Resum

El problema tractat en el present projecte és una variant del problema de programació del taller mecànic. En aquesta variant hi ha una disposició de les màquines en paral·lel. A més, no tots els tipus de peces es poden produir a cadascuna de les màquines. I finalment es consideren temps de pre-procés i post-procés en les peces i temps de preparació de les màquines quan es passa a processar peces d'un altre nivell diferent al de la peça anteriorment processada.

Aquest projecte té com a objectiu l'estudi i millora de l'instant d'acabament de l'última peça (C_{max}) de sistemes on n peces s'han de produir en m màquines a k nivells. S'estudiarà, en particular, l'escenari on els elements, màquines i peces, estan separats en $k = 3$ nivells (alt, mig i baix) i es consideraran les dates d'arribada de les peces al sistema (temps de pre-procés) així com també el seu temps d'acabat (temps de post-procés). Cada màquina podrà produir les peces del seu nivell i els seus inferiors però considerant l'afectació del temps de preparació de la màquina, si aquesta és necessària.

El problema es resoldrà plantejant dos tipus d'algorismes: per una banda, un mètode heurístic amb dos variants, i per altra, un metaheurístic. Aquest últim, es divideix en dues parts: una primera que dóna una solució segregada i a la qual posteriorment se li apliquen dues fases que milloren aquesta solució segregada; la segona part consisteix en l'aplicació d'un algoritme genètic utilitzant la solució trobada en la primera part. Seguidament, i com a resultat de la comparativa entre algorismes, es realitzaran unes modificacions al mètode metaheurístic per millorar el seu comportament.

Com a resultat dels procediments, s'obté una seqüència factible per a cada màquina que millora C_{max} respecte la situació inicial on les màquines només produirien les peces del seu nivell.

Amb la finalitat de verificar el correcte funcionament i l'eficiència dels procediments de resolució, s'acompanya la resolució del problema amb una experiència computacional que permet resoldre problemes de fins a 10 màquines i 200 peces.

Sumari

1. Glossari

n; nombre de peces

j; índex de referència per les peces ($j = 1,...,n$)

m; nombre de màquines

i; índex de referència per les màquines ($i = 1,...,m$)

k; índex de referència pels nivells de les peces i les màquines ($k = 1,...,3$)

u_j; nivell de la peça *j*

p_j; temps de producció de la peça *j*

r_j; temps de pre-procés de la peça *j*

q_j; temps de post-procés de la peça *j*

$s_{kk'}$; temps de preparació dependent de la seqüència. Temps necessari per preparar una màquina quan una peça del nivell *k* precedeix una peça de nivell *k'*.

C_i; instant de finalització de l'última peça de la màquina *i*

C_{max}; índex d'eficiència que mesura l'instant de sortida de la última peça del sistema

σ; vector de dimensió *n* que determina els instants de començament de cada peça

J_R ; subconjunt de peces que seran programades al començament de la seqüència

J_Q ; subconjunt de peces que seran programades al final de la seqüència

$F_{i,l}$; instant en què la màquina *i* comença un temps ociós *l*

$W_{i,l}$; durada d'un temps ociós *l* en la màquina *i*

$Kp_{i,l}$; nivell al que pertany la peça anterior a un temps ociós *l* en la màquina *i*

$Kf_{i,l}$; nivell al que pertany la peça posterior a un temps ociós *l* en la màquina *i*

P_{size}; dimensió de la població inicial a l'algoritme genètic

t; índex de referència pels individus que conformen la població inicial dels algoritmes genètics ($t = 1,..., P_{size}$)

P_c; probabilitat d'aplicar l'operació d'encreuament

P_m; probabilitat d'aplicar l'operació de mutació

P_{ls}; probabilitat d'aplicar l'operació de cerca local

2. Prefaci

2.1. Origen del projecte

Tradicionalment, els problemes en la programació d'operacions es resolien de forma manual i utilitzant criteris basats en l'experiència del personal. Aquest mètode no presentava una base objectiva i per tant no garanteix la qualitat de la solució escollida, factor important en un escenari de gran competitivitat com l'actual.

Cada vegada més, les indústries necessiten ser capaces de produir un determinat nombre de productes per satisfer unes necessitats, uns requisits i unes expectatives dels clients de la manera més competitiva possible, adaptant-se ràpidament als continus canvis de les línies productives, dels mercats o de la cartera de productes en el menor temps possible **[1]**.

Aquesta tendència provoca que s'hagin de millorar contínuament els processos productius amb la finalitat de minimitzar la durada de les operacions que no aporten valor al producte com per exemple els temps de preparació de les màquines entre operacions. Això comporta disminuir d'aquesta manera el temps de permanència de les peces en el procés productiu.

Un possible camí per tal de millorar l'eficiència de les plantes pot ser l'automatització de la programació d'operacions per a problemes de dimensió gran. Encara que no es garanteixi obtenir una solució òptima, s'obtenen solucions de qualitat en un temps computacional adequat. Això possibilita a les empreses satisfer la demanda respectant els costos i els temps d'entrega (entre d'altres) per superar la competència del sector. En aquest estudi, els procediments implementats permeten escollir l'ordre de producció de les peces a processar en un conjunt de màquines en paral·lel per minimitzar el valor de C_{max} (*makespan*).

Per trobar la solució a problemes de complexitat exponencial com aquests existeixen procediments exactes (*Programació entera i mixta o Branch & Bound*) els quals proporcionen una solució òptima. El principal problema d'aquets procediments és que la dimensió dels problemes que generalment les empreses han de solucionar requereixen uns temps de càlcul i uns recursos informàtics excessius. Com a conseqüència, moltes

vegades s'ha d'optar per procediments heurístics i metaheurístics que permeten trobar solucions no necessariament òptimes, però d'un bon nivell i en un temps raonable.

En aquest estudi, es busca solucionar el problema de programació mitjançant un algoritme genètic. Aquest procediment es troba classificat dins dels procediments metaheurístics, que busquen resoldre problemes difícils d'optimització combinatòria mitjançant procediments iteratius.

2.2. Motivació

Aquest projecte neix amb el propòsit de satisfer les necessitats actuals de la indústria, que, cada vegada més, necessita ser capaç de produir un major nombre de productes el qual obliga a les empreses a una renovació de la seva maquinària. Aquesta renovació pot produir-se per l'adquisició de noves màquines que poden fer nous productes o com el cas que s'estudiarà en aquest projecte, en l'adaptació de la maquinària existent a la nova realitat sense perdre la seva capacitat inicial.

3. Introducció

3.1. Objectius del projecte

Els objectius d'aquest projecte són:

- Analitzar i estudiar el problema de seqüenciació i assignació de peces en màquines en paral·lel, minimitzant l'instant de finalització de l'ultima peça (C_{max}).
- Dissenyar un algoritme que satisfaci l'objectiu del projecte.
- Implementar un programa informàtic mitjançant l'eina *Microsoft Visual Studio 2010* utilitzant el llenguatge de programació C# i que proporcioni una solució factible a aquest tipus de problemes.
- Analitzar els resultats obtinguts com a resultat d'una experiència computacional.

3.2. Abast del projecte

El procediment de resolució proposat és apte per a la resolució de la programació de peces en tots els processos industrials en els quals intervinguin màquines en paral·lel, limitacions de produir certes peces en certes màquines i temps de preparació de les màquines. L'índex d'eficiència utilitzat és l'instant de finalització de l'última peça (C_{max}).

En el cas particular d'aquest projecte, el nombre màxim de màquines i peces amb el que es treballarà serà 10 i 200 respectivament. Els temps de producció es troben compresos entre valors de 1 i 10 segons i els temps de pre-procés i post-procés segueixen una distribució uniforme amb valors entre [1, K (n/m)], on K és un enter positiu igual a 3 i 5. Pel que fa als temps de preparació, es mouen en una escala entre 1 i 19 segons.

3.3. Estructura del projecte

El cos d'aquest projecte està estructurat en tres parts principals, la primera de les quals tracta la introducció amb els conceptes previs. Aquesta primera part està formada pels **capítols 1, 2, 3 i 4**.

La segona part, que descriu i resol el problema plantejat, està formada pel **capítol 5**, que formula el problema; en els **capítols 6 i 7** es defineix i presenta el procediment de

resolució utilitzat; en els **capítols 8 i 9** es descriu la situació inicial del problema i el mètode de resolució; i en el **capítol 10** es detalla l'experiència computacional.

En la tercera i última part del projecte es poden trobar el pressupost pel projecte en el **capítol 11**, l'estudi d'impacte ambiental en el **capítol 12** i les conclusions a les quals s'han arribat.

En un altre dossier apart, es tenen un conjunt d'annexos que contenen les aplicacions i les dades amb les que s'ha treballat.

4. Programació d'Operacions

En aquest apartat s'introdueixen els conceptes fonamentals de la programació d'operacions, la classificació dels problemes, els índexs utilitzats per calcular l'eficiència d'un programa de producció, les regles de seqüenciació per assignar les tasques als recursos, els tipus de flux que poden donar-se, les diferents configuracions productives existents i la seva nomenclatura.

4.1. Definició

La programació d'operacions és l'àmbit general en què es situa aquest projecte. Abans d'introduir en els següents punts la classificació dels problemes de programació, es interessant conèixer la definició de la programació d'operacions. A continuació es presenta la definició segons Companys **[2]**:

"La programació d'operacions té com a finalitat definir de forma concreta en quin dels recursos disponibles s'executarà cadascuna de les operacions necessàries per la realització de les ordres de treball emeses i els instants (dates) en què tindran lloc aquestes execucions. S'associen a la programació d'operacions diferents subfuncions entre les quals cal destacar la càrrega, la seqüenciació i la temporització".

La programació d'operacions consisteix en resoldre les tres subfuncions següents **[3]**:

- **Càrrega:** consisteix en l'assignació de les operacions a les estacions de treball.
- **Seqüenciació:** consisteix en establir l'ordre o la seqüència en què es processaran les operacions assignades a una estació de treball.
- **Temporització:** consisteix en determinar els instants d'execució, és a dir, les dates d'inici i d'acabament de cada operació.

4.2. El problema del taller mecànic

El prototip de problema de programació acostuma a denominar-se *problema del taller mecànic* i el seu enunciat és el següent **[2]**:

"n peces (lots de peces, comandes o ordres de treball) han de realitzar-se en m màquines (recursos, seccions, llocs de treball). La realització de cada peça implica l'execució, en cert ordre establert, d'una sèrie d'operacions prefixades on cada operació està assignada a una de les m màquines i té una durada (temps de procés) determinada i coneguda; ha d'establir-se un programa, és a dir, la seqüència d'operacions en cada màquina i el interval temporal d'execució de les operacions, amb l'objectiu d'optimitzar un índex determinat que mesura l'eficiència del programa."

Problemes com el descrit s'originen en els sectors més diversos, encara que no tinguin cap relació ni amb la mecànica ni amb els tallers. De tota manera, per uniformar la nomenclatura, s'utilitzen les paraules "peces" i "màquines" per indicar els conceptes equivalents en aquests problemes.

4.3. Problemes estàtics, semidinàmics i dinàmics

Els problemes del taller mecànic poden dividir-se en problemes estàtics, semidinàmics i dinàmics **[4]**. Cadascun d'aquests té les seves característiques.

4.3.1. Problema estàtic

- El nombre de peces és finit i determinat, i aquestes peces han de realitzar-se en un taller amb un nombre finit de màquines.
- En l'instant de programació es coneixen diferents paràmetres com la ruta de cada peça, les operacions per les quals ha de passar, a quina màquina s'ha de realitzar cada operació i la duració de cada operació.
- Totes les màquines i peces estan disponibles a l'instant 0.
- La seva finalitat és optimitzar un índex d'eficiència.

4.3.2. Problema semidinàmic

- El nombre de peces és finit i determinat, i aquestes peces han de realitzar-se en un taller amb un nombre finit de màquines.
- En l'instant de programació es coneixen diferents paràmetres com la ruta de cada peça, les operacions per les quals ha de passar, a quina màquina s'ha de realitzar cada operació i la duració de cada operació.

- Els instants de disponibilitat de totes les peces i/o màquines no són iguals, però sí coneguts en l'instant en què es realitza la programació.
- La seva finalitat és optimitzar un índex d'eficiència (en el cas del present projecte és busca minimitzar el temps de finalització de totes les peces o C_{max}).

4.3.3. Problema dinàmic

- L'horitzó del taller és il·limitat cap al futur.
- El nombre de peces pot ser il·limitat mentre que el nombre de màquines és finit.
- Les peces que haurà de tractar el taller en el futur no estan definides en un instant determinat, sinó que la definició de les peces es coneix progressivament a mida que transcorre el temps i les peces arriben al taller.
- Quan les peces acaben de ser processades al taller, l'abandonen donant pas a altres peces que arriben per ser processades.
- Donat que les peces van entrant i sortint del sistema, un únic programa no és suficient, sinó que s'ha d'anar actualitzant i reprogramant.
- La seva finalitat és establir un procediment de programació i jutjar la qualitat d'un procediment. En aquest cas, els índex d'eficiència s'associen a les característiques mitjanes dels programes al llarg d'un interval de temps suficientment gran.

4.4. Hipòtesis de Conway, Maxwell i Miller per a problemes estàtics

Conway, Maxwell i Miller (1967) **[5]** van establir una llista d'hipòtesis generalment acceptades en problemes estàtics del taller mecànic:

1. Cada màquina està disponible contínuament des d'un instant $t \geq 0$, fins T, amb T arbitràriament gran. No es consideren intervals de no-disponibilitat per avaries o per manteniment.
2. En les rutes de les peces no es produeixen convergències (muntatges) ni divergències (partició en lots). Cada operació té una única operació immediatament precedent (excepte la primera operació de cada peça) i una sola operació immediatament següent (excepte l'última operació de cada peça).
3. Cada operació pot executar-se en un sol tipus de màquina del taller.
4. Només hi ha una màquina de cada tipus en el taller.

5. No s'admeten interrupcions, és a dir, quan una operació ha començat ha d'acabar-se abans d'efectuar-ne una altra en la mateixa màquina.
6. No poden solapar-se dues operacions de la mateixa peça (en la mateixa màquina o en màquines diferents).
7. Cada màquina pot efectuar una única operació a la vegada.
8. L'única restricció activa en el taller és relativa a les màquines. No existeixen restriccions relatives a la disponibilitat de la mà d'obra, dels materials etc.

4.5. Índexs d'eficiència

Els índex d'eficiència permeten classificar els programes segons la seva finalitat (minimitzar l'instant de finalització de l'última peça, minimitzar els retards, etc.) i a més quantificar la qualitat d'un programa respecte un altre.

Un índex d'eficiència és *regular* i pot formular-se en funció dels instants de sortida de les peces C_i i és tal que, augmenta (empitjora) si com a mínim un del valors de C_i augmenta, i disminueix (millora) si com a mínim un dels valors de C_i disminueix:

$$\gamma = f(c_1, c_2, \dots, c_n)$$

Alguns índex d'eficiència regulars són els següents:

- F_{max} , el major dels temps de permanència de les peces.
- F_{med} , el valor mitjà dels temps de permanència de les peces.
- C_{max} , l'instant de finalització de la última peça (*Makespan*).
- C_{med} , el valor mitjà dels instants de finalització de les peces.
- T_{max} , el major dels retards de les peces.
- T_{med} , el valor mitjà dels retards de les peces.

4.6. Configuracions productives

La configuració dels problemes està estretament lligada al tipus i al nombre de màquines de les diferents configuracions. A continuació, es repassen les principals configuracions de màquines que es donen en el problema del taller mecànic **[6]**:

4.6.1. Una màquina

La configuració productiva més senzilla està constituïda per una única màquina en la qual es realitza una única operació per cada peça.

Totes les peces es processen en aquesta màquina, és a dir, la subfunció de càrrega es realitza automàticament, i només cal programar la seqüenciació i la temporització. Donat un exemplar amb *n* treballs a realitzar, les solucions possibles seran *n!*.

Figura 4.1: Configuració productiva amb una màquina

4.6.2. Màquines en paral·lel

Una generalització de la configuració productiva precedent és la que preveu *m* màquines en paral·lel. En aquest cas el procés productiu està format per una única operació, però existeixen diversos centres de treball alternatius que la poden realitzar.

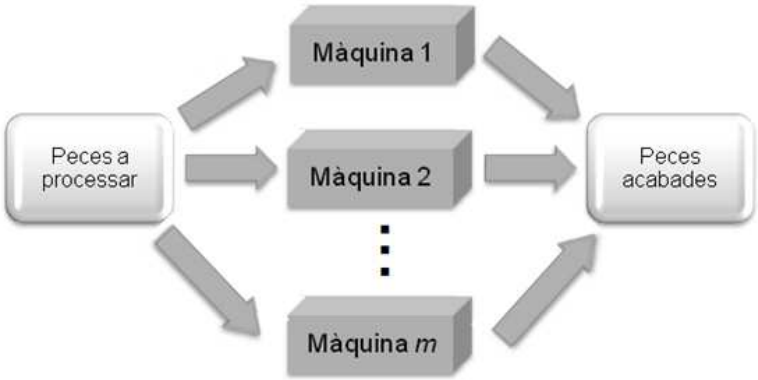


Figura 4.2:Configuració productiva d'una etapa amb m màquines en paral·lel

En aquest cas és necessari resoldre dos problemes: assignar les peces a les màquines i seqüenciar les peces assignades a una mateixa màquina. Existeixen tres casos diferents segons la velocitat de les màquines:

- **Màquines idèntiques:** *m* màquines idèntiques en paral·lel. Totes les màquines tarden el mateix temps en realitzar la mateixa operació.
- **Màquines uniformes:** *m* màquines en paral·lel amb velocitats diferents però proporcionals. Cada màquina té associat un factor de velocitat, és a dir, pot realitzar les operacions amb una velocitat diferent respecte a les altres.
- **Màquines diferents:** *m* màquines en paral·lel amb velocitats diferents no relacionades. El temps de procés és completament arbitrari entre una màquina i una

altra. Una màquina donada pot ser ràpida en una operació però lenta en una altra tasca en comparació amb les altres màquines.

Al contrari del que succeeix en configuracions per a una etapa amb una única màquina, en problemes amb màquines en paral·lel és necessari desenvolupar també la subfunció de càrrega. Un desenvolupament interrelacionat de càrrega i seqüenciació permet aconseguir generalment programes millors, però, a causa de la dificultat que això comporta, sovint es resolen independentment les dues subfuncions.

4.6.3. Flow-shop

El sistema *flow-shop* és una de les problemàtiques més estudiades. En aquest tipus de sistemes s'ha de processar un conjunt de *n* peces en *m* etapes de treball disposades en sèrie on cada etapa disposa d'una única màquina.

La ruta de producció de totes les peces és idèntica, és a dir, totes les peces recorren les màquines en un mateix ordre. Les peces *j* es processen en cada màquina *i* una sola vegada i cada operació té una durada determinada anomenada temps de procés (p_{ij}).

Figura 4.3: Representació gràfica d'un flow-shop

En el **flow-shop tradicional** es compleix:

- *m* etapes en sèrie.
- Una única màquina en cada etapa.
- Totes les peces segueixen el mateix ordre: etapa 1, etapa 2,..., etapa *m*.

Quan totes les peces no han de passar necessàriament per totes les màquines, sinó que poden saltar algunes etapes de la línia de producció, ens trobem davant d'una línia de producció flexible. Aquest fenomen es coneix amb el nom de *stage skipping*. Un problema de flow-shop amb possibilitat de *stage skipping* és conegut com **flow-shop flexible**.

4.6.4. Flow-shop híbrid

En aquest tipus de sistema també s'han de processar un conjunt de *n* peces en *m* etapes de treball disposades en sèrie. La diferència amb el flow-shop tradicional és que en el **flow-shop híbrid** cada etapa disposa d'una o varies màquines en paral·lel.

Les característiques del **flow-shop híbrid** són les següents:

- *m* etapes en sèrie.
- Cada etapa està formada per M_i màquines en paral·lel, i en almenys una de les etapes $M_i > 1$. Les màquines disponibles per realitzar una operació poden ser idèntiques, proporcionals o diferents.
- Totes les peces segueixen la mateixa ruta: etapa 1, etapa 2, ..., etapa *m*.

4.6.5. Job-shop

En aquest sistema les màquines estan disposades amb una certa configuració i cada peça segueix la seva pròpia ruta a través de la línia de producció. Així doncs, en el **job-shop** tradicional es compleixen les següents condicions:

- Hi ha *m* etapes en sèrie.
- Hi ha una única màquina en cada etapa.
- Cada peça segueix una ruta determinada.

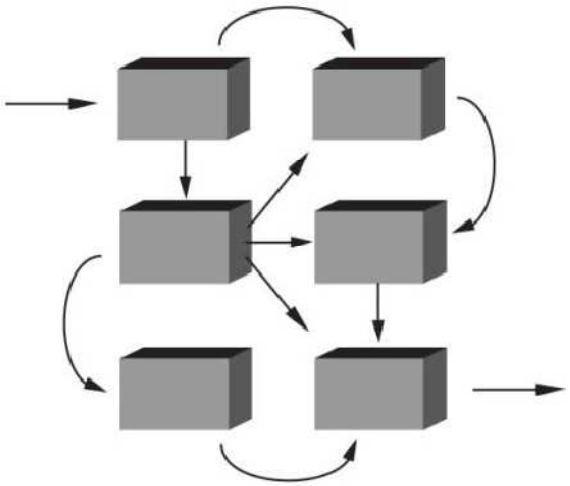


Figura 4.4: Exemple d'una possible configuració job-shop

4.7. Nomenclatura

4.7.1. Nomenclatura de Conway, Maxwell i Miller

Conway, Maxwell i Miller (1967) **[5]** van proposar una notació per als problemes de seqüenciació que consta de quatre símbols:

n / m / A / B

- **n:** En els problemes estàtics i semidinàmics és el nombre de peces (per indicar-ne un nombre arbitrari s'utilitza el símbol n), mentre que en els problemes dinàmics és una llei de probabilitat que indica l'arribada de les peces al taller.
- **m:** Indica el nombre de màquines (per indicar-ne un arbitrari s'utilitza el símbol m).
- **A:** Indica el tipus de flux de les peces en el taller:
  - R: flux aleatori
  - F: flux regular (flow-shop)
  - P: flux permutacional
  - G: flux general (job-shop)
- **B:** Indica l'índex d'eficiència que avalua la qualitat dels programes (F_{max}, C_{max}, T_{max}, etc.).

Aquest tipus de nomenclatura és ràpida i senzilla, però no és exhaustiva, donat el gran nombre de variants del problema del taller mecànic estudiades en la literatura.

4.7.2. Nomenclatura de Lawler

Més endavant, Lawler et al. (1982) **[7]** va proposar una notació per al problema del taller mecànic que consta de tres símbols:

α | β | γ

- **α:** Descriu l'entorn de les màquines i conté una única inscripció:
  - 1: una màquina
  - P: màquines idèntiques en paral·lel
  - Q: màquines en paral·lel amb velocitats diferents però proporcionals
  - R: màquines en paral·lel amb velocitats diferents no relacionades
  - F: flow-shop amb *m* màquines en sèrie
  - FF: flow-shop flexible

  - HF: flow-shop híbrid
  - Jm: job-shop
- **β:** Detalla les característiques del procés i pot contenir una, vàries o cap inscripció:
  - r_i: dates de disponibilitat de les peces diferents de 0
  - p: tots els temps de procés són iguals
  - d: totes les dates de lliurament són iguals
  - $s_{h,i}$: temps de preparació dependents de la seqüència
  - prmp: interrupcions (*preemptions*)
  - bkdw: no disponibilitat de la màquina (*breakdowns*)
- **γ:** Representa l'índex d'eficiència que avalua la qualitat dels programes (ja s'ha comentat amb el símbol **B** de la nomenclatura de l'apartat 4.7.1).

4.7.3. Nomenclatura de Vignier

Al 1999, Vignier et al. **[8]** proposa una notació específica per a problemes flow-shop híbrids. Aquesta notació segueix la de Lawler, però desglossant el camp α en quatre paràmetres de la manera següent:

$$\alpha = \alpha_1\alpha_2\left(\alpha_3{\alpha_4}^{(1)}, \alpha_3{\alpha_4}^{(2)}, \dots, \alpha_3{\alpha_4}^{(\alpha_2)}\right)$$

On:

- α_1: Equival al paràmetre α de la nomenclatura de Lawler.
- α_2 : Indica el nombre d'etapes en un flow-shop.
- α_3 i $\alpha_4$es repeteixen per cadascuna de les α_2 etapes:
  - $\alpha_3^{(k)}$ indica el tipus de màquines en l'etapa *k*. Valdrà *0* si hi ha únicament una màquina en l'etapa, *P* si hi ha màquines idèntiques en paral·lel, *Q* si les màquines són proporcionals o *R* si hi ha màquines en paral·lel no relacionades.
  - $\alpha_4^{(k)}$ indica el nombre de màquines en l'etapa *k*.

La notació de Vignier et al. és la més completa entre les analitzades i permet descriure sistemes més complexos i realistes.

5. Definició del problema considerat

En aquest capítol es descriu en detall el problema objecte d'aquest projecte: la programació de peces en màquines en paral·lel a múltiples nivells i amb temps de preparació.

5.1. Introducció

L'actual projecte sorgeix de la necessitat de solucionar el problema en la programació de peces que es planteja en la indústria actual. Per tant, és indispensable ser conscient dels factors que influeixen i de les variables que es volen millorar. Aquest fet ens porta en primer lloc a conèixer i plantejar en detall el problema en qüestió.

El projecte consisteix en el desenvolupament i la implementació d'un procediment que sigui capaç de programar un conjunt de *n* peces en *m* màquines. Tant les màquines com les peces estan dividides en *k* = 3 nivells, *alt, mig i baix*. Els subíndex *a, m* i *b* fan referència als nivells alt, mig i baix respectivament. Per tant, podem definir *n* i *m* com la suma de peces i màquines de cada nivell, on n (= n_a+ n_m+ n_b) i m (= m_a+ m_m + m_b).

En aquest cas s'estudiarà el cas en el qual les màquines són capaces de produir les peces del seu nivell i les de tots els seus inferiors.

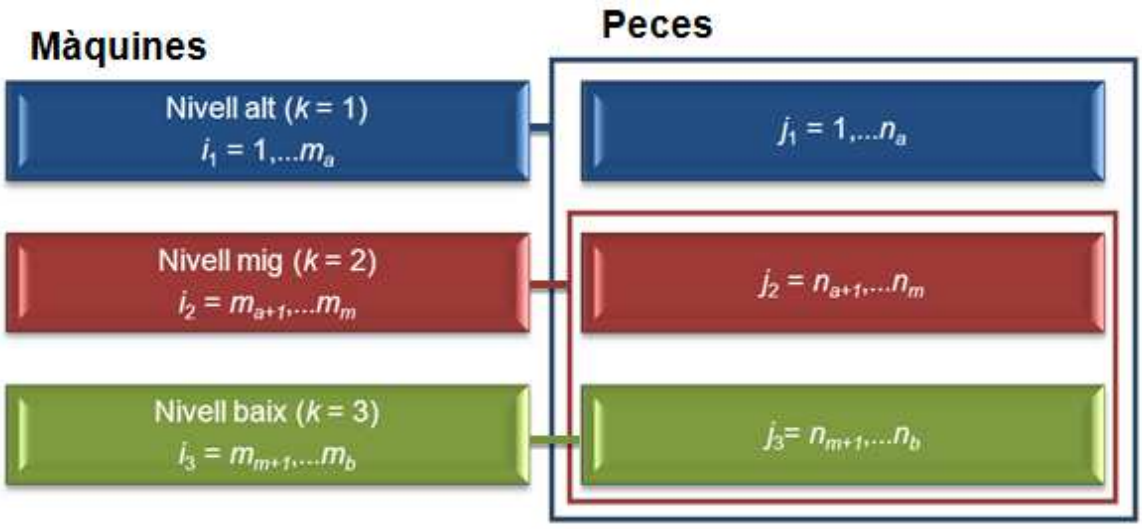


Figura 5.1: Conjunt de peces que es poden seqüenciar en cada màquina

El procediment que es proposa té com a objectiu millorar l'instant en què la última peça acaba el seu post-procés i, per tant, l'instant en què s'han acabat de produir totes les peces (C_{max}).

5.2. Dades de les màquines

Una màquina està definida pel seu nivell, que pot ser alt, mig i baix.

Nivell	Màquina
Alt (k = 1)	$i_1 = 1,...m_a$
Mig (k = 2)	$i_2 = m_{a+1},...m_{a+m}$
Baix (k = 3)	$i_3 = m_{a+m+1},...m_{a+m+b}$

Taula 5.1: Nomenclatura de les màquines

5.2.1. Temps de preparació

Associat a la producció de peces de diferents nivells tenim els temps de preparació de les màquines ($s_{kk'}$). Aquest temps fa referència al temps necessari per adaptar una màquina per poder processar peces d'un altre nivell.

		Nivell		
		Alt	Mitjà	Baix
Màquina	Alt	✓	✓	✓
	Mitjà	✗	✓	✓
	Baix	✗	✗	✓

Taula 5.2: Possibles canvis de nivell de les màquines

5.3. Dades de les peces

Les peces j (j = 1,...,n) estan definides pel seu nivell k (a,m,b) en funció de quines màquines les poden elaborar, per un instant d'arribada al sistema o temps de pre-procés (r_j), un temps de producció (p_j) i per un temps de post-procés (q_j).

Per a una peça qualsevol s'entén com a temps de producció el temps en que està ocupant un recurs (màquina). Per tant, podem resumir que el temps de producció ocupa uns recursos mentre que ni el temps de pre-procés ni el de post-procés n'ocupen.

Nivell	Peça	Instant arribada	Temps de pre-procés	Temps de post-procés
Alt (k= 1)	j_1 = 1,...n_a	r_{j_1}	p_{j_1}	q_{j_1}
Mig (k =2)	j_2 = n_{a+1},...n_m	r_{j_2}	p_{j_2}	q_{j_2}
Baix (k=3)	j_3= n_{m+1},...n_b	r_{j_3}	p_{j_3}	q_{j_3}

Taula 5.3: Nomenclatura de les peces

5.4. Suposicions per al problema

Per a poder resoldre el problema i limitar-ne l'abast, s'han fet les següents suposicions:

- Totes els màquines estan classificades en tres nivells: alt, mig i baix.
- Totes les peces estan classificades en tres nivells: alt, mig i baix.
- Totes els màquines processen cada peça a la mateixa velocitat.
- El nombre de màquines i peces és conegut inicialment.
- Tots els temps de procés són enters.
- Cap màquina és capaç de processar més d'un treball alhora.
- No es permet la interrupció d'un treball un cop ha estat començat.
- Els temps de preparació de les màquines per passar de produir una peça classificada en un nivell diferent del nivell de l'anterior.
- Cada peça es comença a processar en un instant de temps igual o superior al de la seva arribada (o temps de pre-procés).

- Les màquines poden acabar amb la configuració de l'última peça que han processat, no han de tornar a la seva configuració inicial.

La gran majoria d'aquestes suposicions pertanyen a les que van fer Conway, Maxwell i Miller(1967), però cal fixar-se que, per donar sentit al problema que es tracta no s'han tingut en compte les següents:

- Cada operació pot ser tractada en un sol tipus de màquina del taller.
- Només hi ha una màquina de cada tipus al taller.

5.5. Objectiu del problema

L'objectiu considerat és reduir el major d'entre els instants d'entrega més gran de totes les peces (C_{max}); és a dir, quan la peça ja ha estat sotmesa a les operacions de post-procés després d'haver-se produït. Donat que es disposa d'una solució inicial, en la qual la màquina d'un nivell només produeix peces d'aquest nivell, la millora es calcularà respecte aquesta situació (presentada en el capítol 8) poc eficient, però molt habitual en els tallers actuals.

6. Procediments generals de resolució

Els problemes de programació – entre els que es troba el problema tractat en aquest projecte – són de tipus combinatori, tal com succeeix en molts dels problemes que s'originen en l'àmbit de l'organització industrial.

Resoldre un problema combinatori consisteix en trobar la solució òptima dins d'un conjunt finit d'alternatives, assumint que la qualitat de la solució és quantificable i comparable amb qualsevol de les altres solucions **[9]**. El problema és que en la majoria dels casos no és viable aplicar aquest mètode de comparació amb qualsevol altre solució per a la resolució de problemes combinatoris ja que la complexitat d'aquests obligaria a temps de càlcul massa llargs per a obtenir la millor solució.

Es parla que un problema és **P** si existeix un algoritme que en un temps polinomial és capaç de trobar la solució; si, pel contrari, no existeix aquest algoritme el problema és **NP**. A més a més, el problema es classifica com **NP-difícil** si tot problema de NP és polinòmicament reductible a ell. La majoria dels problemes del taller mecànic són d'aquest tipus.

Així, per exemple, si es preveu resoldre un problema combinatori per enumeració amb n = 20 i es requereix una permutació, tindríem 20! = $2{,}43*10^{18}$ possibilitats. A un µs per possibilitat, es necessitarien més de 77 anys per poder analitzar totes les solucions. Òbviament, no és factible encara que s'acoti l'espai de cerca de la solució; a menys que es trobi un conjunt prou gran de solucions dominades, s'acostuma a recórrer a mètodes de resolució heurístics.

Així, a causa de la inviabilitat d'avaluar en un temps raonable totes les combinacions possibles, s'han desenvolupat diversos mètodes per optimitzar la cerca de solucions d'aquets problemes. En aquest capítol es presentaran els mètodes exactes, els mètodes heurístics i els mètodes metaheurístics.

6.1. Procediments exactes

Els mètodes exactes són els que garanteixen trobar la solució òptima d'un problema en un temps finit, sempre i quan aquesta existeixi.

El desavantatge d'aquest tipus de mètodes és que el temps invertit pot arribar a ser d'un ordre de magnitud molt superior al dels mètodes heurístics i, per tant, inaplicable en la indústria. Aquests mètodes només s'acostumen a aplicar en exemplars de dimensió molt reduïda, ja que el temps de càlcul requerit augmenta exponencialment amb la dimensió de l'exemplar.

6.1.1. Branch and Bound

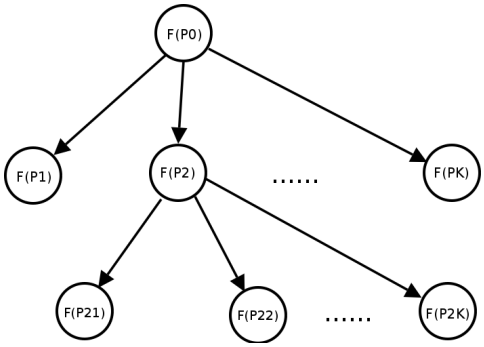


Figura 6.1: Exemple de resolució del mètode B&B

El *branch and bound* (B&B) és un mètode de resolució que utilitza la separació i acotació de les solucions.

És un procediment enumeratiu de cerca, ja que "enumera" totes les solucions possibles fins arribar a l'òptima, tot i que pel camí es van descartant solucions, demostrant a *priori* que no són òptimes. En aquest tipus d'algoritme l'espai de les solucions es divideix en unes branques (fase de ramificació), s'avalua cada branca, i es continua investigant només sobre les branques susceptibles d'arribar a solucions òptimes mitjançant un indicador de qualitat de la millor solució en cada branca (fase de poda).

6.1.2. Programació entera (PLE), binaria (PLB) o mixta (PLM)

Quasi la totalitat dels problemes combinatoris admeten una formulació com programes lineals enters o binaris, i especialment com mixtos. L'inconvenient d'aquest mètode resideix en la gran quantitat de variables que sorgeixen en la majoria dels problemes.

6.1.3. Programació dinàmica (PD)

En comptes d'enfocar el problema com l'optimització d'una funció global, la programació dinàmica resol el problema per etapes, en cadascuna de les quals es pot prendre una decisió independentment de les decisions considerades amb anterioritat.

6.1.4. Programació dinàmica acotada o *bounded dynamic programming* (BDP)

L'estructura és la mateixa que en el procediment anterior. En aquest mètode, es coneix una cota inferior i superior dels elements que s'han d'integrar per passar d'un estat a un altre. Es compara el millor valor esperat a partir de l'estat considerat amb un d'una solució heurística, cancel·lant els estats que no ofereixen garanties de millora.

6.2. Mètodes heurístics

Els procediment heurístics existeixen per la inviabilitat de trobar la solució òptima per a la majoria de problemes de programació en un temps acceptable. D'aquesta manera, en els últims anys la investigació s'ha dirigit cap al desenvolupament de procediments que, si be no asseguren l'òptim, sí permeten trobar solucions de qualitat en un temps acceptable.

La definició d'un mètode heurístic és la següent **[10]**:

"Un procediment senzill, solvent i basat en el sentit comú, que proporciona una solució de bon nivell (no necessariament òptima) a problemes difícils de forma fàcil i ràpida."

Els mètodes heurístics solen ser mètodes de resolució de problemes molt útils quan es donen una o més d'una de les següents circumstàncies **[11]**:

- Quan no existeix cap mètode exacte per a la resolució o, si existeix, el seu ús és molt costós perquè requereix masses recursos (temps de càlcul, memòria, etc.).
- Quan hi ha limitacions de temps i/o d'espai (per a l'emmagatzematge de dades).
- Quan no es necessita la solució òptima.
- Quan les dades disponibles són poc fiables. Si s'han simplificat les dades, no té sentit buscar la solució exacta ja que les dades no són reals.
- Com un pas intermedi per a l'aplicació d'altres algoritmes. Moltes vegades la solució aportada per un mètode heurístic es pren com a punt de partida d'altres procediments (per exemple, metaheurístics).

Els algoritmes de tipus heurístic poden classificar-se en diversos tipus. A continuació se'n detallen alguns **[11]**.

6.2.1. Mètodes constructius

Son aquells mètodes que afegeixen components individuals a una solució parcial fins que s'obté una solució factible. El més popular d'aquests mètodes és l'algoritme *Greedy* (golós), el qual construeix la solució afegint-hi en cada pas només un element, buscant el màxim benefici en cada pas.

6.2.2. Mètodes destructius

En els mètodes destructius es tenen tots els elements en una solució infactible i es van eliminant els que allunyen la solució de l'òptim, fins a obtenir-ne una factible.

6.2.3. Mètodes de descomposició

Aquests mètodes divideixen el problema en subproblemes més petits, essent la sortida d'un l'entrada del següent i així successivament. Així, quan s'han resolt tots els subproblemes, s'obté també la solució del problema global.

6.2.4. Mètodes de reducció

Els mètodes de reducció identifiquen alguna característica que podria tenir la solució òptima i d'aquesta manera simplifiquen el problema. Per exemple, la detecció d'alguna variable amb certs valors o correlació.

6.2.5. Mètodes de manipulació del model

Els mètodes de manipulació del model modifiquen les estructures del model per tal de fer-lo més senzill de resoldre, deduint, a partir de la solució del problema modificat, la solució del problema original.

6.2.6. Mètodes de cerca per entorns

Aquest tipus d'heurístiques parteixen d'una solució inicial factible (probablement obtinguda a partir d'una altra heurística) i mitjançant alteracions de la solució de forma iterativa, emmagatzemen la millor de les solucions trobades com a millor solució coneguda. A aquesta categoria pertanyen dues de les heurístiques clàssiques: els algoritmes de descens exhaustiu i no exhaustiu.

En l'heurística de descens exhaustiu (AED), tots els veïns es generen a partir de la solució en curs i s'avaluen. Si el millor d'ells és millor que la solució en curs, s'emmagatzema com a nova solució en curs i es repeteix el procediment. En cas contrari, si el millor veí és pitjor o igual a la solució en curs, el procediment es dóna per acabat.

En canvi en el mètode de descens no-exhaustiu (ANED), tots els veïns es generen a partir de la solució en curs en un cert ordre i s'avaluen. Si un d'ells és millor que la solució en curs, s'emmagatzema com a nova solució en curs (sense acabar la generació dels veïns de la solució primitiva) i es continua aplicant el procediment als veïns de la nova solució en curs. Quan s'han generat tots els veïns d'una determinada solució en curs sense que cap sigui millor, el procediment es dóna per acabat.

6.3. Mètodes metaheurístics

En els procediments heurístics és difícil escapar dels òptims locals, que poden estar molt lluny de l'òptim global. Per escapar d'aquests òptims locals a vegades es necessiten moviments que empitjorin (o no millorin) la funció objectiu sense repetir solucions ja analitzades i aplicant un cert criteri de parada.

Les metaheurístiques són estratègies iteratives que combinen diferents conceptes "intel·ligents" per explorar l'espai de cerca escapant d'òptims locals. En cada iteració manipulen una única solució o un conjunt d'aquestes i a mesura que el procés avança, la solució o les solucions van millorant.

Utilitzant la definició d'Osman i Kelli (1996) **[12]**:

"Els procediments metaheurístics són una classe de mètodes aproximats que estan dissenyats per resoldre problemes difícils d'optimització combinatòria, en els quals els heurístics clàssics no són ni efectius ni eficients."

A continuació, s'exposen els procediments metaheurístics més utilitzats.

6.3.1. Cerca Tabú (TS, *Tabu Search*)

La cerca tabú, tècnica proposada per Glover (1989) **[13]**, utilitza la cerca per entorns seleccionant el millor dels moviments en cada pas i prohibint-ne altres en un determinat horitzó temporal.

És una evolució del mètode de descens (AED o ANED), en el qual es parteix d'una solució inicial i s'analitzen les solucions veïnes buscant una solució en què la funció objectiu tingui un valor inferior respecte a la solució actual. Si en el veïnatge no hi ha cap solució millor, el procediment s'acaba però la solució trobada podria ser un mínim local que està molt lluny de la solució optima.

El TS permet moure's a solucions veïnes pitjors perquè la solució no es quedi atrapada en mínims locals, i marca les últimes solucions analitzades com a solucions prohibides o *tabú*, per evitar tornar a caure en el mateix mínim local en la cerca de veïns següent.

6.3.2. GRASP (Greedy Randomized Adaptive Search Procedure)

El GRASP és una metaheurística de tipus iteratiu desenvolupada per Feo i Resende (1989) **[14]**. En aquests algoritmes es repeteixen iterativament dues fases. En la primera es construeix cada vegada una solució inicial utilitzant una heurística *Greedy* aleatoritzada, és a dir, es tria la solució a base que en cada pas de manera aleatòria s'escull un element entre una llista de candidats (normalment els millors). En la segona fase de l'algoritme, les solucions factibles generades passen per un procés de millora local mitjançant un procediment d'intercanvi i la millor solució obtinguda es guarda com a resultat del procediment.

6.3.3. Recuita Simulada (SA, *Simulated Annealing*)

L'SA parteix de la idea que, si la solució en curs es troba en un mínim local respecte el seu veïnatge, l'algorisme només podrà escapar-ne d'aquest si es pren com a nova solució en curs un veí pitjor, el qual potser tindrà veïns millors que l'antic mínim local.

Per aconseguir-ho, utilitza una estratègia heurística de cerca local en la qual l'elecció del nou element de l'entorn es realitza de manera aleatòria.

Es parteix d'una solució inicial determinada per una heurística i mitjançant l'exploració del seu entorn, tracta de millorar la solució. En cada iteració, el procediment considera alguns veïns de la solució en curs. Si el valor de la seva funció objectiu és igual o millor que el de la solució en curs, el veí passa a ser la nova solució en curs; en canvi, si el veí és pitjor que la solució en curs, només es prendrà el veí com a nova solució en curs segons una certa probabilitat. Aquest pas es repeteix fins que es dóna una determinada condició de parada (per exemple, un nombre determinat d'iteracions o un cert temps de càlcul).

La probabilitat d'acceptar una solució veïna com a nova solució en curs segueix la distribució de Boltzmann:

$$P(\partial E, T) = e^{-\frac{\partial E}{T}} = e^{-\frac{Z(s')-Z(s)}{T}} \quad \text{(Eq. 6.1)}$$

On:

- **Z(*s'*):** és el valor de la funció objectiu de la solució veïna (*s'*)
- **Z(*s*):** és el valor de la funció objectiu de la solució en curs (*s*)
- **T:** és el paràmetre temperatura. Generalment el seu valor es va reduint a llarg de l'aplicació del procediment.

Aquesta probabilitat és menor quan pitjor sigui el valor del veí candidat a ser acceptat com a nova solució en curs, és a dir, com més gran sigui la diferència Z(s')- Z(s), i quan menor sigui el valor del paràmetre T.

6.3.4. Algoritme Genètic (GA, *GeneticAlgorithm*)

El GA, introduïts per Holland (1975) **[15]**, és un tipus d'algoritmes de cerca basat en la selecció natural i la genètica. En lloc d'una única solució en curs s'arrossega una població de solucions. Aquesta població evoluciona i la nova població s'obté mitjançant el creuament i/o la mutació dels progenitors de la població inicial.

Aquest és el mètode seleccionat per resoldre aquest problema i com s'explicarà amb molt més detall en el següent capítol. Es seleccionen els progenitors, es creuen i/o muten i s'obtenen els descendents regenerant la població inicial.

6.3.5. Cerca Local Iterativa (ILS, *Iterated Local Search*)

Segons Hoos i Stützle (2005) **[16]**, l'ILS és una de les metodologies més senzilles i eficaces per evitar l'estancament de la solució al voltant d'òptims locals. En els mètodes ILS es parteix d'una solució inicial, que pot ser generada de forma aleatòria o mitjançant una heurística. A continuació, comença la part iterativa de l'algoritme, la qual consta de tres fases: la cerca local del veïnatge, el criteri d'acceptació i la pertorbació de la solució per sortir de mínim local, explorar altres regions de l'espai de les solucions i trobar altres solucions millors.

6.3.6. Cerca per veïnatges variables (VNS, *Variable Neighborhood Search*)

La Cerca en Veïnatges Variables (*Variable Neighborhood Search*, VNS) és una metaheurística recent per a resoldre problemes d'optimització que té com a idea fonamental el canvi sistemàtic del veïnatge o de l'entorn dins d'una cerca local **[17]**.

6.3.7. Ant Colony Optimization (ACO)

El comportament col·lectiu d'una colònia de formigues ha servit per inspirar aquesta tècnica. El mètode consisteix en simular la comunicació directa (feromones) que utilitzen les formigues, insectes gairebé cecs, per establir el camí més curt entre el seu niu i l'aliment.

En un principi, si existeixen dos camins que un grup de formigues pot prendre per arribar a un aliment, on un podria ser més llarg que l'altre, en un principi, aproximadament un 50% de les formigues aniran pel camí curt i la resta pel llarg. Durant els seus moviments cap al menjar i de tornada al niu van dipositant una substancia química (feromones) i instintivament, segueixen el camí on hi ha més feromones. Atès que la feromona s'evapora més ràpid en el camí llarg que en el curt, aquest últim atraurà a més formigues en els moviments futurs de les formigues entre l'aliment i el niu.

7. Algoritmes Genètics

En el present capítol és presenta d'una forma teòrica l'algoritme que s'ha decidit implementar per solucionar el problema plantejat en el present projecte sobre la programació de peces en màquines en paral·lel.

Com ja s'ha comentat en el capítol anterior, els Algoritmes Genètics van ser introduïts per John Holland, investigador de la Universitat de Michigan, el 1970. Aquest mètode està inspirat en el procés que va observar en l'evolució natural dels essers vius. En un principi va denominar aquesta tècnica com *plans reproductius* però es va acabar fent popular sota el nom d'*Algoritmes Genètics* un cop publicat el seu llibre **[15]**.

7.1. Descripció

En un Algoritme Genètic, la informació d'una possible solució es **codifica** en una sèrie de cadenes (**cromosomes**) que descriuen individus que en el seu conjunt formen una població. Cadascun d'aquests cromosomes és avaluat respecte a l'adaptabilitat al seu entorn (**fitness**). Els cromosomes són seleccionats per parelles (**selecció**) amb una major o menor probabilitat en funció del seu grau d'adaptació, donant origen a nous cromosomes com a resultat del intercanvi d'informació entre les parelles de cromosomes seleccionades (**encreuament**), i podent patir alteracions aleatòries (**mutació**). L'Algoritme Genètic aplica un procediment iteratiu en el qual la població que es guarda per a la següent iteració (nova població) està composada per certs cromosomes escollits segons els criteris marcats pel procés de **regeneració**. En el cas de la població utilitzada en la primera iteració (**població inicial**), en el capítol 9 s'ha detallat com s'obté.

L'esquema general d'un Algoritme Genètic es el següent, basat en **[18]**:

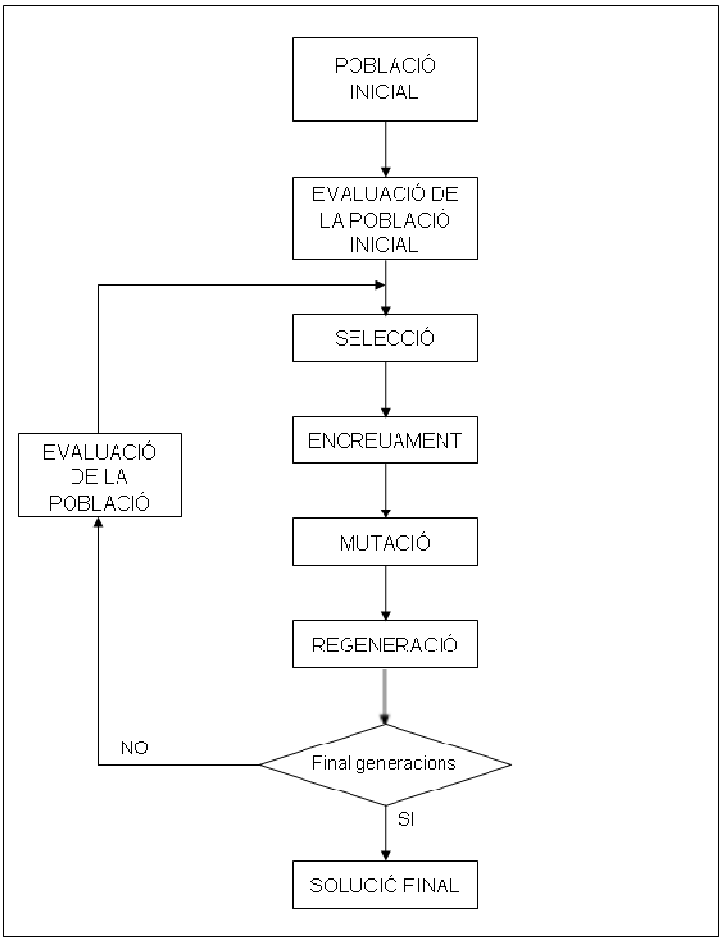


Figura 7.1: Esquema general d'un Algoritme Genètic

7.2. Elements de l'Algoritme Genètic

Per tal de portar a la pràctica l'esquema comentat anteriorment i transformar-lo en un algoritme, cal especificar els següents elements:

- Una representació cromosòmica a mode de codi
- Una població inicial
- Una mesura d'avaluació
- Un criteri de selecció de cromosomes
- Una o diverses operacions de recombinació o encreuament
- Una o diverses operacions de mutació
- Una o diverses regeneracions de la població
- Una condició de final de l'algoritme

7.2.1. Codificació

La primera decisió que s'ha de prendre al dissenyar un algoritme genètic és la codificació. L'objectiu és permetre que els elements característics del problema es puguin representar de tal forma que resulti senzilla la seva implementació i comprensió.

La codificació més comú es mitjançant cadenes binàries, tot i que també es poden utilitzar números reals o lletres, vectors, arbres o grafs. El primer d'aquests esquemes, el que té més popularitat, va ser proposat originàriament per Holland (1975) **[15]**, és fàcil d'implementar i la seva convergència està provada. De totes formes, normalment, l'elecció d'un o altre mètode depèn de la complexitat del problema.

7.2.2. Població inicial

Per constituir la població inicial, que serà la població base de les successives generacions, existeixen diferents mètodes. Sol ser concebuda aleatòriament, tot i que últimament s'estan utilitzant mètodes heurístics per generar solucions inicials de bona qualitat. En aquest cas, és important garantir la diversitat estructural d'aquestes solucions per tenir una "representació" de la major part de la població i evitar una convergència prematura **[19]**.

Una qüestió important és la relacionada amb la dimensió idònia de la població, no pot ser ni molt petita ni molt gran. En el primer cas correríem el risc de no cobrir tot l'espai de cerca, i en el segon cas podríem tenir problemes relacionats amb l'excessiu cost computacional.

Goldberg (1989) **[20]** va demostrar que la dimensió òptima de la població, amb codificació binaria, creix exponencialment amb la longitud del cromosoma. Estudis posteriors coincideixen en afirmar que una població inicial no superior a 30 individus assegura trobar una solució factible al problema. En canvi, el que sí sembla segur és que la dimensió de la població inicial ha d'estar relacionada amb la longitud dels cromosomes i la complexitat del problema.

7.2.3. Mesura d'avaluació o *fitness*

El *fitness* és una mesura d'avaluació que assigna a cada cromosoma un número real que reflexa el seu nivell d'adaptació al problema. Aquest valor també rep el nom de funció d'aptitud, funció adaptació o directament funció objectiu. És la base per determinar quines solucions tenen major o menor probabilitat de sobreviure.

Aquesta funció no ha de crear diferències molt grans entre individus per no provocar una convergència prematura ni diferències molt petites que puguin produir un estancament en el procés de cerca de la solució.

La regla general per construir un bon *fitness* és que aquest ha de reflectir el valor de l'individu de manera "real". El problema és que en molts problemes, on existeix una gran quantitat de restriccions, una part dels punts de l'espai de cerca representen individus no vàlids.

Per evitar aquesta situació, existeixen diverses solucions tals com no considerar aquells individus que no verifiquen les restriccions i seguir iterant fins obtenir individus vàlids, assignar a aquests individus un *fitness* igual a zero o reconstruir aquells individus que no verifiquen les restriccions per mitjà d'un nou operador que s'acostuma a denominar *reparador*. La idea general consisteix en dividir el *fitness* de l'individu per una quantitat (la penalització) que guardi relació amb les restriccions que aquest individu viola, o bé amb el cost associat a la conversió d'aquest individu en altres que sí compleixin les restriccions.

7.2.4. Selecció

La selecció és el procés pel qual s'escullen una o diverses parelles d'individus de la població inicial perquè exerceixin el paper de progenitors, encreuant-se posteriorment i obtenint la descendència.

Només es poden seleccionar dos individus perquè es regenerin de entre tots els individus de la població o es pot optar per aparellar a un major número d'individus que formen la població. La primera és la que s'aproxima més al model biològic, però la segona és la que obté millor resultats en un temps menor.

Els quatre mètodes més típics de selecció de parelles són **[21]**:

1. **Fit-Fit:** els més competitius s'encreuen amb els que ho són menys, i els menys competitius entre ells. En una llista ordenada de forma decreixent segons el *fitness* dels individus, el primer s'aparellaria amb el segon, el tercer amb el quart i així successivament.
2. **Fit-weak:** els individus amb millor *fitness* s'encreuen amb els de pitjor. El millor avaluat s'encreua amb el pitjor avaluat, el segon amb millor *fitness* amb el segon pitjor, i així successivament.

3. **Random:** l'aparellament de progenitors és totalment a l'atzar, de forma que les probabilitats que tenen els individus per encreuar-se són equiprobables.
4. **Roulette:**la probabilitat de cada individu a ser seleccionat per encreuar-se és ponderada segons el valor del *fitness* que posseeix. Rep el nom de "Ruleta" per l'analogia que existeix entre aquest mètode i una "ruleta de la sort" on aquells individus amb una millor mesura d'avaluació són els que ocupen una major superfície a la ruleta i, tenen per tant més possibilitats de ser seleccionats.

$$P_{sel_t} = \frac{1/_{fitness_t}}{\Sigma\left(1/_{fitness_t}\right)} \qquad \text{(Eq. 7.1)}$$

7.2.5. Encreuament

L'operació d'encreuament permet el intercanvi d'informació entre individus d'una població, recombinant els cromosomes i donant lloc a nous individus. L'encreuament s'aplica d'acord a una probabilitat d'encreuament p_c (es creuen si el número generat aleatoriament és menor o igual a p_c) i en un punt, o punts, escollits aleatòriament.

L'objectiu fonamental d'aquesta operació és aprofitar les millors qualitats de tots els individus. Així, si un individu té un subgrup de gens que el fan més competitiu i es creua amb un altre que posseeix un altre conjunt de gens igualment competitius, la addició d'aquets dos grups de gens pot produir un descendent amb un major nivell de *fitness* **[22]**.

Si el nou cromosoma és més competitiu, tindrà més possibilitats de sobreviure quan se li torni a aplicar el *fitness*. En cas contrari, tindrà menors possibilitats de sobreviure a la competència. Els models més habituals d'encreuament són els següents:

1. **Un punt d'encreuament (o bàsic):** s'escull aleatòriament un punt de ruptura dels progenitors i s'intercanvien els seus gens.
2. **Dos punts d'encreuament:** s'escullen aleatòriament dos punts de ruptura per intercanviar els seus gens.
3. **Uniforme:** a cada gen s'escull a l'atzar un progenitor perquè contribueixi en el gen del seu descendent, mentre que el segon descendent rep el gen d'un altre progenitor.
4. **Encreuament per permutació (PMX, SEX, CX, etc):** són operadors més sofisticats com a resultat de mesclar i aleatoritzar els anteriors.

7.2.6. Mutació

L'operació de mutació s'aplica mitjançant l'encreuament amb l'objectiu d'incrementar la diversitat de la població. Es defineix com una variació elemental de les informacions contingudes en de codi genètic. La mutació s'explica d'acord a la possibilitat de mutació p_m. En general, es recomanen baixos percentatges de mutació ja que si no es correria el risc d'aproximar l'evolució a una cerca aleatòria intensament explorativa. Cal destacar la possibilitat d'aplicar aquesta operació tantes vegades com es vulgui en un mateix individu.

Aquesta operació permet augmentar l'exploració en l'espai de cerca cap a nous entorns al produir un increment de la diversitat poblacional i també evita l'aparició d'algoritmes bloquejats o de poblacions degenerades (iguals o similars).

Els models més habituals de mutació són els següents:

1. **Mutació suau:** consisteix en intercanviar, al atzar, dos gens consecutius de la seqüència.
2. **Mutació intercanvi:** consisteix en intercanviar, al atzar, dos gens no consecutius de la seqüència.
3. **Mutació inversa:** consisteix en escollir a l'atzar dos punts de la seqüència i fer que els gens continguts entre els dos punts inverteixin la seva posició.

7.2.7. Regeneració i condició final

Després d'aplicar els encreuaments i les mutacions, s'ha de regenerar la població. La forma més habitual és que els individus nous passin a formar part de la població substituint, o bé els individus menys competitius de la població, o bé els pares que els han engendrat. També es pot escollir directament als individus més competitius sense tenir en compte la seva descendència, o seleccionar, segons un paràmetre establert, una part de la població inicial i una part de la població filial.

Aquesta població servirà de població inicial per a la següent generació, tornant a repetir el procés que formen la selecció, l'encreuament, la mutació i la regeneració, i sempre amb l'objectiu de trobar individus més aptes a cada pas.

El nombre d'iteracions o regeneracions dependrà fonamentalment de la complexitat del problema. A major nombre de regeneracions, major serà la probabilitat de trobar una solució

factible o pròxima a l'òptim al final del procés, però s'ha de tenir en compte que si aquest valor és molt elevat, el cost computacional pot ser molt elevat. Per tant, en ocasions el nombre de regeneracions depèn directament del temps d'execució del problema. Una altra possibilitat consisteix en parar l'algoritme quan s'arriba a l'òptim (sempre que sigui conegut), o bé quan es considera que una solució en curs és satisfactòria (respecte la millor solució coneguda).

7.2.8. Altres operadors genètics

Amb el temps, s'han incorporat nous operadors que no formen part de l'Algoritme Genètic bàsic, però la seva utilització aporta millors prestacions al llarg de l'algoritme.

- **Elitisme:** tracta d'evitar que en el transcurs de l'algoritme, individus amb unes excel·lents qualitats puguin no generar descendència o ser aquesta reduïda, perdent una informació genètica valuosa. Mitjançant aquest operador, el millor o els millors individus d'una població s'integren directament a la població filial. Aquest operador permet, per tant, probabilitats d'encreuament més altes.
- **Reinicialització:** aquest operador permet executar de nou l'algoritme, després de la convergència de l'algoritme utilitzat, prenent com a punt de partida una població aleatòria nova a la qual s'insereix l'individu de millor *fitness* de l'evolució finalitzada. Aquest operador és útil si es vol que la solució sigui molt pròxima a l'òptim i es disposa de temps suficient.
- **Repoblació:** s'usa quan l'algoritme s'estanca, i té com a finalitat augmentar la diversitat. Consisteix bàsicament en preservar els individus més adaptats, generant a partir d'ells una nova població que elimina l'anterior aplicant la mutació. Una bona pràctica consisteix en generar una part de la població d'aquesta forma, i la resta de forma aleatòria. Una altra possibilitat, quan es treballa amb estratègies elitistes, consisteix en aplicar aquest operador cada cert nombre de generacions, independentment de què l'algoritme s'estanqui o no, per accelerar la convergència a l'òptim global.
- **Clonació:** consisteix en la duplicació de l'estructura genètica d'un cromosoma per a la generació següent, amb la finalitat de què aquest exemplar sobrevisqui intacte per competir en la nova generació. Aquesta tècnica és especialment útil per aquells algoritmes en els que un excel·lent descendent pot "eliminar" a un no tant bo, però sí un bon pare, que podria resultar útil per posterior aparellaments.

7.3. Problemes dels Algoritmes Genètics

Els Algoritmes Genètics poden presentar una sèrie de problemes que s'han de tenir en compte, tant en el moment de portar a terme la programació de l'algoritme com en la interpretació dels resultats finals. A continuació es comenten tres d'aquests problemes.

7.3.1. Convergència prematura

Aquest és el problema més comú. Radica en trobar un màxim o un mínim relatiu en l'espai de solucions i concentrar tota la cerca en aquest entorn sense buscar en la resta de l'espai de solucions on es podrien trobar millors resultats.

Les possibles solucions estan relacionades amb la variació de diferents operadors: incrementar la probabilitat de mutació, augmentar la dimensió de la població o disminuir el grau de selecció. Totes aquestes mesures tenen com objectiu intentar que l'espai d'exploració de solucions sigui el més extens possible.

7.3.2. Finalització lenta

L'ús dels Algoritmes Genètics no és garantia de trobar sempre una solució factible en un temps adient. Per aqueta raó, moltes vegades s'ha de trobar el compromís entre el grau de factibilitat de la solució i el cost computacional emprat per trobar la solució. Aquest compromís depèn dels operadors, de la dimensió de la població i del nombre de regeneracions.

7.3.3. Obtenció de solucions no factibles

Per problemes amb les restriccions, és habitual trobar que moltes solucions no respecten les restriccions. L'opció més utilitzada és la introducció d'un coeficient de penalització a la funció objectiu en cas de què no compleixi alguna de les restriccions que s'imposen en el problema. S'ha de tenir en compte que una penalització proporcional al grau de factibilitat podria degradar la funció objectiu i una penalització massa gran faria que molts individus no poguessin arribar mai a ser seleccionats, podent perdre un valuosa informació. Per això, en el nostre cas s'ha procurat tenir en comte les restriccions a l'hora de generar les solucions, i evitar així arribar a aquesta situació.

8. Situació inicial

En el present capítol, es descriu el mètode utilitzat per tal de trobar una solució factible en un cas del problema simplificat. Cada peça és fabricada per una màquina del seu tipus; per tant, en aquest cas no interfereixen els temps de preparació de la màquina.

Per tal de trobar una solució factible, s'ha utilitzat l'algoritme presentat per Gharbi i Haouari (2002) **[23]**.

8.1. Definició

La situació de partida descriu un cas real en l'actualitat on es té un conjunt de *m* màquines, definides a l'apartat 5.2, separades en $k = 3$ nivells (alt, mig i baix) que han de processar *n* peces definides a l'apartat 5.3, dividides en aquests mateixos nivells.

Cada màquina inicialment produeix únicament les peces del seu nivell, tal i com es mostra gràficament a la figura 8.1, i per tant no són necessaris els temps de preparació de les màquines.

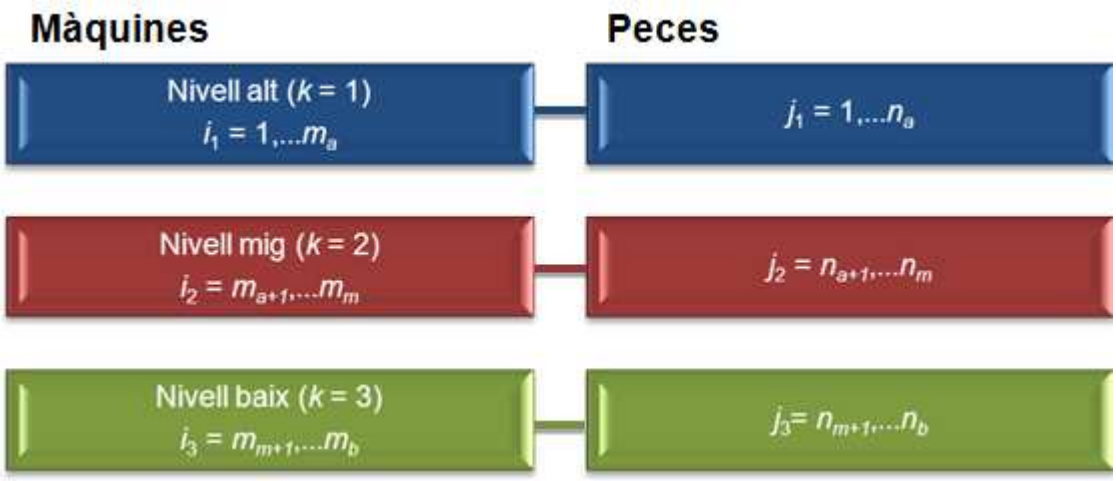


Figura 8.1: Conjunt de peces que es programa en cada tipus de màquina (situació inicial)

La disponibilitat de les màquines és, per a totes, l'instant d'inici de la programació; és a dir, totes les peces poden ser programades a partir del temps $t = 0$. També cal destacar que totes les màquines estan inicialment configurades per poder fabricar peces del seu nivell.

Un aspecte molt important a tenir en compte és la disponibilitat de les peces i el temps de post-procés. La disponibilitat de les peces es representa per un temps de pre-procés en què

les peces no ocupen cap màquina, però no es poden començar a produir fins passat aquest temps. El temps de post-procés és posterior a la utilització de les màquines i tampoc resten ocupades, però no poden ser entregades fins a finalitzar aquest temps. Cada peça està definida pel seu temps de procés p_j; pel pre-procés r_j i pel post-procés q_j .

Per resoldre aquesta situació, s'ha pres com a referència el treball de Gharbi i Haouari (2002) **[23]** *Minimizing makespan on parallel machines subject to realase dates and delivery times*. El text té en compte els temps de pre-procés i post-procés , però només té en compte un sol tipus de màquines, és a dir, màquines i peces d'un sol nivell.

Una peculiaritat d'aquest problema és la seva simetria, ja que compta amb uns temps abans i després de processar les peces.

8.2. Mètode de resolució

En aquest punt ens podem trobar dos casos diferenciats depenent del nombre de màquines disponibles a cada nivell:

1. Una sola màquina d'un determinat nivell (m_k = 1). Per aquest cas es proposa un mètode heurístic per assignar les peces.
2. Més d'una màquina d'un determinat nivell (m_k > 1). Per aquest cas es seguirà l'algoritme proposat per Gharbi.

En aquest primer pas es troba un bon resultat, òptim en el cas que s'utilitzi l'algoritme de Gharbi, amb les màquines classificades per nivells. Òbviament el resultat pot ser millorable quan es comencin a barrejar peces dels diferents nivells, però això s'estudiarà en el següent capítol.

8.2.1. Mètode heurístic de la fase 1 (m_k = 1)

El mètode consisteix en assignar les peces al començament i al final de la seqüència depenent dels seus temps de pre-procés i post-procés. Per assegurar el millor resultat possible, encara que no sigui l'òptim, es plategen dues heurístiques basant-se una en els valors mínims dels temps de pre-procés i post-procés i l'altra en els valors màxims:

a) Heurística 1

La primera heurística assigna al començament les peces que arriben primer, és a dir, el seu temps de pre-procés és més petit i es col·loquen al final de la seqüència les peces que tarden menys a entregar-se, és a dir, amb un temps de post-procés inferior. Amb aquest mètode, s'evita que una peça que tardi molt a entregar-se es produeixi de les últimes de la seqüència.

Sigui un conjunt *J* de *n* peces definides cadascuna per r_j , p_j i q_j ,la metodologia a seguir és la següent:

1) $s_0 \in J$ és una peça del conjunt tal que r_{s_0} o q_{s_0} correspon a min $\{r_j; q_j\}$ $\forall j = 1,...,n$. En cas d'empat de r_{s_0}, s'agafa la de q_{s_0} major i en cas d'empat de q_{s_0}, s'agafa la de r_{s_0} major.
2) Si s_0 ha estat escollida pel seu temps de pre-procés, la peça passa a la primera posició lliure de la seqüència. Si s_0 ha estat escollida pel seu temps d'entrega, la peça passa a la última posició lliure de la seqüència.
3) $J = J \setminus \{s_0\}$. Si $J = \emptyset$, **fi**. En cas contrari, es torna al **punt 1**.

b) Heurística 2

Aquesta segona heurística està basada en l'**heurística 1** però assignant al començament les peces que tarden més a entregar-se, és a dir, amb un temps de post-procés superior i es col·loquen al final de la seqüència les peces que arriben més tard, és a dir, el seu temps de pre-procés és més gran. Amb aquest mètode s'evita, com en l'heurística 1, que una peça que tardi molt a entregar-se es produeixi de les últimes de la seqüència.

La metodologia és la següent:

1) $s_0 \in J$ tal que r_{s_0} o q_{s_0} correspon a max $\{r_j; q_j\}$ $\forall j = 1,...,n$. En cas d'empat de q_{s_0}, s'agafa la de r_{s_0} menor i en cas d'empat de r_{s_0}, s'agafa la de q_{s_0} menor.
2) Si s_0 s'ha escollit pel seu temps de pre-procés, la peça passa a la última posició lliure de la seqüència. Si s_0 s'ha escollit pel seu temps de post-procés, la peça passa a la primera posició lliure de la seqüència.
3) $J = J \setminus \{s_0\}$. Si $J = \emptyset$, **fi**. En cas contrari, es torna al **punt 1**.

El acabar d'aplicar ambdues heurístiques s'obtenen dos vectors no-negatius $\sigma = (t_1,t_2,...,t_n)$ on cada t_j dóna l'instant de començament del treball j ($t_j \geq r_j \ \forall j = 1,...,n$). Per a continuar treballant, s'agafa el vector amb una C_{max} mínima.

Exemple 1

Un exemple ajudarà a clarificar l'aplicació de les heurístiques. Les dades de les peces queden reflectides en la següent taula:

Peça (*j*)	7	8	9	10	11
Nivell (u_j)	3	3	3	3	3
Temps de pre-procés (r_j) [s]	4	4	0	3	5
Temps de procés (p_j) [s]	8	6	5	4	2
Temps de post-procés (q_j) [s]	6	2	1	7	3

Taula 8.1: Dades de l'exemple 1

Es disposa d'una màquina de nivell baix per assignar els treballs. S'aplica l'heurística 1:

1) $J = \{7;8;9;10;11\}$; $s_0 = \{9\}$ i el min és r_s
2) Com que el min és r_s, la peça 9 s'assigna al començament de la seqüència
3) $J = \{7;8;10;11\}$. Es torna al P1
1) $s_0 = \{8\}$ i el min és q_s
2) Com que el min és q_s, la peça 8 s'assigna al final de la seqüència
3) $J = \{7;10;11\}$. Es torna al P1
1) $s_0 = \{10\}$ i el min és r_s
4) Com que el min és r_s, la peça 10 s'assigna a la primera posició lliure de la seqüència
2) $J = \{7;11\}$. Es torna al P1
1) $s_0 = \{11\}$ i el min és q_s
2) Com que el min és q_s, la peça 11 s'assigna a la última posició lliure de la seqüència
3) $J = \{7\}$. Es torna al P1
1) $s_0 = \{7\}$ i el min és r_s
2) Com que el min és r_s, la peça 7 s'assigna a la primera posició lliure de la seqüència
3) Fi

Es resol fins que $J = \emptyset$. Si el vector resultant és: $\sigma = (9;10;7;11;8)$ i l'instant de finalització C_{max} és 27. Anàlogament i amb les mateixes dades, es resol la segona heurística donant el

vector resultant σ = (10;7;9;8;11) i l'instant de finalització C_{max} és 31. Com que la seqüència obtinguda amb l'heurística 1 té menor C_{max}, és la que s'utilitzarà per a les posteriors fases del mètode de resolució. La figura 8.2 mostra la seqüència resultant de l'heurística 1.

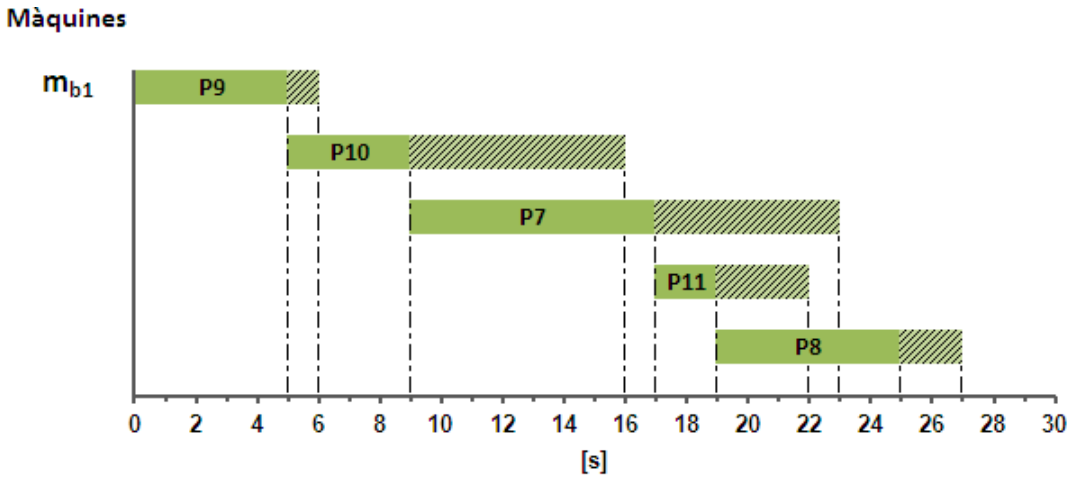


Figura 8.2: Seqüenciació i temporització de peces en una màquina de nivell baix de l'exemple 1

En la figura 8.2 es pot veure com la última peça deixa lliure la màquina a l'instant 25 i després del post-procés, s'entrega en el 27. Si s'estudien les dues peces anteriors, la 7 i la 11, es pot observar com la 7, tot i començar primer, s'acaba entregant més tard. L'objectiu és que això no passi amb la última peça, ja que voldria dir que una peça amb un temps de post-procés elevat s'hauria hagut de començar abans.

8.2.2. Algoritme de Gharbi

Gharbi i Haouari (2002) **[23]** presenten en el seu text *Minimizing makespan on parallel machines subject to realase dates and delivery times* un algoritme per minimitzar el temps de realització d'un conjunt *J* de *n* treballs que s'ha de realitzar en *m* màquines ($m \geq 2$) i dóna com a resultat un vector no-negatiu $\sigma = (t_1,t_2,...,t_n)$ on cada t_j dóna l'instant de començament de la peça *j*.

Resolució de l'algoritme de pre-processament

Per tal de simplificar el problema, es proposa l'aplicació d'un algoritme de pre-processament que permet reduir el nombre de peces a ser programades.

Es tenen les peces ordenades per ordre no-decreixent del temps de pre-procés (en cas d'empat primer el de p_j menor) i s'anomena $j_{(k)}$ al treball que té la k-èssima data d'arribada i

es defineix $J_{r,m} = \{j_{(1)}, j_{(2)}, \dots, j_{(m)}\}$ i $j_0 \in J_{r,m}$ tal que $r_{j_0} + p_{j_0} = min_{j \in J_{r,m}}(r_j + p_j)$. S'assumeix que es compleix la **condició 1**:

$$r_{j_0} + p_{j_0} \leq r_{j_{(m+1)}}$$

Per exemple, si hi ha dues màquines, el conjunt $J_{r,m} = \{j_{(1)}, j_{(2)}, \dots, j_{(m)}\} = \{j_{(1)}, j_{(2)}\}$. El que diu la condició a complir és que el $min\{r_{j_1} + p_{j_1}; r_{j_2} + p_{j_2}\} \leq r_{j_3}$. Això significa que hi haurà una de les dues peces que es podrà produir (i per tant, deixar lliure la màquina) abans no arribi la següent peça al sistema.

Un fet molt interessant que es dóna al tenir en compte el temps de pre-procés i el temps de post-procés és que el problema es pot considerar simètric. Aquest es troba canviant-se les funcions temps de pre-procés i temps de post-procés.

Una conseqüència immediata d'aquesta simetria és la definició de $J_{q,m} = \{j_{(1)}, j_{(2)}, \dots, j_{(m)}\}$ i $j_0 \in J_{q,m}$ tal que $q_{j_0} + p_{j_0} = min_{j \in J_{q,m}}(q_j + p_j)$. $J_{q,m}$ és el resultat d'ordenar les peces per ordre no-decreixent del temps d'entrega (en cas d'empat primer el de p_j menor). S'assumeix que es compleix la **condició 2**:

$$q_{j_0} + p_{j_0} \leq q_{j_{(m+1)}}$$

El significat d'aquesta segona condició és el mateix que la primera però mirat des del punt de vista dels temps de post-procés en comptes dels temps de pre-procés.

Per a resoldre el problema, s'utilitza un algoritme de classificació prèvia que dividirà el conjunt de treballs de J en tres; J_R, J_Q i J_P. El primer conjunt acollirà les peces que es processin al principi, el segon les últimes peces a processar i el tercer, al grup la resta de les peces.

L'algoritme de classificació prèvia queda de la següent manera:

0) En un pas previ de l'algoritme s'inicialitzen els tres conjunts on es repartiran els treballs. Inicialment es té: $J_R = \emptyset$, $J_Q = \emptyset$ i $J_P = J$.
1) Es prova la **condició 1**.
   1.1) Si el nombre de peces de J_P és inferior o igual al nombre de màquines es para. Dit d'una altra manera:
      - Si $|J_P| \leq m$ **stop**.

1.2) Si cap peça satisfà la **condició 1**, es va al **P2**. En cas contrari $J_P = J_P \setminus \{j_0\}$ i $J_R = J_R \cup \{j_0\}$ i anar al **P1.1**.

2) Es prova la **condició 2**.

2.1) Si el nombre de peces a J_P és inferior o igual al nombre de màquines es para. Dit d'una altra manera:

- Si $|J_P| \leq m$ **stop**.

2.2) Si cap peça satisfà la **condició 2** es va al **P3**. En cas contrari $J_P = J_P \setminus \{j_0\}$ i $J_Q = J_Q \cup \{j_0\}$ i anar al **P2.1**.

3) Si cap peça és moguda en el **P2**, **stop**. En cas contrari es va al **P1**.

Al finalitzar el pre-procés es tindran les peces classificades a J_R, J_Q i J_P.

Exemple 2

Per entendre millor l'algoritme es proposa resoldre un exemple considerant dues màquines de nivell baix i les peces de la taula 8.2.

Peça (j)	7	8	9	10	11
Nivell (u_j)	3	3	3	3	3
Temps de pre-procés (r_j) [s]	6	4	0	3	8
Temps de procés (p_j) [s]	8	5	2	1	4
Temps de post-procés (q_j) [s]	6	2	1	4	7

Taula 8.2: Dades de l'exemple 2

0) Inicialització dels tres conjunts: $J_R = \emptyset$, $J_Q = \emptyset$ i $J_P = J$.

1) Es prova la condició 1. En una primera instància, es té $J_{r,m} = \{9,10\}$. $min_{j \in J_{r,m}}(r_j + p_j) = \min\ (2,4) = 2 \rightarrow j_0 =$ {9}. Es prova la condició $r_{j_0} + p_{j_0} \leq r_{j_{(m+1)}}$ amb els valors $2 \leq 4$ condició complerta. Conseqüentment, $J_R = J_R \cup \{9\}$ i els conjunts queden definits com $J_R = \{9\}$; $J = \{7;8;10;11\}$ i $J_Q = \{\emptyset\}$.

Com s'ha satisfet la condició, es torna a provar la condició 1.

1) Es prova la condició 1. Es té $J_{r,m} = \{10,8\}$. $min_{j \in J_{r,m}}(r_j + p_j) = \min\ (4,9) = 4 \rightarrow j_0 =$ {10}. Es prova la condició $r_{j_0} + p_{j_0} \leq r_{j_{(m+1)}}$ amb els valors $4 \leq 6$ condició complerta. Conseqüentment, $J_R = J_R \cup \{10\}$ i els conjunt queden definits com $J_R = \{9;10\}$; $J = \{7;8;11\}$ i $J_Q = \{\emptyset\}$.

Com s'ha satisfet la condició, es torna a provar la condició 1.

1) Es prova la condició 1. Es té $J_{r,m} = \{8,7\}$. $min_{j \in J_{r,m}}(r_j + p_j) = \min\ (9,14) = 9 \rightarrow j_0 =$ {8}. Es prova la condició $r_{j_0} + p_{j_0} \leq r_{j_{(m+1)}}$ amb els valors $9 \leq 8$ condició NO complerta. Conseqüentment, es passa al punt 2.
2) Es prova la condició 2. Es té $J_{q,m} = \{7,8\}$. $min_{j \in J_{q,m}}(q_j + p_j) = \min\ (14,7) = 7 \rightarrow j_0 =$ {8}. Es prova la condició $q_{j_0} + p_{j_0} \leq q_{j_{(m+1)}}$ amb els valors $7 \leq 7$ condició complerta. Conseqüentment, $J_Q = J_Q \cup \{8\}$ i els conjunt queden definits com $J_R = \{9;10\}$; $J = \{7;11\}$ i $J_Q = \{8\}$.

En aquest punt, s'acaba l'algoritme de classificació prèvia, ja que tot i que en aquesta última iteració s'ha complert la **condició 2** i s'ha pogut assignar una peça a J_Q , es té que $|J_P| = m$, **stop**.

Resolució de l'algoritme de Gharbi

Per a poder resoldre el problema s'han de seguir els següents passos:

Pas 1:

Es reparteixen els treballs de J_P a les màquines. S'assigna el treball que primer arribi a la màquina que primer estigui disponible. Amb les dades que s'han extret de l'exemple anterior, on $J = \{7;11\}$. Com que el primer dels dos en arribar és el treball 7, aquest s'assignarà a la primera màquina de nivell baix que estigui buida. El següent treball, l'11, s'assigna a l'altra màquina.

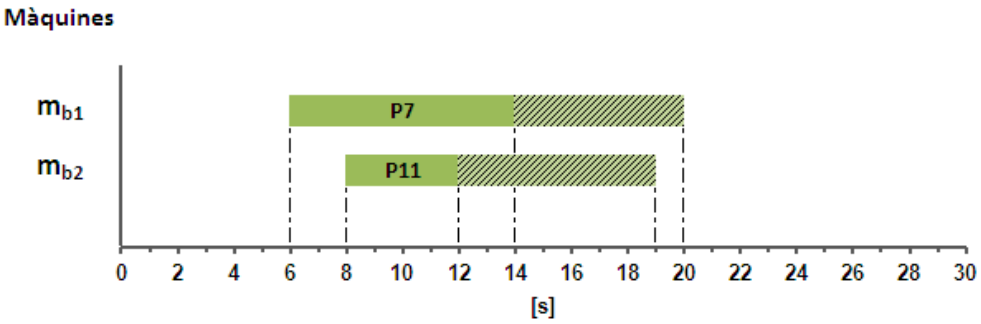


Figura 8.3: Gràfica resultant del Pas 1

Un cop assignades, el vector $\sigma = (6,0,0,0,8)$ amb una $C_{max} = 20$.

Pas 2:

Un cop assignades les peces de *J* , es defineixen els conceptes de:

- $u_i(\sigma)$ és l'instant en què la màquina *i* està disponible en la programació σ .
- σ^{-1} és la programació simètrica de σ.

Per assignar els conjunts J_R i J_Q, aquest pas consta de dues parts: una primera, on s'assignen les peces de J_Q i, una segona, on s'assignen les de J_R. Tot seguit, es presenta l'algoritme proposat per Gharbi:

1) En aquesta primera part s'assignen totes les peces que formen el conjunt J_Q
   1.1) Si $J_Q = \emptyset$ es va al **punt 2**.
   1.2) La peça $j_0 \in J_Q$ és aquella tal que $q_{j_0} + p_{j_0} = max_{j \in J_Q}(q_j + p_j)$.
   1.3) S'anomena m_0 a la màquina amb disponibilitat $u_0 = min_{i=1,\dots,n} u_i(\sigma)$.
   1.4) Es programa la peça j_0 a la màquina m_0 i s'actualitza $u_0 = max(u_0, r_{j_0}) + p_{j_0}$.
   1.5) $J_Q = J_Q \setminus \{j_0\}$ i es torna a anar al **punt 1.1**.
2) $\sigma = \sigma^{-1}$
3) En aquesta segona part, comença la segona part i s'assignen les peces de J_R.
   3.1) Si $J_R = \emptyset$ es va al **punt 4**.
   3.2) La peça $j_0 \in J_R$ és aquella tal que $r_{j_0} + p_{j_0} = max_{j \in J_R}(r_j + p_j)$.
   3.3) S'anomena m_0 a la màquina amb disponibilitat $u_0 = min_{i=1,\dots,n} u_i(\sigma)$.
   3.4) Es programa la peça j_0 a la màquina m_0 i s'actualitza $u_0 = max(u_0, q_{j_0}) + p_{j_0}$.
   3.5) $J_R = J_R \setminus \{j_0\}$ i es torna a anar al **punt 3.1**.
4) $\sigma = \sigma^{-1}$

En el **punt 1**, es tracten les peces que estan en el conjunt de J_Q . Per ordre de temps de procés més temps de post-procés decreixent s'assignen, tot fent iteracions, a la primera màquina que està disponible. Es requereix aquest ordre en l'assignació de peces, ja que si no es segueix, no es complirà el temps de cicle proposat. En cada iteració de l'algoritme, la peça assignada es treu del conjunt J_Q.

Un cop assignades les peces de J_Q, es fa el simètric de la programació per tal de poder assignar les peces pertanyents al conjunt J_R, que s'assignaran a la primera màquina que estigui disponible per ordre decreixent de la suma dels temps de pre-procés i de procés.

Igual com en l'altre conjunt, cada cop que s'assigna una peça, aquesta es treu del conjunt J_R.

Si s'aplica l'algoritme a les dades de l'exemple anterior, els resultats queden de la següent manera:

Exemple 2 (cont):

En aquesta part de la resolució de l'exemple es veurà el pas 2 de l'algoritme de Gharbi.

1) Primerament s'assignaran les peces de $J_Q = \{8\}$.
   1.1) Com que $J_Q \neq \emptyset$ es va al P1.2.
   1.2) La peça $j_0 \in J_Q$ és aquella tal que $q_{j_0} + p_{j_0} = max_{j \in J_Q}(q_j + p_j)$. Com que només hi ha una peça, $j_0 = \{8\}$ amb $q_{j_0} + p_{j_0} = 7$.
   1.3) S'anomena m_0 a la màquina amb disponibilitat $u_0 = min_{i=1,\dots,n} u_i(\sigma)$. Donat que σ = (6,0,0,0,8), es té que l'instant la disponibilitat $u_i(\sigma) = (14,12)$ més petita és de 12 i pertany a la màquina m_{b_2}.
   1.4) Es programa la peça 8 a la màquina m_{b_2} i s'actualitza $u_0(m_{b_2}) = max(u_0, r_{j_0}) + p_{j_0} = 12 + 5 = 17$, per tant $u_i(\sigma) = (14,17)$ i al calcular el nou vector σ = (6,12,0,0,8). La seva assignació es veu a la figura 8.4.
   1.5) $J_Q = J_Q \setminus \{j_0\}$ i es torna a anar al P1.1.
   1.1) Com que $J_Q = \emptyset$ es va al pas 2.

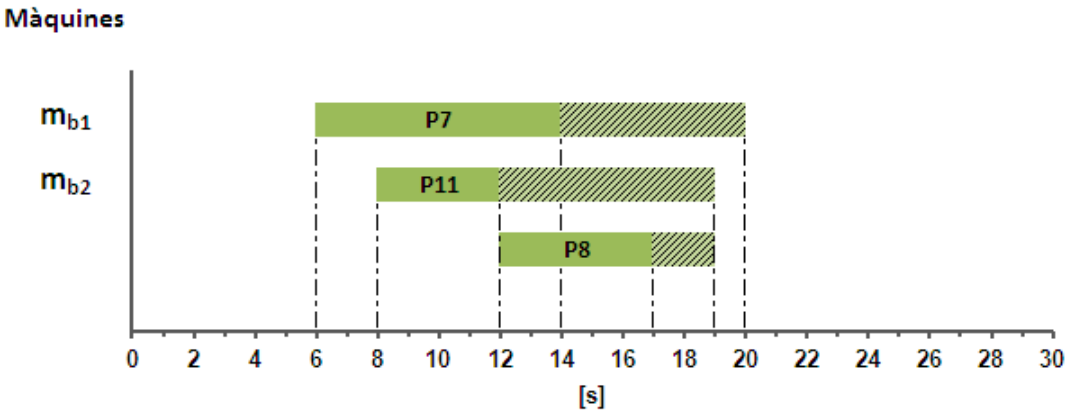


Figura 8.4: Gràfica resultant de l'exemple 2 (cont)

2) S'estableix $\sigma = \sigma^{-1} = (14,8,20,20,12)$. Per fer l'invers de σ, cal mirar l'instant de finalització p_j de cada peça, des de C_{max} ; 20 en aquest exemple.
3) En aquest punt, comença la segona part i s'assignen les peces de $J_R = \{9;10\}$.
   3.1) Si $J_R = \emptyset$ es va al P4.
   3.2) La peça $j_0 \in J_R$ és aquella tal que $r_{j_0} + p_{j_0} = max_{j \in J_R}(r_j + p_j) = max\{0 + 2, 3 + 1=4$ i pertany a la peça 10.
   3.3) S'anomena m_0 a la màquina amb disponibilitat $u_0 = min_{i=1,\dots,n} u_i(\sigma)$, amb $u_i(\sigma) = (14,12)$, $u_0 = 12$ i pertany a la màquina m_{b_2}.
   3.4) Es programa la peça 10 a la màquina m_{b_2} i s'actualitza $u_0 = max(u_0, q_{j_0}) + p_{j_0} = max(12,4) + 1 = 13$, per tant $u_i(\sigma) = (14,13)$ i σ = (14,8,20,13,12).
   3.5) $J_R = J_R \setminus \{j_0\}$ i es torna a anar al P3.1.

Es torna a repetir el punt tres sencer per assignar la peça 9, que s'assigna a la màquina m_{b_2}, ja que és la que esta disponible abans, $u_i(\sigma) = (14,13)$, quedant σ = (14,8,15,13,12). Finalment queda tornar a invertir σ.

4) $\sigma = \sigma^{-1} = (6,12,5,7,8)$.

El resultat es mostra gràficament en la figura 8.5:

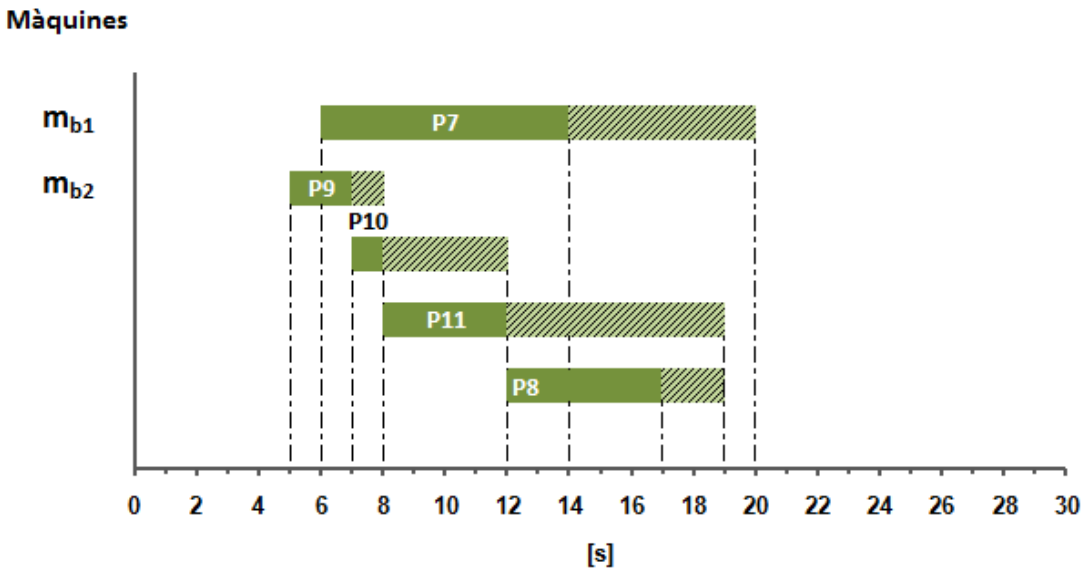


Figura 8.5: Gràfica resultant de l'exemple 2 amb l'aplicació completa de l'algoritme de Gharbi

9. Procediments de resolució proposats

Per resoldre el problema plantejat en el capítol 5, s'han desenvolupat dos procediments de resolució: el primer basat en un mètode heurístic i el segon basat en un metaheurístic (algoritme genètic).

9.1. Presentació del problema

La situació inicial (del capítol anterior) es compararà amb la situació on les màquines de nivell alt són capaces de produir les peces de nivell alt, mig i baix; les de nivell mig són capaces de produir les de nivell mig i baix i les de nivell baix les peces de nivell baix. En resum, cada màquina podrà produir les peces del seu nivell i els seus inferiors. La diferència amb el problema de l'apartat anterior és que es té en compte el temps de preparació de les màquines per tal d'adaptar-les i que puguin produir peces d'altres nivells. Aquesta explicació es mostra a la figura 9.1.

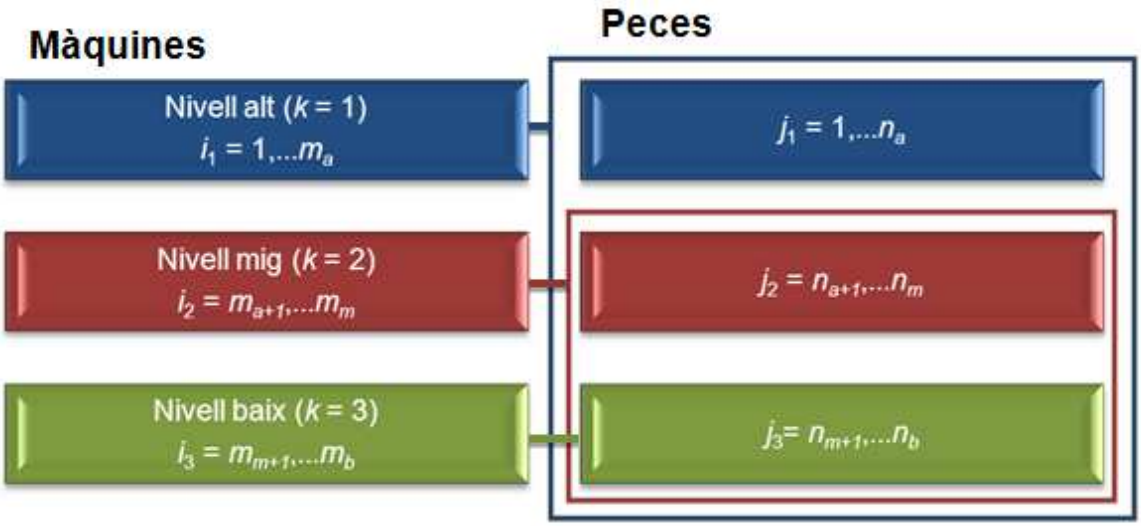


Figura 9.1: Conjunt de peces que es poden seqüenciar en cada màquina

9.2. Procediment de resolució heurístic

El procediment de resolució heurístic es basa en la generació d'un conjunt de 12 solucions mitjançant una efectiva heurística. Aquesta heurística, coneguda com a *Multiple Insertion* (MI), consisteix en ordenar les peces en una matriu seguint un criteri determinat (en cas d'empat, per ordre lexicogràfic) o mitjançant un mètode aleatori. A continuació, per ordre de com s'han ordenat, cada peça és programada en cadascuna de les posicions de les màquines on sigui factible i finalment es programa en aquella posició en què resulta un menor instant d'entrega de l'última peça (C_{max}). En cas d'empat de C_{max}, la peça és programa en aquella posició en la qual el temps de preparació sigui menor.

En aquest cas, els criteris seguits per ordenar les peces són els següents:

- Menor temps de procés.
- Menor temps de pre-procés.
- Menor temps de post-procés.
- Major temps de procés.
- Major temps de pre-procés.
- Major temps de post-procés.
- La resta dels 6 solucions és generen de forma aleatòria.

La solució final és la millor de les 12 solucions obtingudes mitjançant aquest mètode. Aquest mètode l'anomenarem **Heurística 2.**

9.2.1. Procediment heurístic simplificat

Es proposa estudiar un procediment simplificat d'aquest procediment heurístic. En aquest cas únicament es generen un conjunt de 6 solucions sense aplicar el mètode d'ordenació aleatòria. Aquest procediment ens permetrà determinar la influència dels individus aleatoris en la solució del problema. Aquest mètode l'anomenarem **Heurística 1.**

Exemple 3

Per tal d'entendre millor aquest procediment, a continuació es presenta un exemple que simula la segona i la tercera forma de crear una solució. En aquest cas es seguirà la regla d'ordenar les peces per menor temps de pre-procés amb $m_a = 1$, $m_m = 1$, $m_b = 1$. Les peces queden definides en la taula 9.1 i els temps de preparació en la taula 9.2.

Peça (j)	1	2	3	4	5	6	7	8	9
Nivell (u_j)	1	1	1	2	2	2	3	3	3
Temps de pre-procés (r_j) [s]	2	8	11	1	5	5	0	6	7
Temps de procés (p_j) [s]	3	2	3	3	11	4	6	2	6
Temps de post-procés (q_j) [s]	7	3	0	5	3	0	1	2	1

Taula 9.1: Dades de les peces de l'exemple 3

$k \setminus k'$	1	2	3
1	-	1	3
2	1	-	2
3	3	2	-

Taula 9.2: Dades dels temps de preparació (en segons) de l'exemple 3

S'ha de tenir present que les màquines inicialment estan preparades per processar les peces del seu nivell. En canvi, acaben en la configuració de l'última peça que processen.

1) S'ordenen les peces per menor temps de pre-procés:

7	4	1	5	6	8	9	2	3

2) La peça 7 es programa a la màquina de nivell baix, al no necessitar temps de preparació.
3) La peça 4 es programa a la màquina de nivell mig, al no necessitar temps de preparació.
4) La peça 1 es programa a la màquina de nivell alt, al no necessitar temps de preparació.
5) La peça 5 es prova de programar en cadascuna de les posicions factibles i es fixa en aquella que produeix un menor C_{max}, a la màquina de nivell mig i després de la peça 4.
6) La peça 6 es prova de programar en cadascuna de les posicions factibles i es fixa en aquella que produeix un menor C_{max}. En aquest cas el millor C_{max} s'obté quan es programa a la màquina de nivell alt, després de la peça 1.

Se segueix aquest procediment fins a programar totes les peces, resultat que es mostra en la figura 9.2.

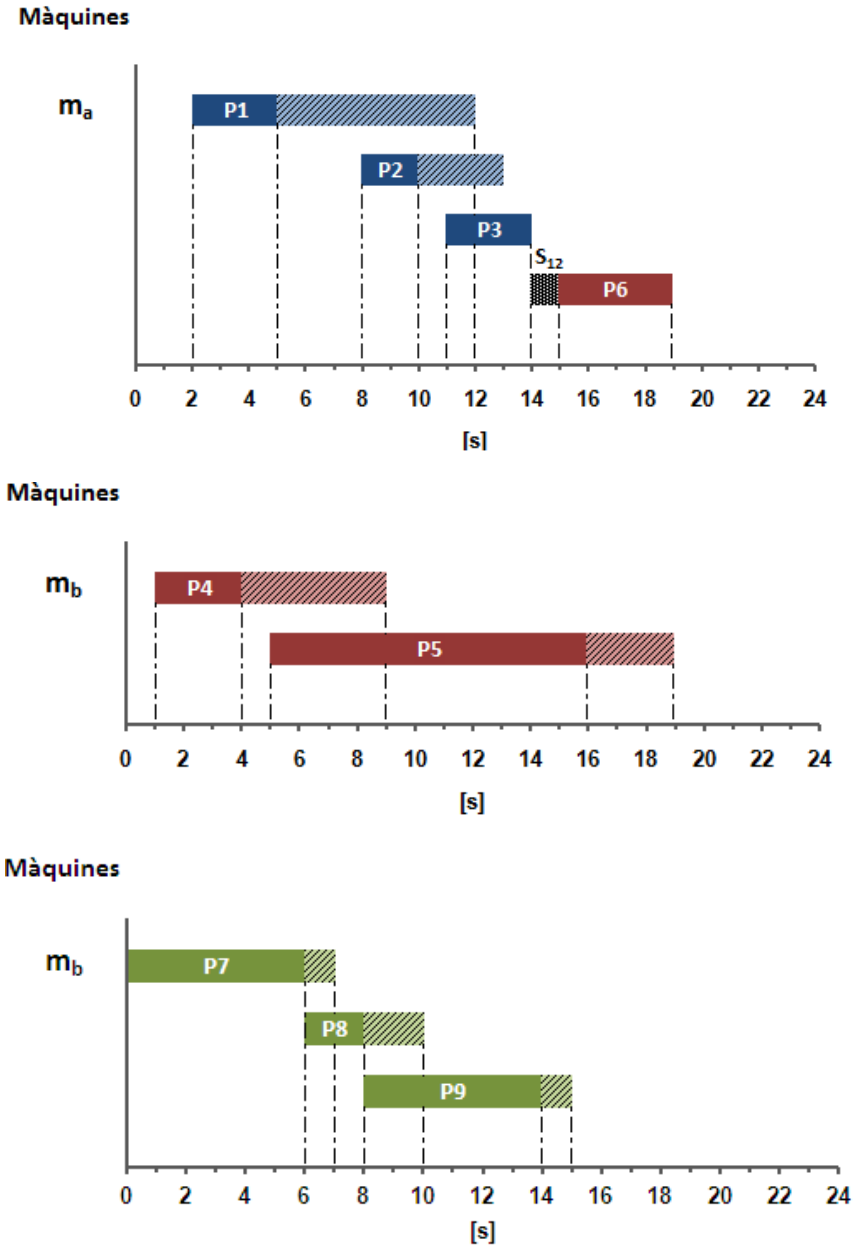


Figura 9.2: Diagrames de Gantt de les màquines de l'exemple 3 després d'aplicar el mètode heurístic

9.3. Procediment de resolució metaheurístic

En el capítol 7 s'ha presentat l'algoritme utilitzat pel segon procediment de resolució del problema des d'un punt de vista teòric i general, mentre que en el present apartat, es descriu més en detall alguns aspectes pràctics de l'algoritme.

El mètode utilitzat es divideix en dues parts; la primera, explicada en la *Fase I* del capítol 8, es troba una solució factible utilitzant l'algoritme presentat per Gharbi i Haouari (2002) **[23]** i posteriorment s'aplica dues fases de millora que intenten omplir els possibles temps morts de la màquina que ha deixat la primera solució; la segona consisteix en l'aplicació de l'Algoritme Genètic utilitzant la solució trobada en la primera part com a membre de la població inicial.

9.3.1. Mètode de resolució (1ª part)

Tot seguit es presentarà el mètode, amb les seves tres fases, utilitzat per resoldre la primera part del problema.

9.3.1.1. Fase I: Generació d'una solució segregada

Aquesta part és idèntica a la descrita en el capítol 8.2 on es busca una solució a un cas simplificat en què totes les peces es fabriquen en màquines del seu mateix tipus. A partir de la solució trobada en aquesta primera fase, s'apliquen les dues següents fases que busquen una millora de la solució tal com mostra la següent figura.

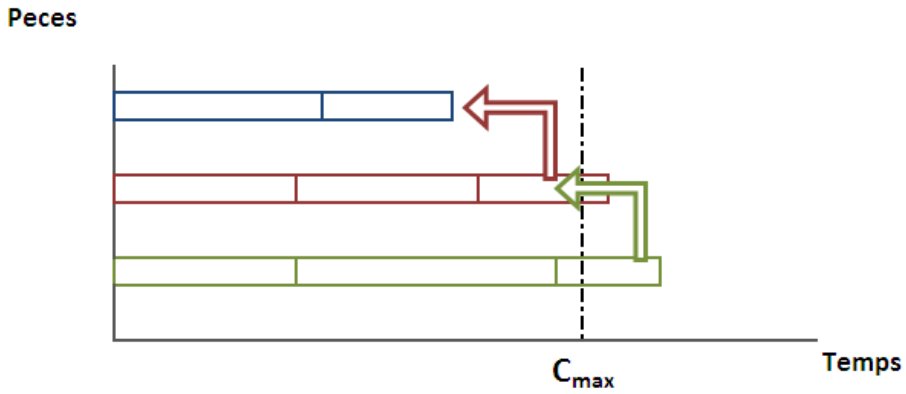


Figura 9.3: Desplaçament de peces per aconseguir reduir C_{max}

9.3.1.2. Fase II: Millora estàtica

Un cop es tenen les peces assignades a màquines del seu propi nivell, es té una possible solució (segregada) del problema. Aquesta solució, però, és molt probable que presenti temps ociosos en la seva programació, és a dir, períodes de temps en què una màquina no està produint. Per a poder obtenir una millora s'utilitzarà la capacitat que tenen les màquines de poder fabricar les peces de nivells inferiors al seu. Per a poder-ho fer, abans es definiran els temps ociosos que tenen les màquines a través d'un vector de quatre dimensions (f_i^k,w_i^k,s_i^k,r_i^k); on f_i^k representa l'instant on la màquina k comença a no produir (període

ociós), w_i^k la durada en què la màquina *k* està inactiva, s_i^k el nivell de la peça abans d'aquest període i r_i^k el nivell de la peça a processar un cop acabat el període ociós, *k: 1,...,m*.

Recordem que en cas que no hi hagi cap més peça, la màquina pot acabar configurada en el nivell corresponent al d'aquesta última peça.

Una peça (j_0) d'un nivell es podrà "pujar" (assignar a una màquina d'un nivell superior) quan es compleixin aquestes dues condicions:

C1. $r_{j_0} + p_{j_0} + s_{j_0 i}^k \leq f_i^k + w_i^k$. Aquesta condició vol dir que la durada en què la màquina està parada ha de ser més gran o igual que la duració de la peça j_0 més els temps de preparació necessaris.

C2. $r_{j_0} \geq f_i^k + s_{i j_0}^k$. L'instant d'arribada de la peça al sistema, i que per tant pot ser programada, ha de ser igual o més gran que l'instant en què la màquina queda lliure i preparada per processar aquest tipus de peça.

Aquestes dues condicions fan d'aquest un problema estàtic, ja que només es mou la peça que canvia de nivell. Es proposa la següent metodologia per "pujar" peces de nivell:

1) Per a cada màquina de nivell alt es calcula el seu respectiu vector de temps lliure (f_i^k,w_i^k,s_i^k,r_i^k).
2) S'aplica una cerca en totes les peces de nivell mig i es compara cada peça amb els vectors de les màquines de nivell alt.
   2.1) Si una peça compleix les condicions C1 i C2, s'assigna a la màquina de nivell alt **(sense introduir els temps de preparació)** i s'actualitzen els vectors de temps ociosos de la màquina on s'ha assignat. S'estudia la següent peça de la llista i així successivament; en cas d'haver-les mirat totes, anar al **punt 3**.
   2.2) Si una peça no compleix les condicions es busca la següent peça de nivell mig; en cas d'haver-les mirat totes, anar al **punt 3**.
3) Es determinen els temps ociosos de les màquines de nivell mig.
4) S'aplica una cerca en totes les peces de nivell baix i es compara cada peça amb els vectors de els màquines de nivell alt, primerament, i mig, després.
   4.1) Si una peça compleix les condicions C1 i C2, s'assigna a la màquina corresponent **(sense introduir els temps de preparació)** i s'actualitzen els vectors de temps ociosos de la màquina on s'ha assignat i es busca la següent

peça i així successivament; en cas d'haver-les mirat totes, anar al **punt 5**. En cas que sigui possible, s'avancen les peces de la màquina de nivell baix.

4.2) Si una peça no compleix les condicions es busca la següent peça de nivell baix; en cas d'haver-les mirat totes, anar al **punt 5**.

5) S'assignen els setup times entre les peces que sigui necessari de totes les màquines.

6) Avançar tant com es pugui les peces que siguin possibles de les màquines de nivells alt i mig.

Aquest mètode consisteix en, primerament, intentar col·locar treballs de nivell mig a màquines de nivell alt per tal de deixar el màxims de temps ociosos a les màquines de nivell mig i seguidament, col·locar els treballs de nivell baix a les màquines de nivells alt i mig. Cal notar que una peça de nivell baix s'intentarà seqüenciar primer a una màquina de nivell alt que a una de nivell mig, ja que altrament caldria tornar a aplicar una iteració més a l'algoritme per tornar-la a canviar de nivell. És a dir, la idea és intentar carregar al màxim les màquines més flexibles sense penalitzar C_{max} , primer amb peces del seu nivell, després amb peces del seu nivell inferior immediat i per finalitzar amb peces del nivell més baix.

En l'últim pas (el 6), es té en compte que el fet de moure peces genera nous períodes de temps lliures en les màquines i en alguns casos aquests períodes poden permetre un avançament de les peces, i aconseguir millorar C_{max} respecte la fase 1.

Exemple 4

Per entendre millor aquesta fase es presenta un exemple amb $m_a = 1$, $m_m = 1$, $m_b = 1$ i les peces queden definides en la taula 9.3 i els setup times a la taula 9.4.

Peça (*j*)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Nivell (u_j)	1	1	1	2	2	2	3	3	3
Temps de pre-procés (r_j) [s]	1	9	11	1	6	5	5	5	7
Temps de procés (p_j) [s]	3	1	3	3	2	9	1	2	6
Temps de post-procés (q_j) [s]	7	3	0	5	3	0	2	1	1

Taula 9.3: Dades de les peces de l'exemple 4

k \ *k'*	1	2	3
1	-	1	1
2	1	-	1
3	1	1	-

Taula 9.4: Dades dels setup times (en segons) de l'exemple 4

Després d'aplicar la fase 1, *Generació d'una solució segregada*, les peces queden programades tal com mostra la figura 9.4, donant C_{max} =17 .

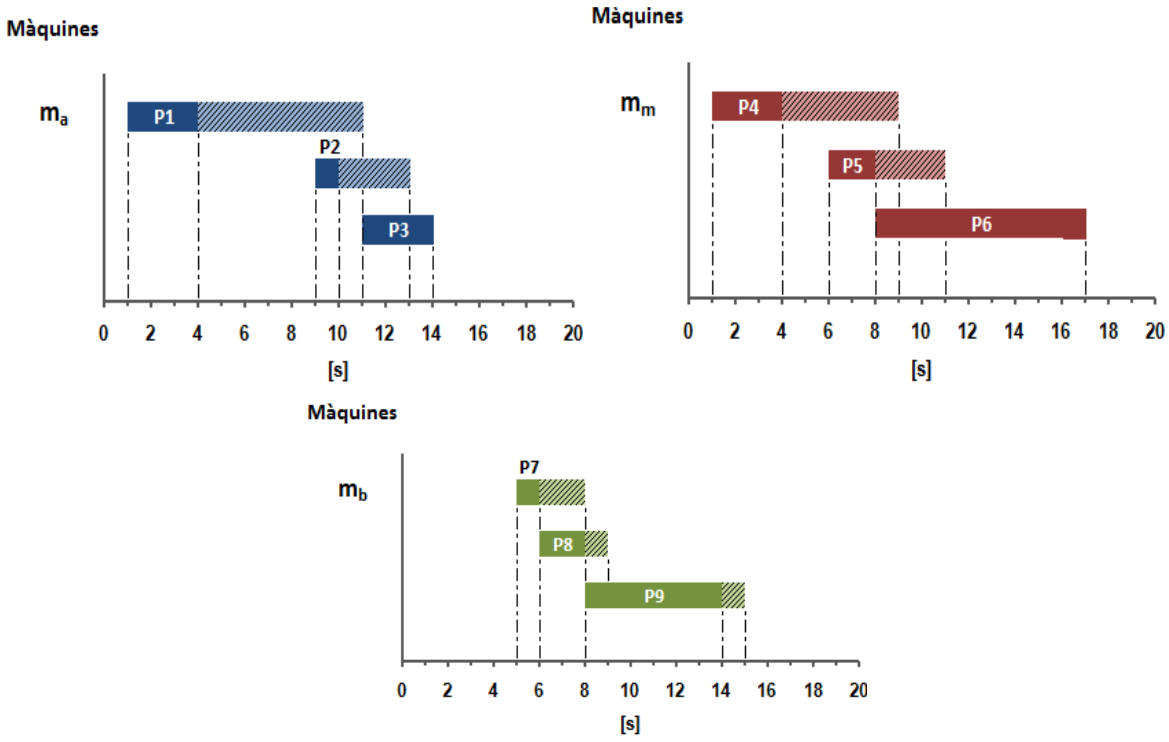


Figura 9.4: Diagrames de Gantt resultats després d'aplicar la fase I a les dades de l'exemple 4

En analitzar el diagrama de Gantt i sense parar molta atenció a les dades de les peces, es poden treure conclusions bastant interessants. Per exemple, es pot observar com les peces de nivell alt ja no es poden programar abans al haver-hi temps ociosos entre totes elles, és a dir, totes les peces comencen en el moment en què estan disponibles, al acabar el pre-procés (r_j). Un altre fet a destacar és que les peces amb major temps de post-procés són les que comencen a ser processades al principi de la seqüència, fet que permet no endarrerint l'instant d'acabament del conjunt de les peces.

A continuació es detallen els passos que s'han seguit per aplicar el segon pas (Fase II). Els passos són els següents:

1) Llistar els vectors de períodes de temps ociós (f_i^k,w_i^k,s_i^k,r_i^k) de les màquines de nivell alt. (f_1^1,w_1^1,s_1^1,r_1^1) = (0,1,1,1) ; (f_2^1,w_2^1,s_2^1,r_2^1) = (4,5,1,1) ; (f_3^1,w_3^1,s_3^1,r_3^1) = (10,1,1,1); (f_4^1,w_4^1,s_4^1,r_4^1) = (14,∞,1,-).
2) S'aplica la cerca per a les peces de nivell mig. Estudi de la peça **4**.
   2.2) Al comprovar les condicions 1 i 2, es veu que compleix la condició 2 $\left(r_{j_0} \geq f_1^k + s1j0k \rightarrow 1 \geq 0+1\right.$, però no la 1 $rj0+pj0+sj01k \leq f1k+w1k \rightarrow 1+4+1 \leq 0+1$. La següent peça a estudiar és la **5**.

   2.1) Al comprovar les condicions C1 i C2, es veu que compleix tant la condició 1 $\left(r_{j_0} + p_{j_0} + s_{j_0 2}^k \leq f_2^k + w_2^k \rightarrow 6+2+1 \leq 4+5\right)$ com la C2 $\left(r_{j_0} \geq f_2^k + s_{2j_0}^k \rightarrow 6 \geq 4+1\right)$ per al segon període ociós de la màquina de nivell alt. Conseqüentment, la peça 5 s'assigna a la màquina de nivell alt i s'actualitza la llista de períodes ociosos de la màquina de nivell alt; ara (f_1^1,w_1^1,s_1^1,r_1^1) = (0,1,1,1) ; (f_2^1,w_2^1,s_2^1,r_2^1) = (4,2,1,2); (f_3^1,w_3^1,s_3^1,r_3^1) = (8,1,2,1); (f_4^1,w_4^1,s_4^1,r_4^1) = (10,1,1,1); (f_5^1,w_5^1,s_5^1,r_5^1) = (14,∞,1,-). La següent peça a estudiar és la **6**.
   2.2) Si s'estudia la peça 6, dóna el resultat que no es pot canviar a la màquina de nivell alt.

Fent el mateix estudi per les peces de nivell baix amb els períodes ociosos de les màquines de nivell alt i mig, tampoc es pot moure res.

5) S'assignen els temps de preparació entre les peces que sigui necessari de totes les màquines. En aquest cas, a la màquina de nivell alt per passar de peces de nivell alt a mitjà i viceversa.
6) A la màquina de nivell mitjà, s'avança la peça 6 fins a començar-la a l'instant 5 i es redueix C_{max} en dos unitats, ja que la programació de peces a la màquina mitjana deixa de ser el coll d'ampolla per ser-ho la màquina de nivell baix.

La figura 9.5 mostra les màquines en les quals ha canviat la programació de les operacions de peces després d'aplicar la fase II.

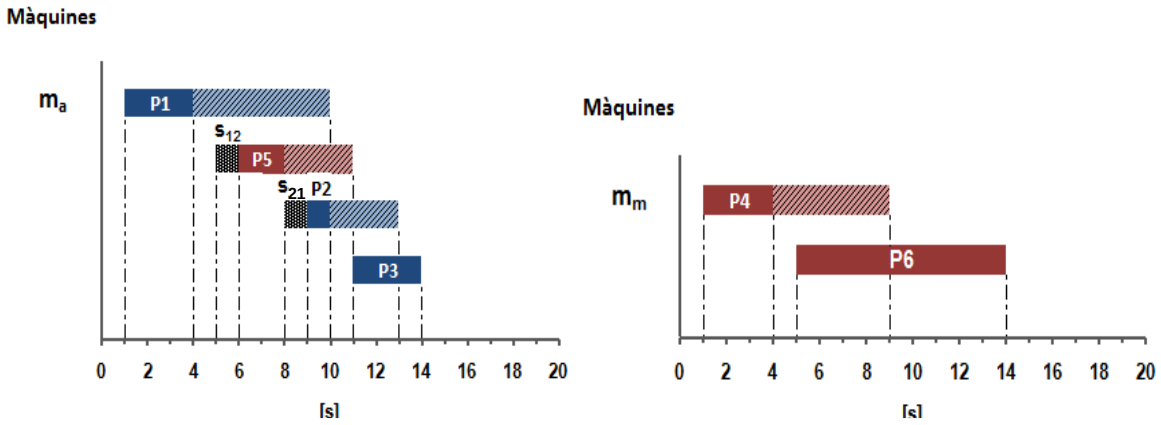


Figura 9.5: Diagrames de Gantt de les màquines que han patit variacions en la fase II respecte la fase I

9.3.1.3. Fase III: Millora dinàmica

Es proposa una tercera fase, on es transformarà el problema en "*dinàmic*". Es parteix d'una situació on les màquines de nivell alt poden tenir peces de tot tipus i les de nivell mig poden haver absorbit alguna peça de nivell baix. No obstant, les peces de nivell alt segur que no han canviat la seva programació respecte la finalització de la fase I, i les peces de nivell mig, és probable que tampoc. Per això, és necessària una fase final del procés de millora que sigui capaç d'un possible avançament de la programació les peces.

Donades les característiques del problema, on les màquines poden produir peces del seu nivell i els inferiors, l'objectiu intrínsec és passar peces de nivells més baixos a màquines de nivells més alts per a fer que totes les màquines estiguin tan carregades com sigui possible, per acostar-se a la càrrega del coll d'ampolla del problema.

El mètode proposat consisteix, igual que el mètode proposat en la fase II, en anar "pujant" peces a màquines de nivells més elevats. En aquesta fase, però, canvien les condicions per a poder "pujar" una peça. Una peça es podrà pujar si arriba abans que el començament d'un període ociós a la màquina i el fet de pujar-la millora el C_{max}, obtenint C'_{max}. Una peça és candidata a "pujar" si compleix les condicions presentades a continuació:

C3. $r_{j_0} \leq f_i^k + w_i^k$

C4. $C'_{max} \leq C_{max}$

Cal fixar-se que en les condicions no hi apareix el fet que la durada de la peça sigui més petita que la durada del període ociós en la màquina, ja que la fase III permet el desplaçament de peces sempre i quan es millori el C_{max}.

La metodologia per canviar peces de nivell és la següent:

0) Es llisten els instants de finalització de cada màquina i s'escull la màquina que condueixen al C_{max} global. Per altra banda, s'eliminen els temps de preparació que existeixin entre les peces (sense afectar a la posició de les peces ni al C_{max}). D'aquesta forma seria possible col·locar peces del mateix nivell sense duplicar temps de preparació.
1) Selecció del cas depenent del tipus de màquina on estigui el C_{max}. Si C_{max} pertany a una màquina de nivell alt, **Fi**. Si C_{max} pertany a una màquina de nivell mig, anar al punt 2. Si C_{max} pertany a una màquina de nivell baix, anar al punt 3.
2) Es llisten tots els temps ociosos de les màquines de nivell alt i de les màquines de nivell mig que no continguin C_{max} (f_i^k,w_i^k,s_i^k,r_i^k). Els espais ociosos de cada màquina s'ordenaran per ordre creixent de f_i^k.
   2.1) S'aplica una cerca en totes les peces de la màquina que conté C_{max} i es compara cada peça amb els vectors dels temps ociosos generats al punt 2, començant primer pels vectors de les màquines de nivell més alt.
   2.2) Si una peça compleix les condicions C3 i C4, s'assigna **(sense introduir els setup times)**, i s'actualitza la llista de temps ociosos i la llista d'instants de finalització de les màquines; si no, es passa a la següent peça. En cas d'haver acabat la llista de peces, es va al punt 4.
3) Es llisten tots els temps ociosos de les màquines de nivell alt, mig i de les màquines de nivell baix que no continguin C_{max} (f_i^k,w_i^k,s_i^k,r_i^k). Els espais ociosos de cada màquina s'ordenaran per ordre creixent de f_i^k.
   3.1) S'aplica una cerca en totes les peces de la màquina que conté C_{max} i es compara cada peça amb els vectors dels temps ociosos generats al punt 3, començant primer pels vectors de les màquines de nivell més alt.
   3.2) Si una peça compleix les condicions C3 i C4, s'assigna **(sense introduir els setup times)**, i s'actualitza la llista de temps ociosos i la llista d'instants de finalització de les màquines; si no es passa a la següent peça. En cas d'haver acabat la llista de peces, es va al punt 4.
4) Si s'ha fet algun canvi, anar al punt 1; si no, anar al punt 5.

5) S'assignen els temps de preparació que sigui necessari entre la programació de les peces de totes les màquines, **Fi**.

Els passos de la fase III són bastant similars als de la fase II. En els dos casos l'algoritme compara llistes de peces amb llistes de d'espais ociosos. La diferencia recau en l'actualització de C_{max} per part del tercer pas.

La fase II seria prescindible, ja que els canvis de màquina fets en la fase II també es podrien haver fet en la fase III. El fet d'eliminar la segona fase, però, implicaria que la fase III hauria de fer més iteracions, i conseqüentment, actualitzacions i càlculs de C_{max} que no són necessaris en el cas d'aplicar la fase II.

Exemple 4 (cont)

Es proposa continuar resolent l'exemple previ amb l'aplicació de la fase III.

7) Es llisten els instants de finalització de cada màquina i s'obté la següent llista {14;14;15} .
8) Max {14;14;15} = 15, que pertany a una màquina de nivell baix. Per tant, es va al punt 3.
3) Es llisten els temps ociosos de les màquines de nivell alt: $(f_1^1,w_1^1,s_1^1,r_1^1)$ = (0,1,1,1) ; $(f_2^1,w_2^1,s_2^1,r_2^1)$ = (4,2,1,2); $(f_3^1,w_3^1,s_3^1,r_3^1)$ = (8,1,2,1); $(f_4^1,w_4^1,s_4^1,r_4^1)$ = (10,1,1,1); $(f_5^1,w_5^1,s_5^1,r_5^1)$ = (14,∞,1,-). Seguidament el de les màquines de nivell mig $(f_1^2,w_1^2,s_1^2,r_1^2)$ = (0,1,2,2) ; $(f_2^2,w_2^2,s_2^2,r_2^2)$ = (4,1,2,2); $(f_3^1,w_3^1,s_3^1,r_3^1)$ = (14,∞,2,-). Com que no hi ha més màquines de nivell baix que la que conté la C_{max} , ja es tenen tots els temps ociosos computats.
   3.1) S'aplica una cerca en totes les peces de la màquina que conté C_{max} (m_b) i es compara cada peça amb els vectors dels temps ociosos generats en el punt 3.
   3.2) La peça 7 no compleix la condició C3 quan s'estudia el primer temps ociós $(f_1^1,w_1^1,s_1^1,r_1^1)$ = (0,1,1,1); en canvi al estudiar el segon $(f_2^1,w_2^1,s_2^1,r_2^1)$ = (4,2,1,2) es compleix la condició C3 i també la C4. Per tant, la peça 7 es programa a la màquina de nivell alt i s'actualitzen les llistes de temps ociosos de les màquines i la llista d'instant de finalització de cada màquina.
   3.2) Pel que fa a la peça 8, s'arriba a la conclusió que no es pot programar en cap altra màquina per millorar C_{max}.
   3.2) Tampoc la peça 9 es pot programar en cap altra màquina per millorar C_{max}.

4) Com que hi ha hagut una actualització, es passa al punt 1.
1) Amb la nova llista de màquines, s'obté max {14;14;14} =14 , que pertany a una màquina de nivell alt; per tant, fi.

El resultat de l'aplicació de la fase III queda representat en la figura 9.6.

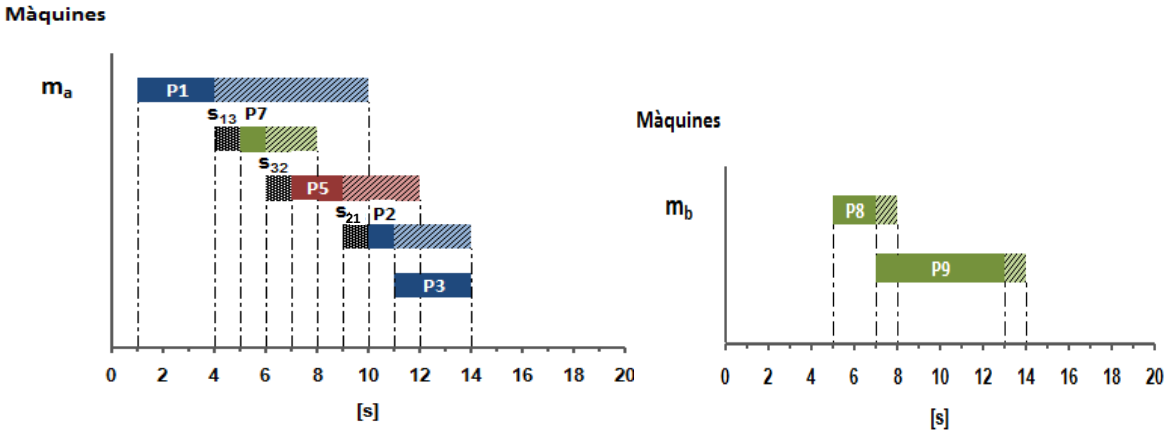


Figura 9.6: Diagrames de Gantt de les màquines de l'exemple 4 després d'aplicar la fase III

9.3.2. Mètode de resolució (2ª part, Algoritme Genètic)

Mentre en la primera part de la resolució del problema s'obté una solució factible al aplicar l'algoritme de Gharbi i Haouari i les dues fases de millora, l'estàtica i la dinàmica, en aquesta segona part es presenta un Algoritme Genètic per resoldre el problema que parteix de la solució aconseguida a la primera part.

Eva Vallada i Rubén Ruiz (2011) **[24]** presenten en el seu text *A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times* un algoritme genètic (GA) de cerca local ràpida i un de cerca local millorada per mitja de l'operació d'encreuament. Aquest algoritme contempla un conjunt de *n* peces que s'han de processar en una de les *m* màquines disposades en paral·lel i es requereix d'un temps de preparació entre peces.

En la Figura 9.7 de la següent pàgina, es mostra l'esquema de l'Algoritme Genètic aplicat en el problema en qüestió per tal de tenir una idea general del procediment aplicat. En les posteriors pàgines, es descriu com s'aplica l'Algoritme Genètic. En aquest cas és necessari prendre una sèrie de decisions que afecten a cadascun dels passos de l'algoritme, des de la codificació a la finalització del problema.

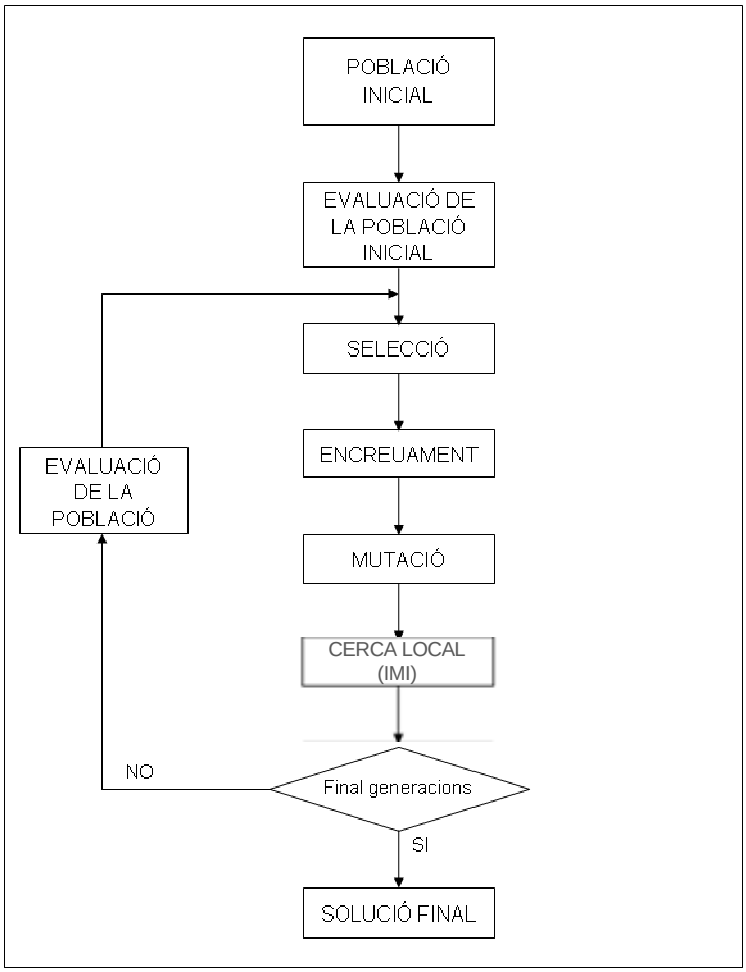


Figura 9.7: Esquema general d'un Algoritme Genètic

9.3.2.1. Codificació

La solució es representa mitjançant una matriu on cada fila correspon a la seqüenciació de peces en una màquina i cada columna correspon a l'ordre de fabricació d'una peça en la màquina de la fila.

Per exemple, en el cas que tinguem 12 peces a programar en 2 màquines, una possible solució podria quedar representada de la següent forma:

Màquina 1	1	3	4	5	10	11	12
Màquina 2	2	7	6	8	9		

Figura 9.8: Representació d'una solució formada per 2 files (una per cada màquina)

Per tant, cada vector de la solució podria tenir una longitud màxima de *n* peces, cas en què totes les peces es programarien en una sola màquina.

9.3.2.2. Població inicial

Pel que fa a la població inicial hi ha dos factors que són determinants, i que s'han de decidir a l'inici: la dimensió i la seva composició.

Pel que fa a la dimensió, és important que garanteixi que la solució trobada al final de l'algoritme sigui factible en un temps raonable. Ha de variar segons la complexitat del problema, i per tant ha de mantenir una relació amb el nombre de peces *n* i de màquines *m*. En aquest cas, i basant-se en les conclusions obtingudes per Eva Vallada i Rubén Ruiz (2011) **[24]** després de realitzar dos calibratges, es creu convenient crear una població inicial (P_{sizes}) formada per 80 solucions.

Amb la finalitat de realitzar una exploració acurada de tot l'espai de cerca, la composició de la població inicial ha de ser heterogènia. Una composició totalment aleatòria suposaria començar amb solucions molt allunyades de la solució del problema, que no condueixen a una solució factible al final de l'algoritme.

La població inicial es generarà de tres formes diferents:

- Un primer individu serà el trobat mitjançant la primera part de l'algoritme, format per tres fases, una primera basada en l'algoritme de Gharbi i dos posteriors fases de millora.
- Una altra part de la població serà generada mitjançant una heurística molt similar a la *Multiple Insertion* (MI) utilitzada per a la primera resolució del problema. En aquest cas es basa en un procediment més simple. Consisteix en ordenar les peces en un vector seguint un criteri (en cas d'empat, per ordre lexicogràfic). A continuació, per ordre de com s'han ordenat, cada peça es programa en l'última posició de cadascuna de les màquines en què sigui factible i finalment es programa en aquella màquina en què resulta un menor instant d'entrega de l'última peça (C_{max}). En cas d'empat de C_{max}, la peça és programa en aquella posició en la qual el temps de preparació sigui menor.

  En aquest cas, els criteris seguits per ordenar les peces són els següents:
  - Menor temps de procés.
  - Menor temps de pre-procés.

- Menor temps de post-procés.
- Major temps de procés.
- Major temps de pre-procés.
- Major temps de post-procés.

- La resta dels individus és generen de forma aleatòria però basant-se en el procediment heurístic explicat en el punt anterior. Això implica que el vector de peces es genera de forma aleatòria, però a continuació l'assignació de les peces a les màquines segueix un procediment igual a l'anterior.

Exemple 5

Per tal d'entendre millor com es forma la població inicial, a continuació es presenta un exemple que simula la segona i la tercera forma de crear un individu. En aquest cas es seguirà la regla d'ordenar les peces per menor temps de pre-procés amb $m_a = 1$, $m_m = 1$, $m_b = 1$. Les peces queden definides en la taula 9.5 i els setup times en la Taula 9.6.

Peça (*j*)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Nivell (u_j)	1	1	1	2	2	2	3	3	3
Temps de pre-procés (r_j) [s]	2	8	11	1	5	5	0	6	7
Temps de procés (p_j) [s]	3	2	3	3	2	9	6	2	6
Temps de post-procés (q_j) [s]	7	3	0	5	3	0	1	2	1

Taula 9.5: Dades de les peces de l'exemple 5

k* \ *k'	**1**	**2**	**3**
1	-	1	3
2	1	-	2
3	3	2	-

Taula 9.6: Dades dels temps de preparació (en segons) de l'exemple 5

1) S'ordenen les peces en ordre no decreixent del temps de pre-procés:

7	4	1	5	6	8	9	2	3

2) La peça 7 es programa a la màquina de nivell baix, al no necessitar temps de preparació.

3) La peça 4 es programa a la màquina de nivell mig, al no necessitar temps de preparació.
4) La peça 1 es programa a la màquina de nivell alt.
5) La peça 5 es prova de programar en la última posició de cadascuna de les dues màquines factibles i es fixa en la que el C_{max} és menor, la màquina de nivell mig.
6) La peça 6 es prova de programar en la última posició de cadascuna de les dues màquines factibles i es fixa en aquella que produeix un menor C_{max}. En aquest cas el millor C_{max} s'obté quan es programa a la màquina de nivell alt.

Se segueix aquest procediment fins a programar totes les peces, resultat que es mostra a la figura 9.9.

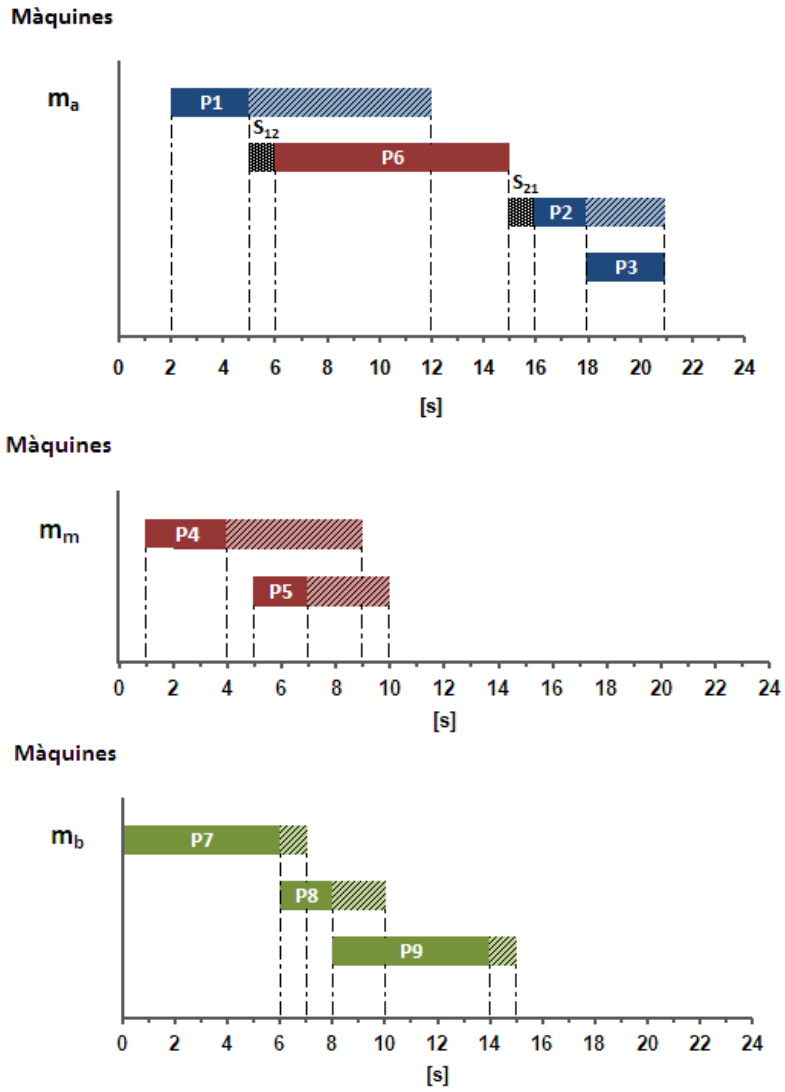


Figura 9.9: Diagrames de Gantt de programació de les màquines de l'exemple 5

9.3.2.3. Mesura d'avaluació o *fitness*

L'objectiu principal del problema és minimitzar l'instant de lliurament de l'última peça (C_{max}). Per tant, la funció objectiu o *fitness* que avalua cadascun dels individus ha de reflectir aquesta dada. S'ha optat per no incloure cap altra variable.

$$Fitness = C_{max} \quad \text{(Eq. 9.1)}$$

9.3.2.4. Selecció

La selecció és el procés mitjançant el qual es determinen les parelles d'individus que s'encreuaran per tal d'obtenir descendència.

S'ha decidit aplicar un procés de selecció en el qual es formen tantes parelles com progenitors hi ha en la població inicial. Els criteris per formar les parelles són el següent:

- El 10% de la descendència és resultat de l'aparellament entre l'individu de millor *fitness* i un altre individu seleccionat aleatòriament entre el 10% dels individus amb millor *fitness* de la població. Aquesta forma d'aparellament té un caràcter elitista, al centrar-se únicament en els individus de millor *fitness*.
- El 90% restant de la descendència és resultat de l'aparellament mitjançant la regla de la ruleta. Consisteix en què un individu té més probabilitat de ser seleccionat com major sigui el seu *fitness*. Concretament, la probabilitat de què un individu sigui seleccionat es determina de la següent forma:

$$P_{sel_t} = \frac{1/fitness_t}{\Sigma\left(1/fitness_t\right)} \quad \text{(Eq. 9.2)}$$

  Amb aquest mètode, primer és seleccionarà un individu, i posteriorment un altre, formant una parella. Per tant, qualsevol individu de la població pot ser seleccionat per tenir descendència.

9.3.2.5. Operació d'encreuament i mutació

A continuació és presenten les operacions d'encreuament i mutació que serveixen per millorar els individus que configuren la població inicial.

Cal destacar que els descendents creats pels operadors d'encreuament i mutació són sotmesos posteriorment a una cerca local per a la millora de la solució.

9.3.2.6. Encreuament

Un cop s'han seleccionat els individus que formen cada parella, s'aplica l'operació d'encreuament d'acord a una probabilitat P_c que serveix per decidir si una parella de progenitors es creua i té descendència o no. Es genera un nombre aleatori entre 0 i 1, si aquest és menor o igual a P_c hi ha encreuament; en cas contrari no n'hi ha. En aquest cas, i basant-se en les conclusions obtingudes per Eva Vallada i Rubén Ruiz (2011) **[24]**, després de realitzar dos calibratges, es creu convenient treballar amb un valor de P_c de 0,5.

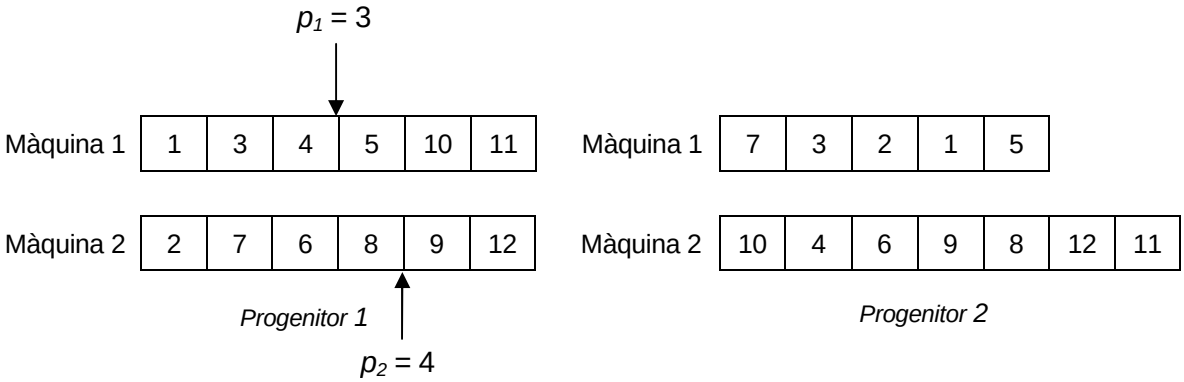
Aquesta operació permet l'intercanvi d'informació entre parelles d'individus, recombinant els cromosomes, per donar lloc a nous individus que intenten aprofitar les millors qualitats de cadascun dels individus de la parella que es creua.

L' operació d'encreuament utilitzada consisteix en seleccionar un punt *p* aleatori per a cada màquina. Les primeres *p* peces del progenitor 1 són copiades al primer descendent, en la mateixa màquina en la que estan en el progenitor 1; les peces restants són copiades al segon descendent. A continuació, les peces del progenitor 2 que encara no estan contingudes en cap de les màquines del seu descendent, s'introdueixen en el mateix ordre i en la mateixa màquina en la que estan en el progenitor 2.

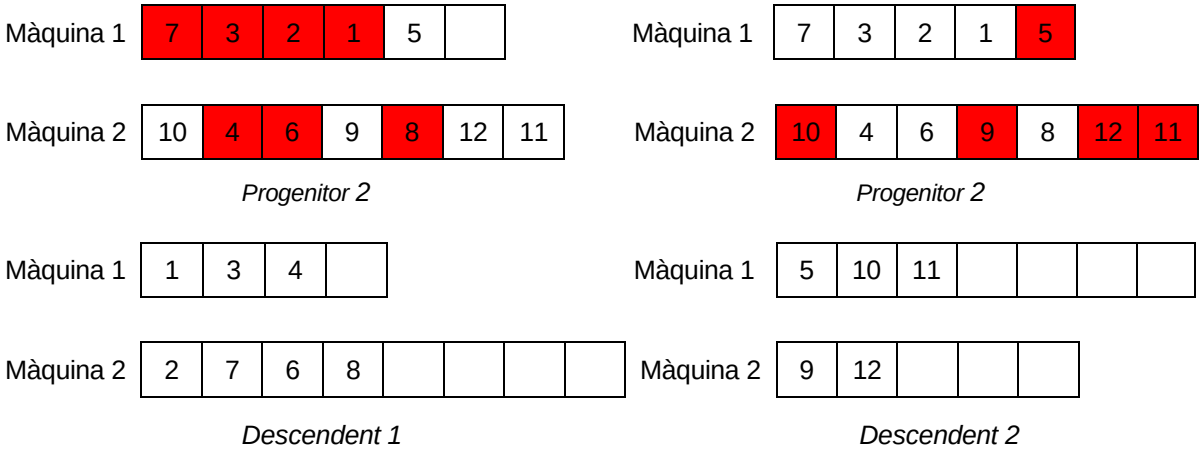
Una altra opció és realitzar aquest mateix pas, però incorporant una **cerca local limitada**. Les peces procedents del progenitor 2 són introduïdes a cadascuna de les posicions possibles de la mateixa màquina en la que estan programades en el progenitor 2 i, finalment, es programa en aquella que resulta un temps de finalització mínim de les peces d'aquella màquina tenint en compte els temps de preparació.

Exemple 6

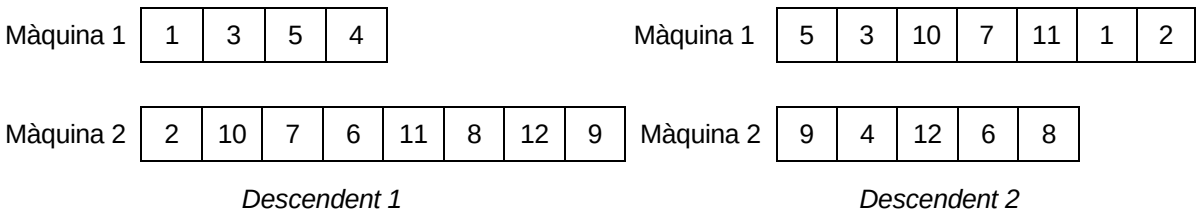
Es presenta un exemple per tal d'entendre millor l'operació d'encreuament. En aquest cas hi ha 12 peces programades en 2 màquines:



(a) Progenitors seleccionats i punts de creuament



(b) Copia de les peces del progenitor 1 i inserció de les peces del progenitor 2



(c) Resultats de l'encreuament

Figura 9.10: Exemple de l'encreuament

9.3.2.7. Mutació

Un cop s'han creuat els progenitors i s'han obtingut els dos descendents, es sotmeten als descendents a un procés de mutació amb l'objectiu d'originar variacions elementals a la població. Aquesta operació es duu a terme d'acord a una probabilitat P_m. En aquest cas, i basant-se en les conclusions obtingudes per Eva Vallada i Rubén Ruiz (2011) **[24]**, després de realitzar dos calibratges es creu convenient treballar amb un valor de P_m de 0,5.

Per a aquest problema, el tipus de mutació que aporta millors resultats consisteix en seleccionar una màquina i una peça d'aquesta màquina aleatòriament i reintroduir-la a l'atzar en una posició diferent de la mateixa màquina. D'aquesta manera, no hi ha perill de crear solucions no factibles al moure les peces únicament entre diferents posicions d'una mateixa màquina.

Exemple 7

Per entendre millor l'operació de mutació, donats els descendents 1 en la solució obtinguda a l'exemple 6, es presenta l'exemple que hi ha a continuació:

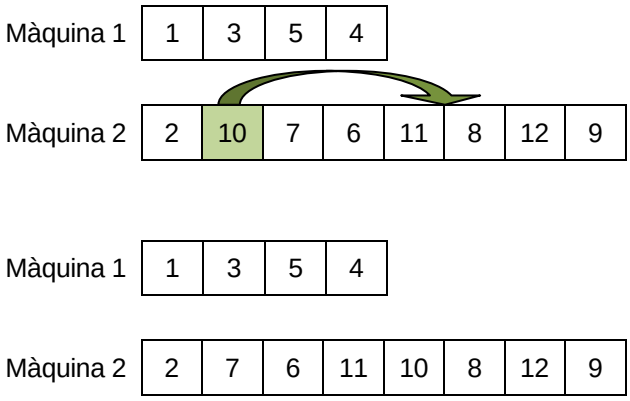


Figura 9.11: Exemple de l'operació de mutació

9.3.2.8. Procediment de cerca local

Aquest procediment de millora consisteix en la cerca de solucions veïnes i es coneix com *inter-machine insertion neighborhood* (*IMI*). Consisteix en programar cadascuna de les peces en cadascuna de les posicions de les màquines en les que poden ser programades per complir la factibilitat de la solució.

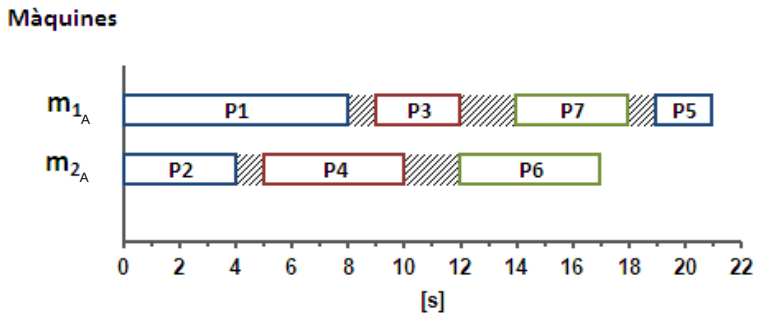
Cal destacar que, en aquest cas, una solució obtinguda amb aquest procediment passarà a formar part de la població inicial només si és millor que la pitjor solució que forma part de la població i al mateix temps és única, es a dir que no existeixi cap altra solució de C_{max} igual a aquesta que ja formi part de la població inicial.

Amb la finalitat de reduir l'esforç computacional necessari, és possible introduir un simple però molt eficient procediment: quan una peça és programada, no és necessari avaluar la seqüència completa per obtenir el temps en què la màquina acabarà totes les peces.

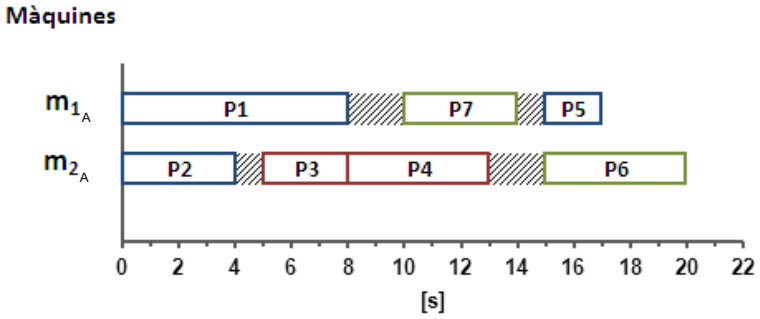
Exemple 8

Per explicar aquest procediment, s'utilitza un exemple en el que hi ha 7 peces i dues màquines. Una possible solució es mostra a la figura 9.11. Els espais en gris representen els temps de preparació entre peces.

Es pot observar en la figura 9.12(a) com el temps de finalització de les peces és l'instant 21. L'objectiu és reduir aquest temps i, per tant, es pot moure la peça 3 i programar-la a la segona posició de la màquina 2. Per obtenir el temps de finalització de la programació, s'ha d'eliminar el temps de preparació entre la peça 1 i 3, el temps de procés de la peça 3 i el temps de preparació entre la peça 3 i 7. Posteriorment, en la màquina 2 s'ha d'eliminar el temps de preparació entre la peça 2 i 4 i inserir el temps de preparació entre la peça 2 i la 3, el temps de procés de la peça 3 i el temps de preparació entre la peça 3 i la 4. Per tant, s'obté el nou temps de finalització de les peces efectuant 3 sostraccions i 3 addicions. Realitzant aquest procediment, s'obté que el nou temps de finalització de les peces passa a ser de 20, millor que l'inicial.



(a) Programació inicial de les 7 peces en 2 màquines



(b) Resultat de la programació després d'assignar una peça que estava a la màquina 1

Figura 9.12: Exemple del procediment IMI amb 7 peces i 2 màquines

En general, per cada moviment, el nombre de sostraccions i addicions seran quatre i quatre respectivament, sempre i quan la peça no es programi en la primer o la última posició. En aquest cas, el nombre de sostraccions i addicions seria de tres cadascun. Cal destacar una situació particular: en el cas que la primera peça programada a una màquina no correspongui amb el nivell d'aquesta caldrà un temps de preparació al principi de la seqüència. En total, la cerca local proposada pot contenir el següent nombre de iteracions:

$$\sum_{i \in M} \sum_{l \in M, l \neq i} n_i(n_l + 1) \qquad \text{(Eq. 9.3)}$$

on n_i i n_l són el nombre de peces assignades a la màquina *i* i *l*, respectivament. Això vol dir que cada peça de la màquina *i* pot ser inserida en $n_l + 1$ posicions de la màquina *l*.

Un altre tema a tractar són els moviments que durant el procediment són acceptats. Aquest algoritme, basat en el procediment *IMI*, només s'ha plantejat per ser dut a terme entre dues màquines. Per tant, les peces assignades a la màquina 1 s'han de programar en totes les posicions possibles de la màquina 2 i aplicar els criteris d'acceptació. Posteriorment, realitzar el mateix procediment programant les peces en totes les posicions possibles de la màquina 3, i així progressivament amb totes les màquines. En aquests casos, s'ha de tenir en compte que hi ha peces que no poden ser programades en altres màquines, perquè es crearien solucions no factibles.

Per a cada parella de màquines, els criteris per decidir si acceptem o no els moviments són:

- El temps per completar les peces en les dues màquines disminueix. En aquest cas el moviment és **acceptat**.
- El temps per completar les peces en una màquina disminueix i en l'altra màquina augmenta. En aquest cas s'ha de comprovar si el temps reduït en una màquina compensa l'augment de l'altra.

- C_i, C'_i: temps per completar les peces de la màquina *i* abans i després del moviment, $i \in M$.
- C_l, C'_l: temps per completar les peces de la màquina *l* abans i després del moviment.
- C_{max}, C'_{max}: temps total per acabar totes les peces abans i després del moviment.

En aquest cas el moviment serà **acceptat** si es compleixen totes les següents condicions:

$$C_i' < C_i \quad \text{(Eq. 9.4)}$$

$$C_l' > C_l \quad \text{(Eq. 9.5)}$$

$$C_i - C_i' > C_l' - C_l \quad \text{(Eq. 9.6)}$$

$$C'_{max} \leq C_{max} \quad \text{(Eq. 9.7)}$$

Amb l'expressió (Eq. 9.4), es comprova si el temps de finalització de la màquina *i* s'ha reduït. L'expressió (Eq. 9.5) mostra si el temps de finalització de la màquina *l* s'ha incrementat. L'expressió (Eq. 9.6) compara la reducció en la màquina *i* respecte l'increment en la màquina *l*; la reducció en la primera màquina ha de ser major que l'increment de la segona. Per últim, l'expressió (Eq. 9.7) comprova que el temps de finalització de **totes les màquines** després del moviment hagi disminuït.

- El temps per completar les peces en les dues màquines augmenta. En aquest cas el moviment **NO** és **acceptat**.

D'altra banda, el procediment IMI de cerca local s'aplica fins a un òptim local. Això vol dir que si després d'una iteració de la cerca local un o més moviments són acceptats, o sigui impliquen millores, s'ha de dur a terme el procediment IMI per a cadascun d'aquests possibles moviments.

En existir tantes possibilitats de moviments, s'ha de limitar l'aplicació d'aquest procediment d'acord a una probabilitat P_{ls} per al millor individu del conjunt de solucions inicials i per als seus descendents. En aquest cas, i basant-se en les conclusions obtingudes per Eva Vallada i Rubén Ruiz (2011) **[24]** després de realitzar dos calibratges, es creu convenient treballar amb un valor de P_{ls} d'1. Per tant, es pot assegurar que és un pas molt important per aconseguir arribar a una solució amb un bon resultat.

10. Experiència Computacional

En el present capítol, s'ha dut a terme l'experiència computacional basada en els procediments descrits ens els dos capítols anteriors.

10.1. Generació dels exemplars

Per comprovar la viabilitat i l'aptitud de l'algoritme proposat s'han generat un conjunt d'exemples de manera similar a la utilitzada per Gharbi **[23]**. Pel que fa als temps de preparació de les màquines, s'ha seguit una manera molt similar a la utilitzada en **[24]**.

El nombre de peces *n* a tractar són 20, 50, 100 i 200. Per a cada quantitat de peces s'han generat 100 jocs de peces. El nombre de màquines *m* ha estat de 4, 5, 6, 8 i 10 i en cap cas cap conjunt de màquines de cap nivell ha estat superior a 4. Les diferents distribucions de les màquines queda representat en la taula 10.1. Els temps de procés s'han generat mitjançant una distribució discreta uniforme entre [1, 10]. S'ha considerat que les peces de nivell alt formaran entre el 10 i el 30% del total de les peces, les de nivell mig, entre el 10 i el 50% i les de nivell baix entre el 40 i el 60%.

	4a	4b	4c	5a	5b	5c	6a	6b	6c	6d	6e	6f	6g
m_a	2	1	1	2	2	1	3	3	2	1	2	1	2
m_m	1	2	1	2	1	2	2	1	3	3	1	2	2
m_b	1	1	2	1	2	2	1	2	1	2	3	3	2

	8a	8b	8c	8d	8e	8f	10a	10b	10c	10d	10e	10f
m_a	4	2	2	3	3	2	4	3	3	4	4	2
m_m	2	4	2	3	2	3	3	4	3	4	2	4
m_b	2	2	4	2	3	3	3	3	4	2	4	4

Taula 10.1: Taula amb la distribució de les màquines

Els temps de pre-procés i post-procés s'han extret d'una distribució discreta uniforme similar a la que planteja Gharbi i té la forma [1, *K* (*n*/*m*)], on *K* és un enter positiu igual a 3 i 5. El paràmetre *K* afecta per tant, a la distribució dels temps de pre-procés i post-procés.

Pels temps de preparació s'han generat 3 subconjunts seguint una distribució discreta uniforme entre [1, 3], [1, 9] i [1, 19]. Aquests subconjunts s'han escollit per poder obtenir uns resultats representatius per temps de preparació menors als temps de procés, de magnitud similar i superiors respectivament.

Pels casos amb 20 peces, s'han descartat els estudis amb 8 i 10 màquines i pels casos amb 50 peces s'han descartat els estudis amb 10 màquines.

10.2. Aplicacions i software programari utilitzat

Amb la finalitat d'agilitzar i automatitzar els càlculs, s'ha desenvolupat una aplicació informàtica en llenguatge C# basada en el compilador *Microsoft Visual Studio* versió *2010*. L'aplicació del primer i del segon estudi del problema es troben en el fitxer adjunt *Primer estudi.exe* i *Segon estudi.exe* respectivament(veure annex E). Les aplicacions han estat executades en un Intel(R) Core(TM) i5 de 3,33GHz i 4GB de RAM.

Per aplicar el mètode basat en l'algoritme genètic, s'ha limitat el temps de processament de l'ordenador. Aquest temps depèn del nombre *n* i *m* de peces i màquines respectivament, però falta realitzar un estudi per determinar el temps que es vol implementar. El temps de processament estarà limitat pel valor que s'obté de la següent operació:

$$n \cdot \left({}^{m}/_{2}\right) \cdot (temps)\ ms \qquad \text{(Eq. 10.1)}$$

La variable *temps* es determina en l'estudi realitzat al punt 10.3, on es tracta de buscar el valor més adequat d'aquesta variable.

10.2.1. Funcionament de l'aplicació

Els passos a seguir per a l'obtenció dels resultats són els següents:

1. Descriure els paràmetres en els rangs descrits anteriorment. Aquests paràmetres són:
    - Número de peces
    - Número de màquines total i per cada nivell
    - Temps de preparació

    Aquests paràmetres es defineixen en l'arxiu *GeneraExemplars.cs* que hi ha dins la carpeta *Programa \ Arxius*. Dins d'aquest arxiu, s'han de definir el nombre de màquines de cadascun dels nivells i el nombre de peces per generar les dades d'un

conjunt de 100 instàncies (Figura 10.1) i la distribució del temps de preparació amb el que es vol treballar i inhabilitar els altres dos (Figura 10.2).

```
niv = 3;
Console.Write(niv + "  ");
Console.WriteLine();

mach1 = 1;
mach2 = 1;
mach3 = 1;
mach = mach1 + mach2 + mach3;
Console.Write(mach1 + "  "); Console.Write(mach2 + "  "); Console.Write(mach3 + "  ");
Console.WriteLine();

exemplars1 = 50;
exemplars = exemplars1 + exemplars1;
Console.Write(exemplars + "  ");
Console.WriteLine();

semilla = (int)DateTime.Now.Ticks;
pec1 = new Random(semilla);
semilla1 = (int)DateTime.Now.Kind;
p1 = new Random(semilla1);
semilla2 = (int)DateTime.Now.Millisecond;
q1 = new Random(semilla2);
semilla3 = (int)DateTime.Now.Millisecond;
pec2 = new Random(semilla3);
semilla4 = (int)DateTime.Now.Ticks;
r1 = new Random(semilla4);

peces = 20;
pecess = (double)peces;
a1 = pecess * (0.2);
a2 = pecess * (0.3);
a3 = pecess * (0.5);
a4 = 3 * (pecess / mach);
a5 = 5 * (pecess / mach);
```

Figura 10.1: Definició del nombre de maquines de cadascun dels nivells i del nombre total de peces

```
Random r = new Random();

//Seleccionar un dels següents tres casos

Setup[1, 2] = r.Next(1, 3);
Setup[1, 3] = r.Next(1, 3);
Setup[3, 2] = r.Next(1, 3);

/*
Setup[1, 2] = r.Next(1, 9);
Setup[1, 3] = r.Next(1, 9);
Setup[3, 2] = r.Next(1, 9);
*/
/*
Setup[1, 2] = r.Next(1, 19);
Setup[1, 3] = r.Next(1, 19);
Setup[3, 2] = r.Next(1, 19);
*/

Setup[2, 1] = Setup[1, 2];
Setup[3, 1] = Setup[1, 3];
Setup[3, 2] = Setup[3, 2];
```

Figura 10.2: Distribució del temps de preparació seleccionat

2. Generar el conjunt d'instàncies per cadascuna de les combinacions explicades en el punt 10.1.
3. Executar el programa per generar les solucions.

10.3. Influència i determinació del temps de processament

Com ja s'ha comentat anteriorment, la determinació del temps de processament busca un compromís entre el cost computacional i l'eficiència de la solució. A major temps de processament, major nombre de regeneracions, i per tant el cost computacional augmenta i s'espera un millor valor C_{max} de la solució.

Per trobar el millor compromís entre aquests dos factors s'ha considerat que el temps en el qual el programa ha d'arribar a una solució ha d'estar relacionat amb el nombre *n* i *m* (peces i màquines respectivament) tal i com es mostra en l'equació 10.1.

Per la realització d'aquest estudi s'analitzarà el comportament de les solucions de totes les combinacions de peces amb la distribució de màquines dels casos on $m_a \geq m_m \geq m_b$. En el nostre cas són *4a, 5a, 6a, 8a, 8d, 10a i 10 d* exposats a la Taula 10.1. Es consideren aquests casos al ser més propensos a produir moviments entre màquines, i per tant a requerir major temps de processament.

En el cas dels temps de preparació, es realitza l'estudi utilitzant la primera dsitribució (valors entre 1 i 3 segons), cas en què són més petits i que per tant afavoreix el moviment de les peces entre diferents nivells.

Pel que fa als valors de la variable *temps* (Eq. 10.1), s'ha partit d'uns valors de 30, 60 i 120 ms basant-nos en **[24]**. Un cop realitzada l'anàlisi dels resultats obtinguts d'aquest primer estudi, s'ha cregut necessari fer una segons anàlisi augmentant els valors de la variable *temps* fins a 240 i 300 ms.

Per tal de simplificar aquest estudi, treballarem amb la mitjana del *fitness de* 10 instàncies per cada combinació de peces i màquines quedant reflectits els resultats a la Taula 10.2. En negreta s'indica el millor resultat obtingut per a cada combinació de màquines.

En la Taula 10.3 apareix un valor que indica, per cada instant de temps, quantes de les 10 instàncies de cada combinació es troben entre les millors d'aquell cas. Per últim, a la Taula

10.4, s'indica el nombre de solucions que han millorat en augmentar el valor de la variable *temps*. Els valors complerts d'aquest estudi es troben a l'annex B.

n	20					50				
temps	30	60	120	240	300	30	60	120	240	300
4a	44,6	**44,4**	**44,4**	**44,4**	**44,4**	105,1	104,4	**104,1**	**104,1**	**104,1**
5a	37,2	**37,1**	**37,1**	**37,1**	**37,1**	82,4	82,2	82	**81,9**	**81,9**
6a	**31**	**31**	**31**	**31**	**31**	70,9	**70,8**	**70,8**	**70,8**	**70,8**
8a	-	-	-	-	-	**55,5**	**55,5**	**55,5**	**55,5**	**55,5**
8d	-	-	-	-	-	55	**54,9**	**54,9**	**54,9**	**54,9**
10a	-	-	-	-	-	-	-	-	-	-
10d	-	-	-	-	-	-	-	-	-	-

Taula 10.2: Valor de C_{max} per a diferents temps màxims de processament (part 1)

n	100					200				
temps	30	60	120	240	300	30	60	120	240	300
4a	212,2	210,3	208,5	**207**	**207**	426	419,5	413,9	410	**409,2**
5a	167,9	166,5	164,6	**163,7**	**163,7**	342,4	335,2	334,1	327,6	**326,2**
6a	138,8	137,4	136,6	135,5	**135,4**	280,8	275,3	271,8	269	**268,5**
8a	108,5	108,1	**107,9**	**107,9**	**107,9**	220,4	213,8	210,8	**208,8**	**208,8**
8d	107,1	106,5	105,6	**105,5**	**105,5**	214,2	209,9	206,7	204,4	**204,3**
10a	**86,8**	**86,8**	**86,8**	**86,8**	**86,8**	174,5	170,7	169,2	**169,1**	**169,1**
10d	87,7	**87,4**	**87,4**	**87,4**	**87,4**	174	171,2	169	168,2	**168,1**

Taula 10.2: Valor de C_{max} per a diferents temps màxims de processament (part 2)

n	20					50					100					200				
temps	30	60	120	240	300	30	60	120	240	300	30	60	120	240	300	30	60	120	240	300
4a	9	**10**	**10**	**10**	**10**	6	8	**10**	**10**	**10**	5	5	5	**10**	**10**	4	5	5	7	**10**
5a	9	**10**	**10**	**10**	**10**	8	8	9	**10**	**10**	5	5	7	**10**	**10**	3	4	5	7	**10**
6a	**10**	**10**	**10**	**10**	**10**	9	**10**	**10**	**10**	**10**	5	5	6	9	**10**	3	3	4	8	**10**
8a	-	-	-	-	-	**10**	**10**	**10**	**10**	**10**	7	9	**10**	**10**	**10**	4	5	6	**10**	**10**
8d	-	-	-	-	-	9	**10**	**10**	**10**	**10**	6	6	9	**10**	**10**	5	5	6	9	**10**
10a	-	-	-	-	-	-	-	-	-	-	**10**	**10**	**10**	**10**	**10**	5	5	9	**10**	**10**
10d	-	-	-	-	-	-	-	-	-	-	8	**10**	**10**	**10**	**10**	5	6	7	9	**10**

Taula 10.3: Nombre d'instàncies millors entre les 10 de cada combinació per a diferents temps màxims de processament

Temps	30 → 60	60 → 120	120 → 240	240 → 300
n = 20	**2**	**0**	**0**	**0**
n = 50	**8**	**3**	**1**	**0**
n = 100	**22**	**17**	**13**	**1**
n = 200	**37**	**34**	**28**	**10**
Total	**69 / 220**	**54 / 220**	**42 / 220**	**11 / 220**

Taula 10.4: Nombre d'instàncies que milloren el seu C_{max} en incrementar el valor temps

A la vista dels resultats mostrats a les taules 10.2 i 10.3 es pot destacar la major influència que té el temps de processament en els casos en què es disposa d'un nombre elevat de peces a programar en poques màquines. Observant més en detall les solucions presentades a l'annex B, i simplificat a la Taula 10.4, s'observa com les 11 solucions millorades al passar d'un valor de *temps* de 240 a 300 ms s'han millorat en cadascun els intervals de temps, indicant que són exemplars "difícils" que necessitarien més temps per resoldre'ls.

Tot i existir una sèrie d'instàncies "difícils" de resoldre, els valors de la Taula 10.4 indiquen que el nombre d'instàncies que milloren les seves solucions en augmentar el valor de la

variable *temps* va disminuint progressivament. A partir dels resultats de la Taula 10.4 es pot concloure que amb 20 peces, el valor de *temps* que seria suficient per arribar al millor resultat és de 60 ms; en el cas de 50 peces, amb 120 ms seria suficient; en el cas de 100 peces, 240 ms; i en el cas de 200 peces, potser el valor hauria de ser major que 240 ms.

En un estudi general, s'observa com en els tres primers increments de la variable temps existeix un 31,4%, 24,5% i 19,1% de millora del valor de la solució, respectivament. En canvi al passar la variable temps d'un valor de 240 a 300 ms només es millora en un 5% el valor de la solució. Per tant, es conclou que un valor adequat de la variable *temps* pot ser el de 240 ms.

Per últim, observant els valors complerts d'aquest estudi a l'annex B, s'observa un fet rellevant: la poca o nul·la variació del temps de finalització de les peces si augmenta el temps de processament en els casos on K = 5 (temps de pre-procés i post-procés més grans).

Observant l'evolució que presenten les solucions obtingudes durant l'execució de l'algoritme, s'observa com en els casos amb K = 5, al tenir temps de pre-procés i post-procés majors, es permet menys moviments de les peces. Per tant, amb molt poques iteracions s'obté un valor de C_{max} que no es millora per més iteracions que es facin.

10.4. Anàlisi dels resultats

En aquesta part es presentarà i estudiarà els resultats obtinguts després d'executar l'aplicació pels exemplars creats. A la memòria es presenten els resultats ja tractats dels estudis realitzats. Per a més informació es poden consultar els annexos C i D.

L'estudi dels resultats busca:

- Determinar la millora en les solucions que suposa l'aplicació dels procediments proposats (capítol 9) comparades amb les de la situació inicial (capítol 8).
- Determinar si un dels dos procediments proposats es comporta millor que l'altre (capítol 9).

Com es veurà, una cop fet un primer estudi del procediment, s'ha cregut necessari fer un segon estudi per analitzar les noves millores incorporades en l'Algoritme Genètic.

10.4.1. Primer estudi del problema

Els estudis realitzats en aquest apartat busquen determinar l'eficiència de l'algoritme plantejat en funció del nombre de peces, de la proporció de màquines, del paràmetre *K* i del temps de preparació.

Les següents taules presenten els resultats mitjans d'aplicar els algoritmes presentats en el capítol 9 als jocs d'instàncies de 20, 50, 100 i 200 peces. Primerament s'ha realitzat un estudi comparatiu de la millora que proporciona en el valor de C_{max} els dos algoritmes proposats en funció del nombre de peces i posteriorment en funció de la proporció de màquines de cada nivell.

La millora de la solució obtinguda per un algorisme *alg* respecte la situació inicial (cada màquina només processa peces del seu nivell) es presenta en forma porcentual. Una millora M_{alg} es calcula tal com mostra l'equació 10.2.

$$M_{alg} = \frac{C_{max_{situaciónicial}} - C_{max}\ aplicacióalgoritme}{C_{max_{situaciónicial}}} \qquad \text{(Eq. 10.2)}$$

A. Estudi en funció del nombre de peces

La Taula 10.5 mostra el resultat de l'estudi en funció del nombre de peces després de l'aplicació dels algoritmes a totes les combinacions de màquines mostrades a la Taula 10.1. A la primera columna es visualitzen el nombre de peces; a la segona i a la tercera, la millora obtinguda en el cas d'aplicar els procediments heurístics 1 i 2, M_{H1} i M_{H2} respectivament (apartat 9.2); i a la última, la millora obtinguda en el cas d'aplicar el procediment metaheurístic M_{AG} (apartat 9.3).

n	M_{H1}	M_{H2}	M_{AG}
20	28,0%	28,3%	**28,7%**
50	30,8%	**31,0%**	30,8%
100	32,7%	**32,8%**	32,4%
200	**33,4%**	**33,4%**	32,6%

Taula 10.5: Resultats de M_{H1}, M_{H2} i M_{AG} de l'estudi segons el nombre de peces de la instància (n)

Com es pot comprovar en la Taula 10.5, les millores més grans de la C_{max} es produeixen en tots els casos amb l'aplicació de l'Heurística 2 excepte en els casos de 20 peces. De totes maneres, es comprova com els resultats obtinguts amb l'Algoritme Genètic s'aproximen molt als obtinguts a través dels mètode heurístic. Per tant, es considera que els dos procediments són vàlids per millorar la programació de peces proposada en el present estudi.

Per altra banda, també podem comprovar com l'aleatorietat introduïda a l'Heurística 2 aporta certes millores que no es poden menysprear. Per tant, entre l'Heurística 1 i la 2 és preferible escollir la segona.

B. Estudi per proporció de màquines

Un cop estudiada la influència de la quantitat de peces i veure que tant l'Heurística 2 com l'Algoritme Genètic aporten millores importants, és interessant estudiar si la millora és més o menys important en funció de la proporció de màquines de cada tipus sense tenir en compte la quantitat de peces. S'han considerat els següents casos:

1. $m_a > m_m$; $m_a > m_b$. Aquest cas està format per les distribucions de màquines 4a, 6a, 6b, 8a i 10a.
2. $m_m > m_a$; $m_m > m_b$. Aquest cas està format per les distribucions de màquines 4b, 6c, 6e, 8b i 10b.
3. $m_b > m_a$; $m_b > m_m$. Aquest cas està format per les distribucions de màquines 4c, 6d, 6f, 8c i 10c.
4. $m_a < m_m$; $m_a < m_b$. Aquest cas està format per les distribucions de màquines 5a, 6e, 6f, 8d i 10d.
5. $m_m < m_a$; $m_m < m_b$. Aquest cas està format per les distribucions de màquines 4b, 6b, 6d, 8e i 10e.
6. $m_b < m_a$; $m_b < m_m$. Aquest cas està format per les distribucions de màquines 4c, 6a, 6c, 8f i 10f.
7. $m_a = m_m = m_b$. Aquest cas és el 6g.

Els resultats queden presentats en la Taula 10.6.

Cas	M_{H1}	M_{H2}	M_{AG}
1	**39,3%**	**39,3%**	38,1%
2	37,1%	**37,2%**	36,7%
3	18,1%	18,2%	**18,8%**
4	20,1%	20,3%	**20,6%**
5	**29,8%**	**29,8%**	29,0%
6	**47,9%**	**47,9%**	46,9%
7	30,7%	**30,8%**	30,0%

Taula 10.6: Resultats de M_{H1}, M_{H2} i M_{AG} de l'estudi segons la proporció de màquines per nivell

En aquest cas, s'obtenen uns resultats molt similars als de l'estudi anterior. Tot i que a primera vista sembla que l'Heurística 2 és el procediment que aconsegueix una major reducció de la C_{max}, es pot observar com l'Algoritme Genètic no es queda massa lluny d'aquestes millores, superant en alguns casos els resultats de l'Heurística 2.

Estudiant més en profunditat els resultats de la taula, s'observa com les millores més grans (39,3% i 47,9%) es produeixen en els casos amb un elevat nombre de màquines de nivell alt (cas 1) i un baix nombre de màquines de nivell baix (cas 6).

C. Estudi en funció del temps de preparació (*setup time*)

El temps de preparació o *setup time* pot afectar a la mobilitat de peces entre màquines de diferents nivells. En aquest estudi es pretén observar l'afectació o no de la dimensió temps de preparació en l'eficiència de cadascun dels algoritmes. Els resultats es mostren a la Taula 10.7.

Temps preparació	M_{H1}	M_{H2}	M_{AG}
1	**33,1%**	**33,1%**	32,7%
2	32,7%	**32,8%**	32,2%
3	32,2%	**32,3%**	31,7%

Taula 10.7: Resultats de M_{H1}, M_{H2} i M_{AG} de l'estudi segons la distribució dels temps de preparació

En aquest cas es pot observar com les millores més grans s'obtenen amb l'aplicació de l'Heurística 2. De totes maneres si s'observen les millores obtingudes amb l'aplicació de l'Algoritme Genètic, els valors són molt similars als anteriors estudis. Aquests no queden molt allunyats dels obtinguts amb l'Heurística 2.

D. Estudi en funció del paràmetre *K*

El paràmetre *K* afecta a la distribució dels temps de pre-procés i post-procés; com més gran és *K*, més grans poden ser els temps de pre-procés i el post-procés. En aquest estudi es pretén observar, l'afectació o no del paràmetre *K* en l'eficiència dels algoritmes presentats. Els resultats es mostren a la Taula 10.8.

n	K = 3			K = 5		
	M_{H1}	M_{H2}	M_{AG}	M_{H1}	M_{H2}	M_{AG}
20	31,9%	32,3%	**33,0%**	24,2%	24,4%	**24,6%**
50	37,9%	**38,2%**	37,8%	24,4%	24,5%	**24,6%**
100	41,5%	**41,6%**	40,7%	24,9%	24,9%	**25,0%**
200	43,1%	**43,2%**	41,5%	24,6%	24,6%	**24,7%**
Mitjana	41,6%	**41,7%**	40,5%	24,6%	24,7%	**24,8%**

Taula 10.8: Resultats de M_{H1}, M_{H2} i M_{AG} de l'estudi segons el paràmetre K utilitzat

Els resultats d'aquest estudi és el que aporta més dubtes. En els casos en què *K* = 3 sembla que el millor mètode és l'Heurístic 2. En canvi, en els casos de *K* = 5 s'observa que el mètode que sembla que millor es comporta és l'Algoritme Genètic.

Observant les dades més en detall, es pot observar com les millores obtingudes per cadascun dels mètodes són molt similars, i es fa difícil decidir quin dels dos procediments té un millor comportament en tots i cadascun dels casos plantejats.

10.4.2. Segon estudi del problema

Un cop analitzat el primer estudi, es creu convenient dur a terme un segon estudi que intenti aprofitar els punts forts de cadascun dels dos mètodes que s'han comportat millor en el primer estudi (l'Heurística 2 i l'Algoritme Genètic). Aquestes millores consisteixen en utilitzar l'Heurística 2 a l'hora de generar la població inicial de l'Algoritme Genètic per tal d'aprofitar el bon funcionament d'aquest procediment heurístic.

En aquest cas l'estudi realitzat treballa amb subconjunts de 20 instàncies de les 100. Així, es pot reduir el temps d'execució necessari. Les millores s'han calculat tal com es mostra a l'equació 10.2 (veure el primer estudi).

Aquest nou procediment el coneixerem com a Algoritme Genètic 2'. També cal destacar que aquest és un estudi independent del primer i per tant, no són comparables els resultats dels dos estudis.

A. Estudi en funció del nombre de peces

La Taula 10.9, mostra les millores obtingudes en l'aplicació del procediment heurístic M_{H2} i del procediment metaheurístic millorat $M_{AG2'}$.

n	$M_{H2'}$	$M_{AG2'}$
20	27,7%	**28,6%**
50	32,4%	**33,2%**
100	34,1%	**34,5%**
200	33,6%	**34,0%**

Taula 10.9: Resultats de $M_{H2'}$ i $M_{AG2'}$ de l'estudi segons el nombre de peces de la instància (n)

En aquest cas es pot comprovar un fet que ja s'esperava, que l'Algoritme Genètic 2' sempre millora els resultats del procediment Heurístic 2' al partir dels resultats obtinguts per l'heurística. Amb les dades de la taula, també es manté el millor comportament de l'Algoritme Genètic 2' respecte el procediment heurístic quan el nombre de peces a programar és reduït.

B. Estudi per proporció de màquines

S'ha realitzat el mateix estudi que en l'apartat 10.4.1 considerant les mateixes combinacions de màquines però en aquest cas comparant l'Heurística 2' i l'Algoritme Genètic 2'. Els resultats es mostren a la Taula 10.10.

Cas	$M_{H2'}$	$M_{AG2'}$
1	39,9%	**40,0%**
2	38,1%	**38,4%**
3	18,9%	**20,0%**
4	20,7%	**21,6%**
5	30,4%	**30,7%**
6	48,6%	**48,7%**
7	31,2%	**31,5%**

Taula 10.10: Resultats de $M_{H2'}$ i $M_{AG2'}$ de l'estudi segons la proporció de màquines per nivell

Si dintre de l'anàlisi per proporció de màquines es mira quin procediment obté una major millora, és conclou que l'Algoritme Genètic 2' és el que te les millores més elevades en tots els casos.

Els resultats mostren la mateixa tendència que en el primer estudi. Les millores més grans es produeixen en les combinacions de màquines que tenen com a màxim les màquines de nivell alt (cas 1) o com a mínim les màquines de nivell baix (cas6). En aquest últim cas les millores arriben a ser properes al 50%.

C. Estudi en funció del temps de preparació (setup time)

En aquest estudi es pretén observar l'afectació o no de la variabilitat del temps de preparació en l'eficiència de cadascun dels algoritmes. Els resultats es mostren a la Taula 10.11.

Temps preparació	$M_{H2'}$	$M_{AG2'}$
1	33,7%	**34,1%**
2	33,4%	**33,8%**
3	33,1%	**33,6%**

Taula 10.11: Resultats de $M_{H2'}$ i $M_{AG2'}$ de l'estudi segons la distribució dels temps de preparació

La Taula 10.11 mostra que la major millora de C_{max} es produeix en l'aplicació de l'Algoritme Genètic 2' per la mateixa raó que s'ha comentat en l'estudi segons el nombre de peces. Amb les dades d'aquesta taula també es pot concloure la nul·la afectació d'aquesta distribució de temps en la millora aconseguida pels dos procediments comparats.

D. Estudi en funció del paràmetre K

En aquest estudi es pretén observar l'afectació o no del paràmetre *K* en l'eficiència dels algoritmes presentats. Els resultats es mostren a la Taula 10.12.

n	*K* = 3		*K* = 5	
	$M_{H2'}$	$M_{AG2'}$	$M_{H2'}$	$M_{AG2'}$
20	32,8%	**34,3%**	23,0%	**23,4%**
50	39,4%	**40,8%**	26,2%	**26,3%**
100	42,3%	**43,1%**	26,6%	**26,8%**
200	38,7%	**39,1%**	24,9%	**25,1%**
Mitjana	39,6%	**40,2%**	25,5%	**25,6%**

Taula 10.12: Resultats de $M_{H2'}$ i $M_{AG2'}$ de l'estudi segons el paràmetre K utilitzat

Ara els resultats són molt més clars: a diferència del primer estudi l'Algoritme Genètic és el que aporta les majors millores en totes les situacions.

També és destacable la influència que té el valor de *K* a l'hora d'aconseguir millorar la programació respecte la Situació inicial. En el cas de *K* = 3 (menor amplitud dels temps de pre-procés i post-procés), les millores aconseguides sempre superen el cas de *K* = 5.

10.5. Conclusions de les comparatives

A continuació es presenten les conclusions de les comparacions en pels dos estudis realitzats.

A. Primer estudi

Un cop estudiada l'afectació als resultats dels tres procediments en diferents situacions, s'arriba a les següents conclusions:

- En línies generals l'Heurística 2 aporta majors millores en el $C_{max.}$
- L'Algoritme Genètic millora en alguns casos els resultats obtinguts amb l'Heurística 2 i en els altres obté uns resultats molt pròxims als del citat procediment heurístic.
- La proporció de màquines de cada nivell afecta al resultat, obtenint millores més elevades en els casos amb una relació $^{m_a}/_{m_b}$ elevada. Aquest fet és degut a la distribució de peces: normalment hi ha més peces de nivell baix que d'altres, i amb moltes màquines de nivell baix es podrà produir sense necessitat d'haver d'assignar-ne a les màquines de nivell alt.
- Com més gran és el paràmetre *K* (major amplitud de la distribució dels temps de pre-procés i post-procés), més difícil és millorar els resultats de la Situació Inicial.
- És convenient dur a terme un segon estudi combinant els punts forts de cadascun dels dos mètodes que millor s'han comportat en el primer estudi (l'Heurística 2 i l'Algoritme Genètic).

B. Segon estudi

Les conclusions que s'extreuen després d'aplicar l'Algoritme Genètic 2' amb una població inicial més bona són:

- L'Algoritme Genètic 2' demostra aprofitar bé les millores introduïdes de l'Heurística 2.
- Igual que en el primer estudi, la proporció de màquines de cada nivell afecta al resultat, obtenint millores més elevades en els casos amb una relació $^{m_a}/_{m_b}$ elevada.
- Com més gran és el paràmetre K (major amplitud de la distribució dels temps de pre-procés i post-procés), costa més millorar els resultats de la Situació inicial.

11. Pressupost

En aquest apartat es realitza una valoració econòmica del projecte, analitzant les fases que el componen i el professional requerit per desenvolupar-la, un Enginyer Industrial o un Enginyer Informàtic per les tasques de programació. Les fases són les següents:

1. **Definició del problema**
   - 1.1. **Estudi del cas particular:** cerca i estudi d'informació i publicacions que tracten problemes del taller mecànic, en particular el cas de màquines en paral·lel a múltiples nivells i amb temps de preparació.
   - 1.2. **Estudi dels mètodes de resolució:** estudi dels mètodes més utilitzats i eficients per resoldre aquest tipus de problema. Anàlisi dels avantatges i dels inconvenients.
2. **Disseny del programa**
   - 2.1. **Disseny de l'algoritme:** elecció del mètode més apropiat per resoldre el problema i definició dels passos de l'algoritme. Determinació de l'estratègia i de paràmetres a ajustar per tal d'aconseguir els millors resultats.
   - 2.2. **Implementació del software:** traducció de les operacions de l'algoritme al codi del llenguatge escollit, en aquest cas el C#.
   - 2.3. **Creació de les dades:** creació de jocs de dades amb exemplars de diferents característiques (nombre de peces i nombre de màquines per nivell) per a la posterior avaluació del programa.
   - 2.4. **Proves i depuració:** verificar el correcte funcionament del programa i depurar els possibles errors que es puguin haver generat.
3. **Resultats**
   - 3.1. **Obtenció de resultats:** avaluació del programa sobre els jocs de dades creats.
   - 3.2. **Anàlisi de resultats i propostes de millora:** anàlisi dels resultats obtinguts en 3.1 i determinació dels paràmetres que proporcionen els millors resultats.
   - 3.3. **Elaboració d'un informe:** elaboració d'un informe on es detallen les fases del projecte, el funcionament del programa, l'anàlisi dels resultats i les conclusions.

L'Enginyer Industrial és el responsable de realitzar totes les fases menys la 2.1, realitzada únicament per un Enginyer Informàtic. La fase 2.4 es realitza conjuntament entre l'Enginyer Informàtic i l'Enginyer Industrial.

En la taula 11.1. es pot observar que s'ha reservat una columna pel nombre d'hores previstes per cada tasca i una altra columna pel salari per hora de la persona encarregada d'aquella tasca. Finalment s'ha establert una última columna amb el cost pressupostat per realitzar la tasca indicada. Tots els salaris per hora s'han basat en el sou de professionals amb experiència i en el cost per l'empresa de contractar aquella persona, no el sou net que rep la persona contractada.

Concepte	Categoria Professional	Salari (€/h)	Temps estimat (hores)	Cost Total (€)
1. Definició del problema				
1.1. Estudi de cas particular	Enginyer	60	100	6.000
1.2. Estudi dels mètodes de resolució	Enginyer	60	50	3.000
2. Disseny del programa				
2.1. Implementació del software	Enginyer	60	30	1.800
2.2. Disseny de l'algoritme	Enginyer	60	100	6.000
2.3. Creació de les dades	Informàtic	50	80	4.000
2.4. Proves i depuració	Enginyer	60	20	1.200
	Informàtic	50	20	1.000
3. Resultats				
3.1. Obtenció de resultats	Enginyer	60	50	3.000
3.2. Anàlisi de resultats i propostes de millora	Enginyer	60	50	3.000
3.3. Elaboració d'un informe	Enginyer	60	80	4.800
TOTAL				33.800 €
TOTAL (21% IVA Inc.)				**40.898 €**

Taula 11.1: Pressupost per la implementació del projecte

A continuació citarem alguns exemples on es podrien generar reduccions en els costos en cas d'implementar l'aplicació desenvolupada:

- Mà d'obra directa: al disminuir els temps de permanència de les peces a la fàbrica i optimitzar la gestió en la programació, es podria permetre requerir menys hores de mà d'obra i per tant reduir el cost en personal.

- Gestió dels estocs: al poder programar les peces abans, aquestes generen menys estoc de producte entre operacions, i per tant es precisa de menys espai per a l'emmagatzematge.
- Maquinària: si s'optimitza la seqüenciació de peces, es redueix el temps de funcionament de les instal·lacions, disminuint el consum elèctric de les instal·lacions.

12. Estudi del impacte ambiental

En l'entorn industrial actual, un requisit important per a totes les empreses és dur a terme una política que recolzi el desenvolupament sostenible. Una empresa sostenible és aquella que crea un valor econòmic (un bé o un producte) compatible a curt i llarg termini amb una realitat socioeconòmica intel·ligent, pròspera i respectuosa amb el medi ambient. Per tant, un aspecte important a tenir present durant la realització de tot projecte és l'impacte sobre el medi ambient.

En l'estudi de la sostenibilitat d'aquest projecte no s'han detectat danys causats per la hipotètica implantació en una empresa del programa informàtic dissenyat. En canvi, és possible identificar-hi diversos beneficis. En aquest cas el benefici és doble, al mateix temps que es troba una seqüència de peces que permet millorar el temps d'ocupació de les instal·lacions, es redueix el consum d'energia. En cas de què la maquinària de la línia de producció sigui molt sorollosa, la reducció de la duració total del procés també implica una disminució de la contaminació acústica.

El fet de començar a produir quan arriben les peces al sistema ajuda també a la reducció de magatzems dins de les plantes amb tots els estalvis energètics que això comporta com sistemes d'il·luminació, calefacció i refrigeració.

Finalment, el fet d'utilitzar una eina informàtica permet disminuir les necessitats d'un suport físic per la presa de dades i càlculs, amb la conseqüent reducció de l'ús de paper.

Conclusions

El problema tractat en aquest projecte és el de la programació d'un conjunt de peces classificades en 3 nivells a un conjunt de màquines en paral·lel classificades també en els mateixos nivells.

La classificació per nivells va sorgir amb el propòsit de satisfer les necessitats actuals de la indústria que, cada vegada més, necessita ser capaç de produir un major nombre de productes amb la menor inversió possible. Sovint, cada màquina produeix únicament les peces del seu nivell; per tant, no són necessaris canvis en la configuració de les màquines. Una solució a aquest problema és que es puguin produir peces en màquines d'altres nivells amb l'objectiu de minimitzar l'instant d'acabat de l'última peça.

A més, per adaptar el problema a la realitat industrial, s'ha tingut en compte que en la fabricació de qualsevol producte les peces no sempre estan disponibles a l'instant inicial ni quan surten de la màquina es consideren producte acabat. Per això s'han introduït els conceptes de temps de pre-procés i post-procés.

En aquest projecte, s'ha estudiat les millores que suposa que les màquines d'un determinat nivell puguin produir les peces del seu nivell i els seus inferiors tenint en compte l'afectació del temps de preparació per canviar la configuració de les màquines. S'ha comprovat que la programació de peces a diferents nivells permet homogeneïtzar el temps de funcionament del conjunt de màquines, i per tant fer més eficient el sistema productiu.

En una primera part del projecte, s'ha plantejat comparar l'aplicació de dos mètodes heurístics i un metaheurístic. El mètode metaheurístic utilitzat es divideix en dues parts; la primera, es troba una solució factible utilitzant l'algoritme presentat per Gharbi i Haouari (2002) **[23]** i posteriorment s'aplica dues fases de millora que intenten omplir els possibles temps morts de la màquina que ha deixat la primera solució; la segona consisteix en l'aplicació de l'Algoritme Genètic utilitzant la solució trobada en la primera part com a membre de la població inicial.

Un cop realitzat i analitzat el primer estudi, s'ha decidit modificar l'Algoritme Genètic per aprofitar els punts forts de cadascun dels dos mètodes que millor s'han comportat en el

primer estudi (l'Heurística 2 i l'Algoritme Genètic). Aquestes millores han consistit en introduir el procediment de l'Heurística 2 en la generació de la població inicial de l'Algoritme Genètic.

Després de l'aplicació de l'algoritme desenvolupat a diversos exemplars de peces, s'ha pogut comprovar que la major millora s'aconsegueix quan hi ha més màquines de nivell alt i quan el nombre de màquines de nivell baix és mínim, aconseguint unes millores de l'instant de finalització de l'última peça properes al 50%. Quan el nombre de màquines de nivell baix és màxim i el de nivell alt és mínim, la millora obtinguda és més reduïda i es mou amb valors al voltant del 20%, tot i que pot ser nul·la en alguns exemplars.

També s'ha arribat a la conclusió que com més gran és el paràmetre *K* (major amplitud de la distribució dels temps de pre-procés i post-procés), es fa més difícil millorar els resultats de la Situació inicial.

En canvi, la quantitat de peces amb les que es treballa i la dimensió del temps de preparació no tenen una influència gaire diferent sobre els resultats. En ambdós estudis s'han aconseguit millores del voltant del 30%.

Es pot concloure, doncs, que amb l'aplicació de l'Algoritme Genètic 2 s'aconsegueix que l'acabament de totes les peces sigui el més d'hora possible, i per tant, s'aconsegueix en certa mesura que els temps morts de les màquines disminueixin.

Possibles vies d'estudi

Un cop finalitzat el present projecte, penso que hi ha múltiples vies en les quals es podria continuar estudiant aquest problema. Entre elles m'agradaria citar-ne dues, que són les que trobo més interessants:

- La verificació / modificació de les variables de decisió de l'Algoritme Genètic. En el cas que ens acompanya, s'han donat per bones les obtingudes per Eva Vallada i Rubén Ruiz (2011) **[24]**.
- La verificació mitjançant un mètode exacte, com pot ser una *Programació Dinàmica Fitada* o un *Branch and Bound*, del bon comportament de l'Algoritme Genètic 2'.

Agraïments

En primer lloc, m'agradaria agrair al meu professor i tutor Manel Mateo tota la seva ajuda, col·laboració i implicació en aquest projecte i pel seu suport en tots els aspectes.

També voldria agrair especialment a un amic, Antar Moratona la seva inestimable ajuda en la fase de programació de l'algoritme i en general a totes les persones que m'han ajudat a realitzar el projecte amb les seves crítiques, comentaris i idees.

Finalment, considerant aquest projecte com el final d'una etapa, donar les gràcies a la meva família i amics pel suport rebut durant aquests anys.

Bibliografia

12.1. Referències bibliogràfiques

[1] HUANG, W.; CHUNG, P.W.H. *Scheduling of pipeless batch plants using constraint satisfaction techniques.* Computers & Chemical Engineering. Vol 24 (2-7), 2000, pàg.377-383.

[2] COMPANYS, R. *Secuenciacion*. Barcelona, ETSEIB – CPDA, 2003, pàg. 99-100.

[3] COMPANYS, R., COROMINAS. *Organización de la producción II. Dirección de operaciones 4*. Barcelona, Edicions UPC, 1996, pàg. 11-12, pàg. 24, pàg. 27-29.

[4] COMPANYS, R. *Secuenciación, programación de proyectos y de taller. Equilibrado y secuenciación de líneas*. CPDA – ETSEIB (Publicacions d'abast S.L.L.). pàg. 193-347.

[5] CONWAY, R.W.; MAXWELL, W.L.; MILLER, L.W. *Theory of scheduling*, Addison-Wesley, 1967, pàg. 5-8.

[6] PINEDO, M.L, *Planning and Scheduling in Manufacturing and Services*, 2005.

[7] LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G., y SHMOYS, D. B. *Recent developments in deterministic sequencing and scheduling: A survey.* In Lenstra Dempster & Kan Rinnooy (Eds.), *Deterministic and Stochastic Scheduling*, Dordrecht, Netherlands: D. Reidel Publishing Company, 1982, pàg. 35-74.

[8] VIGNIER, A.; BILLAUT, J.-C.; PROUST, C. *Scheduling problems of hybrid flowshop type: state of the art*, RAIRO Operations Research, 1999, Vol. 33 (2), pàg. 117–183.

[9] MORALES, E. *Búsqueda, Optimización y Aprendizaje*. Marzo 1999.

[10] ZANAKIS, S.H., EVANS, J.R., *Heuristic "Optimization": Why, When and How to Use It, Interfaces*, 1981, Vol.11(5).

[11] DÍAZ, A., [et al.], *Optimización Heurística y redes Neuronales en Dirección de Operaciones e Ingeniería*, Madrid, Editorial Paraninfo, 1996, pàg. 24-27.

[12] OSMAN, I.H.; KELLY, J.P. *Metaheuristics: Theory & Applications*, Kluwer Academic Publishers, Boston, USA, 1996.

[13] GLOVER, F. *Tabu Search — Part I*, ORSA Journal on Computing, 1989, Vol. 3, pàg.190-206.

[14] FEO, T.A.; RESENDE, M.G.C.; *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters, 1989, Vol. 8, pàg. 67-71.

[15] HOLLAND, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[16] HOOS, H.H.; STÜTZLE, T. *Stochastic local search: foundations and applications*, Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[17] HANSEN, P.; MLADENOVIC, N.; MORENO, J.A. *Variable Neighbourhood Search*. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial, Nº 19, 2003, pàg.77-92.

[18] GARCÍA LAUSIN, M. *Procediments heurístics de programació de comandes en diferents màquines amb temps de preparació dependents de la seqüència*. Proyecto Fin de Carrera. Escola Tècnica Superior d'Enginyeria Industrial de Barcelona, 2005.

[19] MARTÍ, R. *Meta Heurísticas en Optimización Combinatoria: Algoritmos Genéticos*. 2000.

[20] GOLDBERG, D.E. *Genetic Algorithms in Search, Optimization &Machine Learning*, Ed. Addison-Wesley, 1989.

[21] DAVIS, L. *Practical Handbook of Genetic Algorithms. Applications*. New York Van Nostrand Reinhold, 1991, pàg. 43-45.

[22] YOLIS, E. *Tesis de grado en ingeniería informática: Algoritmos Genéticos aplicados a la categorización de documentos*. Abril 2003.

[23] GHARBI, A.; HAOUARI, M. *Minimizimg makespan on parallel machines subject to release dates and delivery times*. Journal of Scheduling. 2002; 5: pàg. 329-355.

[24] VALLADA, E.; RUIZ, R. *A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times.* European Journal of Operational Research, 2011, pàg. 612-622.

12.2. Bibliografia complementària

- ALLAHVERDI, A. [et al.] *A survey of scheduling problems with setup times or costs.* European Journal of Operational Research, 2008, pàg. 985-1032.
- ALLAHVERDI, A. [et al.] *A review of scheduling research involving setup considerations.* Omega, Int. J. Mgmt Sci., 1999, pàg. 219-239.
- CENTENO, g.; ARMACOST, R. *Minimizing makespan on parallel machines with release time and machine eligibility restrictions.* Int. J. Rod. Res., 2004, VOL. 42, NO. 6, pàg. 1243-1256.
- LU, L.; YUAN, J.; ZHANG, L. *Single machine scheduling with release dates and job delivery to minimize the makespan.* Theoretical Computer Science 393, 2008, pàg. 102-108.
- RUIZ, R.; ANDRÉS-ROMANO, C. *Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times.* Int. J. Adv. Manug. Technol, 2011, pàg. 777-794.
- MONTOYA-TORRES, J. [et al.] *Machine Scheduling with Sequence-dependent Setup Times using a Randomized Search Heuristic.* IEEE, 2009, pàg. 28-33.
- DUNSTALL, S.; WIRTH, A. *Heurisitc method for the identical parallel machine flowtime problem with set-up times.* Computers & Operations Research, 2005, pàg. 2479-2491.
- CEBALLOS, F. J. *Enciclopedia de Microsoft Visual C#.* Ra-Ma Editorial, 2006.
- GONZÁLEZ, A. *Programación de Bases de Datos con C#.* Ra-Ma Editorial, 2010.
- Programació en C#

  http://www.dotnetperls.com/
- Visual Studio 2012

  http://msdn.microsoft.com/en-us/library/dd831853.aspx