Csoportos technológiák

Molnár Tamás
Eötvös Loránd Tudományegyetem
Alkalmazott matematikus szak

Témavezető:
Vizvári Béla
Operációkutatás Tanszék

2001. július 5.

Kivonat

Ipari üzemek tervezésekor gyakran felmerül az az igény, hogy a gépek fizikai elhelyezkedése tükrözze a gyártandó termékeken végzendő műveletek miatt közöttük fellépő természetes kapcsolatokat. Műszaki és gazdasági okok miatt sokféle gyakorlati szempont felmerülhet, ami alapján egyes gépcsoportokat összetartozónak nyilváníthatunk. A dolgozat áttekinti, miért szükséges ilyen csopotosításokat végezni, majd azt, hogy a probléma már egyszerű alakjában is matematikailag nehéz. A továbbiakban bemutatja a feladat megoldására jelenleg alkalmazott elterjedt ötleteket, és végül néhány gyakorlatban megvalósított módszer vizsgálatával felméri ezek felhasználhatóságának korlátait, egymáshoz viszonyított hatékonyságát. A 6. fejezet számpéldákon szemlélteti az algoritmusokat.

Tartalomjegyzék

1. A feladat eredete

A modern ipari üzem gyakran szembesül valamely gyártható terméke iránti kis mennyiségre szóló kereslettel, amely kielégítése után bizonyos idővel újra generálódik, miközben a kereslet eltérő, de hasonló termékek felé fordul. A vállalatnak a fennmaradása, profittermelő képessége megtartása érdekében készen kell állnia bármelyik potenciális termékének gyártására. Ha két termék a gépek egymástól diszjunkt halmazain készül, akkor nyilván előnyös igény esetén ezeket a termékeket egyszerre legyártani, mert a vállalatok közti versenyben a gyorsaság döntő tényező lehet. Ha lehetséges, akkor egy-egy gépcsoport tagjait célszerű egymás közelébe helyezni, mert ezzel elkerülhető például, hogy a félkész termékek egymás gyártását akadályozzák. Viszont nem mindig könnyen látható, van-e lehetőség ilyen felosztásokra, illetve egyenesen rá lehetünk kényszerítve, hogy mindig több terméket gyártsunk egyszerre, amelyeknek esetleg többféle lehetséges gyártási útvonala van, és az útvonal(ak) nem teszik lehetővé egymástól elkülönített gépcsoportok kialakítását. Ilyenkor is hasznos azonban legalább közelítőleg megkísérelni, mert így is az üzemen belüli szállítás jelentős része megtakarítható. Ezzel a gyártási folyamat olcsóbb, gyorsabb és könnyebben konrollálható lesz. A kialakított struktúra alapul szolgálhat rugalmas gyártó rendszerek tervezéséhez, konkrét ütemezések előállításához. Ha egyes termékek többféle módon is legyárthatók, akkor elkerülhetők az olyan helyzetek, amikor egyes gépek túlterheltek, míg mások állnak, és egyenletes terhelés valósítható meg. Egyes modellek magukba foglalják a gépek számának megváltoztathatóságát, így ezek arra is alkalmasak, hogy egy új üzem telepítésekor a rendelkezésre álló, kívánalmaknak megfelelő lehetőségek közül a legolcsóbbat keressék. Egyébként a modellek alkalmasak akár egy új üzem belső szerkezetének megtervezésére, akár egy meglévő üzem átszervezésére. A termékek adott gépcsoporthoz rendelése lehetőséget nyújt annak világos körülírására, hogy ki miért felelős. Ezáltal biztosítható a folyamatos és ellenőrzött működés, mert minden terméket figyelemmel kísérnek, és a vállalati döntéshozó számára is adott, bizonyos u-ta-sí-tá-so-kat kinek kell eljuttatnia. A csoportos technológia filozófiáját megvalósító vállalatok jelentős költségmegtakarításról és hatékonyságnövekedésről számolnak be. Egyesült államokbeli megfigyelések szerint a termelés be-in-dí-tá-sá-nak fixköltsége 20-60, a szerszámköltségek 20-30, a selejtarány 15-75 és a gyártási műveletekkel szorosan kapcsolatos egyéb költségek 20-50 százalékkal csökkentek. Egy vállalat saját felmérése 32 százalékos eladásnövekedést, 44 illetve 83 százalékos csökkenést mutatott ki a raktárkészletekben és a késedelmes teljesítéseiben.[1] Sajnos a feladat egyrészt NP-nehéz, másrészt a probléma mé-re-té-nek növekedésével az elvégzendő számítások mennyiségének gyors növekedése miatt a feladat közvetlenül nem támadható. Nem véletlen tehát, hogy a csoportos technológiákkal kapcsolatos kutatások napjainkban is intenzíven folynak, amit a csaknem minden egyes hónapban megjelenő új publikációk is jeleznek.

2. Egzakt modellek

Ebben a fejezetben bemutatásra kerül néhány modell, amelyek megoldása a bennük megfogalmazott követelmények tekintetében optimális megoldást ad. A konkrét igények alapján sokkal többféle modell kifejlesztésére lehet szükség, mint ahány itt előfordul, de ezek legtöbbje is felépíthető az itt szereplő ötletek alapján.

2.1. Általános jellemzők

A modell feltételeinek ki kell fejeznie, hogy itt különböző alternatívák közötti választásokról van szó. Emiatt természetes a bináris változók megjelenése. A gépek számára vonatkozó változók a jelentésünél fogva csak egész értékkel értelmesek. A cél egy költségfüggvény (pénzbefektetés, időköltség, szállítási költség) minimalizálása. A probléma kezelése tehát egy egészértékű programozási feladat megoldását igényli. A gépcsaládokat mindig ki kell alakítanunk, a termékcsaládokat nem feltétlenül, mert ha a gépeket már beosztottuk, és minden termékre van egy választott útvonal, akkor már adott, hogyan kell termelni. A termékcsaládok kialakításának az lehet az előnye, hogy lehetővé válik a hasonló kezelést igénylő feladatok rögzített munkáscsoporthoz rendelése. Ha mindkét csoportosítás megtörténik, akkor a modellalkotó kétféle alapelképzelés közül választhat: a szekvenciális megközelítésben először kialakítja az egyik csoportosítást, majd a másodikat úgy oldja meg, hogy a még csoportosítatlan gépeket/termékeket a termékek/gépek kész csoportjai közül valamelyikhez hozzárendeli. Szimultán megközelítésben a két csoportosítást egyszerre végzik el. A felírandó modellek hasonlóak, csak a szimultán modell várhatóan nagyobb méretű, és költségesebb megoldani. Előnye viszont, hogy nem szolgáltat rosszabb megoldást, mint a szekvenciális modell, mert ez utóbbi megoldása a szimultán modellben is megengedett megoldás. Ezért előnyben részesítjük a szimultán modelleket. A továbbiakban a gépek csoportjaira gyakran használjuk a megszokott cella elnevezést.

2.2. I. modell

- Szimultán modell,[2].
- Egy termékre több gyártási útvonal is lehetséges.
- Lehetséges, hogy egy adott gyártási művelet több gépen is elvégezhető.
- Adott típusú gépből több példányt megengedünk akár egy cellán belül is. Tehát ugyanolyan típusú gépeket nem kezelünk különbözőnek, így a gyártási útvonalakban géptípusok szerepelnek, nem egyéni gépek.
- Az adott típusba tartozó gépek száma nem kötött, csak azt követeljük meg, hogy minden elvégzendő művelethez legyen egy alkalmas gép abban a cellában, amihez a műveletet igénylő terméket rendeljük.
- A gépekbe történő tőkebefektetést minimalizáljuk, miközben az üzemeltetési összköltségre megállapított korlátot sem lépjük túl.

Jelölések. Indexek:

k	$= 1, 2, ..., K$ termékek	
m	$= 1, 2, ..., M$ gépek	
p	$= 1, 2, ..., P_k$ gyártási tervek a k. termékre	
s	$= 1, 2, ..., S(k,p)$ részműveletek a (k,p) párosításra azaz amikor a k terméket p terve szerint gyártjuk	
c	$= 1, 2, ..., C$ cellák	

Döntési változók:

N_{mc} = az m típusú gépek száma a c cellában

$Y_{kp} = \begin{cases} 1, & \text{ha k terméket a p terv szerint gyártjuk} \\ 0 & \text{különben} \end{cases}$

$X_{ms}(k,p)$ =1, ha az m gépet használjuk az s művelet elvégzéséhez a (k,p)párosítás mellett.

$r_{kc} = \begin{cases} 1, & \text{ha a k termék a c cellához tartozik} \\ 0 & \text{különben} \end{cases}$

Együtthatók:

$c_{ms}(k,p)$ = s művelet elvégzési költsége m gépen (k,p) párosításnál.

$t_{ms}(k,p)$ = s művelet elvégzési ideje az m gépen (k,p) párosításnál.

$a_s(k,p) = \begin{cases} 1, & \text{ha az s műveletet el kell végezni a k termék} \\ & \text{p terv szerinti gyártásakor} \\ 0 & \text{különben} \end{cases}$

$\alpha_{ms} = \begin{cases} 1, & \text{ha m gép képes az s művelet elvégzésére} \\ 0 & \text{különben} \end{cases}$

C_m = m gép egységára

OC = pénzbeli korlát a működési költségekre (adott időperiódus alatt)

b_m = egyetlen m típusú gép maximális működtetési ideje (adott időperiódusban)

d_k = a k termék iránti kereslet

MAX_c = a cellák maximális mérete

Mivel a modell felépítésekor a szimultaneitás miatt $r_{kc}X_{ms}(k,p)$ alakú szorzatok is megjelennek a feltételekben, és a két döntési változó szorzata elrontja a linearitást, a linearitás megtartására be kell még vezetnünk további változókat ezen probléma elkerülésére. Minden ilyen lehetséges szorzatra legyen az új folytonos változó $L_{cms}(k,p)$

A modellt leíró formulák.

$$\sum_{mc} C_m N_{mc} \to \ min$$

$$\sum_p Y_{kp} = 1 \quad \forall k$$

Pontosan egy gyártási útvonalat választunk minden termékhez.

$$\sum_m \alpha_{ms} X_{ms}(kp) = a_s(kp) Y_{kp} \quad \forall s,k,p$$

Amelyik mûvelet szükséges, azt végre is hajtjuk valahol.

$$\sum_{kps} d_k L_{cms}(kp) t_{ms}(kp) \leq b_m N_{mc} \quad \forall m,c$$

A cellán belüli gépek kapacitását nem lépjük túl.

$$\sum_{kpms} d_k X_{ms}(kp) c_{ms} \leq OC$$

A költségkorlátot nem lépjük túl.

$$r_{kc} + X_{ms} - L_{cms}(kp) \leq 1 \quad \forall k,c$$

$$r_{kc} \geq L_{cms}(kp) \quad \forall c,m,s,k,p$$

$X_{ms}(kp) \geq L_{cms}(kp) \quad \forall c, m, s, k, p$

$L_{cms}(kp)$ *modellbe foglalása miatt szükségesek.*

$\sum_c r_{kc} = 1 \quad \forall k$

Minden terméket hozzárendelünk pontosan egy cellához. $\sum N_{mc} \leq MAX_c \quad \forall c$

A cellák mérete nem lépi túl a maximálisan megengedettet.

$L_{cms}(kp) \geq 0 \quad \forall c, m, s, k, p$

$X_{ms}(kp) \in \{0, 1\} \quad \forall m, s, k, p$

$Y_{kp} \in \{0, 1\} \quad \forall k, p$

$0 \leq N_{mc} \in N \quad \forall m, c$

2.3. II. modell

- Szekvenciális modell -először a cellákat alakítja ki, majd ezekhez hozzárendeli a termékeket,[3].
- Gép-gép hasonlósági mértéket (távolsági mértéket) alkalmaz.
- A kialakítandó cellák számát előre kéri.
- A cél az eltérő cellába kerülő gépek egymástól vett távolságai összegének minimalizálása.
- Azonos típusú gépekből többet is megengedünk. Egy termékre egy gyártási útvonal van, és a gépek ezen belüli sorrendjét nem vesszük figyelembe.

Két gép távolságát az egyik gépen igen, de a másikon nem megforduló termékek számával definiáljuk. Így azon gépek távolsága lesz kicsi, amelyek sok termékhez együtt szükségesek, és ezért a külön cellába helyezésük várhatóan sok cellaközti szállítást generálna. Ha a kapacitások elégtelenek, akkor a szükséges gépeket megduplázzuk, azaz új, de tulajdonságaiban valamelyik már létezővel teljesen ekvivalens gépeket veszünk be. El szeretnénk kerülni, hogy két teljesen azonos gép egy cellába kerüljön, mert feltételezés szerint ezek a gépek alapműveleteket végeznek a résztermékek nagy részén. Ezért ezek távolságát végtelennek definiáljuk.

Jelölések.

$m =$	a gépek száma
$B_i =$	az i. típusú gépek száma
$n =$	a termékek száma
$p =$	az előre megszabott cellaszám
$G_k =$	egy cellán belül a gépek maximális száma
$a_{it} =$	$\begin{cases} 1, & \text{ha a t termék hsználja az i gépet} \\ 0 & \text{különben} \end{cases}$
d_{ij}	az i. és j. gépek távolsága:
$d_{ij} = \sum_{t=1}^{n} \delta(a_{it}, a_{jt})$	ahol
$\delta(a_{it}, a_{jt}) =$	$\begin{cases} 1, & \text{ha } a_{it} \neq a_{jt} \\ 0 & \text{különben} \end{cases}$

$$x_{ik} = \begin{cases} 1, & \text{ha az i. gépet a k. cellába helyezzük} \\ 0 & \text{különben} \end{cases}$$

$$y_{tk} = \begin{cases} 1, & \text{ha a t. terméket a k. cellához rendeljük} \\ 0 & \text{különben} \end{cases}$$

A modell formulái.

1. lépés: cellaformálás.

$\sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \sum_{k=1}^{p} d_{ij} x_{ik} x_{jk} \to min$

$\sum_{i=1}^{m} x_{ik} \leq G_k \quad \forall k$

Biztosítja, hogy a maximális cellanagyságot ne lépjük túl.

$\sum_{k=1}^{p} x_{ik} = B_i \quad \forall i$

Minden gépet elhelyezünk valahová.

$x_{ik} \in \{0,1\} \quad \forall i,k$

2. lépés: a termékek elhelyezése. Mindegyik terméket ahhoz a cellához rendeljük, amelyben rajta a legtöbb művelet történik.

$$\sum_{i=1}^{m} a_{it} x_{ik} \to max \qquad \forall t$$

A szekvenciális modell sokkal egyszerűbben felirható, mint egy szimultán. Igaz, ez jelen esetben a kevesebb figyelembe vett tényezőnek is köszönhető, de érezhető az a különbség, amelyik azonos feladatokat megoldó szekvenciális és szimultán modellek között van.

2.4. III. modell

- Szimultán modell,[4].
- Megengedett több példány azonos típusú gépekből, és ezeket különböző cellákba próbáljuk helyezni.
- A műveletek időigényét figyelembe vesszük.
- Ha egy termék nagy mennyisége miatt valamelyik gép napi gyártókapacitását túllépnénk, akkor megengedett ezt a mennyiséget (legfeljebb) két részre osztani, és a két részt egymástól függetlenül kezelni. Előre meg kell határozni, mely termékekre szükséges a kettéosztás.
- a cellán belüli és a cellák közötti minden szállítást figyelembe veszünk, csak különböző súlyokkal.
- a cellák száma előre eldöntendő, a maximális cellaméret adott

Jelölések.

$i = 1, 2, ..., m$	gépek
$j = 1, 2, ..., n$	termékek
NLS=	a megosztást nem kívánó termékek halmaza
LS=	a megosztást kívánó termékek halmaza
n_1=	NLS számossága
n_2=	LS számossága
$n =$	$n_1 + n_2$
$l = 1, 2, ..., c$	cellák
θ_1=	a cellaközti szállítások szorzósúlya (0.7 ajánlott)
θ_2=	a cellán belüli szállítások szorzósúlya (0.3 ajánlott)

$$y_{jl} = \begin{cases} 1, & \text{ha a j terméket az l cellához soroljuk} \\ 0 & \text{egyébként} \end{cases}$$

$$z_{jkl} = \begin{cases} 1, & \text{ha a j termék k műveletét az l cellában végezzük} \\ 0 & \text{egyébként} \end{cases}$$

$$x_{ipl} = \begin{cases} 1, & \text{ha az i típusú gép p. példánya az l cellában van} \\ 0 & \text{különben} \end{cases}$$

p_{jki}=	j termék k műveletének elvégzési ideje az i gépen
k_j=	a j terméken végzendő műveletek száma
op_j=	j azon műveleteinek halmaza, amelyeket osztott módban végzünk $j \in LS$
k'=	a második részkötegen végzett művelet, azaz $k' \in op_j$
n_l=	a cellák maximális mérete, minden cellára azonos
m_i=	az i típusból szükséges gépek száma
c_{ip}=	az i típusú gép p. példányának kapacitása
MM	a legalább két példánnyal bíró géptípusok halmaza
PP	a legalább két műveletet igénylő termékek halmaza
$m(j,k)$=	a géptípus, ahol j termék k műveletét végre kell hajtani

A modell egyenletei. A célfüggvény a kettéosztási lehetőség miatt bonyolult. A j termék cellaközti mozgásai:

$$A(j) = \sum_{l=1}^{c} z_{j1l}(1 - y_{jl}) + \sum_{k=1, j \in PP}^{k_j - 1} \sum_{l=1}^{c} z_{jkl}(1 - z_{j(k+1)l})$$

ha $j \in NLS$, és

$$A(j) = \sum_{l=1}^{c} z_{j1l}(1 - y_{jl}) + \sum_{l=1}^{c} z_{jk'l}(1 - y_{jl})|k' = 1$$

$$+ \sum_{k=1, j \in PP}^{k_j - 1} \left(\sum_{l=1}^{c} z_{jkl}(1 - z_{j(k+1)l}) + \sum_{l=1}^{c} z_{jk'l}(1 - z_{j(k'+1)l})|k' = k \right)$$

ha $j \in LS$. A j termék cellán belüli mozgásai:

$$B(j) = \sum_{k=1, j \in PP}^{k_j - 1} \sum_{l=1}^{c} z_{jkl} z_{j(k+1)l}$$

ha $j \in NLS$, és

$$B(j) = \sum_{k=1, j \in PP}^{k_j - 1} \left(\sum_{l=1}^{c} z_{jkl} z_{j(k+1)l} + \sum_{l=1}^{c} z_{jk'l} z_{j(k'+1)l}|k' = k \right)$$

ha $j \in LS$
Így a feladat:

$\sum_{j \in NLS} \theta_1 A(j) + \sum_{j \in LS} \theta_2 B(j) \to min$

$\sum_{l=1}^{c} y_{jl} = 1 \quad \forall j$

Minden terméket pontosan egy cellához hozzárendelünk.

$\sum_{l=1}^{c} x_{ipl} = 1 \quad \forall i, p$

Minden gépet pontosan egy cellában elhelyezünk.

$\sum_{l=1}^{c} z_{jkl} = 1 \; j \in NLS \; \cup \; LS \; k = 1..k_j$

$\sum_{l=1}^{c} z_{jk'l} = 1 \; j \in LS \; k' \in op_j$

Minden szükséges mûveletet pontosan egy cellában végrehajtunk,

megosztás esetén is.

$\sum_{p=1}^{m_i} c_{ip} x_{ipl} \geq p_{jkl} z_{jkl} \quad \forall l, j; \; k = 1..k_j \; i = m(j,k)$

$\sum_{p=1}^{m_i} c_{ip} x_{ipl} \geq p_{jk'l} z_{jk'l} \quad \forall l; j \in LS \; k' \in op_j \; i = m(j,k)$

$\sum_{p=1}^{m_i} c_{ip} x_{ipl} \geq \sum_{j \in NLS} \sum_{k=1}^{k_j} p_{jkl} z_{jkl} + \sum_{j \in LS} \sum_{k=1}^{k_j} (p_{jkl} z_{jkl} + p_{jk'l} z_{jk'l} | k' = k)$

$\forall l \; i \in MM$

$\sum_{i=1}^{m} \sum_{p=1}^{m_i} x_{ipl} \leq n_l \quad \forall l$

Biztosítja a cellaméretre vonatkozó kikötés betartását.

A magyarázat nélküli egyenletek biztosítják, hogy minden cellában bármely ottani művelethez akadjon a művelet elvégzésre alkalmas gép. Ha adott géptípusból egy van, akkor ez elég is; a gép éppen ott lesz, ahol szükséges. Ha több példány is van, akkor elegendő számú gépnek kell a cellában lennie a szükséges kapacitás biztosításához.

2.5. IV.modell

Szekvenciális megközelítés esetén, ha először a termékcsoportokat alakítjuk ki, kedvezőtlen esetben nem tudunk egyes termékcsoportokat egyetlen gépcsoporthoz rendelni a kapacitások elégtelensége miatt. Akadály lehet a szerszámhelyek vagy félkész termékek tárolóhelyének kicsisége. Ha ilyen probléma merül fel, akkor egy lehetséges megoldás a termékcsalád alcsaládokra osztása. Ez annyit jelent, hogy az azonos gépigényű termékek egyes csoportjait időben elkülönülve gyártjuk, ciklikusan változtatva az éppen gyártandó halmazt.[5]

- Egyetlen cellánk van, amelyben az összes terméket gyártjuk.
- Minden termékből egyetlen egységnyit tételezünk fel gyártandónak, különben többször számítjuk különböző termékként.
- Amikor egy adott alcsalád van soron, minden tagjából egy egységnyit gyártunk, majd a következő alcsalád jön.

- Előre megadandó, hány alcsaládot engedünk meg. Ha túl kis számot adunk meg, akkor a feltételek teljesíthetetlenek lesznek.
- A cél a gépek összes állásidejének minimalizálása.

Jelölések.

P	a termékek halmaza
M	a gépek halmaza
S	az alcsaládok halmaza
U	a szerszámok halmaza
p, m, s, u	termékek, gépek, alcsaládok,szerszámok
b_u	az u szerszám által elfoglalt hely nagysága
B_m	az m gép szerszámtárának kapacitása
t_{pm}	p termék kapacitásigénye az m géptől.
C_s	az s család átfutási ideje
	$C_s = max_{m \in M} \sum_{p \in P} t_{pm}$

Változók.

$$x_{ps} = \begin{cases} 1, & \text{ha a p termék az s cellához tartozik} \\ 0 & \text{különben} \end{cases}$$

$$y_{mus} = \begin{cases} 1, & \text{ha az m gépnek szükséges u szerszám az s gyártásához} \\ 0 & \text{különben} \end{cases}$$

$$a_{mup} = \begin{cases} 1, & \text{ha az m gépnek szüksége van u szerszámra p gyártásához} \\ 0 & \text{különben} \end{cases}$$

A modell egyenletei.

$$\sum_{s \in S} \left(|M| \, C_s - \sum_{m \in M} \sum_{p \in P} t_{pm} x_{ps} \right) \to min$$

$$\sum_{p \in P} t_{pm} x_{ps} \leq C_s \quad \forall m, s$$

Alsó korlát a gyártási idôre.

$$\sum_{s \in S} x_{ps} = 1 \quad \forall p$$

Minden terméket egyértelmûen helyezünk alcsaládba.

$$\sum_{u \in U} b_u y_{mus} \leq B_m \quad \forall m, s$$

A tárolóhely mindig elég a szerszámoknak.

$$a_{mus} x_{ps} \leq y_{mus} \quad \forall u, p, m, s$$

A definíció miatt szükséges.

2.6. V. modell

- Szekvenciális modell,[6].
- P-medián modell, azaz csoportosításkor a csoportot annak egy rögzített eleme köré szervezi.
- Hasonlósági mértékeket használ, vagyis ugyanazon típusú tényezők párjai közötti hasonlóságot egyetlen számmal jellemzi.

- A bemutatott módszer előnye korábbi hasonló típusú munkákhoz képest, hogy kevesebb változót használ.
- Egy termékhez több gyártási útvonal megengedett.

Jelölések.

n, m	gépszám és termékszám
$a_{ijr} =$	$\begin{cases} 1, & \text{ha j termék r terve használja az i gépet} \\ 0 & \text{különben} \end{cases}$
$R_j =$	j termék gyártási terveinek halmaza
$t = \sum_{j=1}^{n} \|R_j\| =$	a tervek összes száma
$y_{hi} =$	$\begin{cases} 1, & \text{ha a h gép az i cellához tartozik} \\ 0 & \text{különben} \end{cases}$
$\beta_{hij} =$	$\begin{cases} n, & \text{ha } a_{hjr} = a_{ijr} = 1 \text{ valamilyen } r \in R_j \text{ -re} \\ -1, & \text{ha } a_{hjr} \neq a_{ijr} \quad \forall r \in R_j \\ 0 & \text{egyébként} \end{cases}$
$s_{hi}=$	$\begin{cases} \sum_{j=1}^{n} \beta_{hij} \text{ ha } h \neq i \quad i = 1, ..., m \\ 0 \quad \text{egyébként} \end{cases}$
L=	minimális cellaméret
U=	maximális cellaméret

A hasonlóság mértéke két gép között: s_{hi}. Értelme: nagy a hasonlósága azoknak a gépeknek, amelyek sok termék útvonalában szerepelnek együtt. El kell kerülnünk azonban olyan gépek egy cellába helyezését, melyek sohasem szükségesek együtt; ezért lehet a hasonlóság negatív.

A modell egyenletei.

$$\sum_{h=1}^{m} \sum_{i=1}^{m} s_{hi} y_{hi} \to max$$

$$\sum_{i=1}^{m} y_{hi} = 1 \quad h = 1..m$$

Minden gépet pontosan egy cellában helyezzünk el.

$$\sum_{h=1}^{m} y_{hi} \geq L y_{ii} \quad i = 1..m$$

$$\sum_{h=1}^{m} y_{hi} \leq U y_{ii} \quad i = 1..m$$

A cellaméretek a kívánt határok közé esnek.

$$y_{hi} \in 0, 1 \quad h, i = 1..m$$

A modell nem igényli a cellaszám előzetes korlátozását, mert y_{hi}-k gépszámnyi sok cellát is megengednek, ennél több nem lehet. Így az optimális cellaszámot is megkapjuk. Ezután még meg kell oldanunk minden termékre az útvonalválasztást és a cellához rendelést, de mint egy korábbi szekvenciális modellel kapcsolatba látható volt, ez egyszerű feladat. A cél a legkevesebb szállítást eredményező tervek kiválasztása lehet, a leggyakoribb célt követve.

2.7. VI. modell

Az eddigi modellek mérsékelt méretűek voltak, cserébe viszont csak a jellemzők egy kis hányadát vették figyelembe. Előfordulhat, hogy valamely problémára az eddigi leegyszerűsítések túl soknak bizonyulnak. Például ha a terveket tekintjük, nem csak az előforduló gépek számbavétele lehet szükséges a kielégítő eredményhez, hanem a gépek sorrendje is. Korábban jeleztük, hogy a célunk is nagyon sokrétű lehet. Egyetlen célfüggvény egyes részcélokat kiemelhet, de másokat teljesen figyelmen kívül hagy. A következő modell elemei hasonló gondolatmenetet követnek, mint az eddigiek, de sokkal több jellemzőt vesz figyelembe. A rövidség kedvéért az előzetes leírást elhagyjuk, az egyenletek magyarázata jelzi a modell szerkezetét. [7]

Változók.

x_{ij}:	0-1 változó, 1, ha az i termék a j cellához tartozik
λ_j:	0-1 változó, 1, ha a j cella nem üres
E_{kj}:	0-1 változó, 1, ha a k gép előfordul j cellában
γ_{klj}:	0-1 változó, 1, ha k gépet az l elé helyeztük fizikailag a j cellában (nem okvetlenül módon)
N_{kj}:	k típusú gépek száma a j cellában
m_{kj}:	k típusú gép sorszáma a j cellában
L_{kj}:	a j cellában lévő k típusú gépek várható terhelése
AL_j:	az átlagos terhelés a j cellában
TL_j:	a teljes terhelés a j cellában
EL:	a cellák átlagos terhelése
S_i:	i termék cellán belüli szállításainak száma (cellák közti nincs)

Paraméterek.

p:	cellák száma
n:	termékek száma
M:	a géptípusok halmaza
DA_{cd}:	c és d termékek különbözőségi mértéke formai tulajdonságaik alapján
OS_{cd}:	c és d termékek különbözőségi mértéke gyártási útvonalaik alapján
Y_{ikl}:	0-1 paraméter:1, ha i termék valamikor a k gépen van, közvetlenül mielőtt az l gépre kerülne
B_{cd}:	0-1 paraméter:1 ha c és d termékeknek nem ajánlatos egy cellához tartozniuk ellentétes gyártási tervük miatt
α_{kj}:	alsó korlát a k típusú gép kihasználtságára a j cellában
β_{kj}:	felső korlát ugyanerre
P_i:	átlagos igény az i termékre
t_{ik}:	i termék egy darabja által a k típusú gépen töltött idő
A_k:	k típusú gépből rendelkezésre álló kapacitás
C_k:	k típusú gép fix költsége
CS_j:	a gépek maximális száma a j cellában
U:	egy "nagyon nagy" konstans

Impliciten: egy termékre egy technológia (gyártási útvonal) van.

Célok.

$$\sum_{c=1}^{n}\sum_{d=1}^{n}\sum_{j=1}^{p} DA_{cd}x_{cj}x_{dj} \to min \tag{1}$$

$$\sum_{c=1}^{n}\sum_{d=1}^{n}\sum_{j=1}^{p} OS_{cd}x_{cj}x_{dj} \to min \tag{2}$$

$$\sum_{j=1}^{p}\sum_{k\in M} (N_{kj}C_k + E_{kj}) \to min \tag{3}$$

$$\sum_{j=1}^{p}\sum_{k\in M} \frac{\left[\frac{L_{kj}}{N_{kj}} - AL_j\right]^2}{\sum_{k\in M} E_{kj}} \to min \tag{4}$$

$$\sum_{j=1}^{p} \frac{(TL_j - EL)^2}{p} \to min \tag{5}$$

$$\sum_{i=1}^{n} S_i \to min \tag{6}$$

$$\tag{7}$$

Minimalizáljuk az alaki különbözőségeket csoporton belül, minimalizáljuk a gyártási útvonalak különbözőségét, minimalizáljuk a befektetési költséget, a cellán belüli terhelések szórását, a cellák terhelésének szórását, és végül a cellák közti szállítások számát.

Az egyenletek. A magyarázatokat lásd lentebb.

$$\sum_{j=1}^{p} x_{ij} = 1 \qquad \forall i \tag{8}$$

$$\sum_{j=1}^{p} \lambda_j \le p \tag{9}$$

$$x_{ij} \le \lambda_j \qquad \forall i, j \tag{10}$$

$$B_{cd}(x_{cj} + x_{dj}) \le 1 \qquad \forall c, d, j \tag{11}$$

$$x_{ij}Y_{ikl}(m_{lj} - m_{kj}) \ge 1 \qquad \forall k, l, j \tag{12}$$

$$(m_{lj} - m_{kj}) + U(1 - \gamma_{klj}) \ge 1 \qquad \forall k, l, j \tag{13}$$

$$(m_{kj} - m_{lj}) + U\gamma_{klj} \ge 1 \qquad \forall k, l, j \tag{14}$$

$$L_{kj} - \sum_{i=1}^{n} t_{ik}P_i x_{ij} = 0 \qquad \forall k, j \tag{15}$$

$$L_{kj} - N_{kj}A_k \le 0 \qquad \forall k, j \tag{16}$$

$$\alpha_{kj} \le \frac{L_{kj}}{N_{kj}} \le \beta_{kj} \qquad \forall k, j \tag{17}$$

$$N_{kj} - E_{kj}U \le 0 \qquad \forall k, j \tag{18}$$

$$\sum_{k\in M}\sum_{l\in M} x_{ij}Y_{ikl}(m_{ij} - m_{kj} - 1) - S_i = 0 \qquad \forall i, j \tag{19}$$

$$TL_j - \sum_{k\in M} L_{kj} = 0 \qquad \forall j \tag{20}$$

$$AL_j - \frac{\sum_{k \in M} \frac{L_{kj}}{N_{kj}}}{\sum_{k \in M} E_{kj}} = 0 \qquad \forall j \tag{21}$$

$$EL - \frac{1}{p} \sum_{j=1}^{p} TL_j = 0 \tag{22}$$

$$\sum_{k \in M} N_{kj} \leq CS_j \qquad \forall j \tag{23}$$

$$x_{ij}, \lambda E_{kj}, \gamma_{klj} \in 0, 1 \tag{24}$$

m_{kj}, S_i, N_{kj} egészek.

8: minden terméket pontosan egy cellához hozzárendelünk. 9: legfeljebb p darab cella van. 10: csak megnyitott cellához rendelünk gépet (vagy: megnyitott az, amiben van legalább egy gép). 11: két ellentétesen haladó terméket nem gyártunk ugyanazon cellában. 12: ha valamit gyártunk egy cellában, és útvonala szerint két gép közvetlenül egymás után következik, akkor ezek fizikailag is egymás után helyezkednek el (nem biztosan közvetlenül). 13, 14: két gép nem foglalhatja el ugyanazon sorszámú helyet a cellában. 15,16: a gépek terheltsége, és az adott típusú gépből adott cellában szükséges mennyiség kiszámítása. 17: korlát a gépek kihasználtságára. 18: beállítja E_{kj} értékét 1-re, ha szükséges. $E_{kj} = 0$-t, amikor ez kell, a 3. célfüggvény biztosítja. 19: adott termék szállításainak száma. 20: 21: teljes és átlagos gépterhelés a cellákon belül 22: átlagos cellaterhelés 23: a maximális cellaméretet nem lépjük túl.
A körültekintés ára a jelentős méretnövekedés lett, holott cellaközti szállításokat, és alternatív gyártási útvonalakat nem tételeztünk fel.

3. A feladat nehézsége

Valamennyi modellünket diszkrét programozási feladatként fogalmaztuk meg. A leggyakrabban bináris változók szerepeltek, de feltűntek tágabb határok között mozgó egészértékű változók is. Tudjuk, hogy minden ilyen változókkal rendelkező feladat visszavezethető tisztán bináris változókkal felírt feladatra. A paraméterekről feltehetjük, hogy racionálisak, vagy még tovább menve, hogy egész számok, mert a modellezett gyakorlati feladatoknál pontosabb megközelítésre nincs szükség, és mérési tökéletlenségek miatt lehetőség sem. Legyen adott A m*n-es egész mátrix, b m dimenziós egész vektor. Ekkor annak az eldöntése, hogy létezik-e olyan x n dimenziós 0-1 vektor, amelyre $Ax \leq b$, NP-teljes. A bizonyításhoz [12] ad ötletet: vezessük vissza feladatunkra a 3-SAT problémát. Tegyük fel, hogy adott az F 3-konjunktív normálformájú Boole formula. Minden benne szereplő x_i változóhoz vezessük be az x_{i1}, x_{i2} bináris változókat. x_{i1} feleljen meg x_i-nek úgy, hogy pontosan akkor 1, amikor x_i igaz. x_{i2} feleljen meg ugyanígy x_i negáltjának. Feleltessük meg az igaz logikai értéknek az 1-et, a hamisnak a 0-t. Így egyenlőtlenségek formájában felírhatjuk azokat a követelményeket, hogy egy logikai változó az igaz és hamis értékek közül pontosan egyet vesz fel, illetve hogy mikor igaz egy három literálból álló diszjunktív tag a formulában. Ha a literáloknak megfelelő bináris változók C, D, E, akkor

$$-x_{i1} - x_{i2} \leq -1 \tag{25}$$

$$x_{i1} + x_{i2} \leq 1 \tag{26}$$

$$-C - D - E \leq -1 \tag{27}$$

Az egyenleteket összefoglalhatjuk $Ax \leq b$ alakban, ahol b csak $\pm$ 1 elemekből áll, és hossza 2(változók száma F-ben)+(a diszjunktív tagok száma F-ben), A oszlopainak száma 2(változók száma F-ben). Az F formula pontosan akkor elégíthető ki, ha van olyan x bináris vektor, mellyel teljesül $Ax \leq b$. Ezzel visszavezettük a kielégíthetőségi problémát a megengedett megoldás létezésének problémájára. A modellekben megfogalmazott optimalizálási feladatok NP-nehezek, ezért nem várható hatékony megoldó algoritmus létezése. Itt egy példával illusztráljuk, hogy egyes optimalizálási feladatokra más ismerten NP-teljes problémák is visszavezethetők lehetnek. Példánk a III. modell egy speciális esete.[4]

R probléma. Tegyük fel, hogy $3N$ terméket gyártunk ($j = 1, 2, ..., 3N$) N különböző géptípussal ($i = 1, 2, ..., N$). Szintén tegyük fel, hogy minden típusból N darab áll rendelkezésre, és az azonos típusú gépek alkotnak egy-egy cellát, összesen tehát N darabot. Minden egyes terméken egy gyártási műveletet kell végezni minden géptípuson, de a sorrend kötetlen.

$a_{ij} =$ j termék időigénye az i típusú gépen
$at_i =$ az i géptípus egy példányának kapacitása

Néhány pontban kissé eltér a bizonyítás az irodalomtól, például azt is feltesszük, hogy θ, a cellaközti szállítások súlya 1-gyel egyenlő, mert értéke lényegtelen a mostani feladat szempontjából. A célunk az összszállítás minimalizálása a gépkapacitások figyelembevételével. Jelölje T a teljes szállítási mennyiséget.

Az R probléma NP-nehéz.. Az R optimalizációs feladathoz felírunk egy döntési feladatot. (Q): Adott paraméterek mellett létezik-e olyan megoldás, hogy $T(S) \leq Z$? Erre a visszavezetjük az ismerten NP-nehéz 3-partícionálási feladatot.
3-partícionálási feladat: Adottak N, B pozitív egészek, és $A = \{a_1, ..a_{3N}\}$ egészekből álló halmaz úgy, hogy

$$\sum_{j=1}^{3N} a_j = NB, \quad \frac{B}{4} < a_j < \frac{B}{2}, \quad j = 1, .., 3N$$

Létezik-e A-nak olyan három elemű halmazokból álló $(A_1, ..., A_N)$ partícionálása, hogy

$$\sum_{a_j \in A_i} a_j = B \qquad \forall i \ \ ?$$

Legyen N, B, A egy előfordulása a 3-partíciós problémának. Polinomidejű átalakításávall Q-t fogalmazzuk meg a következő módon:

$$a_{ij} = a_j \quad i = 1, .., N \quad j = 1, .., 3N$$

$$at_i = B$$

A Q feladatra akkor és csak akkor létezik S megoldás, amelyre $T(S) \leq 3N(N-1)$, ha van megoldása a 3-partíciós feladatnak.
"Akkor" irány: Tegyük fel, hogy létezik megoldása a 3-partíciós feladatnak. Legyen a 3-partíció a következő:

$$a_1 + a_2 + a_3 = B$$

$$a_4 + a_5 + a_6 = B$$

.

.

$$a_{3N-2} + a_{3N-1} + a_{3N} = B$$

Az egy adott típusból rendelkezésre álló gépek NB kapacitása éppen elégséges a $3N$ termék legyártásához, mert az 1.,2.,3. termékeket gyárthatjuk az első gépen, a 4.,5.,6. terméket a másodikon, és így tovább. A termékeknek valamennyi cellát meg kell látogatniuk, mert egy cellában csak egyféle gép van, így egy termék $N-1$ alkalommal kerül szállításra. Az összszállítás $3N(N-1)$. Tehát az S megoldásra $T(S) = 3N(N-1)$
"Csak akkor": Tegyük fel, hogy létezik egy S_0 megoldása Q-nak
$T(S_0) \leq 3N(N-1)$ szállítással. Mindent legyártunk, ezért $T_(S_0)$ nem is lehet kisebb, mint $3N(N-1)$, mert most is minden termék minden cellában megfordul. Mivel $\frac{B}{4} < a_j < \frac{B}{2}$, és $\sum_{j=1}^{3N} = NB$, nem helyezhetünk nulla, egy vagy kettő terméket egy gépre, mert akkor valahol máshol elégtelennek bizonyulna a kapacitás. Háromnál több terméket egyetlen gépre sem helyezhetünk, mert a kapacitásigény meghaladná B-t. Így minden gépre 3 terméket helyezünk, és mert pontosan elegendő a kapacitásuk, a 3-partíció is létezik. Tehát Q NP-teljes, így a hozzá tartozó optimalizálási feladat NP-nehéz. Ezért nem várható gyors egzakt megoldó algoritmus létezése.
Modelljeink nagyon sok változóból és feltételi egyenletből állhatnak, mint láthattuk, gyakran vezetünk be indikátorváltozókat akkora számban, mint egyes paraméterek szorzata. Az egészértékű változók és korlátozó egyenletek száma már viszonylag kis problémánál is elérhet egy akkora számot, hogy a feladat kezelhetetlenné válik az általános optimalizálási programcsomagok számára.

4. Heurisztikák

A korábbiak miatt kénytelenek vagyunk lemondani a pontos megoldás megkereséséről, és közelítő algoritmusokkal kell dolgoznunk. A probléma gyakorisága miatt nagyszámú eljárást dolgoztak ki és dolgoznak ki jelenleg is. A sokféle lehetséges cél és korlátozó tényező miatt ezen heurisztikák között kisszámú sorolható egy-egy csoportba közös tulajdonságaik alapján. A fejezetben ezért az ismert algoritmusokat nem céljuk, hanem a felhasznált eszközeik szerint fogjuk csoportosítani.

4.1. Mátrixrendező algoritmusok

Ha egy termékre csak egyetlen gyártási útvonal létezik, akkor a gép-termék kapcsolatokat egy gépszám*termékszám méretű incidencia mátrixszal szemléletesen jelezni tudjuk. Az incidenciamátrix i. sorának j. eleme 1, ha a j termék előállításához az i gép szükséges, egyébként minden elem 0. Ez a módszer leginkább akkor használható, ha az útvonalak rövidek, mert a sorrendiség jelzésére nem alkalmas. A korai algoritmusok közül sok alapult azon a felismerésen, hogy ezen mátrix sorainak és oszlopainak ismételt cseréjével olyan alakra hozható, amely információt ad a cellák célszerű kialakításáról, és ha szükséges, a termékek is hozzárendelhetők egy-egy cellához. A cél az, hogy a mátrixot blokkdiagonális alakra hozzuk. Tisztán blokkdiagonális alakot, amikor a főátlón sorakozó blokkok csupa egyesből állnak, és a blokkokon kívül csak 0 elemek állnak, csak ritkán érhetünk el, gyakorlati feladatoknál nem várhatunk mást, mint hogy a blokkokban a 0-k relatív száma minél kisebb legyen, és a blokkon kívüli, kívételes elemek száma szintén alacsony legyen. Ha az egy blokkhoz tartozó gépeket egy cellába helyezzük, akkor elérhetjük, hogy a cellaközi szállítás kis mennyiségű legyen. A termékeket pedig hozzárendelhetjük ahhoz a cellához, ahol rajtuk a legtöbb művelet történik. Hátránya a módszernek, hogy a cellák beazonosítása sokszor emberi utómunkát igényel, mert a legtöbb algoritmus leáll a közel blokkdiagonális alak létrehozása után. [9] áttekint néhány ilyen típusú módszert. Ezek közül a legegyszerűbbet és a (fajtájában a produkált kivételes elemek száma, mint célfüggvény tekintetében) leghatékonyabbat mutatjuk be.

Momentumok módszere -MOM. A módszer fizikai gondolaton alapul. A sorokat (oszlopokat) egykarú emelőkként fogjuk fel, amelyeken egységnyi súlyok találhatók azokon a helyeken, ahol a mátrixban 1 áll. Az A mátrix i. sorának forgatónyomatéka

$$M_{yi} = \sum_{j=1}^{n} j a_{ij}$$

ahol n a termékszám. Ha a forgáspontnak a balszélt tekintjük, és a sorokat a forgatónyomatékuk szerint növekvő sorrendbe állítjuk, akkor várhatóan a mátrix felső részében fognak csoportosulni azok a sorok, amelyek első oszlopaiban vannak inkább egyesek, és alul azok, amelyekben a magas indexű oszlopokban vannak inkább egyesek. Ugyanígy sorbarendezhetjük az oszlopokat is forgáspontnak a mátrix felső szélét tekintve. Ha ezt a rendezést sokszor elvégezzük a sorokra és oszlopokra váltott sorrendben, akkor várhatóan ez közel blokkdiagonális alakhoz fog vezetni, mert a nemnulla elemeket a főátlóhoz közelíti. Speciális esetben előfordulhat, hogy a módszer éppen ellentétes eredményt ad a vártnál, és a blokkokat a főátlótól távol helyezi.

1	1	1	1	1	0	0		0	0	0	0	0	1	1
1	1	1	1	1	0	0		0	0	0	0	0	1	1
0	0	0	0	0	1	1		0	0	0	0	0	1	1
0	0	0	0	0	1	1		0	0	0	0	0	1	1
0	0	0	0	0	1	1		0	0	0	0	0	1	1
0	0	0	0	0	1	1		1	1	1	1	1	0	0
0	0	0	0	0	1	1		1	1	1	1	1	0	0

Ez a probléma kivédhető hasonló fizikai alapú ötletekkel. Elegendőnek bizonyult például annyi változtatás, hogy a rendezést a forgatónyomaték/összes terhelő tömeg hányados alapján végezzük. Azért hatásos a változtatás, mert így egy sok, elszórtan elhelyezkedő 1-et tartalmazó sor nem fog egy olyan másik mögé kerülni, amelyikben csak a forgásponttól távolabbi szélen akad néhány súly.

Differencia-nyomaték módszer -DMOM. Célszerűnek látszik figyelembe venni a rendezésnél nem csupán a súlyokat, hanem a 0 elemeket is. Azzal, hogy belőlük egyes nemnulla elemek között sok vagy kevés áll, befolyásolják a módszerek hatékonyságát. Ha egy 1-es előtt hosszú csupa 0 sorozat áll, akkor várhatóan ő a végső megoldásban kivételes elemet fog jelenteni, és ezért ajánlatos lenne kisebb jelentőséggel figyelembe venni, hogy legalább a többi nemnulla elemet hatékonyan soroljuk blokkokba. Induljunk ki az M_{yi} forgatónyomatékokból. A cikkhez képest kis módosítással:
Mivel $||x| - |y|| \leq |x - y|$,

$$||M_{yi}| - |M_{yk}|| \leq |M_{yi} - M_{yk}| = \left| \sum_{j=1}^{n} (a_{ij} - a_{kj}) j \right| \leq \sum_{j=1}^{n} |a_{ij} - a_{kj}| j$$

$$T_{yik} := \sum_{j=1}^{n} |a_{ij} - a_{kj}| j$$

Hasonlóan definiálhatjuk az oszlopokra a T_{xjk} mennyiségeket. El szeretnénk kerülni a várakozásokkal ellentétes viselkedést, ezért megint leosztunk a sorokban-oszlopokban álló nemnulla elemek összegével (W).

$$DMR_{yik} = \frac{T_{yik}}{W_{yi} + W_{yk}}$$

$$DMR_{xjk} = \frac{T_{xjk}}{W_{xj} + W_{xk}}$$

A DMR mennyiségek szerinti növekvő sorrendbe rendezünk. DMOM várhatóan jobb csoportosítást ad, mint MOM, mert azonos mintázat esetén jobban kiemeli a sorok/oszlopok közötti különbségeket. Hozzájárul ehhez az is, hogy a sorokat/oszlopokat páronként hasonlítja össze.

DMOM klaszterező algoritmus. Legyenek m, n a gépek és a termékek száma.

1. lépés	$W_{yi} = \sum_{j=1}^{n} a_{ij} \quad \forall i$
	M_{yi} kiszámítása $\quad \forall i$
2. lépés	$r_{yi} = \frac{M_{yi}}{W_{yi}} \quad \forall i$
	Rendezzük a sorokat növekvő sorrendbe r_{yi} szerint.
3. lépés	$i = 1$
	Futás közben $1 \le i \le m-1, \ i \le k \le m$
4. lépés	DMR_{yik} kiszámítása $k = i+1, .., m$
5. lépés	DMR_{yik}-k növekvő sorrendben állnak?
	Ha igen, stop, egyébként a 6. lépés jön.
6. lépés	Rendezzük a sorokat növekvő sorrendbe DMR_{yik} szerint.
7. lépés	$i = i + 1$
	Ha $i = m$, akkor a 8., egyébként a 4. lépésre megy.
8-15. lépés	1-7. lépések analogonjai oszlopokra.
16. lépés	Vissza a 3. pontra.

Az algoritmus tartalmaz 1-1 MOM rendezést is, mert ez kevésbé időigényes, és így várhatóan elő tudunk állítani egy elég jó közelítést a DMOM számára, amin gyorsan finomíthat. Amikor már nem első alkalommal rendezzük a sorokat, ezt a MOM lépést kihagyjuk a sorokra, és később értelemszerűen az oszlopokra is. A módszer egy módosításának viselkedésére a későbbiekben még visszatérünk.

Többcélfüggvényű klaszteranalízis a VI. modell közelítő megoldásának kereséséhez. A hat célfüggvény és a nagyszámú feltétel különösen nehezen kezelhetővé teszi a problémát. A vázolandó eljárás egy közelítő megoldást szolgáltat. Csak kezdő lépéseiben jutnak szerephez mátrixok, mégis itt említjük, mert az eljáráshoz egy fontos adatot, a cellák számát mátrixokkal meghatározva kapjuk. Az eljárás szekvenciális, először a termékeket csoportosítja.

- Alkossunk meg két termék-termék különbözőségi mátrixot előre rögzített különbözőségi függvények segítségével. A kombinált különbözőségi mátrix legyen ezek konvex lineáris kombinációja.

- Léteznek statisztikai eljárások, amelyek felső határt szolgáltatnak a termékcsoportok számára, ha bizonyos határ felett vesszük csak figyelembe a mátrixbeli értékeket. Ezután 2 és ezen maximális csoportszám között választhatunk egy tetszőleges p számot a termékcsoportok számára. Megint egy statisztikai eljárással, a k-medoid technikával létrehozhatjuk ezt a p csoportot.[8] A cél ebben a lépésben az átlagos csoporton belüli különbözőség minimalizálása.

- Minden csoporthoz elkezdjük egy gépcella kialakítását. Számbavesszük, milyen típusú gépekre lesz szükség, és ezek milyen sorrendben következzenek. Előfordulhat, hogy egyes csoportbeli termékek útvonalai egymással ellentétesek. Ekkor a gépek fizikai sorrendjének kialakításakor az ilyen darabokat figyelmen kívül hagyjuk. Csak a sorrend kialakítása után vizsgál-juk meg, melyek lettek közülük a többi termékkel ellentétesen haladók, akadályozók.

- Az akadályozó termékek számára megvizsgáljuk az összes gépcellát, és oda helyezzük át őket, ahol a legkevesebb fennakadást okozzák és a legkevesebb többletbefektetést igénylik.

- Miután minden termék valamelyik cellához lett rendelve, kiszámoljuk minden cellára és géptípusra a szükséges kapacitást biztosító gépek számát. Miután egy bizonyos gépkészlet már adott, egyes helyeken a kapacitás kevésnek bizonyulhat. Az ezt okozó termékeket megint megpróbáljuk más, alkalmas cellába áthelyezni. Ezután kiszámítjuk a mindenképpen megvásárolandó gépek értékét.

- Kiszámítjuk a terhelés szórását a cellákon belül és az összes cellára együtt, és a szállítások számát is.

- Értékeljük a lehetőséget a célfüggvények értékeinek egy alkalmasan válasz-tott függvényével.

- Kreáljunk más jelölteket is a 3-7. pont számára a 2. pont eredményéből úgy, hogy rendre kiválasztunk 1-1 terméket, és ezt áthelyezzük abba a termékcsoportba, amelytől való különbözősége a második legkisebb. Értékeljük ki ezen jelölteket is.

- Válasszuk ki a legjobb lehetőséget.

Láthatóan a csoportos technológiák itt csak a filozófiát adják a teendőkhöz, és nagyon sok múlik azon, hogy egyes részlépéseket hogyan, milyen statisztikai módszerrel valósítunk meg.

4.2. Kontrakciós heurisztikák

Az eljáráscsalád nevét arról kapta, hogy a csoportosítás létrehozása közben ha egyes objektumok már egy csoportba kerültek, akkor együtt is maradnak. A kialakult csoportokat egységként kezeljük, és együtt vonjuk össze egyéni objektumokkal vagy más alakuló csoportokkal. Jellemzően ezek az algoritmusok gyorsak, mert hamar eljutunk egy olyan állapotba, amikor már nincs több összevonási lehetőség. A gyors célratörés miatt a probléma finomabb szerkezetét képtelenek érzékelni, így várhatóan más, kifinomultabb módszereknél rosszabb eredményt adnak. A gyakorlati tapasztalatok azonban azt mutatják, hogy ha a tekintett problémák méretét jelentősen növeljük, akkor a kontrakciós heurisztikák teljesítménye megközelítheti, sőt el is érheti más algoritmusokét. Ekkor a kiemelkedő gyorsaságuk révén elsőrangú eszközévé válnak a csoportformálásnak.

Példa kontrakciós heurisztikára a IV. modellel kapcsolatban. Emlékeztetőül: Ebben a modellben már adott volt egyetlen gépcella a benne gyártandó termékekkel, csak kapacitáshiány (szerszámhelyek hiánya) miatt lehetetlen volt valamennyi terméket egyidőben gyártani. A termékeken belül alcsaládokat hozunk létre, és ezeket időben elkülönítve gyártjuk. A cél a gépek összes állásidejének minimalizálása. Ahhoz, hogy erre a feladatra kontrakciós heurisztikát alkalmazzunk, ötletet adhat a következő felismerés: Hacsak nincsenek kényszerítő szerszámozási korlátok, akkor sohasem éri meg egy adott alcsaládot két kisebbre osztani. Legyen bármely s alcsaládra $I(s)$ az s legyártása közbeni összes állásidő. M a gépszám. Legyen most s rögzített, s_1, s_2 pedig s egy partíciója.

$$I(s_1) + I(s_2) = M\left(max_m \in M\{\sum_{p\in s_1} t_{pm}\} + max_{m\in M}\{\sum_{p\in s_2} t_{pm}\}\right) -$$

$$-\sum_{m\in M}\sum_{p\in s_1} t_{pm} - \sum_{m\in M}\sum_{p\in s_2} t_{pm} \geq$$

$$\geq M max_{m\in M}\{\sum_{p\in s} t_{pm}\} - \sum_{m\in M}\sum_{p\in s} t_{pm} = I(s)$$

Így tehát ha nincs erre kényszerítő szerszámozási korlát, akkor az s családot nem érdemes felosztani. A lemma olyan eljárást sugall, amely lépésről lépésre csökkenti az alcsaládok számát. A heurisztika: 1. Kezdetben alkosson minden termék egy külön alcsaládot.
2. Minden s, q alcsaládra számítsuk ki az összevonásból származó potenciális megtakarítást:

$$SAV(s,q) = I(s+q) - I(s) - I(q) \leq 0$$

3. Válasszuk ki a legnagyobb abszolútértékű $SAV(s,q)$-t adó párt. Ha nincs akadályozó korlát, akkor olvasszuk az s, q alcsaládokat egybe. Az egyesített alcsalád kapacitásigénye minden gépen a külön vett kapacitások összege. Ha nem lehetséges az összevonás, tiltottnak nyilvánítjuk.
4. Vegyük elő a következő legnagyobb megtakarítást eredményező párt. Ha minden párt megvizsgáltunk, stop
Kísérleti eredmények szerint az algoritmus erre a problémára vetélytársaihoz képest hatékony, ha az átszerszámozási idők kicsik a gyártási időkhöz képest.

p-medián modellek. A p-medián modellek alapötletüket a telepelhelyezési problémától kapták, de a megvalósított modellek sok eltérő vonást tartalmaznak. Általában nem tesznek meg bizonyos kiválasztott elemeket az alakuló csoportok középpontjává és így szervező magokká, hanem az összevont egységeket együtt kezelik, és a részegységek adataiból számítanak rájuk paraméterértékeket. Ilyen értelemben már az előző algoritmus is p-medián alapúnak tekinthető. Gyakran a cellaközti szállítások minimalizálása a célunk, ekkor kínálja magát, hogy az egyesített elemek szállításai más cellákba és cellákból az elemek külön vett szállításainak összegei legyenek. Cellák összeolvasztása nem növelheti az összszállítást, ezért mindig érdemes lehet elvégezni, ha nem sértünk meg méretkorlátot. A különbség az előző kontrakciós heurisztikához képest annyi, hogy ha szállításokról van szó, akkor csak maguknak a szállításoknak a minimalizálása nagyméretű, egymással gyenge kapcsolatban álló gépekből összeálló cellákat eredményezhet, ezért a "cellaközti szállításon" inkább a bármely irányú szállítások összegének és a két cella gépei számának hányadosát értjük. Ezeket a mennyiségeket összevonáskor nem tudjuk egyszerűen összegezni, ezért a kiindulási adatok ismeretében meg kell határozni egy küszöbértéket, aminek meghaladása esetén összevonunk. A több munkáért kárpótol a könnyebben kezelhető, kis cellákból álló végeredmény.

Kontrakciós heurisztika szállításokkal -CELLA. Csak a cellaközti szállításokkal foglalkozunk.
1. Kezdetben minden gép egy külön cella.
2. Kiszámoljuk a (szállítás(i-ből j-be)+szállítás(j-ből i-be))/(i gépeinek száma + j gépeinek száma) mennyiségeket minden cellapárra.
3. Ha a legnagyobb érték a küszöb felett van, és az összevont csoport mérete nem sért nagysági korlátot, akkor végezzük el az összevonást. Ahol szükséges, számítsuk ki az új szállítási értékeket. Ha az összevonás méretkorlátot sértene, akkor töröljük a két cella közti szállításokat, így megtiltva összevonásukat.
4. Ha van még cellapár küszöb feletti szállítással, akkor ismételjük meg az előző pontot, egyébként stop.

4.3. Hipergráf- és gráfalgoritmusok

A gép-termék kapcsolatokat szemléletesen ábrázolhatjuk hipergráfok segítségével. [10] A csúcsok jelképezik a gépeket, a hiperélek pedig az egy-egy termék gyártásához szükséges gépeket tartalmazzák. Ha a hipergráfban adott egy kiskapacitású vágás, akkor ez olyan felosztását jelzi a gépeknek, ahol a két halmaz között a szállítás csekély. Ezért az a célunk, hogy a hipergráfot kiskapacitású vágásokkal megfelelő méretű részekre partícionáljuk, egyben a cellaközi szállítást is minimalizálva. A hipergráfos reprezentáció hátulütője: nem képes ábrázolni alternatív gyártási útvonalakat. További probléma, hogy hipergráfokra kisszámú kifejlesztett algoritmus létezik. Mindezeket ellensúlyozza azonban az a körülmény, hogy ha sikerül a problémát elemi gráfokra (a továbbiakban: gráf) visszavezetni, akkor a továbbiakban gyors és pontos algoritmusok állnak a rendelkezésünkre.

Hipergráf approximációja gráffal. Két módszert tárgyalunk, a klikk- és a csillagapproximációt. Klikkapproximációnál minden hiperélet egy klikkre cserélünk le, azaz berajzoljuk azt a közönséges teljes gráfot, amelynek csúcsai éppen a hiperél csúcsai. A másik út egy plusz csúcs bevétele a gráfba, és ennek összekötése csillagformában minden egyes hiperélbeli csúccsal.
Az approximáció típusának eldöntése után újabb fontos probléma annak eldöntése, milyen módon rendeljünk súlyokat az élekhez. A kiindulási hipergráfban az élek súlyai jelezhetik a termék kényességét, fontosságát, vagy a mennyiségét. A hiperélekhez egységnyi súlyt is rendelhetünk, hasonló szellemben közelítve, mint az incidenciamátroxokkal. A hiperélek súlyaiból kell a közönséges élek súlyait kiszámolnunk úgy, hogy a hipergráfban és gráfapproximációjában minél inkább egybeessenek a minimális vágások. A tapasztalatok szerint kielégítő eredményeket szolgáltat, ha az n csúcsú klikk éleihez a

$$\lceil \frac{n}{2} \rceil \lfloor \frac{n}{2} \rfloor$$

súlyokat rendelünk. Miután kiszámítottuk az új súlyokat, a többszörös éleket egyetlen éllel helyettesítjük, ennek súlyaként az összevont élek súlyainak összegét definiálva.
A minimális vágások megtalálásának hatékony módja a Gomory-Hu algoritmus

alkalmazása. Eredményként egy fát kapunk, amelyről egyértelműen leolvasható bármely két csúcs között a minimális vágás értéke, és maga a vágás is adott. Csak meg kell keresni a két csúcs közötti egyetlen utat a Gomory-Hu fában, és kiválasztani ennek minimális súlyú elemét. Kapjuk a partíciót és a vágásérték éppen a minimális súly. Ha bármely csúcspárra, illetve ha a Gomory-Hu algoritmus szerint dolgozunk,

$$\frac{m(m-1)}{2},$$

illetve

$$m-1$$

maximális folyamproblémát kell megoldanunk, ahol m a gépek száma.
Csillagapproximáció esetén a kiindulási gráf csúcsai csak az újonnan bevett csúcsokon keresztül vannak összekötve, és nincsenek többszörös élek, ezért azonnal kereshető a minimális vágás. A csillagapproximáció kedvezőbb annyiban, hogy a termékeket is hozzárendeli a gépcsoportokhoz, mert a bevett új csúcsok éppen a termékeket jelképezik, míg a klikkes megközelítésnél ez még egy lépést igényel. A vizsgálatok szerint azonban a két módszer azonos végső teljesítményt mutat, és így a klikkes approximációt kell előnyben részesítenünk, mert az sokkal kevesebb csúccsal dolgozik, kevesebb műveletet és kisebb tárterületet igényel.[10]
A maximális folyam-probléma hatékony megoldhatóságát más algoritmusok is kihasználják, de léteznek teljesen más utat követő heurisztikák is, például a hipergráfban történő szélességi keresés alkalmazása. A következőkben egy olyan gráfos eljárást vázolunk, amely igyekszik több rugalmasságot mutatni, mint a hipergráfok. Például hipergráfban nehézséget okoz annak ábrázolása, ha egy géptípusból több példány van, hiszen ekkor azt is el kell döntenünk, adott termék gyártási útvonalába mely konkrét gépet vesszük be az azonos típusúak közül. Már kisméretű példák is rámutatnak arra, hogy nem hatékony megoldás a többszörös gépeket egyetlen egységként kezelni és egyetlen cellában elhelyezni.

Többszörös gépek figyelembevétele módosított gráffal. A modell ([11]) egy termékre egy gyártási útvonalat tételez fel, viszont kezeli az azonos típusú gépeket. Figyelembe vesszük az egyes termékek által a különböző gépeken lefoglalt kapacitást is.
A rugalmasságot az biztosítja, hogy a több példánnyal rendelkező géptípusok szimbólumaként komplex csúcsot alkalmazunk. A komplex csúcsokon belül pontokat tartunk számon annak érzékeltetésére, hogy adot típusból több egység is van. A pontok száma nem feltétlenül az adott típusú gépek számát jelzi, hanem azt, hogy hány gyártási útvonal igényel a géptípus kapacitásából egy részt. A cellaformálási eljárás során a komplex csúcsokat kisebb darabokra törhetjük, így végül a géptípus példányai több cellában is megtalálhatóak lehetnek. Az egyetlen példánnyal rendelkező géptípusokat szokványos, törhetetlen csúccsal jelképezzük.
Az algoritmus a következő:

- Felvesszük a közönséges és a komplex csúcsokat. Összekötünk két csúcsot, ha létezik olyan termék, amelyik gyártási útvonala szerint megfordul az egyik géptípuson, és onnan közvetlenül a másik típusra kerül. Az él irányítatlan. Egy útvonal ezért egy irányítatlan út a gráfban. Különböző útvonalak egybeeshetnek, például 1-2-3, 1-2-3-2-3-2. Amennyiben komplex csúcsba fut be az él, benne elhelyezünk egy pontot, és ez szolgál az él végpontjaként. Ha az útvonal folytatódik, akkor a következő élet ugyanebből a pontból indítjuk. Ezért az egyes útvonalak a gráfreprezentáció komplex csúcsaiban nem érintkeznek ebben a fázisban, hanem egymással párhuzamosan haladnak. Magyarázó ábrák a dolgozat végén találhatók.

- Egyes speciális élek összevonhatók, ezeket egyesítjük. Ilyenek a két közönséges csúcs közötti többszörös élek, és még néhány eset.
  Végül egy komplex csúcsban a pontok száma az így összevont útvonalak számát jelzi, egyben a géptípus kapacitását megosztó termékcsaládok számát. Nem jelentenek viszont adott típusú géppéldányt.

- Ellátjuk szállításokat jelképező súlyokkal az éleket. Eközben a komplex csúcsokon belüli pontokat ugyanúgy kezeljük, mint a közönséges csúcsokat. Két csúcs, j és k között az él súlya:

$$\sum_{i=0}^{n} d_i(b_{ijk} + b_{ikj})$$

  ahol n a termékszám, d_i az i termék iránti igény, b-k az adott irányú szállítások alkalmának száma. Ezt még tovább szorozhatjuk még például a termék darabtömegével vagy valamilyen költségszorzóval.

- Meghatározzuk, egyes pontok mekkora gépkapacitás igényt jeleznek a rajtuk áthaladó útvonal részéről a géptípusuk iránt. Ez gyorsan megkapható, fel kell összegezni a kapacitásigényt minden, az adott típusú gépen végzendő műveletre.

- Egy gyűjtő és törő algoritmussal kialakítjuk a cellákat. Vegyük a legnagyobb súlyú élet, és húzzuk össze. Ha többszörös élek keletkeznének, akkor ezeket egyesítsük, összeadva a súlyaikat. Ezt komplex csúcsbeli pontokkal is elvégezzük. Ez alkalommal a pontot kiszakítjuk a komplex csúcsból, és ezután vonunk össze. A kiszakítás nem pontosan egy gép kivételét jelenti, hanem azt jelzi, hogy a géptípus jelen van az alakuló cellában, igény szerint több vagy kevesebb mint egységnyi kihasználtsággal. Ezután kiválaszthatjuk megint a maximális súlyú élt, és ismételhetjük az eddigieket. Ha menet közben egy típusból már több példányt igényelnénk, mint amennyi rendelkezésre áll, akkor a típust jelképező csúcsokból összevonunk, tekintet nélkül arra, létezik-e kapcsolat közöttük.

- Az eljárás véget ér, amikor már nem tudunk nagyobb méretű cellákat létrehozni a mérethatár elérése miatt.

A tapasztalatok szerint az algoritmus viszonylagosan nagy hatékonysággal működik.

4.4. Sztochasztikus heurisztikák

A sokszorosan összetett feladatokat determinisztikus algoritmussal nehéz kezelni. Ezért merült fel a gondolat, hogy sztochasztikus módszereket fejlesszenek ki.[1] Remény szerint a véletlen algoritmusok konkrét problémáknál olyan megengedett megoldásokat is elérhetővé tesznek, amelyek esetleg egy másik adott algoritmus hatókörén kívülre esnek. A két legelterjedtebb véletlent felhasználó algoritmuscsalád jellemzőit ismertetjük. Mindkettőre igaz, hogy a már megtalált legjobb megoldás megjegyzése mellett az éppen aktuálisan vizsgált megengedett megoldás valamilyen értelemben vett szomszédjával igyekszik folytatni a keresést, és a lokális optimumból való kimozdulás érdekében bizonyos valószínűséggel rosszabb célfüggvényértékű megengedett megoldásba is továbblépnek.

Szimulált lehülés. A módszer neve eredeti céljából, egy fizikai lehülési folyamat modellezéséből származik.
Ha a szimulált lehülés módszerét szeretnénk alkalmazni, akkor előre el kell döntenünk az alábbiakat:

- Mik a megengedett megoldások?
- Mi egy megengedett megoldás költsége?
- Hogyan állítunk elő egy kiindulási megengedett megoldást?
- Mit tekintünk egy megengedett megoldás környezetének?

Az algoritmus futásidejét az iterációk számával, és az egy iteráción belüli lokális keresések maximális számával befolyásolhatjuk. Szintén befolyásoló a lehülés üteme, azaz hogy hogyan változik az a valószínűség, amellyel egy rosszabb megoldást elfogadunk továbblépésre. Jellemzően ez a valószínűség az algoritmus futása során tart 0-hoz. Megvalósítható ez úgy, hogy minden továbblépéskor a megengedett valószínűséget egy fix értékkel csökkentjük, és a 0 elérésekor már csak határozottan jobb megoldásokat fogadunk el, vagy minden továbblépéskor egy 0 és 1 közötti számmal beszorozzuk. Szomszédos megoldáson rendszerint az aktuálisból egyetlen gép, termék, művelet, szerszám, stb. megengedett áthelyezésével kialakítható megoldást értünk. Áttekintve:
1. Állítsuk be a kezdeti paramétereket.
2. Állítsunk elő egy E kiinduló megoldást. Legyen a célfüggvény értéke Z. Legyen $E^* = E,\ Z^* = Z,\ it = 0$
3. $it = it + 1$ Ha meghaladtuk a maximális iterációszámot, akkor lépjen 7-re, különben $s = 0$, 4.
4. $s = s + 1$ Ha meghaladtuk a maximális lokális keresés-számot, akkor csökkent-sük a hőmérsékletet és 3., különben 5.
5. Állítsunk elő egy E_1 szomszédos megoldást E-ből. Ha $Z_1 < Z$ nem teljesül, akkor 6., egyébként
a)Ha $Z_1 < Z^*$, akkor $E^* = E_1,\ Z^* = Z_1$ és
b)$Z = Z_1,\ E = E_1$ 4-re lép.
6. Generáljunk egy V véletlenszámot (Általában egyenletes eloszlással 0-1 között). Ha $V \leq$ hőmérséklet, akkor $E = E_1,\ Z = Z_1$, 4. lépés.
7. Megáll. Megőrizzük az $E^*,\ Z^*$ párt.

Tabu keresési eljárás. A módszer futása közben ideiglenesen bizonyos megengedett megoldásokat tiltottnak nyílvánít át, így nyitva meg az utat a lokális optimumból való kijutáshoz. Az ideiglenesen tiltott megoldásokat egy tabu listában tarjuk nyilván. További jellemző az aspirációs szint (AL) bevezetése. Ez jelzi azt, hogy milyen feltétellel fogadunk el mégis egy tiltott lépést, ha ez az aktuálisan vizsgált megoldáshoz képest egyértelműen előnyösebb. A szimulált lehülés 4 előzetes döntéséhez további kettő csatlakozik:

- Mekkora legyen a fenntartandó tabu lista (L) maximális mérete?
- Hogyan határozzuk meg az aspirációs szintet?

Szemben a szimulált lehüléssel, ahol szomszédságon egyetlen véletlenül generált megengedett megoldást értettünk, itt a szomszédok az összes, megengedett változtatással nyerhető megoldások halmaza, és ezek legjobbikába próbálunk előrelépni. 1. Állítsuk be a paramétereket. $L = 0$
2. Állítsunk elő egy kiindulási megengedett megoldást, E-t Z

célfüggvényértékkel. $E^* = E,\ Z^* = Z,\ AL = Z,\ t = 0$
3. $t = t + 1$ Ha meghaladtuk a maximális iterációszámot, 6. különben 4.
4. Állítsunk elő megengedett módon egy szomszédos E_1 megoldást, amely legalább egyet kielégít a következő két feltétel közül:
a)E_1 szomszédos, $Z_1 < AL$, minimális a szomszédságban
b)E_1 szomszédos, nincs a tabulistán, Z_1 minimális az ilyen szomszédokra nézve.
HaE_1 nem létezik: 6. különben 5.
5. $E = E_1,\ Z = Z_1\ E_1 \to L$ Ha $Z_1 < Z^*$, akkor $E^* = E_1,\ Z^* = Z_1,$ $AL = Z_1$ Lépés 3-ra.
6. Leáll. Visszaadja E^*-ot és Z^*-ot.

Vállalati tapasztalatok szerint a nagyobb ipari feladatokra mind számításigény, mind eredményesség tekintetében a szimulált lehüléses heurisztikát célszerűbb alkalmazni. Megvalósítástól függően a véletlent használó algoritmusok hajlamosak lehetnek sok közepes méretű cella létrehozására, és így eleshetünk attól a haszontól, amit esetenként a méretkorláthoz közeli nagyságú cellák létrehozatala jelent. Ezért ha ez a haszon létezik, akkor külön figyelmet kell fordítanunk ennek a szempontnak az érvényesítésére.
A továbbiakban legyen m a gépek, n a termékek száma. A mátrixokkal dolgozó algoritmusok inputja legyen $A = (a_{ij})$ mátrix.

4.5. MOM

for l from 1 to $max\_iteracioszam$
do
for i from 1 to $gepszam$
do
$M_{yi} := \sum_{j=1}^{n} j a_{ij}$
od
if sorok M_{yi} szerint növekvő sorrendben vannak
then
RETURN
else
sorok rendezése M_{yi} szerint növekvő sorrendbe
fi
for i from 1 to n
do
$M_{xi} := \sum_{j=1}^{m} j a_{ji}$
od
if oszlopok M_{xi} szerint növekvő sorrendben vannak
then
RETURN
else
oszlopok rendezése M_{xi} szerint növekvő sorrendbe
fi
od

4.6. DMOM

for k from 1 to n
do
$W_{yk} := \sum_{j=1}^{n} a_{kj}$
$M_{yk} := \sum_{j=1}^{n} j a_{kj}$

$r_{yk} := \frac{M_{yk}}{W_{yk}}$
od
sorok rendezése növekvő r_{yk} szerint
$z := 1$
while $z < max\_iteracioszam$
do
$i := 1$
while $i \leq m - 1$
do
for k from $i + 1$ to m
do
$DMR_{yik} := \frac{\sum_{j=1}^{n} |a_{ij} - a_{kj}| j}{\sum_{j=1}^{n} (a_{ij} + a_{kj})}$
od
if DMR_{yik}-k növekvő sorrendben vannak
then
RETURN
else
sorok rendezése növekvő sorrendbe DMR_{yik} szerint
fi
$i := i + 1$
od
if első_oszloprendezés
then
for k from 1 to m
do
$W_{xk} := \sum_{j=1}^{m} a_{jk}$
$M_{xk} := \sum_{j=1}^{m} j a_{jk}$
$r_{xk} := \frac{M_{xk}}{W_{xk}}$
od
rendezzük az oszlopokat növekvő sorrendbe r_{xk} szerint
fi
$i := 1$
while $i \leq n$
do
for k from $i + 1$ to n
do
$DMR_{xik} := \frac{\sum_{j=1}^{m} |a_{ji} - a_{jk}| j}{\sum_{j=1}^{m} (a_{ji} + a_{jk})}$
od
if DMR_{xik}-k növekvő sorrendben vannak
then
RETURN
else
rendezés növekvő DMR_{xik} szerinti sorrendbe
fi
$i := i + 1$
od
$z := z + 1$
od

4.7. CELLA

Legyen S_{ij} a szállított (fizikai) mennyiség i cellából a j cellába.
for i from 1 to m
do
$C_i := \{m_i\}$
od
for i from 1 to m
do
for j from 1 to m
do
$G_{ij} := \frac{S_{ij}+S_{ji}}{M_i+M_j}$
od
od
(i^*, j^*) :=argmaxG_{ij}
while $G_{i^*j^*} \geq küszöb$
do
$k := min\{i^*, j^*\}$
$k' := max\{i^*, j^*\}$
for i from 1 to m
do
$G_{ki} := G_{ki} + G_{k'i}$
$G_{ik} := G_{ik} + G_{ik'}$
$G_{k'i} := 0$
$G_{ik'} := 0$
od
$C_k := C_k \ \cup C_{k'}$
$C_{k'} := \emptyset$
od

4.8. Szimleh

$E, \ Z$
$E^* := E$
$Z^* := Z$
a hőmérséklet beállítása
$it := 0$
while $it \leq max\_iterációszám$
do
$it := it + 1$
$s := 1$
while $s \leq max\_lokális\_keresésszám$
do
$s := s + 1$
E_1 szomszédos megengedett megoldás és Z_1 előállítása
if $Z_1 < Z^*$
then
$E^* := E_1$
$Z^* := z_1$
$Z := Z_1$
$E := E_1$
hőmérsékletbeállítás
elseif rand(1)<hőmérséklet
then

$Z := Z_1$
$E := E_1$
hőmérséklet beállítása
fi
od # lokális keresés vége
hőmérséklet beállítása
od

4.9. TABU

$L := \emptyset$
$E,\ Z$ előállítása
$E^* := E$
$Z^* := Z$
$AL := Z$
$it := 1$
while $it \leq max\_iterációszám$
do
$it := it + 1$
if E_1 megengedett szomszédos megoldás előállítható
then
$E := E_1$
$Z := Z_1$
$L \leftarrow E_1$
if $Z_1 < Z^*$
then
$E^* := E_1$
$Z^* := Z_1$
$AL := Z_1$
fi
fi
od

5. Tesztprogramok

Az irodalmi adatok között szerepelnek teszteredmények a különböző heurisztikák teljesítményére vonatkozóan, de ezek jellemzően csak olyan algoritmusokkal vetik össze az aktuálisan tárgyaltat, amely hasonló alapelvű. Például mátrixrendező algoritmust mátrixrendezővel. Ajánlatos lenne azonban lényegesen eltérő algoritmusok viselkedéséről is információt kapni, mert akkor egy valós feladatnál előre ki tudnánk választani az alkalmazásra érdemes heurisztikát. Ezért négy heurisztikát program formájában megvalósítottam.

5.1. A tesztprogramok közös jellemzői

A megoldandó feladatokban adott gépparkon kell a termékeket gyártanunk. Két program egy gyártási útvonalat enged meg termékenként, kettő pedig többet is. A több útvonalas algoritmusok természetesen az egy útvonalas problémát is megoldják, de ennek csak egyik esetben lesz értelme, mert az egyik többútvonalas program egy egyútvonalas közvetlen általánosítása. Egy géptípusból egy példányt tételezünk fel. A termékek mennyiségét valamilyen közös mértékegységben ismerjük. Tudjuk, hogy adott gyártási művelet elvégzése mennyi időt vesz

igénybe, és feltesszük, hogy a gépek kapacitása elégséges. A kapacitáskorlát bevezethető valamennyi programba, amennyiben szükséges. Korlátozást teszünk a cellák méretére.
A célfüggvény két részből tevődik össze. Az első rész bünteti a cellaközti szállítások tényét, a második ugyancsak a cellaközti szállítást bünteti, de figyelembe veszi azt is, hogy milyen időigényű műveletek között történik a szállítás. Feltételezzük, hogy két időigényes művelet között ártalmasabb az útvonal megtörése, mert vélhetőleg egy munkáscsoport alkalmazása az adott termék gyár-tá-sá-hoz a termék jobb ismerete folytán magasabb gyártási színvonalat eredményez. A két részt kívülről megadható a, b paraméterekkel súlyozzuk.
$a\sum_{j=1}^{n}\sum(j$ terméket cellák között szállítjuk$)(j$ termék mennyisége$)+$
$b\sum_{j=1}^{n}\sum(j$-t cellák között szállítjuk$)(j$ mennyisége$)$(azon műveletek elvégzési idejének összege, amelyek között a szállítás történik)
A programok mindegyike kimenetként megadja a gépcellákat, a termékeket nem rendeli cellákhoz, de minden termékre meghatároz egy ajánlott gyártási útvonalat. Szintén szolgáltatják a célfüggvény értékét. Ha a termékek cellához rendelése is szükséges, akkor ez egyszerűen elvégezhető, hasonlóan, ahogyan a szekvenciális modellek második lépése történt. Valamennyi program a Matlab programcsomag használatával készült. A megoldandó feladatot a feladatgenerátor készítette két file, $adat1$ (gépszám, termékszám, termékmennyiségek) és $adat2$ (gyártási útvonalak) írják le.

5.2. Diffmom

Mátrixrendező algoritmus, a korábban ismertetett MOM és DMOM módszerek ötvözete. Az eredeti algoritmusok $0-1$ mátrixon dolgoztak, és menet közben nem törődtek a célfüggvénnyel, mert nem is voltak ideiglenes celláik, nem volt mit számolni. Bennem felmerült azonban, hogy mégis lehetséges lenne összekapcsolásuk a fentebb leírt célfüggvénnyel. MOM és DMOM közel-blokkdiagonális mátrixot hoztak létre, és a végeredmény jóságát az jelezte, mennyi 1 maradt a blokkokon kívül, mert ott történt cellaközti szállítás. Most a szállítás nem egységnyivel rontja a célfüggvényt, hanem a mennyiség és a műveletidők egy függvényével, de az elv ugyanaz. El kell érni, hogy a blokkokon belüli számok összege a lehető legnagyobb legyen, mert ez jelenti a legnagyobb megtakarítást. Nem lehetséges most sem közvetlenül a célfüggvényt felhasználnunk, helyette a következő MM mátrixon dolgozunk:

$$MM_{ij} = \text{(j termék által i gépen töltött összes idő)(j mennyisége)}$$

Várhatóan kevés lesz a cellaközti szállítás, és nagy időigényű műveletek között nem törjük el az útvonalat.
Tapasztalataim szerint a DMOM sokkal lassabb, mint a MOM, bár jobb a blokkok létrehozásában. Megfigyeltem, hogy MM-en, amely gyakran sok nemnulla elemet tartalmaz és elemei távol állnak az egészértékűségtől, DMOM nem mindig a főátlóra tömöríti a blokkokat, hanem alkalmanként távolabb is kialakít közel téglalap alakú, nemnulla elemekből álló részeket. Az oszlopok rendezése pedig egyenértékű a mátrix transzponáltján a sorok rendezésével. Ezért a felépített MM mátrixra és transzponáltjára felváltva négy-négy alkalommal alkalmaztam az MOM módszert, majd egyszer a DMOM módszert. Leállt a rendezés, ha MOM sokszor találta azt, hogy nem történt változás a gépek sorrendjében (nem az első ilyen esetnél áll le, mert akkor még a transzponálton végzett munka miatt lehet változás), ha a lépésszám elért egy, a feladat méretétől függő határt ($1.25max\{n, m\}$), vagy a beépített DMOM eljárás nem

talált javítanivalót. Jellemzően az iterációszámra vonatkozó korlát miatt fejeződött be a program futása, de ekkor már olyan állapotba került a mátrix, hogy az iterációszám növelése nem hozott javulást. A cellák létrehozatalához megvizsgáltam a szomszédos és másodszomszédos sorok skalárszorzatait, és ezek csökkenő sorrendjében vontam össze, abból kiindulva, hogy ha jó blokkos szerkezet jön létre, akkor a blokkoknál sok nemnulla elem kerül egymás alá a mátrixban. Az eljárás ezen része bizonyára sokat javítható. Nem veszi észre például az olyan szerkezetet, ahol egy nemnulla elemet két nulla követ, és ez a szerkezet minden sorban jobbra eggyel eltolva ismétlődik. A generált véletlen példákban mindenesetre ilyen extrém szerkezet nem volt jellemző.

Hívása:$diffmom(file1, file1, a, b, maxmeret)$

5.3. Egydimenziós szimulált lehülés -Szimleh1

A szimulált lehülés alkalmazásához szükség van egy jó kiinduló megoldásra. Az ezt előállító eljárás egy gépszám méretű négyzetes mátrixon dolgozott. Kiinduláskor minden gép egy cella, ezért a mátrix (i, j) eleme azt mutatja, hogy az i gépről j-re történő szállítások mennyivel járulnak hozzá (rontva) a minimalizálandó célfüggvényhez. $meretmax = \lceil 0.6 maximális\ cellaméret \rceil$ nagy cellákat építünk fel. Ehhez kiválasztjuk azt a gépet, amelyről a legtöbb elszállítás történik, majd hozzáveszünk még $meretmax - 1$-et, azokat, ahová a legtöbb a szálítás. Ezen gépek részét kivágjuk a mátrixból, és megismételjük az eljárást. Közben megfelelően módosítjuk a célfüggvény értékét is. A végén kapunk

$$\left\lfloor \frac{gépszám}{meretmax} \right\rfloor$$

sok közepes méretű cellát, és még néhány egyeleműt.

A megvalósított program (némileg más, mint a korábban szerepelt egyféle lehetséges folyamatábra):

$szimleh1(file1, file2, maxmeret, maxpr, maxpr1, maxpr2, maxsz, a, b)$

Itt a, b a célfüggvény súlyai. A kiindulási megoldásból egymástól függetlenül $maxpr$ alkalommal indulunk ki a sztochasztikus algoritmussal. A kiindulási hőmérséklet (rosszabb célfüggvényértékű megoldás elfogadásának valószínűsége) 0.85. A hőmérséklet minden egyes elfogadáskor $maxpr1$-szeresére változik. Ha a hőmérséklet eléri 10^{-3}-t, akkor a program véget ér. Ez $maxpr1$-től függően százas nagyságrendet határol be az elfogadások számára, azaz ennyiszer lépünk bármilyen új megoldásba. Nagyon nagy problémák esetén ezen változtatni lehet, például alkalmazható a korábbi folyamatábra ötlete, ahol csak bizonyos számú elfogadás után csökkent a hőmérséklet. Egyes megfigyelések szerint azonban a szimulált lehüléses algoritmusok jól válaszott kiinduló megoldást kapva hamar megtalálják végeredményüket. Minden egyes hőmérsékleten legfeljebb $maxpr2 * m$, ($m =$ gépszám) sok próbát teszünk elfogadott lépés megtételére. Ehhez először kiválasztunk véletlenszerűen egy gépet. Ezután legfeljebb 19 alkalommal keresünk számára véletlenszerűen egy alkalmas befogadó csoportot. Ha 19 alkalommal is a méretkorlát miatt nem találtunk befogadó csoportot, akkor ezt a gépet leválasztjuk csoportjáról, és ezután egyedül alkot csoportot. Mind az áttételnek, mind a leválasztásnak feltétele az elfogadás: vagy javul a célfüggvény (feltétel nélküli elfogadás), vagy egy 0-1 közötti egyenletes eloszlásból vett véletlenszám kisebb, mint $e^{-\frac{\Delta f}{T}}$ (feltételes elfogadás), Δf a célfüggvény,megváltozása, T a hőmérséklet. A leválasztás engedélyezése mindig ront, ezért csak feltételesen történhet. Elfogadáskor a hőmérséklet a fentiek szerint csökken.

Bevezettem továbbá egy újítást: feltételeztem, hogy sokszor próbálkozhat a

program olyan cserével, mely az adott lépéskor nem változtatja a célfüggvényt. Ennek az esetnek a kezeléséről nem olvastam, illetve sejthetően csak feltételesen elfogadható, mert nincs javulás. Én ehelyett a rosszabb célfüggvényérték elfogadását kötöttem csak feltételhez, és az azonos célfüggvényértékű állapotot is elfogadtam legfeljebb $maxsz$ alkalommal két feltétel nélküli elfogadás között.
Hívása: szimleh(file1,file2,maxmeret,maxpr,maxpr1,maxpr2,maxsz,a,b)

5.4. Kétdimenziós szimulált lehülés -Szimleh2

A Szimleh2 program a Szimleh1 módosítása azon eset kezelésére, amikor alternatív gyártási útvonalak lehetségesek egy termékre. Az útvonal módosítása hossza miatt nagyobb nehézségbe ütközik, mint a gépcsere, ezért minden gépcserére irányuló próbálkozást öt útvonalváltási kísérlet követ.
Hívása: szimleh2(file1,file2,a,b,maxmeret,maxpr,maxpr1,maxpr2,maxsz)

5.5. Kontrakciós cellázóprogram alternatív útvonalakkal -Cell (CELLA)

A megvalósított algoritmus hűen követi a felvázolt menetet. Nem világos azonban az eddig felírtakból, hogy mit kezdjünk az alternatív útvonalakkal, mert azokkal a heurisztika nem foglalkozik. Azzal a feltevésel élünk, hogy a különböző gyártási útvonalakban gyakran előfordulhatnak közös szakaszaik a gyakorlati feladatoknál, és ezért az átlaguk használható információt ad. Ebből tobábbmenve következik az ötlet, hogy a szállítási mátrixot az útvonalak átlaga alapján számítsuk ki. A mátrix (i,j) eleme azt adja meg, hogy az i gépről j gépre történő szállítások mennyivel rontják a célfüggvény értékét, tehát most is szállítás $\neq$ fizikai szállítás. Ezután sorozatosan összevonjuk azt a két gépcsoportot, amelyek között a szállítás maximális, ha még meghaladja az előre megadott korlátot. A korlát megadása nyilván jelentősen befolyásolja a teljesítményt. Egy bizonyos gyakorlati felhasználó magának kikísérletezheti a legjobbnak tűnő értéket, viszont tesztmérésekhez nem használhatunk légből kapott számértéket. Ezért a programba beépítettem a $pontossag$ nevű változót, amely révén az összevonásokat a generált problémától függetlenül szabályozhatjuk: megtörténik az összevonás, ha a gépcellák közötti kétirányú szállítás összege meghaladja a rendszerben előforduló összes szállítás és a $pontossag$ hányadosát. Az összes szállítás könnyen kiszámítható: ez a kiindulási szállítási mátrix elemeinek összege, mert ekkor még minden gép külön cellát alkot. Tehát $pontossag = 200$ esetén félszázalékos pontossággal számolunk. Hívása:
$cell(file1, file2, merethatar, pontossag, a, b)$

5.6. A feladatgenerátor -generate.m

Hívása: $generate(gepszam, termekszam, minvonal, maxvonal, minhossz,$ $maxhossz, csoportszam, minmenny, maxmenny, erange)$
Itt a paraméterek jelentése a következő:
$gepszam, termekszam$: egyértelmű
$minvonal, maxvonal$: egy termékhez legalább/legfeljebb hány útvonalat generáljunk. Például ha egyútvonalas algoritmus használunk, akkor $minvonal = maxvonal = 1$
$minhossz, maxhossz$: egyes útvonalak minimális/maximális hossza
$csoportszam$: annak a helyzetnek a szimulálására hivatott, amikor a gépek között olyan csoportok szerepelnek, amelyek egymástól teljesen függetlenek, azaz nem létezik olyan gyártási terv, amelyben szerepelne a két csoport gépei

közötti szállítás. Ilyen helyzet a gyakorlatban egyszerűen felismerhető, ezért kevés alkalommal kapott 1-től eltérő értéket.
$minmenny, maxmenny$: egyes termékek fizikai mennyiségeinek alsó és felső korlátja
$erange$: a termékeken végzett műveletek elvégzési idejének szórását szabályozza. Egy művelet elvégzési idejét egyenletes eloszlás szerint választjuk a $[0.1, erange]$ intervallumból.
Az megengedett, hogy egy gép önmaga után szerepeljen, mert egyféle gépen egymástól meglehetősen eltérő müveletek is végezhetőek lehetnek.

Az adatfile-ok szerkezete.

1. adatfile: gépszám - termékszám - termékmennyiségek

2. adatfile:
1. termék útvonalainak száma
1. útvonal hossza - 1. gép - itt töltött idő - 2. gép - itt töltött idő .....
2. útvonal hossza - 1. gép - itt töltött idő - 2. gép - itt töltött idő .....
.
1. termék utolsó útvonalának leírása
2. termék útvonalainak száma
És így tovább.

6. Példák

Néhány futási részeredményen bemutatjuk, hogyan müködnek az egyes algoritmusok. A szimulált lehüléshez csak egydimenziós esetben adunk példát, mert a többdimenziós eset útvonalváltása nem jelent sok minőségi változást.

Adatok az egyútvonalas programokhoz. adat1:
6,12,
15,12,21,21,25,17,19,23,12,12, 19,21,
Tehát ebben a kis példában 6 gép és 12 termék szerepel.
adat2:
1,
2,
1,0.475737,5,0.630079,
1,
3,
4,0.182768,4,0.474399,5,0.919289,
1,
2,
1,0.762474,2,0.669375,
1,
3,
3,0.322335,6,0.750394,5,0.686367,
1,
3,
6,0.345439,3,0.789845,3,0.313997,
1,
2,

1,0.537866,6,0.918287,
1,
3,
4,0.564663,2,0.987978,3,0.339530,
1,
3,
1,0.550636,3,0.349374,6,0.576773,
1,
3,
1,0.785363,5,0.845036,1,0.114281,
1,
3,
4,0.762602,5,0.999512,6,0.309875,
1,
2,
4,0.632002,6,0.470873,
1,
2,
3,0.583574,3,0.358491,

6.1. Diffmom

$diffmom('adat1','adat2',1,1,3)$

7.14	0	16.01	0	0	9.14	0	12.66	10.80	0	0	0
0	0	14.06	0	0	0	18.77	0	0	0	0	0
0	0	0	6.77	27.60	0	6.45	8.04	0	0	0	19.78
0	7.89	0	0	0	0	10.73	0	0	9.15	12.01	0
9.45	11.03	0	14.41	0	0	0	0	10.14	11.99	0	0
0	0	0	15.76	8.64	15.61	0	13.27	0	3.72	8.95	0

A kiindulási mátrix két tizedesre kerekítve az értékeket. Ebből a mátrixból kiindulva végzünk sor- és oszlopcseréket MOM vagy DMOM szerint. Az első MOM lépés eredménye:
Sorcserék:

9.45	11.99	11.03	10.14	0	0	0	14.41	0	0	0	0
0	9.15	7.89	0	0	0	12.01	0	10.73	0	0	0
7.14	0	0	10.80	12.66	9.14	0	0	0	16.01	0	0
0	3.72	0	0	13.27	15.61	8.95	15.76	0	0	8.64	0
0	0	0	0	8.04	0	0	6.77	6.45	0	27.60	19.78
0	0	0	0	0	0	0	0	18.77	14.06	0	0

Ezután a transzponált mátrixon végzünk sorcseréket.

11.03	7.89	0	0	0	0
11.99	9.15	0	3.72	0	0
9.45	0	7.14	0	0	0
10.14	0	10.80	0	0	0
0	12.01	0	8.95	0	0
14.41	0	0	15.76	6.77	0
0	0	12.66	13.27	8.04	0
0	0	9.14	15.61	0	0
0	10.73	0	0	6.45	18.77
0	0	16.01	0	0	14.06
0	0	0	8.64	27.60	0
0	0	0	0	19.78	0

Ezt ismételgetjük, illetve minden ötödik lépésben a DMOM rendezését alkalmazzuk. A leálláskor kapott mátrix szerint elvégezzük az összevonást. Most az eredményvektor:
5 4 4 4 5 5 Jelentése: az 1., 5. és 6., illetve a 2., 3. és 4. gépek alkotnak egy csoportot.

6.2. Szimleh1

$szimleh1('adat1',' adat2', 3, 2, 0.98, 1.5, 4, 1, 1)$

A determininisztikus rész kezdőmátrixa:

0	51.07	43.70	0	63.15	41.75
0	0	44.22	0	0	0
0	0	0	0	0	87.83
0	48.50	0	0	61.87	39.95
23.51	0	0	0	0	27.71
0	0	53.38	0	51.17	0

Összes szállítás: 637.8300

$gepek = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{vmatrix}$

Az első sor k. száma jelzi, hogy a k. gép csoportjában melyik a legkisebb sorszámú gép. Az alatta lévő szám megadja a csoporthoz tartozó gépek számát. Kezdetben minden gép külön csoportot alkot. A gépek a determinisztikus rész után:

$$\begin{vmatrix} 1 & 2 & 3 & 2 & 1 & 6 \\ 2 & 2 & 1 & 2 & 2 & 1 \end{vmatrix}$$

$\lceil 0.6 * 3 \rceil = 2$, ezért legfeljebb 2 méretü csoportok vannak a sztochasztikus algoritmus kiindulási megoldásában. A kiinduló célfüggvényérték:
$cel = 502.6660$
A *legjobb* vektor tartalmazza az eddigi legjobb célfüggvényértéket és a *gepek* jelzővektorát.
$legjobb = 502.6660$ 1 2 3 2 1 6
$p = 0.8500$
$mit = 4$ Ezt a gépet próbáljuk meg eltenni.
$hova = 3$
Jelentése: a *gépcsoportok* közül a harmadikba próbáljuk tenni. *hova* tehát mindig legfeljebb annyi, mint ahany gépcsoport éppen van.
$ujcel = 551.1662$
feltételesen elfogadható
$rnd = 0.9593$

$ex = 1.6580e - 25$

$$\begin{vmatrix} 1 & 2 & 3 & 2 & 1 & 6 \\ 2 & 2 & 1 & 2 & 2 & 1 \end{vmatrix}$$

$legjobb = 502.6660$ 1 2 3 2 1 6
$mit = 2$
$hova = 1$
$ujcel = 500.0973$
javulás

$$\begin{vmatrix} 1 & 1 & 3 & 4 & 1 & 6 \\ 3 & 3 & 1 & 1 & 3 & 1 \end{vmatrix}$$

$legjobb = 500.0973$ 1 1 3 4 1 6
$p = 0.8330$
$mit = 6$
$hova = 2$
$ujcel = 358.8865$
javulás

És így tovább.

6.3. Cella

adat1:
6,12,
15,20,21,24,23,17,21,23,13,25, 25,17,
adat2:
2,
2,
1,0.241208,6,0.235375,
2,
4,0.230500,1,0.238454,
2,
2,
2,0.214125,4,0.356316,
3,
4,0.303440,3,0.213123,4,0.367657,
1,
3,
3,0.349733,6,0.222026,2,0.348879,
1,
3,
1,0.360024,3,0.164808,5,0.360167,
1,
2,
3,0.134884,2,0.197090,
1,
2,
6,0.271197,1,0.305380,
1,
3,

2,0.229285,6,0.244406,1,0.370219,
2,
2,
5,0.139642,6,0.186991,
3,
2,0.301217,6,0.373636,1,0.347269,
2,
2,
5,0.244422,6,0.154025,
3,
6,0.253206,1,0.232049,6,0.201114,
2,
2,
6,0.100822,1,0.185228,
3,
5,0.342956,2,0.267032,5,0.139955,
1,
3,
6,0.174410,5,0.326936,5,0.137211,
2,
3,
4,0.326282,1,0.335708,1,0.121533,
2,
5,0.297761,2,0.151272,
$cell('adat1',' adat2', 3, 200, 1, 1)$
A felépített szállítási mátrix:

0	0	36.5960	0	0	20.3899
0	0	0	15.7044	17.5873	50.2083
0	30.6354	0	15.8078	36.5994	33.0069
25.1441	0	15.1656	0	0	0
0	32.4416	0	0	0	24.3462
106.2291	32.9890	0	0	37.5337	0

$összes\ szállítás = 530.3848$
$elvárt\_javulás = 2.6519$
$maxi = 1$
$maxj = 6$
Jelentése: (1,6) és (6,1) pozíciókon álló értékek adtak maximális összeget, vagyis az 1. és 6. cella közötti szállítás a legnagyobb. $max = 126.6191$
$összevonás\ lehetséges$
$gépek$:

1	2	3	4	5	1
2	1	1	1	1	2

A szállítási mátrixban is elvégezzük az összevonást.

0	21.9927	24.3973	0	25.0224
33.4722	0	0	15.7044	17.5873
22.0046	30.6354	0	15.8078	36.5994
16.7627	0	15.1656	0	0
16.2308	32.4416	0	0	0

$maxi = 1$
$maxj = 2$

$max = 55.4649$
összevonás lehetséges

$$\begin{vmatrix} 1 & 1 & 3 & 4 & 5 & 1 \\ 3 & 3 & 1 & 1 & 1 & 3 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 18.2980 & 7.8522 & 27.5605 \\ 31.8212 & 0 & 15.8078 & 36.5994 \\ 12.5720 & 15.1656 & 0 & 0 \\ 28.3939 & 0 & 0 & 0 \end{vmatrix}$$

$maxi = 1$
$maxj = 4$
$max = 55.9544$
nincs összevonás méretkorlát miatt

$$\begin{vmatrix} 1 & 1 & 3 & 4 & 5 & 1 \\ 3 & 3 & 1 & 1 & 1 & 3 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 18.2980 & 7.8522 & 0 \\ 31.8212 & 0 & 15.8078 & 36.5994 \\ 12.5720 & 15.1656 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$maxi = 1$
$maxj = 2$
$max = 50.1192$
nincs összevonás méretkorlát miatt

$$\begin{vmatrix} 1 & 1 & 3 & 4 & 5 & 1 \\ 3 & 3 & 1 & 1 & 1 & 3 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 0 & 7.8522 & 0 \\ 0 & 0 & 15.8078 & 36.5994 \\ 12.5720 & 15.1656 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$maxi = 2$
$maxj = 4$
$max = 36.5994$
összevonás lehetséges

$$\begin{vmatrix} 1 & 1 & 3 & 4 & 3 & 1 \\ 3 & 3 & 2 & 1 & 2 & 3 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 0 & 7.8522 \\ 0 & 0 & 10.5385 \\ 12.5720 & 10.1104 & 0 \end{vmatrix}$$

$maxi = 2$
$maxj = 3$
$max = 20.6490$
összevonás lehetséges

$$\begin{vmatrix} 1 & 1 & 3 & 3 & 3 & 1 \\ 3 & 3 & 3 & 3 & 3 & 3 \end{vmatrix}$$

$$\begin{vmatrix} 0 & 5.2348 \\ 8.3814 & 0 \end{vmatrix}$$

$maxi = 1$
$maxj = 2$
$max = 13.6162$
nincs összevonás méretkorlát miatt

$$\begin{vmatrix} 0 & 0 \\ 0 & 0 \end{vmatrix}$$

$max = 0$
A végeredmény:

$$\begin{vmatrix} 1 & 1 & 3 & 3 & 3 & 1 \\ 3 & 3 & 3 & 3 & 3 & 3 \end{vmatrix}$$

A szekvenciális algoritmus befejező részében minden termékhez kiválasztjuk a legkevesebb szállítással járó gyártási módszert. Az egyes termékekhez választott technológiák sorszáma:
2,2,1,1,1,1,1,2,2,2,1,2
$cel = 259.8642$

7. Tesztek és eredményük

Megjegyzés. A programok tesztelése az egyetemi UNIX-labor különböző számítógépein történt. Ezért a relatív műveletigény és időigény adatait ajánlatosabb figyelemben részesíteni, a helyenként szereplő időadatok csak tájékoztató jellegűek, és a mi vizsgálataink szempontjából, a gyors számítástechnikai fejlődés közepette kisebb jelentőségúek. Egyféle teszt részeredményei ugyanazon a számítógépen keletkeztek. A lefutási idők másodpercben mértek.
A feladatnagyságot a gépek m számával jellemezzük, a termékek számát leggyakrabban (ha nincs külön jelezve) egyenletes eloszlás szerint választjuk $[2m, 3m]$-ből.
Sokszor fog szerepelni néhány mérés eredményeinek összege. Minden alkalommal E, I, M rendre a célfüggvényértékek összege, a lefutási idők összege, a lebegőpontos műveletek számának összege. Az indexek a programnevek rövidítései: D, S1, C, S2.

1. teszt.

- Összehasonlítjuk a bemutatott MOM és DMOM módszereket együtt hasz-ná-ló Diffmom programot egy kizárólag a DMOM eljárást alkalmazó programmal.
- Vizsgálandó, melyik algoritmus szolgáltat jobb célfüggvényű eredményt, ha a korábban leírt cellázó algoritmus kapcsolódik hozzájuk.
- 20 összehasonlítás
- $generate(gepszam = 10, termekszam = 25, 1, 1, 2, 6, 1, 8, 16, 5)$

Eredmény. 10-10 alkalommal bizonyult jobbnak egyik illetve másik program.

2. teszt.

- A vegyes DMOM-MOM és a tiszta DMOM programok időigényének összehasonlítása.
- Feladatméretenként 1-1 problémát oldunk meg, a trend érzékelhető

Eredmény. idő1,idő2:
a vegyes DMOM-MOM, illetve a tiszta DMOM lefutási ideje másodpercben

m	idő1	idő2	idő2/idő1
12	25.52	86.49	3.39
13	33.74	113.14	3.35
14	32.79	118.68	3.62
15	43.55	148.88	3.42
17	60.13	218.77	3.64
20	97.52	356.58	3.66
23	155.66	574.18	3.69
26	226	826	3.65

A vegyes algoritmus kevesebb, mint harmadannyi idő alatt végez, mint a tiszta, és az időeltérésben előnye enyhén növekedni látszik. Ezt összevetve az 1. teszt eredményével azt láthatjuk, hogy a vegyes Diffmom programot érdemesebb gyakorlati feladatok megoldására felhasználni.

3. teszt.

- Diffmom, Cella, Szimleh1 összehasonlítása a célfüggvény szempontjából.
- 20 próba.
- $m = 15$ méret, $pontossag = 200$
- $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 5)$
- $maxmeret = 6$
- Szimleh1-hez $maxpr = 3,\ maxpr1 = 0.98,\ maxpr2 = 1.4\, maxsz = 4$

Eredmény. Minden esetben Szimleh1 jobban szerepelt, mint Diffmom, és ez jobban szerepelt, mint Cella.

$\frac{E_{S1}}{E_D}$	$\frac{E_C}{E_D}$	$\frac{I_{S1}}{I_D}$	$\frac{I_C}{I_D}$	$\frac{M_{S1}}{M_D}$	$\frac{M_C}{M_D}$
0.71	1.47	0.031	0.014	0.022	0.011

Az idő- és a műveletigény szempontjából Diffmom a sokszorosát igényelte a másik két programénak. Eszerint sajnos még a nem tisztán DMOM eljárás felhasználhatósága is kérdéses.

4. teszt.

- Ugyanolyan, mint a 3. teszt, csak feltételezünk két független gépcsoportot.
- $generate(15, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 2, 10, 20, 5)$

Eredmény.

$\frac{E_{S1}}{E_D}$	$\frac{E_C}{E_D}$	$\frac{I_{S1}}{I_D}$	$\frac{I_C}{I_D}$	$\frac{M_{S1}}{M_D}$	$\frac{M_C}{M_D}$
0.65	1.66	0.031	0.012	0.022	0.011

A célfüggvény szempontjából csekély javulás látszik a Diffmom javára, a műveletszámok aránya nem módosult. Ezért a következő vizsgálatokból Diffmom kimarad.

5. teszt. A 15 gépes probléma meglehetősen kicsi. Érdeklődésre tarthat számot, hogyan működnek a heurisztikák nagyobb méretű feladatokra.

- 10-10 próba minden méretre
- $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 5)$
- $maxmeret = 6$, $pontossag = 200$
- Szimleh1-hez $maxpr = 3$, $maxpr1 = 0.98$, $maxpr2 = 1.4\, maxsz = 4$

Eredmény. Rögzítjük azt is, hogy hány alkalommal szerepel a célfüggvény tekintetében egyik vagy másik algoritmus jobban, illetve a lehüléses algoritmus átlagosan hány másodperc alatt futott le.

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$
20	1.74	0.35	0.46	10	0	0	4.30
25	1.51	0.42	0.57	10	0	0	4.89
30	1.44	0.32	0.44	10	0	0	9.55
35	1.36	0.36	0.46	10	0	0	11.79
40	1.32	0.33	0.45	10	0	0	16.24
45	1.30	0.27	0.42	10	0	0	22.87
50	1.27	0.23	0.36	10	0	0	29.88
65	1.21	0.18	0.32	10	0	0	55.13
80	1.17	0.22	0.30	10	0	0	90.64
95	1.15	0.20	0.26	10	0	0	145.05
130	1.11	0.26	0.23	10	0	0	246.30

Az műveletigényben viszonylagosan egyre olcsóbb Cella a méret növekedésével jelentősen javuló viszonylagos teljesítményt mutat, bár a javulás üteme lassul. A kedvezőbb teljesítménymutató oka az lehet, hogy ha m nő miközben a $maxmeret$ változatlan, akkor csak a legnagyobb szállítások megszüntetésére van lehetőség, ezért mindkét algoritmus csak majdnem ugyanazokat, a leg-ké-zen-fek-vőbb összevonásokat kísérli meg. Esetlegesen javíthatunk Szimleh1 javára, ha benne az előzetesen jelzett módon megnöveljük az iterációszámot, ez viszont a Cella alkalmazását viszonylagosan még olcsóbbá tenné.

6. teszt. Mint az 5. teszt, csak most nagyobb szórást engedélyezünk az úthosszakban.

- a) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{3} \rceil, 1, 10, 20, 5)$
- b) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{2} \rceil, 1, 10, 20, 5)$

Eredmények.

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.58	0.30	0.47	10	0	0	7.01	
25	1.38	0.50	0.59	10	0	0	6.48	a) eset
30	1.37	0.32	0.48	10	0	0	13.22	
35	1.31	0.34	0.46	10	0	0	16.65	

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.49	0.38	0.47	10	0	0	5.88	
25	1.34	0.55	0.57	10	0	0	6.64	b) eset
30	1.33	0.38	0.46	10	0	0	14.29	
35	1.30	0.36	0.42	10	0	0	18.50	

A Cella jobban teljesít, viszont továbbra is minden esetben gyengébb célfüggvényértéket ad, mint Szimleh1, és nem jobb a műveletigény tekintetében. Figyelemre méltó, hogy megismétlődik az 5. teszt által is mutatott kiugróan gyengébb viszonylagos időbeli és műveletszámbeli viselkedés 25-ös gépszámnál. Ez a jelenség kapcsolatban lehet *maxmeret*-tel.

7. teszt. Mint az 5. teszt, de ezúttal a termékmennyiségekben nagyobb szórást engedünk meg.

- a) $generate(m, U(2m, 3m), 1, 1, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 35, 5)$
- b) $generate(m, U(2m, 3m), 1, 1, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 50, 5)$
- c) $generate(m, U(2m, 3m), 1, 1, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 80, 5)$
- d) $generate(m, U(2m, 3m), 1, 1, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 150, 5)$

Eredmények.

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.74	0.58	0.49	20	0	0	0.67	
25	1.50	0.57	0.58	20	0	0	0.80	a) eset
30	1.46	0.49	0.49	20	0	0	1.26	
35	1.38	0.43	0.45	20	0	0	1.80	

$maxmenny = 35$

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.81	0.46	0.48	20	0	0	0.67	
25	1.51	0.55	0.55	20	0	0	0.78	b) eset
30	1.48	0.48	0.48	20	0	0	1.36	
35	1.39	0.40	0.46	20	0	0	1.77	

$maxmenny = 50$

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.76	0.86	0.51	20	0	0	0.68	
25	1.53	0.99	0.61	20	0	0	0.73	c) eset
30	1.49	0.51	0.44	20	0	0	1.45	
35	1.39	0.50	0.50	20	0	0	1.51	

$maxmenny = 80$

m	$\frac{E_c}{E_{S1}}$	$\frac{I_c}{I_{S1}}$	$\frac{M_c}{M_{S1}}$	jobb $S1$	jobb C	egyenlő	$\frac{I_{S1}}{10}$	
20	1.79	0.81	0.50	20	0	0	0.68	
25	1.58	1.00	0.57	20	0	0	0.80	d) eset
30	1.50	0.55	0.44	20	0	0	1.42	
35	1.37	0.48	0.51	20	0	0	1.66	

$maxmenny = 150$

Jelentős változást egyik esetben sem láthatunk, a módszerek egymáshoz viszo-nyí-tott teljesítménye meglehetősen érzéketlennek tűnik a termékmennyiség meg-vál-to-zá-sa-i-ra. Az időeltérés viszont jobban kiemelkedik a 25-ös méretnél.

8. teszt. Mint az 5. tesztben, de *erange*-t csökkentjük. Csak a csökkenési irányt vizsgáljuk, mert az általában használt 5-ös *erange* érték meglehetősen nagy, hogy elképzelhetőek legyenek gyors előkészítéssel párosuló hosszú műveletek is.

- 20-20 próba.
- a) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 3)$
- b) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 1)$
- a) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 0.3)$
- a) $generate(m, U(2m, 3m), 1, 1, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 0.1)$

Eredmények. A táblázati adatok új jelentését a fejléc jelzi.

m	$erange$	$\frac{E_{S1}}{E_D}$	$\frac{E_C}{E_{S1}}$	$\frac{I_{S1}}{I_D}$	$\frac{I_C}{I_{S1}}$	$\frac{M_{S1}}{M_D}$	$\frac{M_C}{M_{S1}}$	$\frac{I_D}{20}$
17	3.0	0.72	1.84	0.034	1.151	0.022	0.464	16.34
20	3.0	0.72	1.74	0.022	0.760	0.016	0.504	31.60
23	3.0	0.51	1.54	0.010	0.555	0.008	0.532	83.17
26	3.0	0.37	1.47	0.007	0.472	0.006	0.490	151.41
17	1.0	0.71	1.81	0.028	0.764	0.019	0.504	18.68
20	1.0	0.42	1.68	0.014	0.548	0.010	0.499	50.33
23	1.0	0.78	1.57	0.020	0.449	0.015	0.453	50.71
26	1.0	0.48	1.43	0.008	0.507	0.007	0.541	133.37
17	0.3	0.74	1.88	0.028	0.600	0.020	0.474	21.78
20	0.3	0.42	1.67	0.012	0.616	0.010	0.518	52.57
23	0.3	0.39	1.55	0.011	0.379	0.009	0.408	100.49
26	0.3	0.82	1.43	0.013	0.516	0.012	0.537	76.51
17	0.1	0.69	1.91	0.039	0.407	0.028	0.386	16.66
20	0.1	0.44	1.63	0.014	0.411	0.012	0.478	58.83
23	0.1	0.40	1.53	0.011	0.407	0.008	0.452	195.44
26	0.1	0.80	1.44	0.017	0.493	0.014	0.476	68.56

Az *erange* megváltoztatásának hatása csekély volt Diffmom és Szimleh1 összehasonlításában, azonban Cellának a másik két programhoz viszonyított teljesítménye javult. Cella Szimleh1 futási idejének töredékrésze alatt, hoz-záve-tő-le-ge-sen fele akkora lebegőpontos műveletigénnyel állítja elő eredményét, és ez az eredmény a feladatméret növekedésével egyre jobb viszonylagos célfüggvényértéket szolgáltat. Ugyancsak javul a célfüggvényértékek hányadosa Cella javára, ha *erange* csökken. Megfigyelhető, hogy Diffmom különösen rosszul teljesít egyes méreteknél, és ha *erange* csökken, akkor ez a méret nő.
A hátralevő tesztek a többutas problémákat kezelő programokat hasonlítják össze.

9. teszt. 9 résztesztet hajtottam végre különböző maximális megengedett útvonalhosszakkal és változó korlátokkal az egyes termékekre létező gyártási útvonalak számára. A (maxhossz,útszám) paraméterek az egyes résztesztekre:

	1:2	1:5	3:5
$\lceil \frac{n}{5} \rceil$	a	b	c
$\lceil \frac{n}{4} \rceil$	d	e	f
$\lceil \frac{n}{3} \rceil$	g	h	i

20-20 próbát végzünk.
$maxpr = 3,\ maxpr1 = 0.98,\ maxsz = 4,\ pontossag = 200,\ maxmeret = 6,\ erange = 2$

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
a) eset	20	0.58	3.29	2.30	20	0	0.445
	25	0.64	3.85	2.25	20	0	1.010
	30	0.67	4.76	2.34	20	0	1.700
	35	0.70	5.86	2.52	20	0	2.524
	40	0.72	6.88	2.59	20	0	3.530
	45	0.75	8.14	2.72	20	0	4.921
	50	0.76	9.76	2.80	20	0	6.463

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
b) eset	20	0.64	2.92	2.36	20	0	0.689
	25	0.71	3.77	2.29	20	0	1.550
	30	0.75	4.34	2.31	20	0	2.711
	35	0.77	5.18	2.43	20	0	3.979
	40	0.80	6.35	2.61	20	0	5.695

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
c) eset	15	0.43	3.21	3.13	20	0	0.492
	20	0.55	3.50	2.66	20	0	1.118
	25	0.70	3.93	2.51	20	0	2.050
	30	0.78	3.73	2.48	19	1	11.720

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
d) eset	20	0.59	4.09	2.50	20	0	0.815
	25	0.66	4.70	2.52	20	0	1.904
	30	0.69	6.07	2.64	20	0	3.446
	35	0.72	7.96	2.92	20	0	5.548
	40	0.74	9.09	2.92	20	0	7.935

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
e) eset	20	0.59	5.13	2.47	20	0	0.745
	25	0.66	5.63	2.53	20	0	1.980
	30	0.70	7.68	2.80	20	0	3.933
	35	0.73	8.27	2.92	20	0	6.848
	40	0.74	9.75	3.01	20	0	9.317

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
f) eset	15	0.53	3.77	3.06	20	0	0.744
	20	0.69	3.92	2.69	19	1	1.810
	25	0.80	4.40	2.67	19	1	3.339
	30	0.85	4.70	2.68	17	3	5.499

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
g) eset	15	0.53	2.87	3.96	20	0	0.885
	20	0.60	3.58	3.38	20	0	1.853
	25	0.67	4.08	3.06	20	0	3.231
	30	0.70	5.28	3.19	20	0	4.764

	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
h) eset	15	0.58	5.15	3.67	20	0	0.602
	20	0.71	4.93	3.22	19	1	1.577
	25	0.77	5.09	2.98	20	0	3.008
	30	0.81	5.97	3.08	20	0	4.888
	m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
i) eset	15	0.67	4.45	3.78	20	0	0.788
	20	0.82	4.34	3.14	14	6	1.990
	25	0.93	4.61	2.97	10	10	3.836
	30	0.94	5.25	3.03	14	6	5.920

Az figyelhető meg, hogy Cella teljesítménye a bonyolultabb feladatosztályokon egyre jobban megközelíti Szimleh2-ét, sőt néhány konkrét példában túl is szárnyalja. Ez azért lehetséges, mert az átlagos szerkezetet Cella jól kiemeli, Szimleh azonban képtelen bármiféle speciális struktúrát figyelembe venni. Eszerint Cella alkalmazása éppen az összetettebb problémák esetén lehet megfontolás kérdése.

10. teszt.

- Viselkedés nagy méretekre
- 10-10 próba
- $generate(m, U(2m, 3m), 1, 5, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, 20, 2)$

m	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
70	0.90	15.18	3.66	10	0	4.74
90	0.93	16.22	3.91	10	0	14.16
110	0.94	20.38	4.63	10	0	13.20

Cella célfüggvényértéke javulva közelíti Szimleh2-t. A néhány százalékos eltéréssel szemben áll kompenzáló tényezőként a Cella javára ugrásszerűen javuló idő- és műveletarány.

11. teszt.

- Nagy szórás a termékmennyiségekben
- 20-20 próba
- $generate(m, U(2m, 3m), 1, 5, \lceil \frac{m}{10} \rceil, \lceil \frac{m}{5} \rceil, 1, 10, maxmenny, 2)$
- $maxmenny = 50\ , 80\ , 110\ , 140$

Eredmények.

m	$maxmenny$	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
30	50	0.77	5.48	2.48	20	0	0.927
40	50	0.81	7.29	2.68	20	0	2.408
30	80	0.76	5.01	2.39	20	0	1.065
40	80	0.80	6.32	2.56	20	0	2.622
30	110	0.79	5.44	2.46	20	0	0.981
40	110	0.80	6.96	2.67	20	0	2.422
30	140	0.79	5.30	2.36	20	0	0.987
40	140	0.81	6.94	2.64	20	0	2.418

A maximális termékmennyiség széles tartományban változva sem befolyásolja az algoritmusok viszonylagos hatékonyságát jelentősen.

12. teszt.

- *erange* változik
- 20-20 próba
- $generate(m, U(2m, 3m), 1, 5, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 20, erange)$
- $erange = 2,\ 1,\ 0.3,\ 0.1$

Eredmények.

m	$erange$	$\frac{E_{S2}}{E_C}$	$\frac{I_{S2}}{I_C}$	$\frac{M_{S2}}{M_C}$	jobb S2	nem jobb S2	$\frac{I_{S2}}{20}$
30	2.0	0.79	4.86	2.27	20	0	0.936
40	2.0	0.82	6.38	2.48	20	0	2.405
30	1.0	0.79	5.15	2.36	20	0	0.983
40	1.0	0.82	7.03	2.66	20	0	2.448
30	0.3	0.77	4.94	2.34	20	0	1.004
40	0.3	0.80	6.99	2.68	20	0	2.445
30	0.1	0.77	7.20	3.22	20	0	1.061
40	0.1	0.80	8.37	3.04	20	0	2.468

erange a célfüggvényértékekre nem volt hatással, annál inkább a végrahajtási időre és a lebegőpontos műveletek számára. A hatásról nem állítható az itt látottak alapján, hogy monoton módon függ *erange*-től, azonban kétségtelen az eredmény $erange = 0.1$ esetén. A Cella számára előnyös képet az okozhatja, hogy ha a műveleti idők azonosak, akkor a szimulált lehüléses algoritmus sok lehetőséget találhat körülbelül azonos nagyságú célfüggvényjavításra, így tovább fut.

13. teszt.

- Megvizsgáljuk, hogy Cella teljesítményét hogyan befolyásolja *pontossag* értéke.
- *pontossag* értéke 520 és 40 között változik
- 20-20 próba, egy legenerált feladatot minden *pontossag*-gal megoldunk
- $generate(m, U(2m, 3m), 1, 5, \lceil\frac{m}{10}\rceil, \lceil\frac{m}{5}\rceil, 1, 10, 20, 2)$

Eredmények.

m	$pontossag$	$\frac{E_C}{20}$	$\frac{M_C}{20}$	$\frac{I_C}{20}$
30	520	8122	173038	1.175
30	480	8122	172999	1.273
30	440	8122	172937	1.386
30	400	8122	172880	1.110
30	360	8124	172776	1.106
30	320	8128	172619	1.089
30	280	8130	172437	1.067
30	240	8135	172078	0.995
30	200	8145	171505	1.044
30	160	8182	169948	1.022
30	120	8302	164954	0.931
30	80	8381	160212	0.809
30	40	8381	159613	0.856
40	520	16956	404850	1.931
40	480	16956	404628	2.059
40	440	16960	404195	2.127
40	400	16978	403515	2.030
40	360	16994	402615	2.117
40	320	17033	400939	2.096
40	280	17085	397504	1.862
40	240	17142	392141	1.874
40	200	17234	384364	1.636
40	160	17327	378642	1.930
40	120	17357	376070	1.747
40	80	17377	375735	1.548
40	40	17377	375735	1.479

Látható, mennyit nyerhetünk a célfüggvényen, ha *pontossag* nagyobb. Százalékban kifejezve a különbség bár egyértelmű, nem nagy. Ezt az okozhatja, hogy 6-ban maximáltuk a csoportméretet. Alátámasztja a feltételezést az, hogy a százalékos javulás a kisebb méretű feladat esetén nagyobb. Ebből azt a következtetést vonhatjuk le, hogy a *pontossag* akkor bír nagyobb szereppel, ha a cellák mérete a gépek számához viszonyítva nagy, mert ekkor foglalkozunk csak a kisebb szállításokkal, hiszen különben a nagy szállítások miatti összevonásokkal hamar elérjük a maximális cellaméretet. Az algoritmus ismeretében könnyen megmagyarázható a csökkenő pontossággal javuló gyorsaság: nagy *pontossag* esetén sokkal több mátrixelemet vizsgálunk meg, összevonást indukál-e.

14. teszt.

- Szimleh2 vizsgálata 4 részteszttel
- Az éppen nem vizsgált paraméterek olyanok, mint korábbi eddigi teszteknél. $maxpr = 3$, $maxpr1 = 0.98$, $maxpr2 = 1.3$, $maxsz = 4$
- 20-20 próba

$maxpr$ **hatása.** $maxpr$ szabályozza, hány alkalommal kezdeje elölről a keresést a sztochasztikus algoritmus.
40 gépnél csak 14 próbát vegzünk.

m	$maxpr$	$\frac{E_{S2}}{20}$	$\frac{M_{S2}}{20}$	$\frac{I_{S2}}{20}$
30	1	7144	245975	3.838
30	2	6781	325526	5.602
30	3	6891	443063	6.649
30	5	6626	612835	10.609
30	7	6609	819065	15.686
40	1	15671	584110	5.390
40	2	15341	887810	11.563
40	3	14953	1077807	14.416
40	5	14843	1507874	22.726
40	7	14710	2055182	31.935

$maxpr$ javasolt értéke 3 és 5 közötti.

$maxpr1$ **hatása.** $maxpr$ szabályozza a lehülés ütemét. Az elvégzési időre vonatkozó adatok nem állnak rendelkezésre, de a számunkra lényegesebb műveletszámok ismertek.

m	$maxpr1$	$\frac{E_{S2}}{20}$	$\frac{M_{S2}}{20}$
30	0.98	6303	410011
30	0.97	6394	404907
30	0.96	6549	387852
30	0.95	6442	388280
30	0.94	6463	368935
30	0.93	6459	353856
30	0.92	6377	351902
30	0.91	6461	342466
30	0.90	6412	348032
30	0.89	6345	329567
30	0.88	6599	314080
30	0.87	6516	311181
30	0.86	6487	295812

Sejtés szerint a célfüggvényértékek romlanak, ha $maxpr1$ csökken, mert kevesebb eséllyel kerülünk ki lokális minimumokból. Az eredmények mégis hullámzanak. Az 1-hez közeli értékeken kívül látszik még egy kedvező tartomány 0.9 körül, de itt gyors a romlás, ha $maxpr$ értékét tovább csökkentjük. Az érzékeny változás miatt a fokozottabb idő- és műveletigény ellenére ajánlatosabb lehet az 1-hez közeli értékeket használni.

$maxpr2$ **hatása.** $maxpr2$ szabályozza a lokális keresések számát.

m	$maxpr2$	$\frac{E_{S2}}{20}$	$\frac{M_{S2}}{20}$	$\frac{I_{S2}}{20}$
40	0.55	15554	723645	8.369
40	0.70	15289	840506	10.249
40	0.85	15399	908674	11.982
40	1.00	15256	899755	10.918
40	1.15	15289	997026	12.894
40	1.30	14916	1050749	14.229
40	1.45	15045	1148810	16.399

A véletlen jelleg miatt nem pontosan követik az számeredmények a logikus várakozásokat: nem monoton javul az eredmény. Mindenesetre kétséges, hogy nagy értéket $maxpr2$ számára célszerű-e választani, mert a haszon nem áll arányban a nyereséggel. Ajánlott:0.7-0.8 vagy 1.3.

$maxsz$ **hatása.** Az általam bevezetett $maxsz$ nagyobb értéke többször engedi meg azonos célfüggvényértékű megengedett megoldás feltétel nélküli elfogadását, így várhatóan nagyobb mozgékonyságot biztosít.

m	$maxsz$	$\frac{E_{S2}}{20}$	$\frac{M_{S2}}{20}$	$\frac{I_{S2}}{20}$
40	0	14796	1073030	14.157
40	1	14762	1053638	13.498
40	2	14733	1015168	12.653
40	3	14662	1043905	12.736
40	4	14606	1045975	13.586
40	5	14632	1074547	14.192
40	6	14661	1108700	14.117
40	7	14657	1063833	14.602
40	8	14568	1037768	12.735

A $maxsz = 0$ esetén az ötlet nem kerül alkalmazásra. A változtatásom e-red-mé-nye-kép-pen a program képes azonos vagy kisebb időigénnyel néhány szá-za-lék-kal jobb eredmény produkálni. Eközben a műveletigény is csökkent. Mindenképpen érdemesnek tűnik tehát beépíteni.

Szimleh1 vizsgálata 4 paramétere szerint. A vizsgálódás tárgya és sorrendje ugyanaz, mint Szimleh2 esetében. Az alapparaméterek is azonosak.

$maxpr$ **hatása.**

m	$maxpr$	$\frac{E_{S1}}{20}$	$\frac{M_{S1}}{20}$
30	1	8481	170605
30	2	8562	168170
30	3	8517	170303
30	5	8416	171236
30	7	8467	170885
40	1	18394	432209
40	2	18003	457730
40	3	18035	485568
40	5	17761	555227
40	7	17721	621664

Alacsonyabb gépszámra 5, nagyobbra 3 ajánlott.

$maxpr1$ **hatása.**

m	$maxpr1$	$\frac{E_{S1}}{20}$	$\frac{M_{S1}}{20}$
30	0.98	8581	174985
30	0.97	8520	174447
30	0.96	8553	171884
30	0.95	8557	173374
30	0.94	8598	172485
30	0.93	8607	171080
30	0.92	8679	168437
30	0.91	8584	169676
30	0.90	8616	172901
30	0.89	8618	171841
30	0.88	8545	176640
30	0.87	8504	171118
30	0.86	8531	173900
40	0.98	18365	484181
40	0.97	18281	485769
40	0.96	18298	483106
40	0.95	18350	481136
40	0.94	18384	474505
40	0.93	18276	483536
40	0.92	18390	483540
40	0.91	18203	480231
40	0.90	18330	488454
40	0.89	18421	478543
40	0.88	18323	483270
40	0.87	18281	480214
40	0.86	18273	486767

Megint előfordulnak jó eredmények gyorsabb lehülési ütem esetén is. Nem jelölhető ki azonban általánosan jó paraméterérték. A műveletigény közelítőleg állandó, és az 1-hez közeli paraméterértékek általában jól teljesítenek, még a 30-as méret 0.98-hoz tartozó eredménye is jobb, mint sok másik, ezért ajánlatosabb nagy értéket választani paraméterül.

$maxpr2$ **hatása.**

m	$maxpr2$	$\frac{E_{S1}}{20}$	$\frac{M_{S1}}{20}$
30	0.55	8367	203214
30	0.70	8180	232146
30	0.85	8167	253273
30	1	8099	265226
30	1.15	8029	296991
30	1.30	7970	331122
30	1.45	7923	340837
40	0.55	17416	534953
40	0.7	17159	608104
40	0.85	17039	645482
40	1	17058	674288
40	1.15	16865	731949
40	1.30	16866	783045
40	1.45	16948	787116

A többdimenziós esettől eltérően a célfüggvényre kedvező hatást gyakorolt a lokális keresések számának növelése. Ezt az okozza, hogy a gépáthelyezések hatását nem másítja meg az útvonalak megváltoztatása.

$maxsz$ **hatása.**

m	$maxsz$	$\frac{E_{S1}}{20}$	$\frac{M_{S1}}{20}$
30	0	9246	166304
30	1	9121	177010
30	2	9090	173410
30	3	9067	172179
30	4	9126	173202
30	5	9242	170460
30	6	9107	174246
30	7	9208	173083
30	8	9221	174215
40	0	18339	478603
40	1	18494	473182
40	2	18238	483868
40	3	18305	481609
40	4	18256	485825
40	5	18222	488752
40	6	18255	484982
40	7	18323	484431
40	8	18089	490645

A módosításomnak csak a bonyolultabb feladatra van csekély hatása. A hatás megléte ugyanazért lehetséges, mint Szimleh2-nél, nagyobb valószínűséggel nyílnak meg az utak a lokális optimumból való kijutás előtt. Meg kell magyarázni azonban, hogy miért jelentkezett Szimleh2-nél mindkétszer a hatás, míg most csak egyszer. A kétdimenziós szimulált lehülésnél az azonos célfüggvényértékű állapotok elfogadása azt jelenti, hogy az útvonalváltoztatások miatt kedvezőtlenné váló gép-gép kapcsolatok elrendeződhetnek cellák közötti gépcserék révén. Ezért $maxsz$ bevezetése hasznos, ha lehetséges az útvonalcsere. Egydimenziós lehülésnél szerepe észrevehetetlen, mert a kevésszámú alternatívát az alapmódszer is nagy valószínűséggel felderíti.

8. Összegzés

A csoportos technológiák alkalmazása jelentős költségmegtakarítást eredményez a termelő vállalatoknál. A felírható egészértékű programozási feladatok NP-nehezek és nagy méretük miatt kezelhetetlenek. Ezért van szükség hatékony heurisztikák kifejlesztésére. Áttekintettük a leggyakoribb megközelítéseket és megvizsgáltunk néhány konkrétan megvalósított algoritmust. Azt találtuk, a körülmények jelentősen befolyasolják a célszerű választást. Mindazonáltal a legszélesebb körben a szimulált lehülés ötletét alkalmazó sztochasztikus programok látszanak alkalmazhatónak.

Hivatkozások

[1] A. J. Vakharia és Y. L. Chang Cell formation in group technology: a combinatorial search approach *International Journal of Production Research* 1997., Vol. 35, NO 7, 2025-2043

[2] Divakar Rajamani, N. Singh és Y. P. Aneja Integrated design of cellular manufacturing systems in the presence of alternative process plans *International Journal of Production Research* 1990., Vol. 28, NO. 8, 1541-1554

[3] C. H. Cheng, M. S. Madan és J. Motwani *International Journal of Production Research* 1996.,Vol. 34, NO. 2,349-361

[4] R. Logendran és P. Ramakrishna Manufacturing cell formation in the presence of lot splitting and multiple units of the same machine *International Journal of Production Research* 1995., Vol. 33, NO. 3, 675-693

[5] M. M. Solomon, R. A. Millen és P. Afentakis The formation of subfamilies for machine loading of flow-line cells *International Journal of Production Research* 1995, ? , 2357-2373

[6] Y. Won New p-median approach to cell formation with alternative processplans *International Journal of Production Research* 2000.,Vol. 38, NO. 1, 229-240

[7] M. S. Akturk és H. O. Balkose Part-machine grouping using a multi-objective cluster analysis *International Journal of Production Research* 1996., Vol. 34, NO. 8,2299-2315

[8] L. Kaufman és P. J. Rousseeuw *Finding groups in data* 1989,Wiley, New York

[9] S. K. Mukhopadhyay, A. Gopalakrishnan és M. K. Kripalani Moments-based clustering techniques for manufacturing cell formation *International Journal of Production Research* 1995.,Vol. 33,NO. 4, 1091-1115

[10] Levent Kandiller A combinatorial optimization tour in cell formation via hypergraphs *Rutcor research report*August 1996, RRR 25-96

[11] Naiqi Wu és Gavriel Salvendy An efficient heuristic for the design of cellular manufacturing systems with multiple identical machines *International Journal of Production Research* 1999.,Vol. 37, NO. 15, 3519-3540

[12] Cormen, Leiserson és Rivest *Algoritmusok* 1997. Műszaki könyvkiadó, Budapest