemezési feladatnak megfelelően rugalmasan átalakíthatóak, vagyis bármilyen ütemezési feladat kezelésére alkalmassá tehetők. A mesterséges intelligencia módszerek kibővített nagyfeladatok esetén is viszonylag rövid idő alatt képesek optimum-közeli megoldás előállítására. Ez indokolja, hogy ütemezési feladatok megoldásakor miért érdemes a mesterséges intelligencia módszereivel kiemelten foglalkozni, és vizsgálni működésüket különböző feladatokon, különböző paraméterek használatával.

2. A FLOW SHOP ÜTEMEZÉSI FELADAT

Az angol terminológia a „*Flow Shop*” elnevezést használja azon ütemezési feladatok esetén, ahol adott erőforrás-környezetben meghatározott gépeken a munkadarabok meghatározott sorrendben haladnak végig (vagyis minden munkadarabra a technológiai útvonal kötött és ugyanaz). A *Flow Shop* típusú ütemezési feladatokat két osztályba sorolják, ezek az előzéses és az előzés-mentes feladatok. Az előzés-mentes feladatok esetén a munkák sorrendje minden gépen megegyezik, míg előzéses esetben a munkák egyes gépeken történő áthaladási sorrendje változhat. ^[2]

2.1. AZ ALAP FLOW SHOP ÜTEMEZÉSI FELADAT

Az alap *Flow Shop* feladat olyan előzés-mentes ütemezési feladat, amelyben azonos technológiai útvonallal rendelkező, adott n számú munkát adott m számú különböző gépen kell végrehajtani úgy, hogy a munkák sorrendjének meghatározása egy adott szempont (pl. a legnagyobb csúszás minimalizálása vagy az átfutási idő minimalizálása) szerint optimális legyen. Az alap *Flow Shop* feladatnak megfelelő műhelymodellt mutatja be az **1. ábra**:



1. ábra Az alap *Flow Shop* feladat műhelymodellje

A gépek között korlátlan méretű átmeneti tárolókat tételezhetünk fel, és elhanyagolhatóak a gépek közötti anyagmozgatási idők, valamint a műveleteket megelőző gépátállítási idők.

A technológiai útvonal és a műveleti idők mellett az alap *Flow Shop* feladatokra az alábbi kikötések érvényesek:

- egy gépen egyszerre csak egy munka lehet;
- egy munkán egyszerre csak egy gép dolgozhat;
- minden munka ugyanazokat az operáció típusokat igényli;
- a folyamatban a gépek és a munkák is várakozhatnak;
- a munkáknak egyedi indítási időpontja, átfutási ideje és elkészülési időpontja van;
- a vizsgált kezdeti időponttól ($T=0$) minden munkadarab rendelkezésre áll (indítható);
- a munkák között nincs megelőzési reláció (kötelező sorrendi korlátozás).

2.2. A RUGALMAS FLOW SHOP ÜTEMEZÉSI FELADAT

A valóságban az alap *Flow Shop* feladat nagyon ritkán fordul elő. Ahhoz, hogy a modell szélesebb körben alkalmazható legyen, érdemes azt úgy kiegészíteni, hogy minél több, a gyakorlatban is megfigyelhető jellemzővel rendelkezzen. Azonban minél pontosabban alkotjuk meg az ütemezési feladat modelljét, annál összetettebb problémához jutunk, amelynek megoldása - nem csak annak eredménye, hanem annak módja és számítási ideje is - jelentősen eltérhet az alap feladat megoldásától.

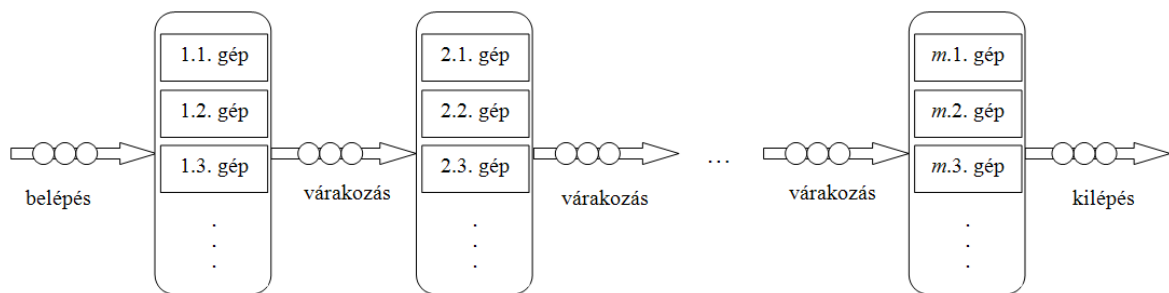
Az alábbiakban az alap feladat néhány kiegészítési lehetőségét mutatom be, amelyek figyelembevételével rugalmas *Flow Shop* feladatot kapunk:

- *határidő*: adott az az időpont, amikorra egy adott munkának el kell készülnie;
- *beállítási idő*: nem hanyagolható el az egyes műveletek végrehajtását megelőző, az egyes gépeken végzett beállítási folyamat időtartama;
- *anyagmozgatási idő*: nem hanyagolható el az az időtartam, amely alatt egy adott munka átkerül a következő műveletet végző gépre, miután a megelőző művelet befejeződött;
- *szakaszos működés*: figyelembe kell venni a gépek rendelkezésre állásának előírt időszakait (pl. rögzített műszakok, karbantartás miatti kiesés, stb.);
- *megszakítható végrehajtás*: szakaszos működés esetén a munkák műveleteinek végrehajtása megszakítható, később pedig folytatható;
- *párhuzamos gépes működés*: adott műveletek végrehajtására egy gépcsoport alkalmas, amelynek gépein valamilyen szempont szerint oszlik el a munkák adott operációinak végrehajtása;
- *előzés*: az egyes gépeknél kialakuló várakozási sorban jelen levő munkák közül valamilyen szempont alapján nem a kezdetben meghatározott sorrend szerinti következő munka kerül végrehajtásra;
- *legkorábbi indítás*: adott az az időpont, amikorra egy adott munka műveleteinek végrehajtásához szükséges feltételek mindegyike teljesül (pl. alapanyag megérkezett);
- *prioritás*: a munkák valamilyen szempont szerinti rangsorolása;
- *sorrendi korlátozás*: a munkák halmazának egy részére vagy egészére vonatkozó megelőzési előírás;
- *stb.*

Dolgozatomban egy olyan rugalmas *FlowShop* ütemezési feladatosztály modellezését és számítógépes megoldását tűztem ki célul, amely támogatja a párhuzamos működésű és eltérő intenzitású gépeket, figyelembe veszi az anyagmozgatási és beállítási időket, az erőforrások szakaszos rendelkezésre állását és a munkákra vonatkozó határidőket.

$$FF / p_{ij}, s_{jik}, d_i, pmtn, A_i / \sum C \text{ vagy } C_{max} \text{ vagy } \sum T \text{ vagy } T_{max}$$

Az így kialakított ütemezési feladatnak megfelelő műhelymodellt a **2. ábra** mutatja be:



2. ábra A rugalmas *Flow Shop* feladat műhelymodellje

2.3. A FLOW SHOP FELADAT MEGOLDÁSI MÓDSZEREI

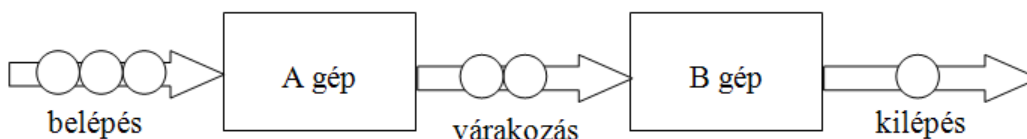
Egy-egy ütemezési feladatot több módszerrel is megoldhatunk, azonban az erősen függ a probléma pontos modelljétől, hogy mely módszerek alkalmazása célszerű. A *Flow Shop* típusú feladatok megoldására is ez jellemző: az alap feladat megoldására alkalmas módszerek a szükséges algoritmusbeli módosításokkal képesek lehetnek rugalmas feladatok kezelésére is, azonban az ily módon előállított megoldások nem feltétlenül optimális vagy kellően optimum közeli eredményekhez vezetnek. Ennek oka, hogy a módosított megoldó módszerek alapját képező algoritmusok több, rugalmas feladatokra jellemző tényezőt is figyelmen kívül hagyhatnak, amelyek pedig jelentős mértékben befolyásolhatják a feladat megoldását.

2.3.1. JOHNSON ALGORITMUS

Az ütemezési feladatok kezelhetősége függ a gépek számától. $m = 2$ gép esetén egy egyszerű módszer, a Johnson-algoritmus használható fel az alap feladat egzakt megoldására, amely bizonyos esetekben $m = 3$ esetre is kiterjeszthető.

Ez az algoritmus olyan sorrendtervezési feladatok megoldására alkalmazható, amelyekben n számú munkát kell két egymást követő géphez hozzárendelni. [2] A feladatban kitűzött cél a munkák átfutási idejének (vagy másképp fogalmazva az egyes gépek állásidejének) minimalizálása.

A **3. ábrán** két gépből álló gyártócella látható, amelyben az időrendileg első megmunkálási művelet az *A*, második megmunkálási művelet a *B* gépen történik.



3. ábra Két gépből álló gyártócella modellje

A kétféle megmunkálási műveletet n számú különböző alkatrészén végzik el, amelyek jellege hasonló, de méreteik különbözőek, továbbá tetszőleges sorrendben jelentkezhetnek. Következésképpen az átfutási idő egy tetszőleges, de választás után rögzített sorozatra igen eltérhet ugyanannak a munkadarab-sorozatnak egy más sorrendben gyártott variánsától.

A Johnson algoritmus segítségével a munkadarabok sorrendje kialakítható úgy, hogy a lehető legkisebb legyen a munkadarab-sorozat átfutási ideje (vagyis a *B* gépen jelentkező veszteségidő minimális). Tehát a módszer a megfogalmazott kétgépes alap *Flow Shop* ütemezési feladatra egzakt megoldást ad.

A Johnson algoritmus a következő lépések szerint működik:

- 1) egy táblázatba felvesszük a job-ok megfelelő gépekhez tartozó műveleti időit;
- 2) a táblázatból kiválasztjuk a legkisebb műveleti időt;
- 3) ha ez az első gépen van, akkor a sorrend elejére (a már korábban *előre* ütemezettek mögé),
ha a második gépen van, akkor a sorrend végére (a már korábban *hátra* ütemezettek elé) kerül a hozzá tartozó job;
- 4) töröljük a táblázatból az elhelyezett job sorát;
- 5) ha van még job a táblázatban, akkor folytatás a 2) ponttól;
- 6) a Johnson sorrend előállt.

A Johnson algoritmus speciális esetben optimális megoldást ad $m = 3$ gépes esetre. Ehhez az szükséges, hogy $\min(A_i) \geq \max(B_i)$ vagy $\min(C_i) \geq \max(B_i)$ teljesüljön, ahol B a középső gépet jelöli. Ha a feltételek közül legalább az egyik teljesül, akkor a 3 gépes problémát 2 gépesre vezetjük vissza úgy, hogy az i -edik job műveleti időit az $A_i + B_i$ és $B_i + C_i$ összegekre módosítjuk. Ha a feltételek egyike sem teljesül, akkor is alkalmazható a Johnson algoritmus az ismertetett módon, de ekkor csupán közelítő megoldást ad. Bár $m > 3$ esetén nem használható a Johnson algoritmus, mégis rendkívül fontos eljárás, mert alapját képezi a nagyobb feladatokhoz kifejlesztett heurisztikus módszereknek.

2.3.2. HEURISZTIKUS MEGOLDÁSOK

Az egyutas, előzés nélküli *Flow Shop* feladat megoldására $m > 2$ esetére számos heurisztikus módszert dolgoztak ki. Az alábbiakban öt ilyen közelítő eljárást ismertetek. ^[2]

Az átfutási idő csökkentésére a következő heurisztikus módszerek alkalmazhatóak:

- A leggyorsabb és legegyszerűbb a **Palmer** által kifejlesztett módszer, amelynek az a lényege, hogy minden egyes munkához prioritási indexet rendel és ezután a munkákat a prioritási indexek nem növekvő sorrendjében ütemezi. Az i -edik job prioritási indexének számítása a következő formulával történik:

$$I_i = - \sum_{j=1}^m \frac{m-(2j-1)}{2} t_{ij} ,$$

ahol m : a gépek száma; j : a gépek indexe;

i : a job-ok indexe; t_{ij} : az i -edik job műveletideje a j -edik gépen.

A formula azt az elvet fejezi ki, hogy ez a módszer azokat a munkákat veszi előre, amelyeknek a megmunkálási idői az első gépeken rövidebbek a többihez képest.

- A **Cambell-Dudek-Smith** (CDS) algoritmus a feladatot $(m-1)$ -számú kétgépes ütemezési problémára vezeti vissza, ahol a futóindex $k=1,2,\dots,m-1$, ekkor k és $k+1$ az aktuálisan vizsgált virtuális gépek sorszámai (a műveleti sorrend szerint rendre egymás után következő szomszédos gépeket tekint a két virtuális gépnek). Ez az algoritmus $m-1$ mesterséges kétgépes problémát generál az eredeti m gépes feladatból, amelyek egyenként megoldhatóak a Johnson-módszerrel. Kiválasztva közülük az átfutási idő szempontjából legjobbat, ez lesz az eredeti m -gépes probléma megoldása.

- **Dannenbring** dolgozta ki az úgynevezett gyors hozzáférés eljárását. Ez a heurisztikus módszer is egy súlyozási sémát használ és szintén a Johnson által megoldott kétgépes problémára vezeti vissza az m gépes feladat megoldását. A végrehajtási időket az alábbi súlyozási séma határozza meg:

$$P_{i1} = \sum_{j=1}^m (m - j + 1) t_{ij} \qquad P_{i2} = \sum_{j=1}^m j t_{ij}$$

ahol P_{i1} : az i -edik job műveleti ideje a kétgépes fiktív feladat első gépén;

P_{i2} : az i -edik job műveleti ideje a kétgépes fiktív feladat második gépén;

m : a gépek száma; j : a gépek indexe;

i : a job-ok indexe; t_{ij} : az i -edik job műveletideje a j -edik gépen.

Az eredeti feladat megoldása ezután a Johnson eljárásnak a fent definiált két fiktív gépre való alkalmazásával áll elő.

Az eddig ismertetett heurisztikus módszerek célfüggvénye az átfutási idő csökkentése volt, azonban vannak olyan módszerek, amelyek más célfüggvényeket vesznek figyelembe.

Legyen most a célfüggvény a legnagyobb csúszás minimalizálása. Ebben az esetben alkalmazható például a következő módszer:

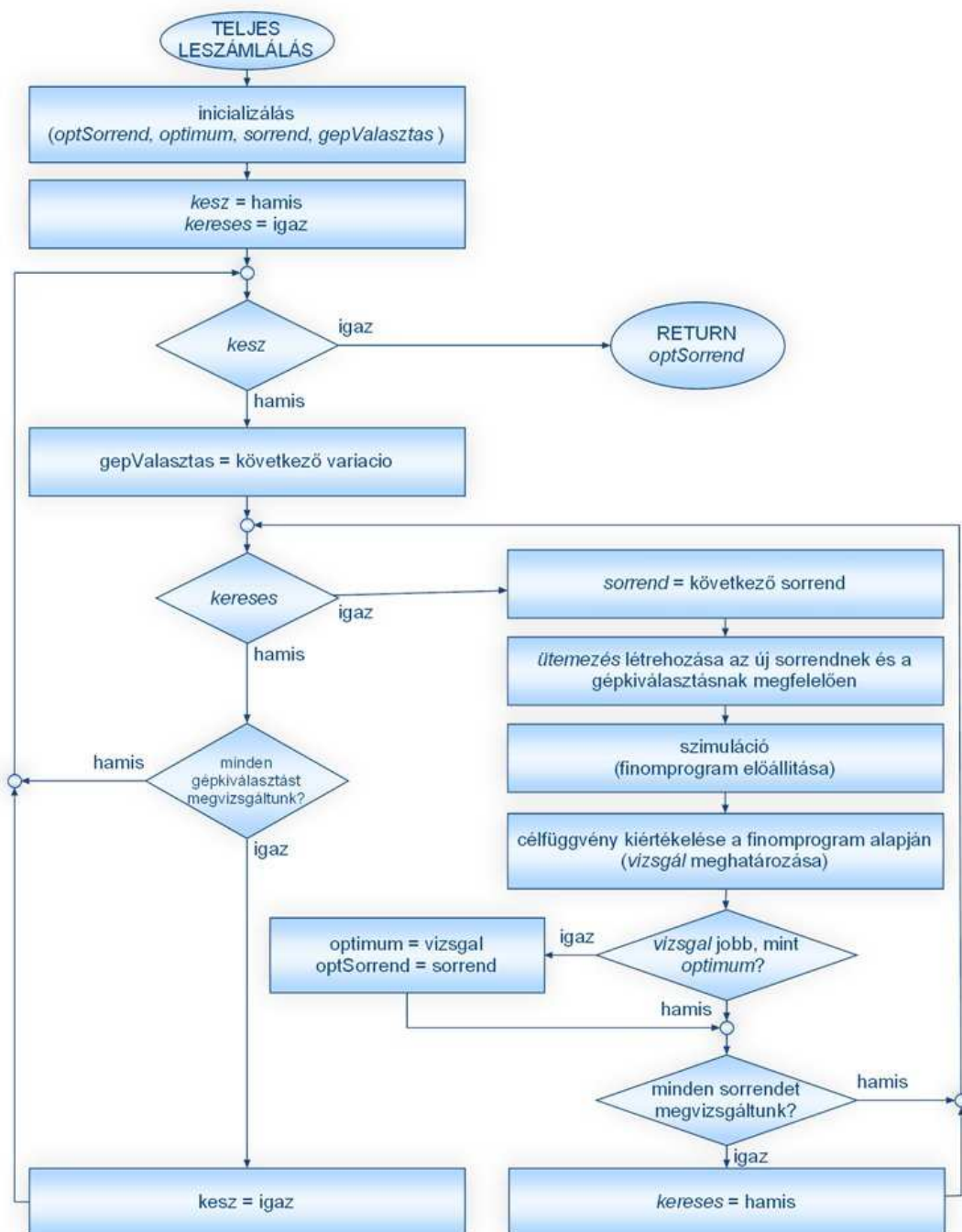
- Az **EDD** (Earliest Due Date) algoritmus működése nagyon egyszerű: a job-okat határidő szerint nem csökkenő sorrendbe rendezve kapjuk az ütemezési sorrendet. Ez a módszer csak egygépes esetben ad optimális megoldást. Többgépes esetben a kapott eredmény csupán közelítő megoldás lehet, mivel ha a várakozási idők nagyon megnövelik a job-ok átfutási idejét, akkor a határidő túllépése is hasonló mértékben nő. ^[6]

Az eddig ismertetett módszerek a munkák sorrendjének meghatározása során csupán a műveletidőket vagy a határidőket vették figyelembe. Ezek az ütemezési stratégiák az alap *Flow Shop* feladatok esetén jól közelíthetik az optimális sorrendtervét, de rugalmas feladatok esetén ugyanez már nem mondható el. Ahhoz, hogy egy rugalmas ütemezési feladat megoldásakor kapott sorrendterv jobban közelítse az optimális megoldást, figyelembe kell venni a feladat további jellemzőit, mint például az átszállítási és beállítási időket, az erőforrások szakaszos működését, az alternatív technológiai útvonalakat, stb.

A következőekben olyan módszereket ismertetek, amelyek már alkalmasak a rugalmas *Flow Shop* feladat kezelésére is.

2.3.3. TELJES LESZÁMLÁLÁS

A módszer működési elve nagyon egyszerű: minden alternatív technológiai útvonalat megvizsgálva előállítja a munkák minden lehetséges sorrendjét (az ütemterveket), és minden ütemtervre kiértékeli a célfüggvény aktuális értékét, végül pedig azt a választja megoldásként, amelyik legjobb célfüggvény értékkel rendelkezik. Ezeket a lépéseket szemlélteti a **4. ábra**.



4. ábra Teljes leszámolás

A módszer előnye, hogy bármilyen célfüggvény esetén megtalálja az optimális megoldást. Hátránya, hogy számításigénye nagy, mivel a módszer alapját az ismétlés nélküli permutáció képezi. Vagyis egyetlen technológiai útvonal és n darab munka esetén $n!$ számú esetet kell megvizsgálni. $n=10$ munkáig az esetek számát a **2. táblázat**ban láthatjuk.

n	1	2	3	4	5	6	7	8	9	10
$n!$	1	2	6	24	120	720	5.040	40.320	362.880	3.628.800

2. táblázat Az ütemezés lehetséges eseteinek száma a munkák számának függvényében

Ha a feladatban alternatív technológiai útvonalak is szerepelnek, akkor a végrehajtás során a munkák sorrendje megváltozhat (előzéses *Flow Shop* feladat). Így ha a munkákon o darab operációt kell végrehajtani és az o_i -edik operáció elvégzéséhez k_i darab gépet rendelünk ($i=1, 2, \dots, o$) ($\sum k_i > o$), akkor a vizsgálandó esetek száma $\prod k_i \cdot n!$ lesz.

A feladat kombinatorikus tulajdonságai miatt a módszernek az alkalmazása csak kis feladatok esetén előnyös. Érdemes megjegyezni, léteznek olyan módszerek, amelyek a kombinatorikus feladatok futási időinek csökkentésére törekszenek. Ilyen például a „Branch and Bound” (szétválasztás és korlátozás) megoldási koncepció, amely a leszámolás során bizonyos ágakat elhagyva csökkenti a futási időt.

2.3.4. MESTERSÉGES INTELLIGENCIA MÓDSZEREK

Az ütemezési feladatok megoldásának egy korszerű eszköze a mesterséges intelligencia módszereinek alkalmazása. Ezek közé tartoznak a keresési módszerek, mint például a szimulált hűtés és a tabu-keresés. ^[7]

Ezek lokális keresési módszerek, vagyis nem garantálják, hogy eredményük valóban egyezni fog az optimummal, de jó esélye van, hogy közel fog esni hozzá. A módszerek előnye, hogy rendelkeznek olyan változtatható paraméterekkel, amelyek megválasztása a keresés eredményét jelentősen befolyásolhatja. Ez a rugalmasság lehetővé teszi, hogy az optimum kereső módszert az ütemezési probléma nehézségének megfelelően állíthassuk be, így növelve a keresés hatékonyságát. Amikor hatékonyságról beszélünk, nem feltétlenül az optimum megtalálásának esélyét értjük alatta, ugyanis vannak olyan esetek, amikor az ütemező algoritmus gyorsasága is számít. Ilyenkor megelégszünk egy kvázi optimális megoldással is annak érdekében, hogy minél hamarabb használható megoldáshoz

jussunk. Ezért a gyakorlati alkalmazhatóság szempontjából az eddig bemutatott heurisztikus módszerekkel szemben jelentős előnyt élveznek a mesterséges intelligencia módszerek. További előnyük, hogy bármekkora feladaton, bármilyen jellemzők (például szakaszos működés) mellett, bármilyen célfüggvény esetén jól alkalmazhatóak.

- **Szimulált hűtés**

A módszer alapötletét egy, a fémöntészetben alkalmazott technika adta, innen ered a szimulált hűtés elnevezés is.

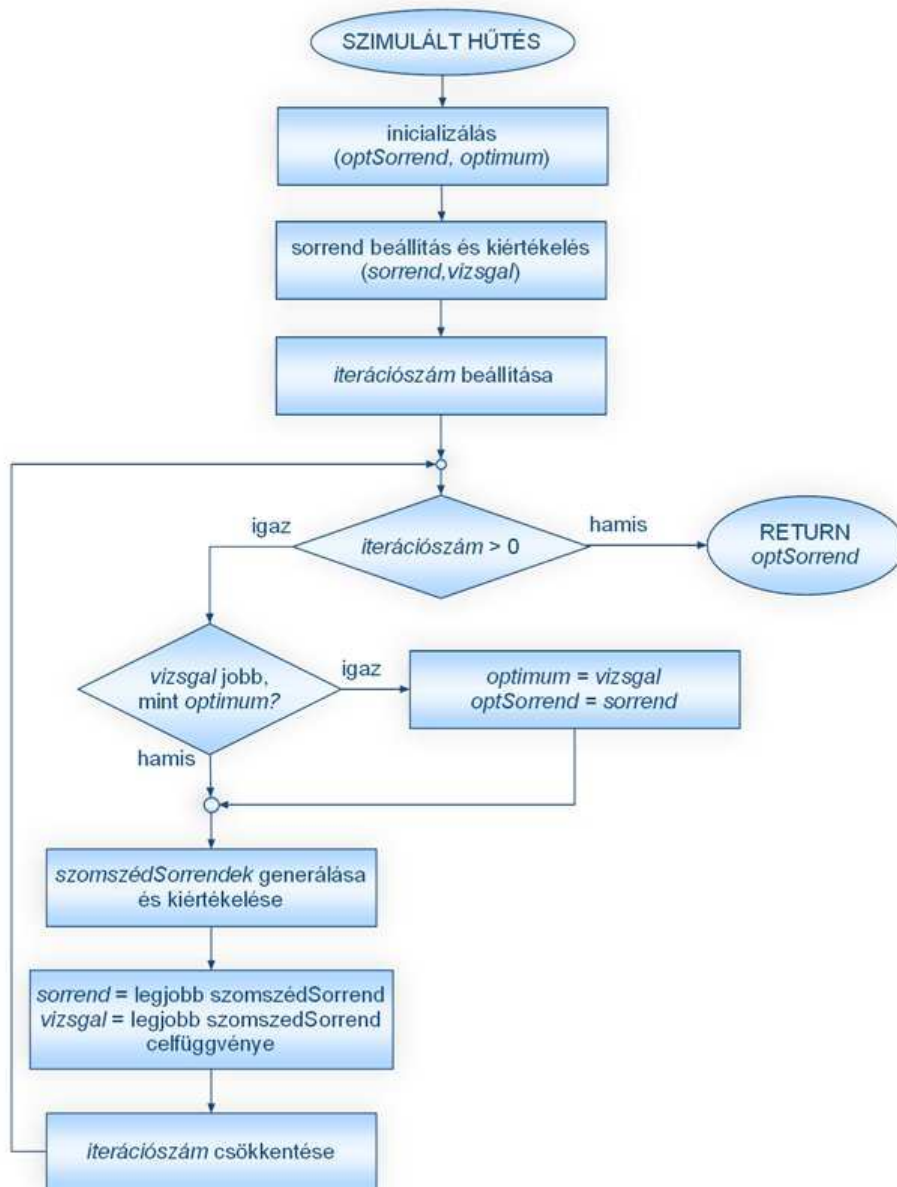
Az öntészet célja, hogy a fém alapállapotba kerüljön, vagyis az anyagi részecskék jól strukturált kristályrácsokba rendeződjenek (ekkor minimális a belső energia). Ezt az állapotot úgy éri el, hogy a fémot megolvasztják, ezáltal az anyagi részecskék szabad mozgását lehetővé téve, majd ezt követően megkezdődik a fém hűtése, aminek hatására az anyagi részecskék kristályrácsokba rendeződnek.

A technika kritikus fontosságú tényezői a kívánt eredmény elérése érdekében, hogy a fém elegendően magas hőmérsékletre legyen hevítve, valamint hogy a hűtés megfelelő lassúsággal történjen.

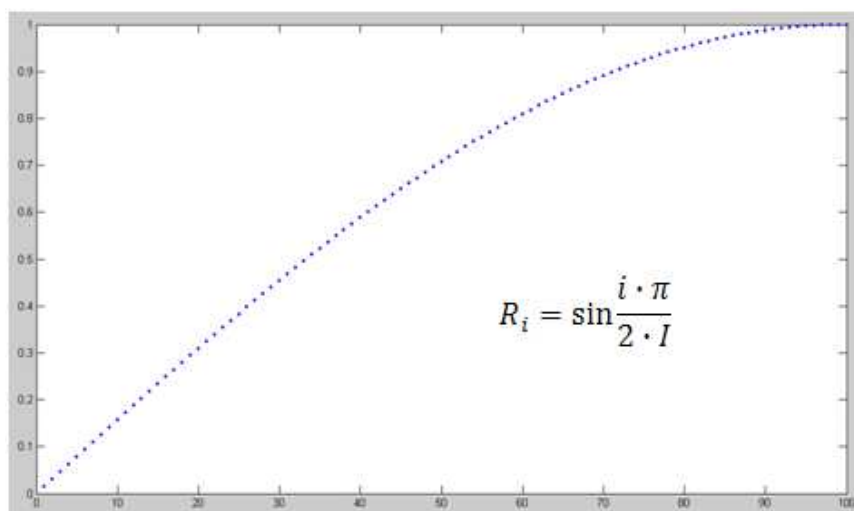
A szimulált hűtés algoritmusát az **5. ábra** szemlélteti.

A szimulált hűtés a lehetséges megoldások közül egy kiválasztott megoldás szomszédjait vizsgálva keresi a sorrendtervezés lehető legjobb megoldását. Ha az éppen vizsgált sorrend jobb célfüggvény-értékkel rendelkezik, mint az eddigi legjobb megoldás, akkor a vizsgált sorrend lesz az eddig talált legjobb megoldás, egyébként pedig a rontási együttható függvényében a módszer rontó lépést tehet, vagyis kiválaszthatja a rosszabb értéket, mint az eddig talált legjobb megoldást. Mivel keresés az aktuális megoldásnál rosszabbal is folytatódhat, ezáltal az is lehetségessé válik, hogy egy lokálisan optimális megengedett megoldásról a keresés tovább lépjen egy rosszabbra, hogy aztán néhány lépés megtétele után az összes eddiginél jobb megengedett megoldást találjon.

A rontási együtthatót meghatározó összefüggést célszerű úgy megválasztani, hogy az algoritmus kezdetben nagyobb valószínűséggel fogadjon el az aktuálisnál rosszabb megengedett megoldásokat is. Az általam implementált algoritmus a **6. ábrán** feltüntetett összefüggést alkalmazza a rontási együtthatók előállításához.



5. ábra Szimulált hűtés



6. ábra R_i értékek $I = 100$ esetén

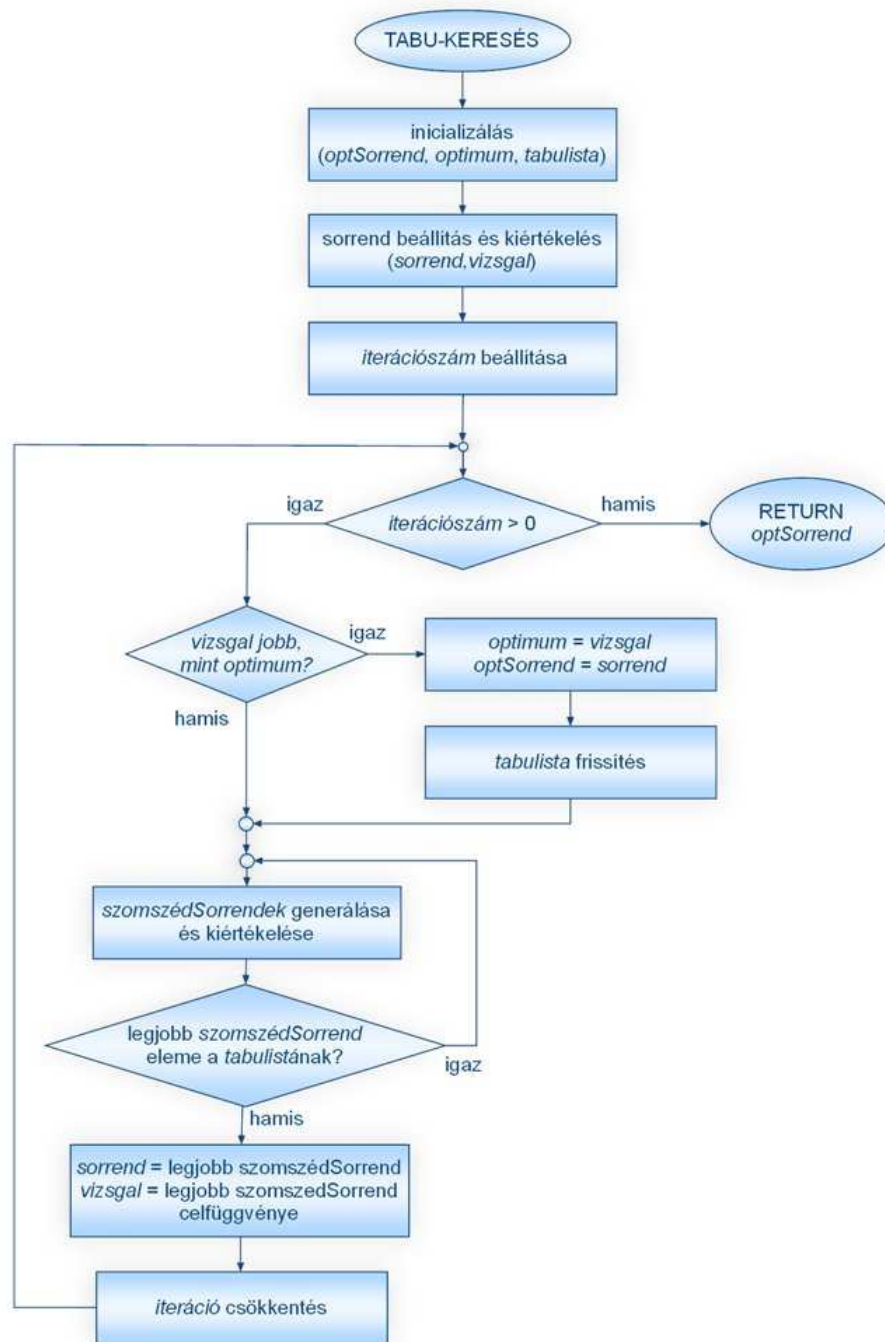
Az általam választott rontási együtthatók alakulását leíró függvény tulajdonképpen a szimulált hűtés iterációinak számához a szinusz függvény egy periódusának első negyedét rendeli. A szinusz függvény tulajdonságának köszönhető, hogy az iterációk első harmadának végrehajtása után már 0,5, a második harmad után már 0,85 fölötti eredményeket állít elő a függvény, vagyis az iterációk előrehaladásával a rontó lépés bekövetkezésének valószínűsége megfelelő mértékben csökken.

- **Tabu-keresés**

A gyakorlatban előszeretettel alkalmazzák ezt a módszert, mivel jól megválasztott paraméterek mellett nagyon hatékony működésre képes: nagy feladatok esetén is viszonylag gyorsan jó eredményt szolgáltat. Érdekesség, hogy erre a jelentős gyakorlati sikerre még nem szolgáltatott általános elméleti magyarázatot.

A tabu-keresés algoritmusát a **7. ábra** vázolja fel.

A módszer működése hasonlít a szimulált hűtés működéséhez, az alapvető különbség az úgynevezett tabu-lista megjelenése. A tabu-lista lehetőséget biztosít a lokális optimumból való továbblépésre, ugyanis ha a lokális optimum közelében nincs kedvezőbb célfüggvény-értékkel rendelkező megoldás, akkor a keresés a „rosszabb” szomszédos megoldás irányába is el tud mozdulni, ha a lokális optimum már szerepel a tiltást jelentő tabu-listán. A tabu-lista tulajdonképpen egy rövid-távú memória, amelyben a nem túl régen vizsgált lehetséges megoldások szerepelnek. Az aktuális megoldás mindig a lista elejére kerül. Mivel a tabu-lista hossza véges, ezért amikor új elemet adunk a listához egy már a listában lévő elem „kiesik”, vagyis a módszer felejt.



7. ábra Tabu-keresés

A vázolt keresési módszerek az ütemezési feladat lehetséges megoldásának keresésekor a vizsgált sorrend-tervekkel szimulációt végezve állítják elő az adott sorrendhez tartozó termelési finomprogramot, amely alapján az ütemezési feladat célfüggvényét kiértékelve hasonlítják össze a munkák sorrendjének vizsgált alternatíváit. Ez a szimulációs keresés teszi lehetővé, hogy a megvalósítástól függően tetszőleges, a rugalmas *Flow Shop* feladatokra jellemző paramétereket is figyelembe véve keressük az ütemezési probléma optimális vagy kvázi-optimális megoldását.

Szimuláció-alapú algoritmusok esetén fontos, hogy az ütemezési feladat méretének megfelelően paraméterezzük fel az alkalmazott módszert, mivel az ütemezési feladatok olyan kombinatorikus problémák, amelyek megoldásának időigénye erősen függ a megfogalmazott feladat munkáinak számától.

2.4. SZIMULÁCIÓ

A sorrendtervező algoritmusokon kívül érdemes még a szimuláció megvalósításával is foglalkozni, amely a munkák végrehajtásának tényleges időbeli meghatározásáért, vagyis a finomprogram előállításáért felelős.

Az eljárás viszonylag bonyolult, ugyanis a munkák sorrendjén, műveletein és a gépek foglaltságán kívül a végrehajtás során figyelembe kell venni az anyagmozgatási időket, a beállítási időket és azt, hogy a beállításhoz szükséges-e a munkadarab jelenléte, továbbá nem lehet figyelmen kívül hagyni az erőforrások szakaszos működését, a műveletek megszakíthatóságát és az alternatív technológiai útvonalakat sem.

A szimuláció során szükséges az indulástól (referencia időtől) eltelt időt, illetve az egyes gépek rendelkezésre állásának intervallumait is nyilvántartani. A helyes működés szempontjából elengedhetetlen ezeknek az időadatoknak a frissítése minden egyes operáció beütemezésekor.

Ennek az összetett szimulációs algoritmusnak a bemutatása előtt érdemes először az egyszerűbb, alap *Flow Shop* ütemezési feladat szimulációs algoritmusát ismertetni. Miután ez megtörtént, a megismert algoritmust lépésenként továbbfejlesztve jutunk el az implementált szimulációs eljárás leírásához.

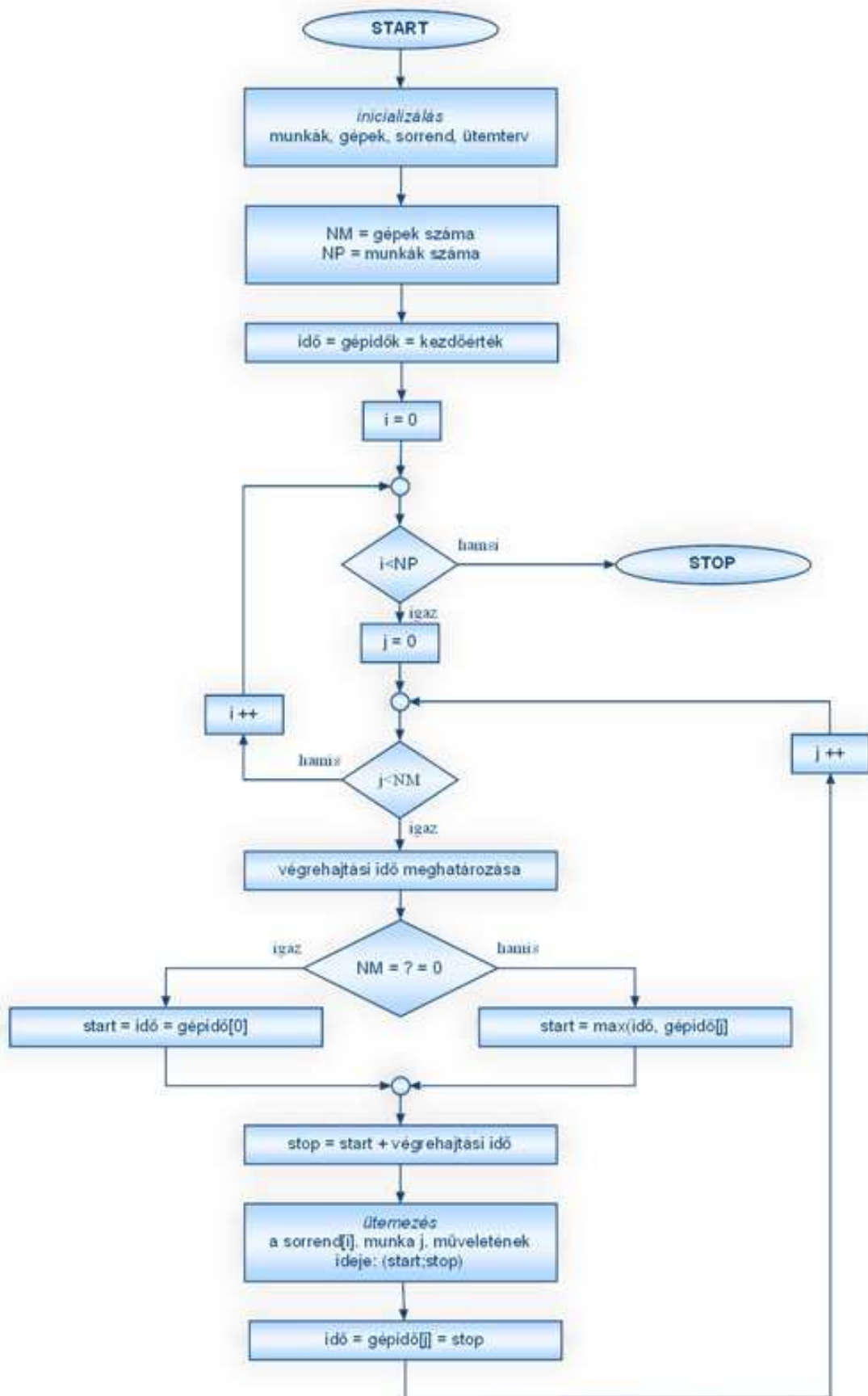
Az alap *Flow Shop* feladat szimulációs eljárása a munkákon egy kapott sorrend alapján halad végig úgy, hogy a munkák operációit egyesével vizsgálja a gépek sorrendjének megfelelően. Minden egyes operációhoz meghatározza a teljes végrehajtási időt (az alap feladat esetén ez az operáció műveletidejével egyenlő), majd pedig a szimuláció kezdetétől eltelt időt és az aktuális gép rendelkezésre állásának kezdeti idejét összehasonlítva kiszámítja az operáció kezdeti és befejezési időpontját is. Ezt követően beütemezi a vizsgált operációt és frissíti a szimuláció kezdete óta eltelt időt és az aktuális gép rendelkezésre állásának kezdeti idejét. Ezt a folyamatot szemlélteti a **8. ábra**.

Az alap feladat szimulációs eljárása egyszerű, könnyen átlátható. Egészítsük ki a feladatot beállítási és anyagmozgatási időkkel! Ez a módosítás nem bonyolítja jelentős mértékben az alap feladatot, csupán az operációk teljes végrehajtási idejének meghatározását befolyásolja. Ebben az esetben a végrehajtási időt a vizsgált operáció műveleti ideje, a műveletet végző gép beállítási ideje és az anyagmozgatási idő fogja meghatározni. A beállítási idő attól függ, hogy melyik gépen milyen típusú munkadarabról milyen típusú munkadarabra kell áttérni. Az anyagmozgatási idő attól függ, hogy melyik gépre milyen típusú munkadarabot kell átszállítani. A munkadarabok típusának szerepe a beállítási idők meghatározása esetén triviálisnak tekinthető (terméktípusonként eltérő szerszámok alkalmazása, eltérő munkadarab rögzítési módok, stb.). Az anyagmozgatási idők meghatározásánál a terméktípusok megkülönböztetését azok fizikai jellemzői és az anyagszállítást végző logisztikai rendszer indokolják: nem biztos, hogy minden terméktípus ugyanazon a módon (pl. targonca, szállítószalag, konvektor, stb.), ugyanazon az útvonalon (pl. egy nagyobb méretű termékkel nem lehet egy szűk, gyártósortok közötti folyosón közlekedni) és ugyanazzal a sebességgel mozgatható egyik megmunkáló helyről a másikra.

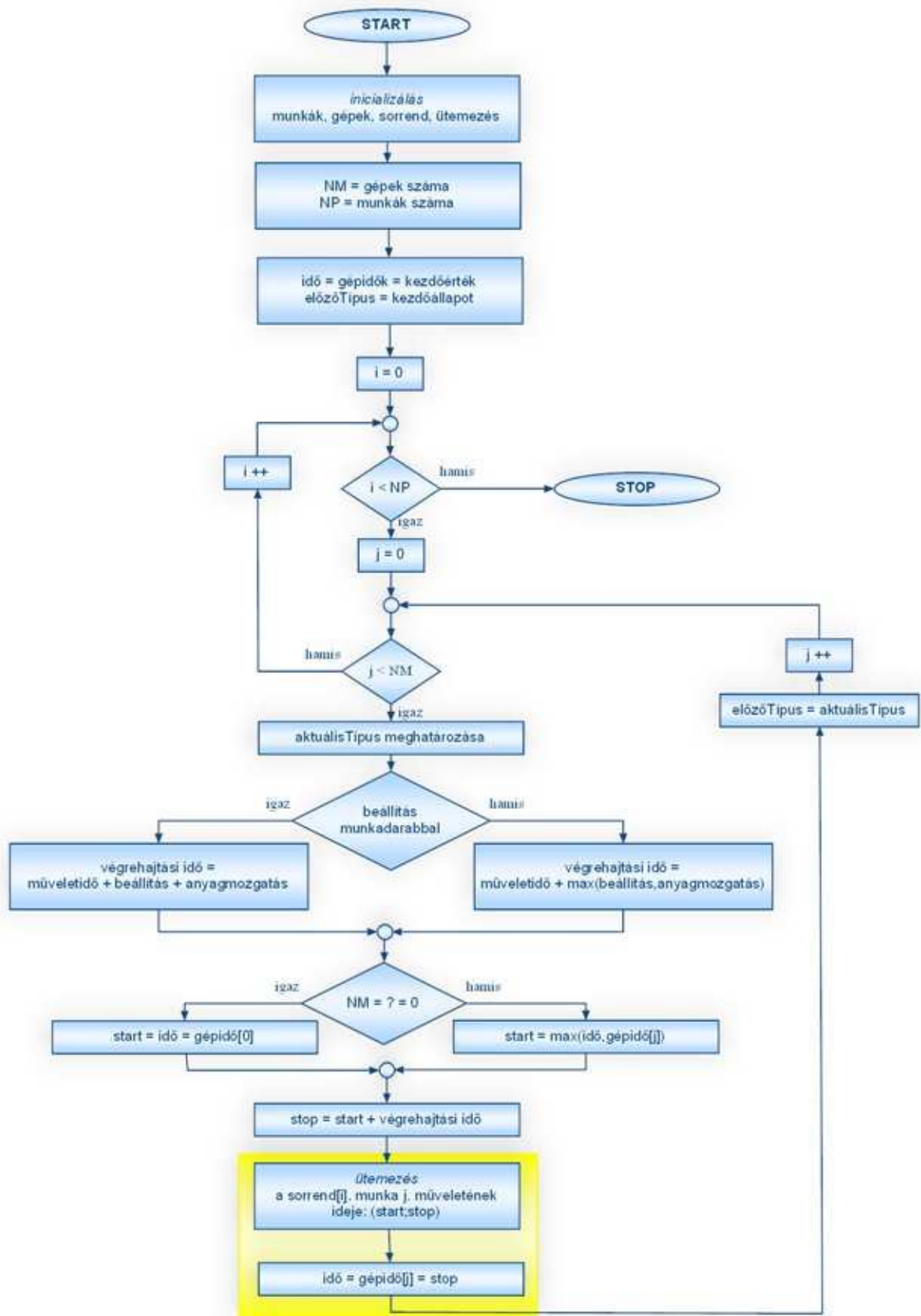
A kiegészített feladat megoldására szolgáló szimulációs módszer először az aktuális művelet teljes végrehajtási idejét határozza meg. Ez a következő módon történik:

- ha a beállításhoz szükséges a munkadarab jelenléte, akkor a teljes végrehajtási idő számításánál a beállítási és anyagmozgatási idők összegét kell képezni (mivel a beállítás csak a munkadarab megérkezése után kezdődhet el) és hozzáadni a műveletidőhöz,
- egyébként pedig a beállítási idő és az anyagmozgatási idő maximumát kell a műveleti időhöz hozzáadni.

Ezt követően az eljárás a szimuláció kezdetétől eltelt idő és az aktuális gép rendelkezésre állásának kezdeti ideje alapján meghatározza az operáció elvégzésének kezdeti és befejezési időpontjait. Végül pedig beütemezi a vizsgált operációt majd pedig frissíti a szimuláció kezdete óta eltelt időt és az aktuális gép rendelkezésre állásának kezdeti idejét. Ezt a folyamatot szemlélteti a **9. ábra**.



8. ábra: Az alap Flow Shop feladat szimulációja



9. ábra: Kiegészített Flow Shop feladat

A feladat kiegészítésének következő lépése az erőforrások szakaszos működésének a figyelembe vétele. Ez az eddigi algoritmusban a **9. ábrán** sárga színnel jelölt részek újratervezését jelenti. A módosított folyamatábra-részletet a **10. ábra** szemlélteti.

Ha a vizsgált gép megszakításokkal dolgozik, akkor figyelembe kell venni, hogy az adott munka műveletei megszakíthatóak-e, vagyis van-e lehetőség a végrehajtás megszakítása után az operációk egy későbbi időpontban történő folytatására.

Ha a műveletek végrehajtása megszakítható, akkor meg kell vizsgálni, hogy a művelet kezdési időpontja a vizsgált gép valamelyik szünet-időintervallumába esik-e. Ha igen, akkor a kezdési időpontot az adott megszakítási időintervallum végére kell beállítani, ennek megfelelően pedig újraszámítani a művelet befejezési időpontját. Ha a befejezési idő alapján a művelet befér az aktuális gép következő rendelkezésre állási időintervallumba, akkor beütemezhető a művelet. Egyébként, ha a vizsgált gépen a következő szünet korábban kezdődik, mint ahogy befejeződne az aktuális művelet, akkor a műveletvégzés teljes idejéből az a legnagyobb részidő kerül ütemezésre, amelyet a vizsgált gép rendelkezésre állása megenged. A művelet megszakítás miatt fennmaradó (részidővel csökkentett) teljes idejének beütemezését a gép következő rendelkezésre állási időintervallumán kell folytatni egészen addig, amíg a művelet teljes egészében beütemezésre nem kerül (vagyis az operáció hátralévő végrehajtási ideje 0 lesz).

Ha a műveletek végrehajtása nem megszakítható, akkor a vizsgált gép szüneteinek végighaladva meg kell vizsgálni, hogy a művelet kezdeti ideje beleesik-e valamelyik szünet-időintervallumba. Ha igen, akkor a művelethez tartozó kezdési időt az adott intervallum végére kell beállítani és újra kell számítani a művelet befejezési idejét is. Ha a kapott befejezési idő nem nagyobb, mint a vizsgált gépen a következő szünet kezdeti időpontja, akkor befér a vizsgált gép adott rendelkezésre állási időintervallumába, vagyis a művelet beütemezhető. Egyébként, pedig ha nem fér be a művelet, akkor a művelet kezdési időpontját a következő rendelkezésre állási intervallum kezdetére kell állítani és újraszámítani a művelet befejezési időpontját, majd pedig újra megvizsgálni, hogy az adott művelet befér-e az éppen vizsgált rendelkezésre-állási időintervallumba. Ezt a vizsgálatot addig kell folytatni, amíg az adott művelet beütemezésre nem kerül.

A feladat következő kiegészítéseként a munkák egyes műveleteihez tartozó státuszjelző kerül bevezetésre, amely meghatározza egy megadott művelet végrehajtási állapotát (befejeződött, folyamatban van, várakozik). Ennek nyilvántartása lehetővé teszi, hogy például egy félkész munkadarab a technológiai folyamat tetszőleges pontján beléphessen, vagy akár egy megkezdett termelési folyamat egy váratlan leállítását következtében újraütemezést után folytatódjon. Ez a változtatás a szimuláció algoritmusában úgy vehető figyelembe, hogy a már befejezett műveletek idejét 0-nak tekintjük, a többi műveletidőt pedig változatlanul hagyjuk.

A feladat utolsó kiegészítése az alternatív útvonalak figyelembevételének megvalósítása. Ennek egyik legegyszerűbb módja, hogy ha az eddigi technológiai útvonalakon szereplő fizikai gépeket virtuális gépekkel helyettesítjük. A virtuális gépek feladata nyilvántartani, hogy az adott művelet elvégzésére mely gépek vannak kijelölve, illetve szintén feladata ezek közül a gépek közül kiválasztani azt az egyet, amely ténylegesen el fogja végezni egy adott munkadarab adott műveletét. A műveletet elvégző gép kiválasztása többféle módon történhet, például kijelöléssel, vagy a gépek terheléseinek figyelembevételével, vagy akár egyszerű lista-algoritmussal (amikor a munkákat sorban osztjuk ki az egyes alternatív gépeken). A szimuláció algoritmusának szempontjából a virtuális gépek bevezetése nem igényel változtatást, mivel az alternatív gépek közötti választást a sorrendtervező módszerek valósítják meg.

3. RUGALMAS FLOW SHOP ÜTEMEZÉSI FELADAT SZÁMÍTÓGÉPES MEGOLDÁSA

A számítógépi program célja a megismert módszerek alapján tetszőleges inputadatokkal rendelkező rugalmas *Flow Shop* ütemezési feladatok megoldása, továbbá a kapott eredményeken keresztül az eljárások összehasonlítása egy kényelmes felhasználói felület segítségével. A program különböző módszerekkel támogatja az alap *Flow Shop* feladat megoldását, továbbá lehetővé teszi a modell kiterjesztését a következő jellemzők figyelembevételével:

- munkadarabonkénti határidők,
- munkák tetszőleges műveletnél történő beléptetése,
- munkák közötti előzés,
- alternatív technológiai útvonalak,
- szakaszos működésű erőforrások,
- terméktípustól függő beállítási idők az egyes gépeken,
- terméktípustól függő anyagmozgatási idők az egyes gépeken.

A kibővített ütemezési modellel megoldható feladatosztály:

$$FF / p_{ij}, s_{jik}, d_i, p_{mtn}, A_i / \sum C \text{ vagy } C_{max} \text{ vagy } \sum T \text{ vagy } T_{max}$$

A programmal a *Flow Shop* feladat megoldására alkalmas módszerek vizsgálhatóak. Összesen kilenc eljárást valósítottam meg, amelyek között az ad-hoc módszeren és a teljes leszámításon kívül öt heurisztikus és két mesterséges intelligencia módszer is szerepel.

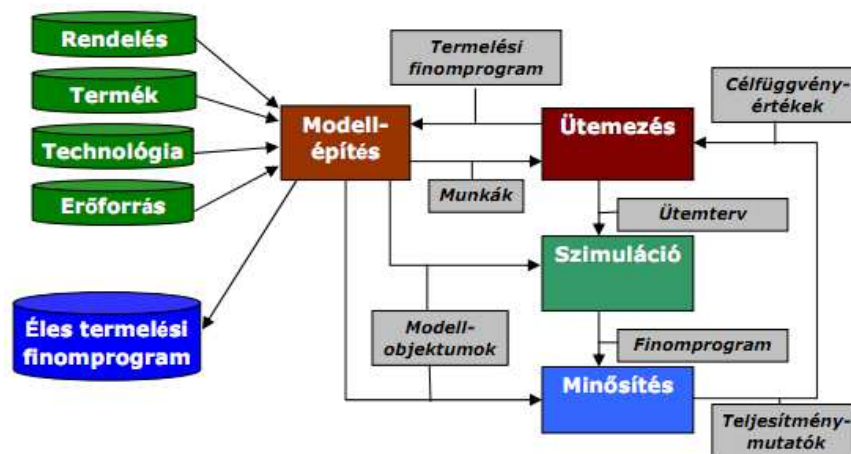
Érdemes kiemelten kezelni ezt a két csoportot és az általuk előállított megoldásokat vizsgálni, ugyanis az ütemezési feladatok megoldásában a mesterséges intelligencia módszereknek egyre hangsúlyosabb szerep jut. A heurisztikus eljárásokkal szemben ezek a módszerek a keresés során talált megoldás iteratív javításával igyekeznek minél kedvezőbb eredményt elérni. A kedvezőbb megoldás iteratív kereséséből adódik, hogy ezeknek a módszereknek az időigénye nagyobb, mint a heurisztikus módszereké. Azonban a heurisztikus ütemezési módszerek az erős modellfüggés miatt csak szűk problémakör esetén alkalmazhatóak, míg a mesterséges intelligencia módszerek könnyen továbbfejleszthetők és rugalmasan alkalmazhatóak bonyolult feladatra. Ez indokolja, hogy miért érdemes a mesterséges intelligencia módszereivel kiemelten foglalkozni és különböző feladatokon, különböző paraméterek használatával vizsgálni működésüket.

3.1. A SZÁMÍTÓGÉPES ALKALMAZÁS

A számítógépi alkalmazás fejlesztéséhez a Java programozási nyelvet választottam, ami olyan objektum-orientált program fejlesztését teszi lehetővé, mely hordozható, platform-független, robosztus, elosztott és többszálú. [8] [9] [10]

3.1.1. AZ ÜTEMEZŐ SZOFTVER FELÉPÍTÉSE

A Java nyelv objektum-orientáltsága lehetővé teszi, hogy az ütemező szoftver felépítése és az osztályok kialakítása az ütemezési feladat entitásait és a megoldás menetét tekintse mintának. Ezt szemlélteti a **11. ábra**.



11. ábra Termelési finomprogram előállítás [4] [6]

Az ütemezési feladatok megoldásának kritikus pontja a feladatnak megfelelő modell létrehozása. Ha ez megtörtént, akkor a következő lépés az ütemterv (vagyis itt a megadott munkák sorrendjének) meghatározása az adott erőforrásokon. Ezt követi a már meghatározott ütemterv alapján a termelési finomprogram előállítása (szimuláció), vagyis az egyes műveletek kezdési és befejezési időpontjának meghatározása a hozzájuk rendelt gépeken. Végül pedig az ütemezési feladat megoldásának minősítésére, a célfüggvényének kiértékelésére kerül sor. Az ütemezési, szimulációs és minősítési lépések szisztematikus, ismételt végrehajtásával egyre jobb megoldások érhetőek el.

A programban kifejlesztett eljárások megvalósítása során saját megoldás készítésére törekedtem. Ehhez nélkülözhetetlen volt a kiválasztott módszerek elméleti alapjainak megismerése és megértése. Ezeket az ismereteket felhasználva saját koncepció alapján dolgoztam ki a megoldási módszerek algoritmusait.

A leírtakat figyelembe véve a következő osztályokat alakítottam ki:

- **Machine:** a gépeknek megfeleltethető osztály, amely egy meghatározott gép paramétereit tárolja (azonosító, rendelkezésre állás, terméktípusonkénti beállításiidők és anyagmozgatási idők, szükséges-e a beállításhoz munkadarab jelenléte).
- **VirtualMachine:** az egyes műveletek elvégzésére alkalmas gépek halmazát tároló osztály.
- **WorkPiece:** a munkadaraboknak megfeleltethető osztály, amely meghatározott munkadarab paramétereit tárolja (azonosító, típus, határidő, műveletidők).
- **Schedule:** a termelési finomprogramok modellezésére alkalmas osztály, amely tartalmazza a munkadarabok sorrendjét és a műveletek kezdeti és befejezési időpontjait a hozzárendelt gépeken.
- **Order:** egységes keretbe foglalja az implementált sorrendtervezési módszereket (ad-hoc, EDD, Johnson, Palmer, CDS, Dannenbring, teljes leszámlálás, szimulált hűtés, tabu-keresés).
- **Simulation:** a termelési folyamat numerikus szimulációját megvalósító osztály, amely egy Schedule objektum sorrend attribútuma alapján a megfelelő szimulációs paraméterek figyelembevételével meghatározza a termelési finomprogramot (kiszámítja az időadatokat).
- **Goal:** az ütemezés célfüggvényének kiértékelését megvalósító osztály, amely egy előállított finomprogram alapján képes meghatározni a $C_{\max}$, ΣC , $T_{\max}$, vagy ΣT célfüggvény-értékeket.

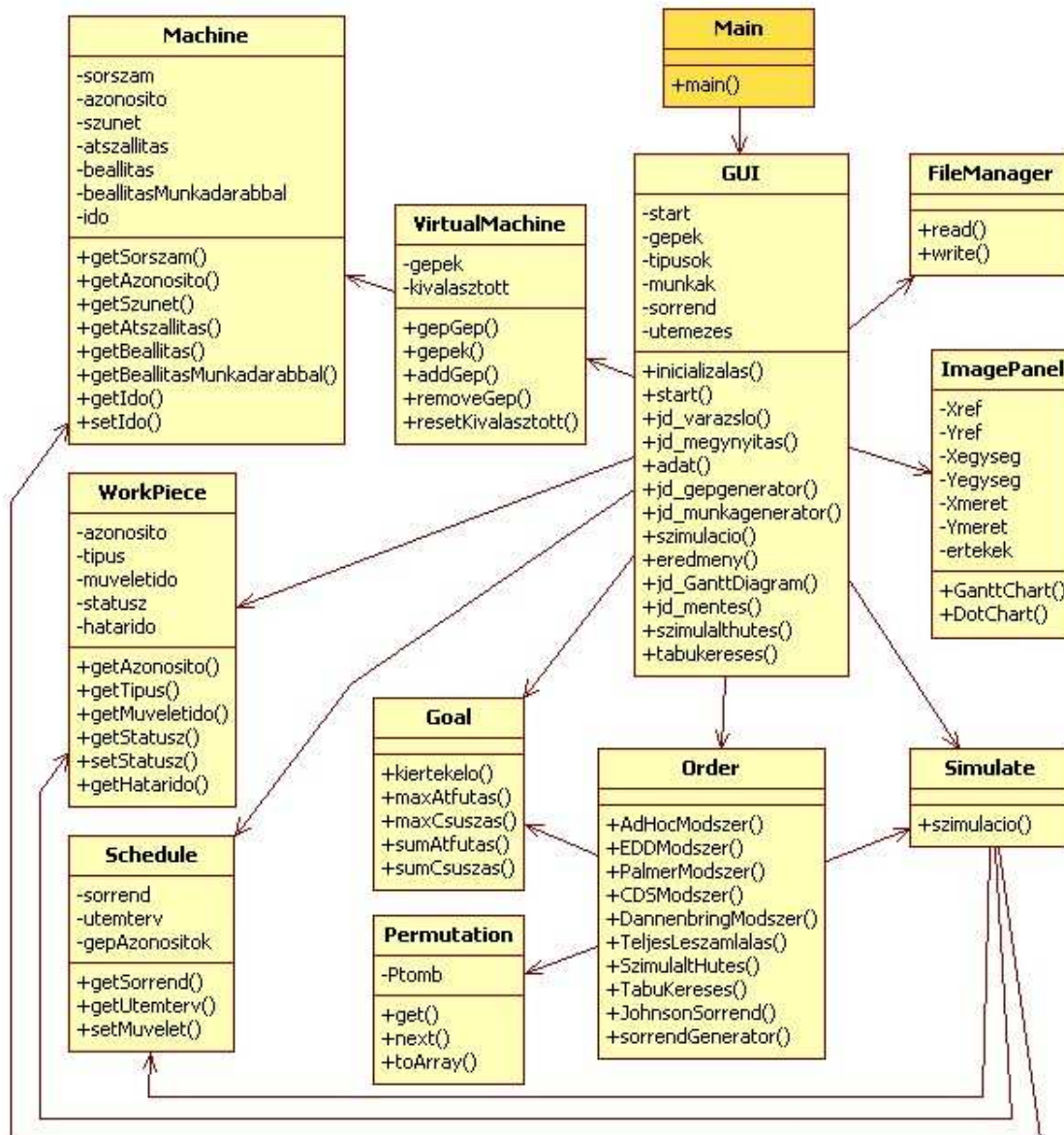
Ezekon az osztályokon kívül a következő osztályok kerültek még implementálásra:

- **Main** (a programot indító osztály),
- **GUI** (a helyes működést és grafikus felhasználói interfészt megvalósító osztály),
- **ImagePanel** (Gantt-diagram rajzoló osztály),
- **FileManager** (fájlkezelő osztály),
- **Permutation** (segédosztály permutáció előállítására).

Összefoglalva a program a következő forrásfájlokból épül fel:

Main.java, GUI.java, Machine.java, VirtualMachine.java, WorkPiece.java, Schedule.java, Order.java, Simulation.java, Goals.java, ImagePanel.java, FileManager.java, Permutation.java.

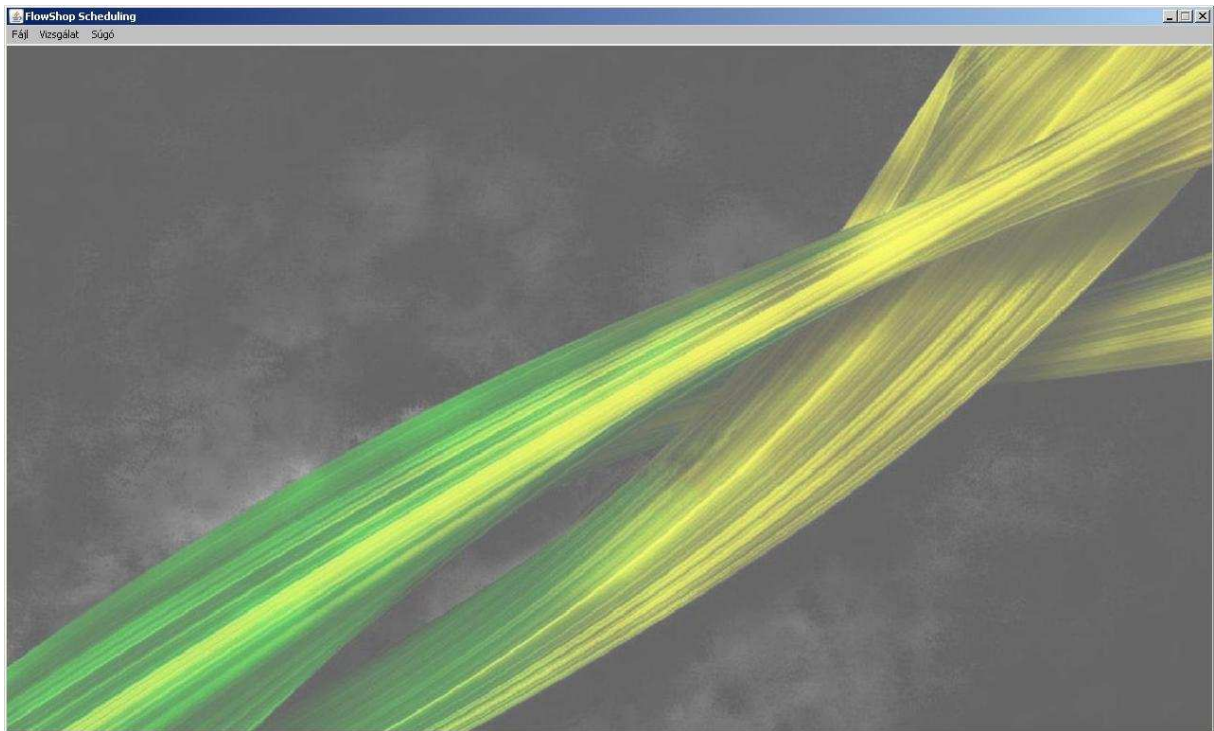
Ezeket az osztályokat és kapcsolataikat ábrázolja a **12. ábra**. Ez az osztálydiagram a fejlesztés tervezési szakaszában felvázolt koncepciót tükrözi. A későbbi tervezés és fejlesztés során az osztályok pontosításra és bővítésre kerültek (ez kimondottan a GUI osztályra igaz, amely a grafikus felület megvalósítása miatt több adattaggal és eseménykezelő metódussal is kiegészült). Ezek a módosítások az átláthatóság érdekében nem szerepelnek az osztálydiagramon.



12. ábra Kezdeti osztálydiagram

3.1.2. AZ ÜTEMEZŐ SZOFTVER BEMUTATÁSA

Ebben a fejezetben az általam készített program azon funkcióját fogom bemutatni, amely az implementált ütemezési módszerek összehasonlítására szolgál.



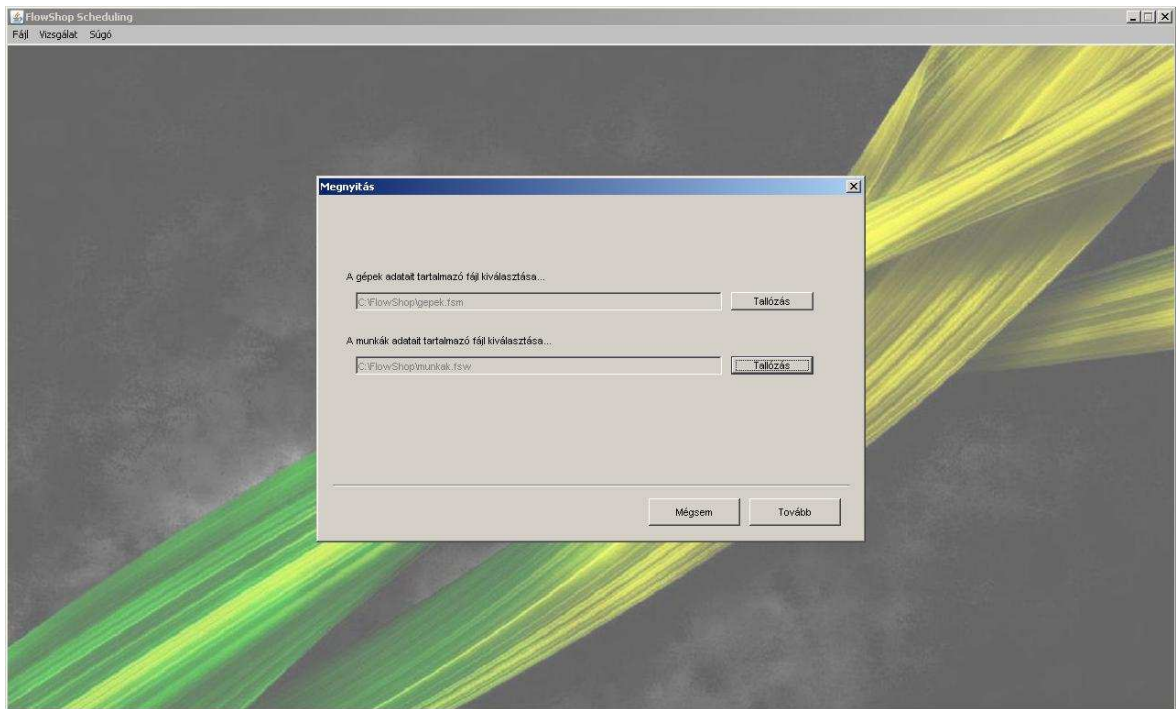
13. ábra: Kezdő képernyő

A programot elindítva a **13. ábrán** látható kezdő képernyő jelenik meg. A program menüjének felépítése a következő:

- Fájl (*módszer-összehasonlító funkció*)
Projekt létrehozása, Projekt megnyitása, Kilépés
- Vizsgálat (*mesterséges intelligencia módszereket vizsgáló funkció*)
Szimulált hűtés, Tabu-keresés
- Súgó
Felhasználói kézikönyv, Névjegy

A **14. ábrán** a *Projekt megnyitása* menüpont kiválasztása esetén felugró ablak látható, amelyen a *Tallózás* gombokra kattintva lehet megadni a gépek és munkák adatait tartalmazó inputfájlokat. A program a következő fájltypusokkal dolgozik:

- FSM (Flow Shop Machines; .fsm kiterjesztéssel)
- FSW (Flow Shop Workpieces; .fsw kiterjesztéssel)



14. ábra: Projekt megnyitása

Egy FSM fájl tartalmazza a gépek adatait, továbbá azt is, hogy az adott gépállomány mely terméktípusok megmunkálására alkalmas. A megmunkálható terméktípusok a fájl első sorában kerülnek felsorolásra. Ezt követően soronként történik a gépek adatainak részletezése. Egy rekord felépítése a következő:

operáció sorszáma | gép azonosítója | terméktípusok száma | szünetek száma | szünetek kezdeti és befejezési idői sorfolytonosan | anyagmozgatási idők terméktípusonként | átállási idők a terméktípusok függvényében sorfolytonosan | beállításhoz szükséges-e a munkadarab | gép rendelkezésre állásának kezdőideje

Egy FSW fájl soronként tartalmazza a munkák adatait. Egy rekord felépítése a következő:

munka azonosítója | típusa | műveletek száma | műveletidők | műveletek státusz jelzői | határidő

Ha minden munkadarabra teljesül, hogy típusa szerepel a szimuláció végrehajtásához választott FSM fájlban felsorolt típusok között, akkor a két fájl kompatibilis adatokat tartalmaz. Ha ez a feltétel nem teljesül (vagyis nem kompatibilisek a fájlok), akkor a szimuláció végrehajtása nem lehetséges, ugyanis a munkadarabok között van olyan típus, amelynek megmunkálására egyik gép sem alkalmas.

Amikor egy tallózó gombra kattintunk, a program alapértelmezett könyvtára kerül kilistázásra (C:\FlowShop). Attól függően, hogy milyen típusú fájl megnyitásáról van szó, csak az annak megfelelő kiterjesztésű elemek és az alkönyvtárak jelennek meg.

A *Tovább* gombra kattintva történik meg az adatok betöltése a programba. Ahhoz, hogy ez ténylegesen megtörténhessen, a kiválasztott fájloknak létezőknek és kompatibiliseknek kell lennie. Ha ezek a feltételek nem teljesülnek, akkor a program egy hibaüzenettel értesít a problémáról. A program hasonlóan hibaüzenettel reagál arra az esetre is, ha úgy akarunk továbblépni, hogy nem adunk meg inputfájlt.

A megadott input fájlok sikeres megnyitása esetén a **15. ábrán** látható felület jelenik meg.

15. ábra: Gépek adatai

A *Gépek* fül felületének felső részén található a feladatban résztvevő gépek listája. Ebben a listában a gépek adatainak kiírása az input fájlok egy sorának megfelelő alakban történik.

A felület alsó részén van lehetőség a gépek adatainak felhasználó-közelibb áttekintésére és szerkesztésére. Ha a gépek listájában kijelölünk egy elemet, akkor ez az adatszerkesztő rész feltöltődik a választott gép adataival.

A *Művelet* legördülő menü segítségével adható meg, hogy az adott gép a munkák hányadik operációjának végrehajtására van kijelölve.

Az *Azonosító* mező tartalmazza a gép egyedi azonosítóját, amelyet csak számok és betűk alkothatnak.

A *Rendelkezésre állás kezdeti ideje* mezőben adható meg az az időpont, amikortól a gépre terheléseket ütemezhetünk.

A *Beállítás munkadarabbal* legördülő menü segítségével adató meg, hogy az adott gépen szükség van-e a munkadarab jelenlétére ahhoz, hogy a gép beállítása elvégezhető legyen. Ha a beállításhoz szükség van a munkadarab jelenlétére az adott gépen, akkor az „Igen”, egyébként pedig a „Nem” opciót kell választani.

A *Szünetek elhanyagolása* és a *Szünetek megadása* rádiógombok segítségével megadható, hogy az adott gép szakaszos működésű vagy sem. Amennyiben a második lehetőség van kiválasztva, akkor lehetőség van a gépek szünet-időintervallumainak szerkesztésére (amikor a gép nem áll rendelkezésre). Új időintervallum felvételéhez a kezdeti és befejezési időpontok megadása után a *Hozzáad* gombra kell kattintani. Meglévő időintervallum módosításához a módosítandó szünet listaelemet kijelölve annak adatainak szerkesztése után a *Módosít* gombra kell kattintani. Meglévő időintervallum törlése pedig a listaelem kijelölése után a *Törlés* gombra kattintva tehető meg. Az időintervallumok szerkesztésére szolgáló mezőkbe csak számok írhatóak. Ha az intervallum megadásakor az első szám nagyobb a második megadott számnál, akkor a *Hozzáadás* vagy a *Módosítás* gomb megnyomása után hibaüzenetet jelenik meg.

A *Beállítási idők elhanyagolása* és a *Beállítási idők megadása* rádiógombok segítségével adható meg, hogy az adott gépen figyelembe vesszük-e a beállítási időket. Az utóbbi lehetőséget választva szerkeszthetjük a beállítási időket. Kizárólag számok írhatóak ezekbe a mezőkbe. Ebben a négyzetes mátrixnak megfeleltethető táblázatban terméktípusonként adhatóak meg a beállítási idők értékei. A sorok és oszlopok száma eggyel több, mint a terméktípusok száma, ennek oka, hogy a táblázat első sora a kiinduló (kezdeti) állapotról történő beállítást, az első oszlop pedig a befejezési állapotba történő beállítást jelölik.

Az *Anyagmozgatási idők elhanyagolása* és az *Anyagmozgatási idők megadása* rádiógombok segítségével adható meg, hogy az adott gépen figyelembe vesszük-e a tranzitidőket. Ebben az oszlopvektornak megfeleltethető táblázatban terméktípusonként adhatóak meg a tranzitidők, vagyis az az idő, amely alatt az adott terméktípusú

munkadarab az előző operáció végrehajtása után a megmunkálás helyéről az aktuális gépre megérkezik. Kizárólag számok írhatóak ezekbe a mezőkbe.

A felület alsó részén végzett műveletek véglegesítéséhez a gépek listája melletti gombokat kell alkalmazni (felső rész). A *Hozzáadás* gomb hatására a gépek listája a szerkesztett adatoknak megfelelően bővül. A *Módosítás* gomb hatására a gépek listájában kijelölt elem a szerkesztett adatoknak megfelelően változik. A *Törlés* gomb hatására a gépek listájában a kijelölt elem törlésre kerül (egyszerre csak egy).

Ha az adatok szerkesztése során hiányosan töltjük ki a mezőket, vagy a gép megadott azonosítója már foglalt, akkor a *Hozzáadás* vagy a *Módosítás* gombra történő kattintás után hibaüzenetet kapunk.

Abban az esetben, amikor egyetlen gép sem szerepel a listában, akkor a *Törlés* és a *Módosítás* gombok nem elérhető funkciókká válnak.

A *Generátor* gombra kattintva érhető el a program azon funkciója, amely tetszőleges számú gép adatait véletlenszerűen generálja a gépeket jellemző paraméterek generálási határértékeinek megadása alapján.

A munkák adatainak megtekintésére és szerkesztésére a *Munkadarabok* fület választva van lehetőségünk. Ezt a felületet mutatja be a **16. ábra**. Ennek a felületnek a felépítése hasonló a *Gépek* fül felületéhez, vagyis a felső részben a feladatban résztvevő munkadarabok kerülnek kilistázásra, az alsó részben pedig lehetőségünk van ezek szerkesztésére. Ebben a listában a munkák adatainak kiírása az input fájlok egy sorának megfelelő alakban történik.

Abban az esetben, ha a feladat egyetlen munkát sem tartalmaz, akkor a *Módosít* és *Töröl* gombok nem elérhető funkciókká válnak. A *Hozzáad* és a *Generál* funkciók mindig elérhetőek maradnak.

Ha kijelölünk egy elemet a listában, akkor a felület alsó részén található adatszerkesztő rész feltöltődik a választott munka adataival.

Az *Azonosító* mező tartalmazza a munka egyedi azonosítóját, amelyet csak számok és betűk alkothatnak.

A *Terméktípus* legördülő menüben választható ki az adott termék típusa. Ebben a listában azok a terméktípusok szerepelnek, amelyek megmunkálására alkalmasak a feladatban felvett gépek.

A *Határidő elhanyagolása* és a *Határidő megadása* lehetőségekkel adhatjuk meg, hogy figyelembe kívánjuk-e venni az egyes munkák határidőit a feladatban. Határidő megadása esetén az arra kijelölt mezőbe kell beírni a kívánt értéket (csak szám lehet).

A *Műveletidők* részénél van lehetőség szerkeszteni egy adott munka műveletidőit. Az oszlopvektornak megfeleltethető táblázatban annyi műveletidő-elem szerepel, ahány gépet előír a betöltött feladat. A műveletidők megadásakor minden mező csak számjegyeket tartalmazhat.

A *Műveletek státusza* részénél adható meg az egyes műveletek végrehajtásának állapotai. A negatív értékek a befejezett műveleteket, a pozitív értékek a még hátralévő, várakozó műveleteket, a 0 pedig az éppen aktuálisan futó műveletet jelölik.

A *Hozzáad* vagy a *Módosít* gombra kattintva, ha hiányosan töltöttük ki az adatszerkesztő részt, akkor hibaüzenetet kapunk, egyébként megtörténik a szerkesztett adat hozzáadása vagy módosítása annak megfelelően, hogy melyik lehetőséget választottuk.

The screenshot shows the 'FlowShop Scheduling' application window. It has a menu bar with 'Fájl', 'Vizsgálat', and 'Súgó'. Below the menu is a tabbed interface with 'Gépek' and 'Munkadarabok' tabs. The main area contains a list of jobs with their details:

Job	Terméktípus	Határidő elhanyagolása	Határidő megadása
w1	T1	☒	30
w2	T2	☐	
w3	T1	☐	
w4	T3	☐	
w5	T1	☐	

On the right side, there are buttons: 'Generátor', 'Hozzáad', 'Módosít', and 'Töröl'. At the bottom, there are two columns of input fields for 'Műveletidők' and 'Műveletek státusza'.

16. ábra: Munkák adatai

A *Generátor* gombra kattintva érhető el a program azon funkciója, amely tetszőleges számú munka adatait véletlenszerűen generálja a munkákat jellemző paraméterek generálási határértékeinek megadása alapján.

A szimulált hűtés és a tabu-keresés módszerei további paraméterek megadását igénylik ahhoz, hogy a módszerek a feladat méretének megfelelően skálázhatóak legyenek. Az *Iterációk száma* határozza meg, hogy az adott módszer hány alkalommal hajtja végre az optimum és az aktuális célfüggvény-értékének összehasonlítását. Az *Iterációként vizsgált esetek száma* adja meg, hogy az algoritmusok az új, vizsgálandó sorrend meghatározásához hány szomszédos sorrendet állítsanak elő. A *Kezdő sorrendnél* felsorolt módszerekkel állítható elő az iteratív javítás kezdő sorrendje. A *Sorrend újragenerálásnál* pedig a szomszédos sorrendek előállítására szolgáló függvények közül választhatunk. A választható módszerek a következők:

- 1. módszer: felcserél két véletlenszerűen választott elemet
- 2. módszer: véletlenszerűen választott részintervallumot középpontosan tükröz
- 3. módszer: előállítja a sorrend egy permutációját
- 4. módszer: véletlenszerűen választott részintervallumot léptet balra a kilépő elem jobb oldalra történő visszahelyezésével
- 5. módszer: véletlenszerűen választott részintervallumot léptet jobbra a kilépő elem bal oldalra történő visszahelyezésével

A tabu-keresés még egy további paraméter megadását írja elő, az pedig a *Tabu-lista hossza*, vagyis a módszer rövid távú memóriájának mértéke.

Ezeknél a kereső algoritmusoknál megadandó paraméterek csak egész számok lehetnek.

Az ütemező módszerek kiválasztásán kívül ezen a felületen van lehetőség az ütemezési feladat célfüggvényének megadására is. A következő célfüggvények választhatóak:

- $C_{\max}$ *A legnagyobb átfutási idő legyen minimális!*
- ΣC *Az átfutási idők összege legyen minimális!*
- $T_{\max}$ *A legnagyobb csúszás legyen minimális!*
- ΣT *A csúszások összege legyen minimális!*

Abban az esetben, amikor az ütemezési feladatban nem minden munka rendelkezik határidővel, akkor az utóbbi két, a munkák csúszására vonatkozó célfüggvény nem alkalmazható.

Ugyanezen a felületen adható meg *A műveletek végrehajtására* vonatkozó előírás, vagyis az, hogy a gépek szakaszos működése esetén a munkák operációi megszakíthatóak vagy sem.

Azokban az esetekben, amikor a feladatban alternatív technológiai útvonalak adottak, elérhetővé válik a *Műveletek gépcsoportonkénti kiosztása* funkció is, amellyel megadható, hogy egyazon művelet végrehajtásához hozzárendelt több alternatíva közül hogyan történjen a munkát ténylegesen végrehajtó gép kiválasztása. Itt két opció közül választhatunk aszerint, hogy figyelembe vesszük-e a gépek aktuális terheléseit vagy sem.

Ezen a felületen, ha a jobb alsó sarokban lévő *Vissza* gombra kattintunk, akkor az ütemezési feladatok adatait megjelenítő felületre jutunk vissza. Ha pedig a *Tovább* gombra kattintunk és minden előírt adatot megadtunk - vagyis választottunk célfüggvényt és végrehajtási módot, valamint szükség esetén kiosztási módszert is, továbbá ha mesterséges intelligencia módszert is alkalmazunk a sorrendtervezésnél, akkor annak minden paraméterét állítottuk -, akkor továbbjutunk az ütemezési feladat eredményeit összesítő felületre, amelyet a **18. ábra** jelenít meg.

INDÍTÁS				
Ad-hoc módszer	Sorrend: w1 w2 w3 w4 w5	Célfüggvényérték: 105	Ad-hoc módszer ütemterve	0m 0.046s
EDO módszer	Sorrend: w3 w5 w4 w2 w1	Célfüggvényérték: 109	EDO módszer ütemterve	0m 0.0s
Johnson módszer	Sorrend:	Célfüggvényérték:	Johnson módszer ütemterve	(...)
Palmer módszer	Sorrend: w1 w2 w5 w3 w4	Célfüggvényérték: 95	Palmer módszer ütemterve	0m 0.0s
CDS módszer	Sorrend: w5 w2 w1 w3 w4	Célfüggvényérték: 95	CDS módszer ütemterve	0m 0.0s
Dannenbring módszer	Sorrend: w1 w2 w3 w4 w5	Célfüggvényérték: 105	Dannenbring módszer ütemterve	0m 0.0s
Szimulált hűtés	Sorrend: w3 w5 w1 w2 w4	Célfüggvényérték: 95	Szimulált hűtés ütemterve	0m 0.032s
Tabu keresés	Sorrend: w1 w2 w3 w5 w4	Célfüggvényérték: 95	Tabu-keresés ütemterve	0m 0.0s
Teljes leszámolás	Sorrend: w1 w2 w3 w5 w4	Célfüggvényérték: 95	Teljes leszámolás ütemterve	0m 0.031s

Vissza Befejezés

18. ábra: Eredmények

Amikor megjelenik ez a felület, akkor az egyes ütemezési módszerekhez tartozó elemek (feliratok, mezők, gombok) még nem elérhetőek. Az ütemezési feladat megoldása a felület felső részén található *INDÍTÁS* feliratú gombbal kezdhető el.

Ha a módszerek végrehajtása befejeződött, akkor elérhetővé válnak a hozzájuk tartozó grafikus elemek, továbbá megjelennek az adott módszerrel elért eredmények

(sorrendek, célfüggvény-érték, futásidő) a megfelelő területeken. Ezen adatok alapján az egyes módszerek akár célfüggvény-érték, akár futásidő alapján is összehasonlíthatóak.

Egy adott módszerrel előállított sorrend Gantt-diagrammjának megtekintése a választott módszer sorában lévő ütemterv gombbal érhető el. Ennek hatására egy felugró ablakban jelenik meg a megfelelő diagram a **19. ábrán** látható módon.



19. ábra: Gantt diagram

A diagramon jól láthatóan vannak jelölve az egyes gépek rendelkezésre állásának időintervallumai (szürke téglalapok), továbbá az operációk végrehajtásának ábrázolásában is megkülönböztetésre kerültek a műveletek előkészítési – vagyis a tranzit és beállítási idők - folyamatai (sárga téglalapok) és a tényleges művelet-végrehajtások (zöld téglalapok).

A Gantt-diagram ábrázolásához saját diagramrajzoló készítettem. Azért döntöttem a saját algoritmus használata mellett, mert a fejlesztés során több, mások által elkészített diagramrajzoló osztályt kipróbálva egyikkel sem voltam igazán megelégedve, ugyanis vagy nem megfelelően voltak paraméterezhetők az egyes tengelyek, vagy - főleg nagyobb feladatok esetén – átláthatatlanok voltak az előállt diagramok. Továbbá egyik osztállyal sem tudtam megvalósítani a gépek rendelkezésre állásának, a munkák előkészítésének illetve tényleges végrehajtásának egységes és átlátható jelölését.

Az általam kialakított diagramrajzoló más diagramrajzolókkal összehasonlítva az általam támasztott kritériumok alapján egyetlen hátránnyal rendelkezik, mégpedig azzal,

hogy még nincs olyan nézete, amelyen a teljes Gantt-diagram látható lenne. A fejlesztés során próbálkoztam olyan függvények létrehozásával, amelyek az átméretezett diagramot jelenítenék meg. Az átméretezés megvalósítása sikerült, de a grafikát megvalósító komponensek nagy ábrák esetén jelentős mértékben torzultak. (Ez a funkció a funkció a dolgozat időpontjában még továbbfejlesztés alatt áll.)

A Gantt-diagramot megjelenítő ablakot bezárva az eredményeket tartalmazó felület jobb alsó részén található *Vissza* gombra kattintva ismét a szimulációs beállításokat tartalmazó felületre jutunk, ahol szükség esetén módosíthatjuk a beállított értékeket. A *Befejezés* gombra kattintva megjelenik egy dialógusablak, amely felkínálja a vizsgált ütemezési feladat input (gépek és munkák) adatainak mentését.

Az ütemezési módszerek összehasonlításának funkcióján végigérve még annyival egészíteném ki a leírtakat, hogy a vizsgálat során nem csak fájlból betöltött adatokkal dolgozhatunk. A *Fájl* menüben az *Projekt létrehozása* lehetőséget választva megjelenik egy felugró ablak, amelyben meg kell adni a feladatban részt vevő *Gépek számát* és a velük megmunkálható *Termékféléseket*. Ez követően a *Tovább* gombra kattintva a már ismertetett, gépek és munkák adatainak szerkesztésére alkalmas felület jelenik meg.

3.2. TESZTFELADATOK KIÉRTÉKELÉSE

A számítógépes alkalmazás konkrét tesztfeladatokkal történő futtatása során három célt határoztam meg:

1. ismert eredményekkel rendelkező tesztesetek segítségével alátámaszom a saját fejlesztésű algoritmusok alkalmazhatóságát és hatékonyságát;
2. tesztfeladatok segítségével összehasonlítsam az implementált ütemezési algoritmusok által szolgáltatott eredményeket;
3. továbbá a tesztfeladatok paramétereinek változtatásával vizsgáljam a feladat megoldásában bekövetkező változásokat.

Tesztelés során Erik Taillard benchmark adatait használtam fel, amelyek a következő honlapon érhetőek el:

<http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html>

- 5 gép, 20 munka	- 10 gép, 20 munka	- 20 gép, 20 munka
- 5 gép, 50 munka	- 10 gép, 50 munka	- 20 gép, 50 munka
- 5 gép, 100 munka	- 10 gép, 100 munka	- 20 gép, 100 munka
	- 10 gép, 200 munka	- 20 gép, 200 munka
		- 20 gép, 500 munka

Az itt található benchmark fájlok 10-10 tesztet tartalmaznak. Ahhoz, hogy ezekkel az adatokkal vizsgálni tudjam a programom működését, először a program által használt fájlformátumokat kellett előállítanom. Ezt a konverziót egy külön erre a célra írt program segítségével valósítottam meg.

Az ütemezési feladatok célfüggvénye a teljes átfutási idő csökkentése (C_{max}). Taillard összefoglalta és az említett honlapon publikálta a tesztadatokkal a valaha elért legjobb eredményeket, amelyek kiváló referenciaértékként szolgáltak saját implementált módszereim hatékonyságának vizsgálata szempontjából.

Az előállított tesztesetekből néhányat kiemelve a **3. táblázatban** összefoglalt eredményeket kaptam. A vizsgálat során a módszereket egyetlen alkalommal és tetszőleges paraméterekkel hajtottam végre. Ezzel az volt a célom, hogy demonstráljam azt az esetet is, amikor a vizsgált mesterséges intelligencia módszer futási paramétereinek beállítása során a feladat jellemzőit nem vesszük figyelembe.

FELADAT	Legjobb eredmény	Legjobb heurisztika	Szimulált hűtés	Tabu-keresés
T_05M-020W_0	1278	1381	1278	1278
T_05M-050W_0	2724	2774	2724	2724
T_05M-100W_0	5493	5730	5493	5493
T_10M-020W_0	1582	1771	1583	1592
T_10M-050W_0	2991	3461	3057	3063
T_10M-100W_0	5770	6161	5836	5785
T_10M-200W_0	10862	11382	10953	10974
T_20M-020W_0	2297	2591	2315	2300
T_20M-050W_0	3771	4272	3944	3950
T_20M-100W_0	6106	7075	6450	6464
T_20M-200W_0	11152	12673	11571	11507
T_20M-500W_0	26040	28131	26831	26791

3. táblázat: Teszteredmények

A kapott eredményeket összehasonlítva az ismert eddigi legjobb eredményekkel kijelenthető, hogy az implementált heurisztikus algoritmusok célfüggvény-értékei kisebb vagy nagyobb mértékben, de minden esetben rosszabbak az ismert legjobb eredményeknél. Ez a tény alátámasztja azt a korábbi megállapítást, miszerint a heurisztikus módszerek önmagukban nem hangolhatóak az ütemezési feladatokra, így nem alkalmasak nagyobb méretű vagy kiterjesztett ütemezési feladatok hatékony megoldására. Ezzel ellentétben a mesterséges intelligencia módszerek minden alkalommal jobb célfüggvény-értéket adtak, mint a heurisztikus módszerek. Voltak esetek, amikor a véletlenszerűen megválasztott futtatási paraméterek mellett a szimulált hűtés és a tabu-keresés megtalálta az eddig ismert legjobb célfüggvény-értéket. Azokban az esetekben, amikor nem sikerült ezt az értéket előállítani az egyszeri futtatás során, akkor is az ismert legjobb megoldás közvetlen közelében maradtak a kapott értékek. A vizsgálatok során a mesterséges intelligencia módszerek esetén a tesztfeladatok ismert legjobb eredményétől vett célfüggvénybeli átlagos eltérés 1.7% (a tapasztalt legnagyobb eltérés 5.5%), míg a heurisztikus módszerek esetén az átlagos eltérés 8.7% (a tapasztalt legnagyobb eltérés 13.7%) volt. Az elvégzett vizsgálatok azt a korábbi feltevést is alátámasztották, hogy a mesterséges-intelligencia módszerek paraméterei jelentősen befolyásolhatják az eredményeket.

Az elemzés során különböző ütemezési feladatokat megoldva nem csak az figyelhető meg, hogy az alkalmazott mesterséges intelligencia módszer az adott futási paraméterek mellett milyen célfüggvény-értékű megoldásokat talál, hanem azok szórásából is következtethetünk a módszer adott paraméterek melletti stabilitására. Stabilitás alatt egy olyan változót értek, amellyel kifejezhető, hogy az adott paraméterek mellett a módszert végrehajtva az eredményül kapott célfüggvény-érték milyen valószínűséggel esik egy meghatározott célfüggvény-intervallumba.

A futásidő szempontjából elmondható, hogy a heurisztikus módszerek minden esetben nagyon gyorsan lefutottak, a mesterséges intelligencia módszerek időigénye viszont érzékelhetően nagyobb volt, de megfelelően megválasztott keresési paraméterek mellett nagy feladatok esetében is a gyakorlat szempontjából kívárható időkorláton belül maradtak.

Az ütemezési feladatok megoldásának vizsgálatakor a munkák teljes átfutási idejét tekintve érdemes az eredményül kapott ütemtervet is tovább elemezni, ugyanis felismerhetőek lehetnek a technológiai útvonalon olyan szakaszok, amelyek szűk keresztmetszetet képezve növelhetik a teljes sorozat átfutási idejét. Ezeknek a szűk

keresztmetszeteknek a kiküszöbölésére szolgálhat egy új erőforrás, vagyis alternatív technológiai útvonal hozzáadása a feladathoz, amellyel megvalósítható a szűk keresztmetszetre eső terhelések elosztása, így az adott helyen az áteresztőképességet növelve a rákövetkező erőforrások (munkafolyamatok) veszteségidejét is csökkenthetjük, ezzel tovább növelve a termelés hatékonyságát.

Ideális megoldás lehet szoftver segítségével felderíteni a lehetséges szűk keresztmetszetek előfordulásának helyét, illetve vizsgálni az alternatív útvonalak hozzáadásának hatását a teljes sorozat átfutási idejének szempontjából. Az általam fejlesztett számítógépes alkalmazás azonban erre az automatikus felismerésre jelenlegi formájában még nincs felkészítve, viszont lehetőséget biztosít az ütemezési feladat manuális módosítására, tehát képesek vagyunk újabb gépek hozzáadásával alternatív technológiai útvonalak kialakítására, így vizsgálva annak a teljes sorozat átfutási idejére gyakorolt hatását. A szoftver tehát szimulációra alapozott döntéstámogató eszközként is használható.

Abban az esetben, ha helyesen ismerjük fel az esetlegesen előforduló szűk keresztmetszeteket, és ennek a felismerésnek megfelelően adunk hozzá a feladathoz alternatív gépeket, akkor a feladat újraütemezését követően a sorozat teljes átfutási idejében javulás következhet be. Azonban előfordulhatnak olyan esetek, amikor a feladatban szereplő gépek halmazának módosítását követően nem tapasztalunk változást a sorozat átfutási idejében. Ilyen esetekben a feladat korlátfeltételeit (mint például az egyes gépek rendelkezésre állása) vizsgálva vonhatunk le feladat-specifikus következtetéseket, amelyek alapján dönthetünk további szűk keresztmetszetek vizsgálatára, vagy a már bevezetett alternatív technológiai útvonalak hanyagolása mellett.

3.3. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Az ütemezési feladatok sokfélesége a program továbbfejlesztésére több lehetőséget is kínál. Ha csak a *Flow Shop* ütemezési feladat megoldásának továbbfejlesztésére gondolunk, akkor is számos lehetőségünk adódik a feladat kiterjesztése szempontjából: korlátozott műveletközi tárolók figyelembevétele, a munkák prioritással rendelkeznek, sorrendi korlátozás vonatkozik a munkák végrehajtására, több célfüggvény egyidejű figyelembevétele, szűk keresztmetszetek automatikus meghatározása és kiküszöbölése, stb.

A feladat továbbfejlesztésekor célszerű figyelembe venni a gyakorlati alkalmazás által támasztott igényeket, amelyek alapján az ütemező szoftver képessé tehető arra, hogy éles ütemezési problémákat megoldva ipari felhasználásra kerüljön. A szoftver jelenlegi formájában rugalmas *Flow Shop* modellel leírható ipari környezetben használható, ahol a párhuzamos gépek szakaszos működtetése kulcsszerepet játszik (tipikus példa erre a kézi és félautomata szerelősorokat üzemeltető végszerelőműhely). A szoftver a könnyen kezelhető grafikus felhasználói felületének, beépített problémagenerátorának és szerkesztőjének köszönhetően alkalmas oktatási eszközként a bemutatott típusú ütemezési feladatok és megoldási módszerek bemutatására, vizsgálatára.

Távlati célkitűzésem egy olyan általános számítógépes alkalmazás kialakítása, amellyel több ütemezési modell is kezelhető és megoldható. Ennek az általánosított modellnek és megoldásainak kialakítása során szintén kulcsfontosságú a gyakorlatban történő alkalmazás követelményeinek feltárása. Ennél a fejlesztésnél már célszerű a lehetséges felhasználási környezeteket is figyelembe venni és annak megfelelően kialakítani az alkalmazást. Célszerű a programot úgy továbbfejlesztetni, hogy az többretegű WEB-alkalmazási keretbe helyezve szolgáltatás formájában valósítsa meg a termelés ütemezését (megoldó motor). Ezáltal jelentősen növelhető az ütemezési feladatok központi megoldását megvalósító és a szüksége egyéb szolgáltatásokat támogató modulok fejlesztésének szétválasztása. Adatbázis-kezelővel - gyártásirányító rendszerrel vagy vállalatirányítási rendszerrel - történő együttműködés szintén fontos új funkció kidolgozását jelentheti.

ÖSSZEFOGLALÁS

Egy termelővállalat működését több cél együttese határozza meg, amely – a korlátfeltételekkel együtt – egy nagyon összetett optimalizálási problémát határoz meg. Ennek a problémának a megoldásához a kiinduló feladat lebontásaként több optimalizálási részfeladat megoldása válik szükségessé. Ilyen részfeladat többek között a termelés ütemezése is. Diszkrét termelési folyamatok esetén a munkadarabokon végzett műveletek és a szükséges erőforrások egymáshoz rendelése, továbbá a műveletek sorrendjének és időbeli lefutásának pontos meghatározása a komplex sokparaméteres kombinatorikus optimalizálási feladatok körébe tartozik.

Az alkalmazási területtől függően az ütemezési feladatok sokfélék és igen változatosak lehetnek. Egy-egy kiválasztott ütemezési modellt tekintve számos eljárás létezik „viszonylag jó” lehetséges megoldás előállítására, azonban ezek közül csak nagyon kevés eredményez optimális vagy optimum-közeli megoldást. Az ütemezési feladat méretének növelésével vagy az ütemezési modell kiterjesztésével az egzakt és az egyszerű heurisztikus módszerek egyre kevésbé képesek a feladat hatékony megoldására. Ilyen NP-hard problémák esetében célszerű bevetni az optimum-közeli megoldások megtalálását hatékonyan támogató mesterséges intelligencia módszereket.

Dolgozatomban az ütemezési feladatot, mint diszkrét termelési folyamattal kapcsolatos kombinatorikus optimalizálási problémát közelítettem meg és a termelésütemezés egy jellegzetes modelljén, a *Flow Shop* ütemezési osztályon keresztül mutattam be. Ezen ütemezési feladatok megoldásának vizsgálatára saját számítógépes alkalmazást fejlesztettem, melynek segítségével a különböző input adatokon végzett vizsgálatok alapján megállapítottam, hogy az implementált sorrendtervező módszerek közül tetszőleges feladat esetén a legjobb megoldást a szimulált hűtés és a tabu-keresés volt képes előállítani kivárható időn belül. Ez egyértelműen igazolta a mesterséges intelligencia módszerek jelentőségét az ütemezési feladatok megoldásában.

A termelés hatékonyságának növelése érdekében - az ütemezési problémakörön túlmutatva, de mégis szoros kapcsolatban vele - dolgozatomban rámutattam az alternatív technológiai útvonalak bevezetésének a termelési célfüggvényre gyakorolt hatására is.

IRODALOMJEGYZÉK

- [1] Tóth Tibor: Termelési rendszerek és folyamatok, Miskolci Egyetemi Kiadó, 2004
- [2] Tóth Tibor: Tervezési elvek, modellek és módszerek a számítógéppel integrált gyártásban, Miskolci Egyetemi Kiadó, 2006
- [3] Vizvári Béla: Diszkrét optimalizáció
<http://compalg.inf.elte.hu/~tony/Elektronikus/Informatikai/I9E.pdf>
- [4] Kulcsár Gyula: Ütemezési modell és heurisztikus módszerek az igény szerinti tömeggyártás finomprogramozásának támogatására, PhD értekezés
http://ait.iit.uni-miskolc.hu/~kulcsar/nyilvanos_vedes/KulcsarGy_ertekezes.pdf
- [5] A termelésinformatika alapjai c. tárgy órai jegyzete
- [6] Diszkrét termelési folyamatok számítógépes tervezése és irányítása c. tárgy órai jegyzete
- [7] Futó Iván: Mesterséges intelligencia, Aula Kiadó, 1999
- [8] Angster Erzsébet: Objektumorientált tervezés és programozás
- [9] Nyékyné Gaizler Judit (szerk.) et al: Java 2 Útikalauz programozóknak 5.0
- [10] Java Platform, JSE6 API Specification
<http://download.oracle.com/javase/6/docs/api/>
- [11] Brucker, P.: Scheduling Algorithms, Springer-Verlag, 1998

KÖSZÖNETNYILVÁNÍTÁS

A bemutatott TDK munka a TÁMOP-4.2.1.B-10/2/KONV-2010-0001 jelű projekt részeként az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.