1. Introducció

1.1. Antecedents

El GREP (grup de recerca de producte, procés i producció) de la UdG actualment disposa d'una eina informàtica desenvolupada en un PFC del 2003 que li permet fer la seqüenciació de la producció d'un taller mecànic amb un màxim de cinc productes, vuit possibles rutes de fabricació per a cada producte i tres màquines.

Aquesta eina és molt resolutiva per a aquests casos, ja que estudia totes les possibilitats i les comprova una per una. No obstant aquest fet presenta una sèrie de limitacions:

a) Només permet treballar amb un número determinat de peces a programar és a dir de 2 a 5 peces.

b) El número de màquines queda fixat a 3 màquines.

c) Si el número de possibilitats és molt elevat el programa necessita molt de temps per a poder calcular-les totes

d) Si el número és molt més elevat (més de 7200) el programa no pot calcular les possibilitats.

e) No es poden afegir rutes per al productes existents.

Degut a aquestes limitacions ens hem proposat desenvolupar una altra aplicació informàtica partint de la ja existent, tal que no estudiï totes les combinacions sinó que només n'estudiï un conjunt suficientment ampli com perquè ens retorni un conjunt de solucions adaptables. Per poder realitzar aquesta operació s'utilitzaran dos tipus d'algoritmes:

- Algoritmes genètics

- Cerca TABU

1.2. Objectius

L'objectiu del següent projecte és realitzar una aplicació informàtica que pugui determinar una seqüenciació òptima de les operacions a realitzar en un taller mecànic. En aquest taller i tindrem "n" peces, on cada peça tindrà varies rutes de fabricació possibles i "m" màquines.

Les principals característiques de l'aplicació informàtica són:

- Poder treballar amb un número variable de peces amb un màxim de 10.
- Poder treballar amb un número variable de màquines.
- Afegir rutes fàcilment a cada producte.
- Afegir nous productes que poder seqüenciar.
- Poder obtenir seqüències acceptables emprant el sistema d'algoritmes genètics i el de recerca TABU.
- Per l'AG, poder escollir la selecció de progenitors, el mètode de creuament i les variables de mutació. A més a més de la mida de la població, el nombre d'iteracions del procés i el màxim de mitjanes iguals entre una població i la seva descendència.
- Per la recerca TABU poder escollir el nombre d'iteracions, del procés.
- Efectuar el procés escollint la regla de seqüenciació (FIFO, RATIO..)

- Desenvolupar la interfície gràfica del programa de manera que sigui fàcil fer el seguiment de les operacions que ha realitzat l'algoritme en qüestió per arribar a la solució.

Degut a la impossibilitat de comprovar totes les combinacions en un cas on tinguem vàries màquines i rutes, compararem la nostra eina amb un cas on l'eina actual que disposa el GREP sí determini la millor opció i així podrem determinar si la nostra eina és capaç de trobar la millor solució o en el seu defecte la solució trobada és satisfactòria.

1.3. Abast

L'abast del següent projecte és la seqüenciació de la producció de qualsevol taller mecànic amb un número variable de màquines, productes i rutes. Amb la implementació d'una eina per l'estudi de rutes i l'optimització de les interaccions entre aquestes per poder avaluar quines accions i polítiques es poden destacar com a millors, després d'un anàlisi de les dades obtingudes amb el programa existent les simulacions d'aquestes en el programa informàtic implementat per aquesta tasca, tot seguint un ordre creixent en la quantitat de peces i rutes a optimitzar, podent obtenir una eina ràpida i útil.

2. Introducció als sistemes de producció.

Hi ha tot un conjunt de factors que determinen com es fabrica un producte, i sembla que les característiques d'aquest seran les de més pes en l'elecció del sistema productiu.

Hi ha tres tipus de sistemes productius.

- Projectes
- Lots
- Contínua

Alhora, aquesta classificació també ordena els projectes en quantitat obtinguda en el procés (de menys (projectes) a més (producció contínua)), també classifica en pes del client (de més a menys en aquest cas), en flexibilitat (creixent),....

Cal dir que la producció per lots es pot alhora dividir en tres tipus de producció: línia, batch i tallers (job-shop). Per copsar les característiques de cada configuració productiva, s'adjunta la taula següent (Taula 4.1):

Configuració	***Homogeneitat***	***Repetitivitat***	***Producte***	***Intensitat del capital***	***Flexibilitat***	***Participació del client***	***Volum***
Contínua	Elevada	Elevada	Estàndar	Alta	Inflexible	Nula	Molt gran
Línia	Mitjana	Mitjana	Diverses opcions	Mitja	Baixa	Baixa	Mig-gran
Batch	Baixa	Baixa	Moltes opcions	Baixa	Mitja	Mitja	Baix
Taller (job-shop)	Molt baixa	Molt baixa	A mida	Baixa	Alta	Alta	Molt baix
Projecte	Nula	Nula	Únic	Nula	Alta	Alta	Un o pocs

Taula 4.1: Qualitats sistemes de producció

2.1. Producció per projectes

La configuració productiva per projectes és la més emprada en l'elaboració de productes (i també serveis) que tindran una baixa repetició, normalment únics. S'empra aquest tipus de sistema de producció amb productes que requereixen una alta coordinació, com són precedències entre tasques i assignació de recursos, motiu pel qual hi ha un equip de coordinació. L'altra tasca d'aquest equip és el control temporal del projecte, per complir les dates establertes.

Una de les característiques del producte final realitzat mitjançant aquest tipus de producció és l'elevat grau d'implicació del client. En els projectes, hi ha un gran fluxe d'informació entre el client i el productor, present en totes les fases del projecte, però especialment en la fase de disseny.

2.2. Producció contínua

La configuració contínua es basa en eliminar els temps ociosos i en fer les mateixes operacions en les mateixes màquines per l'obtenció del mateix producte. La disposició és en línia i l'objectiu és obtenir una producció de flux constant.

Això s'obté dissenyant les màquines per a què "encaixin" en la línia de producció, i fent els processos tant automàtics com sigui possible. En aquesta línia, s'adapta el disseny de la planta a l'ordre de les operacions del producte, fent connexions automàtiques entre màquines (si és possible), per perdre el mínim temps possible.

En els processos continus s'intenta no parar la cadena, així que les operacions realitzades en les màquines haurien de tenir la mateixa durada en tota la cadena. Una possible solució és incorporar més màquines del mateix tipus en la cadena per tal de què el flux no s'obturés en un coll d'ampolla.

2.3. Producció per lots

El tipus de procés productiu intermig, amb versatilitat però amb certa quantitat de peces a fer, és el procés per lots. Les màquines o centres de treball que conformen les eines de la configuració per lots poden executar diferents tipus d'operacions. Així típicament, una màquina podrà realitzar diverses operacions, algunes de les quals podran esser realitzades per altres centres de treball.

Però cal tenir en compte que tot i poder realitzar operacions iguals (o amb un resultat similar) en diferents màquines, el temps d'operació en cada centre de treball serà diferent, amb tota probabilitat. Les màquines disponibles en una configuració productiva per lots tenen un alt cost variable, per la poca automatització, però un baix cost fix degut a la baixa inversió inicial, comparats amb la producció contínua.

En aquest entorn hi ha un elevat nombre de combinacions possibles per la realització dels productes, degut a la versatilitat anteriorment esmentada, així que la gestió del sistema productiu es converteix en una peça clau per obtenir una alta eficiència. I la gestió va determinada per la demanda i les correctes estimacions d'aquesta.

La planificació agregada agrupa els treballs a realitzar en un període de temps, i és en aquest punt on es determinen les necessitats d'equips i treballadors. Un cop s'ha obtingut la programació d'operacions es realitza la càrrega dels tallers o de màquines. Aquesta acció consisteix en assignar a les màquines les operacions que han de realitzar i a quin producte cal fer-les. Així és necessària una regla de prioritat de pas entre les diferents comandes, entre les màquines del taller que permeti gastar la menor quantitat d'inventari amb el menor ús de recursos i el menor temps necessari, obtenint el màxim benefici i complint així el requeriment temporal de les comandes: les dates d'entrega. Tot això quedarà agrupat en el programa detallat d'operacions.

2.3.1. Producció en línia

La tendència de la producció en línia és apropar-se al màxim possible a la producció contínua, per així poder obtenir els avantatges de l'economia d'escala i de l'automatització degut a l'estandardització del producte.

Això és possible en fabricacions de grans lots de productes que tenen poques diferències entre ells, amb un procés productiu molt similar. D'aquesta manera, es poden ubicar les màquines seguint una línia de producció per on van passant els productes.

Els recursos necessaris requereixen d'una alta automatització, que en part és el que es busca en aquest tipus de producció, eliminar temps morts. Aquesta necessitat d'automatitzar el procés implica una inversió elevada, i degut a que les peces no són idèntiques, els temps de preparació augmenten respecte la producció contínua, així com els costos per aquesta tasca. Per motius similars, és necessària mà d'obra més qualificada, amb més especialització i més cost.

2.3.2. Producció en batch.

La producció en batch es basa en fer lots de productes iguals que es tracten com un producte més gros, amb diverses repeticions d'operacions. La clau està en fer els lots de productes de la mida escaient per poder evitar cues (temps perdut en l'elaboració dels productes) i temps ociosos (moments en que un centre de treball no està realitzant cap operació, doncs espera l'arribada d'algun producte).

Els productes tenen un cert grau d'estandarització, però les sèries a fabricar no són suficentment grans per poder produir en línia ni de forma contínua. Les màquines necessiten certa automatització, cosa que requereix més inversió. La programació esdevé complexa quan s'aplica a la realitat, doncs sovint s'han de reconsiderar les prioritats establertes inicialment per la realització de les tasques, degut a què hi ha desviacions en els temps que poden implicar conflictes entre l'ordre preestablert i el requeriment de les dates d'entrega.

2.3.3. Producció en tallers (job-shop)

La producció en tallers, o entorn job-shop, es basa en l'extrema flexibilitat de les comandes, produint objectes "a mida", amb molt de disseny aportat pel client. Normalment, les comandes són reduïdes i els treballadors s'encarreguen del procés d'obtenció de la comanda mitjançant les eines disponibles (màquines o centres de treball).

La sofisticació tecnològica és baixa, però l'elevat grau de flexibilitat de les màquines comporta un enorme ventall de possibilitats de producció per el mateix producte (diverses rutes), complicant la programació.

2.4. Seqüenciació

En tot sistema intermitent, com és el cas d'un entorn job-shop, el que s'intenta definir amb més cura és la seqüenciació, és a dir, l'ordre d'execució d'operacions assignades a un centre de treball. Així doncs, l'objectiu principal de les eines de gestió d'entorns job-shop és l'obtenció d'un llistat de les operacions que es realitzaran en una determinada màquina, ordenat en funció del temps d'entrada en el centre de treball. D'aquesta manera, es poden saber els temps ociosos en un taller i corregir-los aplicant una nova política, si s'escau.

Cal tenir en compte que la decisió final de quin és l'element de control del taller més important, dependrà de cada situació, doncs no sempre serà tan necessari ocupar més les màquines com acabar les peces abans o utilitzar una màquina el menys possible (recurs extern).

El problema de l'entorn job-shop es pot resumir en un conjunt de premisses:

- S'han de realitzar n peces en m màquines.

- Per realitzar cada peça cal fer unes determinades operacions que són conegudes prefixades i amb una durada determinada.

- Una ruta és el camí seguit per una peça per en la seva elaboració. És a dir, és el conjunt d'operacions que ha de realitzar a través de les màquines per ser acabada. També es pot entendre com la visió del procés des del punt de vista de la peça. Cal tenir en compte que en un taller mecànic no hi ha una única ruta per l'elaboració d'una determinada peça.

- Una màquina no pot fer dues peces alhora, ni una peça pot estar en dues màquines en el mateix moment.

- Una operació començada en una màquina no s'interromprà per la realització d'una altra operació d'una altra peça. Les operacions que comencen, s'acaben sense interrupcions.

3. Mètodes de resolució per als problemes seqüenciació.

El problema objecte d'estudi és, com a molts dels que es presenten en el domini d'organització industrial, de tipus combinatori.

Resoldre un problema combinatori es pot resumir 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 altra solució (Morales, 1999). Però en molts problemes, depenent de la seva complexitat, arribar a una solució òptima requereix un procés excessivament llarg.

Si es parla de que un problema és P si existeix un algoritme que en temps polinomial sigui capaç de trobar una solució; si, per el 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 es polinòmicament reduïble a ell. La majoria dels problemes del taller mecànic són d'aquest tipus.

Així per exemple, si es pretén resoldre un problema combinatori per enumeració amb n= 40 i es requereix una permutació, es tindrien $40!=8.16*10^{47}$ possibilitats. A una µs per possibilitat es tardaria molts segles a analitzar totes les solucions. Òbviament, no és factible tot i que s'acoti l'espai de recerca de la solució per tant es recurreix a altres tipus de mètodes de resolució anomenat heurístics.

D'aquesta manera, els procediments de resolució poden ser de dos tipus:

- Exactes: garanteixen trobar una solució òptima, però si el problema és complex, el temps empleat en trobar la solució i garantir-la no és viable.

- Heurístics: no garanteixen trobar la solució òptima buscada, però sí que permeten trobar una solució òptima amb un temps de recerca permissiu.

3.1. Classificació del mètodes heurístics

Una classificació dels diferents mètodes de resolució que es poden aplicar a l'hora de resoldre un problema com l'enunciat podria ser la que es mosta a la figura 3.1.

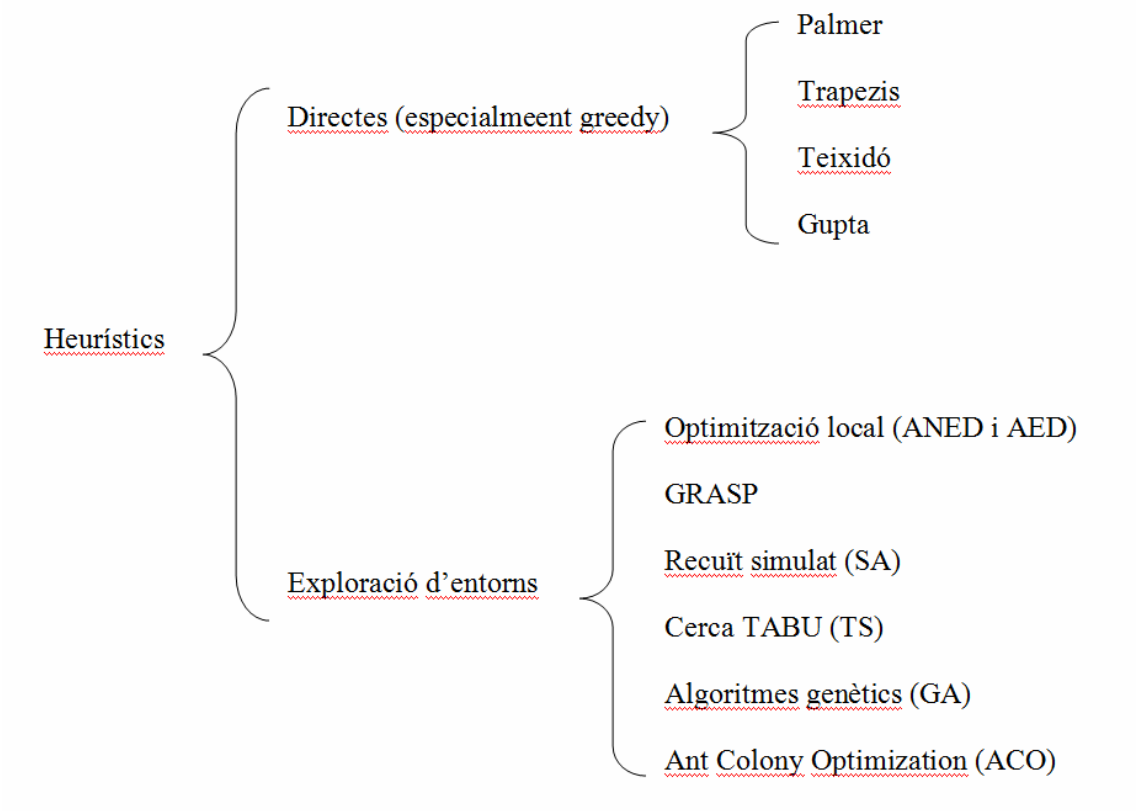


Figura 3.1: Classificació mètodes heurístics

Tal i com s'observa a la figura 5.1 els procediments heurístics, es divideixen en:

- Directes: construeixen una solució de forma progressiva per passos. En problemes de taller mecànic de flux regular destaquen els procediments Palmer, Trapezis, Teixidó i Gupta.

- Exploració d'entorns: a partir de la solució en curs es genera el seu entorn format per solucions veïnes i s'escull entre elles una nova solució en curs guardant al llarg del procés la millor solució possible trobada.

Els diferents procediments d'exploració d'entorns difereixen en la definició de l'entorn, l'elecció de la solució inicial, l'elecció de la nova solució en curs i la detenció del procés. Els procediments més utilitzats en aquest sentit són els següents:

1. Optimització local (ANED y AED)

En l'algoritme exhaustiu de descens (AED), a partir de la solució en curs, es generen i avaluen tots els seus veïns i si algun és millor que la solució en curs, es pren com a nova solució; en cas contrari, el procediment es dona per acabat.

L'algoritme no exhaustiu de descens (ANED) es diferencia només en què s'avaluen en cert ordre els veïns.

2. GRASP (Greedy Ranomized Adaptive Search Procedure)

Es construeix una solució factible generalment mitjançant una heurística greedy. En cada pas, s'estableix una llista de candidats, però en lloc d'escollir el que aporta més, s'escull a l'atzar un els candidats d'una llista restringida. Si la solució obtinguda és millor que la guardada com a tal, s'efectua la substitució.

3. Recuit simulat (SA)

El nom procedeix de l'analogia entre el procediment i el tractament tèrmic que aplica a diversos materials.

Donada una solució en curs, s'obté una veïna de la mateixa. Si és millor (o igual), la substitueix com a tal; si és pitjor, pot passar a ser la solució en curs i el de la veïna i que decreix monòtonament al llarg de l'execució de l'algoritme.

4. Recerca tabú (TS)

Existeix una llista de propietats de solucions. Si alguna característica d'una solució coincideix amb algunes de les propietats de la llista és una solució tabú.

Es generen tots els veïns de la solució en curs, es classifiquen (en no tabú i tabú) i s'avaluen. Si el valor del millor veí és millor que el de la millor solució trobada, substitueix a aquesta i a la solució en curs i s'introdueix una característica de l'antiga solució a la llista tabú.

5. Algoritmes genètics

Aquest procediment es basa en les idees de selecció natural, i és el mètode seleccionat per la resolució del problema.

Com ja s'explicarà amb molt més deteniment a continuació, en lloc d'una solució en curs se n'arrastren varies (població). Es selecciona els pares, es creuen i/o muten i s'obtenen fills regenerant la població inicial.

6. 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 de les formigues, insectes casi cecs, per establir el camí més curt des del seu niu fins a la font d'aliment i tornar.

Tal i com marquen els objectius, aquest projecte es centrarà en dos tipus d'algoritmes que són:

- Els algoritmes genètics
- L'algoritme de cera TABU

3.2. Algoritmes genètics

Els algoritmes genètics varen ser introduïts per John Holland, investigador en la Universitat de Michigan, en 1970, inspirant-se en el procés observat en l'evolució natural dels éssers vius. En un principi va denominar aquesta tècnica com "plans reproductius", però es va fer popular sota el nom "d'algoritmes genètics" després de la publicació del seu llibre (Holland, 1975).

El concepte d'evolució va nèixer en el segle XIX a partir dels treballs de Lamarck i , sobretot, Darwin. Els seus estudis van postular la idea que els organismes existents són el resultat d'un procès desenvolupat al llarg del temps i van eliminar l'antiga creença que les espècies existien i mantenien el seu aspecte immutable a través del temps.

La Teoria de l'Evolució afirma que l'evolució opera en els cromosomes i no en els individus. Mitjançant la selecció natural, els cromosomes amb variacions més fortes es reprodueixen més sovint que els altres. Aquest procés premia les formes més adaptades al medi natural en decliment de les menys adaptades (supervivència o predominància dels més preparats). L'evolucó té lloc en el procés de reproducció amb la combinació (i possible mutació) dels cromosomes dels progenitors.

Holland va establir una analogia entre el conjunt de solucions d'un problema i el conjunt d'individus d'una població natural. En paraules del propi Holland (1975):

"Es poden trobar solucions aproximades a problemes de gran complexitat computacional mitjançant un procés d'evolució simulada"

La correspondència del procés citat amb la seqüència seguida per els algoritmes genètics és la següent:

La informació d'un problema es codifica en una sèrie de cadenes (cromosomes) que formen una població. Per mitjà de l'intercanvi d'informació entre elles (creuament) i/o la variació aleatòria d'aquesta informació (mutació), s'avalua la solució definida per les cadenes, relacionada amb el problema en qüestió, mitjançant aquesta forma d'adaptabilitat a l'entorn (evolució de la funció objectiu o fitness). Els individus

s'escullen en funció de la seva millor adaptació (selecció). Es genera una nova població a partir d'aquests cromosomes d'elit, sobre els que s'hi torna a aplicar aquest procés iteratiu una i altra vegada. D'aquesta manera, la qualitat mitjana dels individus augmenta progressivament.

Actualment, els algoritmes genètics es presenten com un dels més prometedors en el camí cap a la intel·ligència artificial i han demostrat la seva vàlua en múltiples àrea d'aplicació. Són més utilitzats que la resta de mètodes heurístics, probablement degut al poder seductor del neo-Darwinisme i perquè posseeixen un gran interés per l'evident analogia que converteix al programador en un creador de "vida" (Herrán, 1998).

3.2.1. Elements d'un algoritme genètic

Per portar a la pràctica l'esquema anteriorment comentat i concretar-lo en un algoritme, s'ha d'especificar els següents elements:

- Una representació cromosòmica a manera de codi.
- Una població inicial.
- Una mitjana d'evaluació.
- Un criteri de selecció de cromosomes.
- Una o varies operacions de recombinació o creuament.
- Una o varies regeneracions de la població.
- Una condició de fi d'algoritme.

1. Codificació

Decidir el codi és la primera decisió que s'ha de prendre en el moment d'implementar l'algorisme. El principal objectiu de la codificació resideix en què els elements característics del problema es puguin representar de tal manera que resulti senzilla la seva implementació i comprensió.

La codificació més comuna és a través de cadenes binaries, tot i que també es puguin utilitzar números reals o inclús lletres, vectors, arbres o gràfics. El primer d'aquests esquemes és que el gaudeix de més popularitat donat que és el que va propusar originalment Holland (1975), resulta fàcil d'implementar i la seva convergència està provada. Tot i això, en molts problemes poden resultar poc naturals, ineficaces o impossible d'utilitzar-les. L'elecció d'un o altre mètode radica en la complexitat del problema.

2. Població inicial

Per constituir la població inicial, que serà la població base de les successives generacions, existeixen varis 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 així una convergència prematura (Martí, 2000).

Una qüestió important és la relacionada amb la mida idònia de la població. És intuïtiu que les poblacions petites corren el risc de no cobrir adequadament l'espai de recerca, mentre que el treballar amb poblacions de gran mida pot portar problemes relacionats amb la complexitat d'operacions.

Goldberg (1989) va demostrar que la mida óptima d'una població, amb codificació binària, creix exponencialment amb la longitud del cromosoma. Estudis posteriors, com els d'Alander (1992) o Schaffer, coincideixen en afirmar que una població inicial no superior a 30 individus garanteix, en general, donar una solució factible al problema, tot i que s'ha de dir que els problemes que van servir de base per realitzar aquesta afirmació eren de complexitat mitja. En canvi, el que sí ha de semblar segur és que la

mida de la població inicial ha d'estar relacionada amb la longitud dels cromosomes i la complexitat del problema.

3. Mesura d'evaluació o fitness

El fitness o mesura d'avaluació assigna a cada cromosoma un número real, que reflexa el nivell d'adaptació al problema de l'individu representat per el cromosoma. Dit valor també rep el nom de funció 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 diferencies molt grans entre individus i, per tant, provocar una convergència prematura, ni diferencies molt petites que puguin produïr un estancament en el procés de cerca de la solució.

La regla general per construir un bon fitness és que aquest ha de reflexar el valor de l'individu de manera "real", però en molts problemes d'optimització combinatòria, on existeixen gran quantitat de restriccions, bona part dels punts de l'espai de cerca representen individus no vàlids.

Per aquestes situacions, existeixen diverses solucions tals com no considerar aquells individus que no verifiquen les restriccions i seguir iterant fins aconseguir individus vàlids, assignar a dits 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. Però la manera més generalitzada d'enfocar-ho està basada en la penalització de la funció objectiu. L'idea general consisteix en dividir el fitness de l'individu per una quantitat (la penalització) que guarda bé relació amb les restriccions que dit individu viola, o bé amb el cost associat a la conversió de dit individu en un altre que sí que compleixi les restriccions.

4. Selecció

La selecció és el procés per el qual s'escullen una o vàries parelles d'individus de la població inicial per a què desenvolupin el paper de progenitors, creuant-se posteriorment i obtinguent descendència.

Es poden seleccionar només dos individus per què es regenerin d'entre tots els individus que constitueixen la població, o es pot optar per emparellar a un major número d'individus que formen la població. La primera és la que s'adequa més al model biològic, però la segona és la que obté millors resultats en un temps menor.

Els quatre mètodes més típics de selecció de parelles (Davis, 1991) són:

1. Fit-Fit: els més competitius es creuen entre sí, i els pitjors amb els pitjors. En una llista ordenada de forma decreixent segons el fitness dels individus, el primer s'emparellaria amb el segon, el tercer amb el quart i així successivament.

2. Fit-weak: aquells individus amb millor fitness es creuarien amb els que el tenen pitjor. El més fort es creua amb el més dèbil, el segon amb el penúltim i així successivament.

3. Random: l'emparallament dels individus es fa d'una manera completament aleatòria, de manera que les possibilitats

4. Ruleta: la probabilitat de que cada individu sigui seleccionat per creuar-se és ponderada segons el valor del fitness que posseeixi . Rep el nom de "Ruleta" per la semblança que hi ha entre aquest mètode i una "ruleta de la sort" on aquells individus amb millor mesura d'evaluació són els que ocupen una major superfície en la ruleta i, tenen per tant més probabilitat d'esser seleccionats.

$$Psel = \frac{1/fitness}{\sum 1/fitness}$$

5. Creuament

L'operador creuament permet l'intercanvi d'informació entre individus d'una població, recombinant els cromosomes, donant lloc a nous individus. El creuament s'aplica d'acord amb la probabilitat de creuament pc (es creuen si el número generat de forma aleatòria és menor o igual a pc, i no ho fan en cas contrari) i en un punt, o punts, escollits aleatòriament.

L'objectiu fonamental d'aquest operador es aprofitar les millors qualitats de tots els individus. Així, si un individu té un subgrup de gens que el tornen més competitiu i es creua amb un altre individu que posseeixi un altre conjunt de gens igualment profitosos, l'addició de tots dos grups de gens en un descendent podrà produir un individu amb major nivell de fitness (Yolis, 2003).

Si el nou cromosoma creat és més competitiu, tindrà més possibilitats de sobreviure al següent creuament on se li torni a aplicar l'operador fitness. En cas contrari, tindrà menys possibilitats de sobreviure a la competència.

Els models més usuals de creuament són els següents:

- D'un punt de creuament (o bàsic): S'escull aleatòriament un punt de ruptura en els pares i s'intercanvien els gens.

- De dos punts de creuament: s'escullen dos punts de ruptura a l'atzar per intercanviar.

- Uniforme: En cada gen s'escull a l'atzar un pare perquè contribueixi amb el seu gen al del fill, mentre que el segon fill rebrà el gen de l'altre pare.

- Creuaments per permutació (PMX, SEX, CX...): Són operadors més sofisticats fruit de combinar i aleatoritzar els anteriors.

Segons la codificació dels individus, és més aconsellable el possible ús d'un o altre tipus.

6. Mutació

L'operador mutació s'aplica després del creuament amb l'objectiu d'incrementar la diversitat poblacional. Es defineix com una variació elemental de les informacions contingudes en el codi genètic (habitualment, un canvi d'un gen a l'altre). La mutació s'aplica d'acord amb la probabilitat de mutació pm (en aquest cas anàloga a pc). En general, es recomanen baixos percentatges de mutació (pm), ja que taxes elevades podrien aproximar l'evolució a una cerca aleatòria intensament explorativa. Cau la possibilitat d'aplicar aquest operador varies vegades en el mateix individu.

Aquest operador permet, per una part, augmentar l'exploració en l'espai de cerca cap a nous entorns ja que produeix un increment de la diversitat poblacional, i, per altre part, evita l'aparició d'algoritmes bloquejats o de poblacions degenerades (poblacions iguals o similars).

Els models més usuals de mutació són els següents:

- Mutació tova: Consisteix en intercanviar, a l'atzar, dos gens consecutius en la seqüència.

- Mutació intercanvi: Consisteix en intercanviar, a l'atzar dos gens no consecutius en la seqüència.

1. Mutació inversió: S'escullen a l'atzar dos punts de la seqüència. Els gens compresos entre els dos punts inverteixen el seu ordre.

7. Regeneració i condició de fi.

Després dels creuaments i mutacions, s'ha de regenerar la població. La manera més usual de regenerar-la és que els individus concebuts passin a formar part de la població substituint, bé al individus menys competitius de la població, o bé als pares que els van engendrar. 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 altra de la població filial.

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

El número d'iteracions o regeneracions dependrà fundamentalment de la complexitat del problema. A més número de regeneracions, més gran serà la probabilitat de trobar una solució factible o propera a l'òptim final del procés, però s'ha de tenir en compte que, si aquest número és molt gran, el cost computacional pot ser molt elevat. Per això, en ocasions el número de regeneracions depèn directament del temps d'execució del problema. Una altra possibilitat consisteix en parar l'algoritme quan aquest arribi a l'òptim (sempre que aquest sigui conegut), o bé quan es consideri que la solució en curs és satisfactòria (respecte a l'òptim conegut).

8. Altres operadors genètics

Amb el temps, s'han incorporat altres operadors que no formen part de l'algoritme genètic bàsic, però que amb la seva utilització han aportat millores a les prestacions de l'algoritme.

- Elitisme: Tracta d'evitar que en el transcurs de l'algoritme, individus amb excel·lents qualitats puguin no generar descendència o ser aquesta molt reduïda, perdent així una valuosa informació genètica. Mitjançant aquest operador, el millor o millors individus d'una població s'integren directament a la població filial. L'elitisme permet a la vegada l'ús de probabilitat de creuament 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 que s'hi pugui incloure el millor individu de l'evolució finalitzada. Aquest operador és útil si es pretén que la solució sigui molt pròxima a l'òptim i no es disposa de temps suficient.

- Repoblació: S'utilitza quan l'algoritme s'estanca, amb el fi d'augmentar la diversitat. Consisteix bàsicament en preservar els individus més adaptats, generant a partir d'ells una nova població que elimini l'anterior aplicant mutació.

Una bona pràctica consisteix en generar una part de la població d'aquesta manera, i la resta d'una forma aleatòria. Una altra bona possibilitat, treballant amb estratègies elitistes, consisteix en aplicar aquest operador cada cert número de generacions, independentment de que l'algoritme s'estanqui o no, per accelerar la convergència al òptim global.

- Clonació: Consisteix en la duplicació de l'estructura genètica d'un cromosoma per la generació següent amb el fi de que aquest exemplar sobrevisqui intacte per competir a la nova generació. Aquesta tècnica es especialment útil per aquells algoritmes en els que un excel·lent descendent pot "eliminar" a un de no tant bo, però que en canvi serveixi com a bon pare per a posteriors creuaments.

3.2.2. Problemes dels algoritmes genètics

Els algoritmes genètics poden presentar una sèrie de problemes que s'han de tenir molt 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'aquest problemes.

1. Convergència prematura

Aquest és el problema més comú, radica en trobar un màxim o mínim relatiu en l'espai de solucions i concentrar tota la recerca en aquest entorn, deixant la recerca en la resta de l'espai de solucions, on pot ser que hi hagi solucions millors.

Les possibles solucions són vàries i totes elles estan relacionades amb la variació de certs operadors: increment de la taxa de mutació, augment de la mida de població o disminució del grau de selecció. Totes aquestes mesures tenen com a objectiu procurar que l'espai d'exploració de l'espai de solucions sigui el més extens possible.

2. Terminació lenta

L'ús dels algoritmes genètics no garanteix sempre una solució factible en un temps plausible. Per això, moltes vegades s'ha de trobar un equilibri entre el grau de fiabilitat

de la solució i el cost computacional utilitzat per trobar dita solució. Aquest compromís depèn dels operadors de la mida de la població i número de regeneracions.

3.2.3. Algoritmes genètics paral·lels: models illes

Aquest tipus d'algoritmes genètics consisteix en dividir la població en varies subpoblacions, en cada una de les quals es porta a terme un algoritme genètic. Cada subpoblació és una "illa". Cada nombre de regeneracions, s'efectua un intercanvi d'informació genètica d'una "illa" a una altra, en un procés que rep el nom de "migració". La introducció de la "migració" fa que aquest model d'algoritme sigui capaç d'explotar les diferències entre les diverses subpoblacions, obtenint d'aquesta manera una àmplia diversitat genètica. De la taxa de migració pot dependre la convergència prematura de la recerca.

En funció de la comunicació entre les subpoblacions, es poden distingir diferents models d'illes:

1. Comunicació en estrella: La subpoblació amb millor mesura d'evaluació és seleccionada com a "mestra", essent la resta considerades com "esclaves". Les subpoblacions esclaves envien als millors individus cap a la subpoblació mestra, i aquesta envia els seus millors cap a les subpoblacions esclaves.(fig 3.2.3.1).

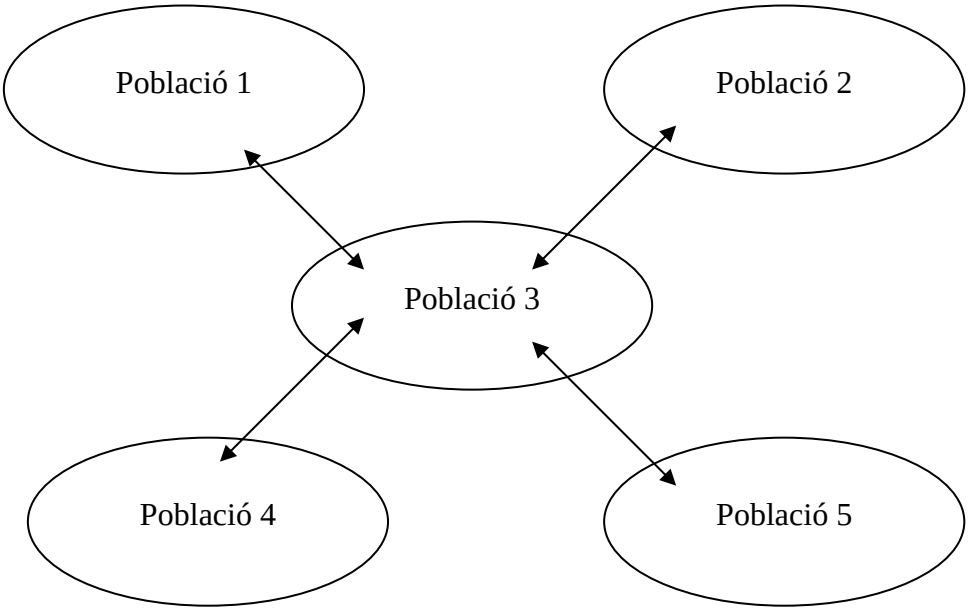


Figura 3.2.3.1: Comunicació de poblacions model illa

2. Comunicació en xarxa: No es crea cap mena de jerarquia entre les illes, i s'intercanvien els seus millors individus entre elles.(fig 3.2.3.2)

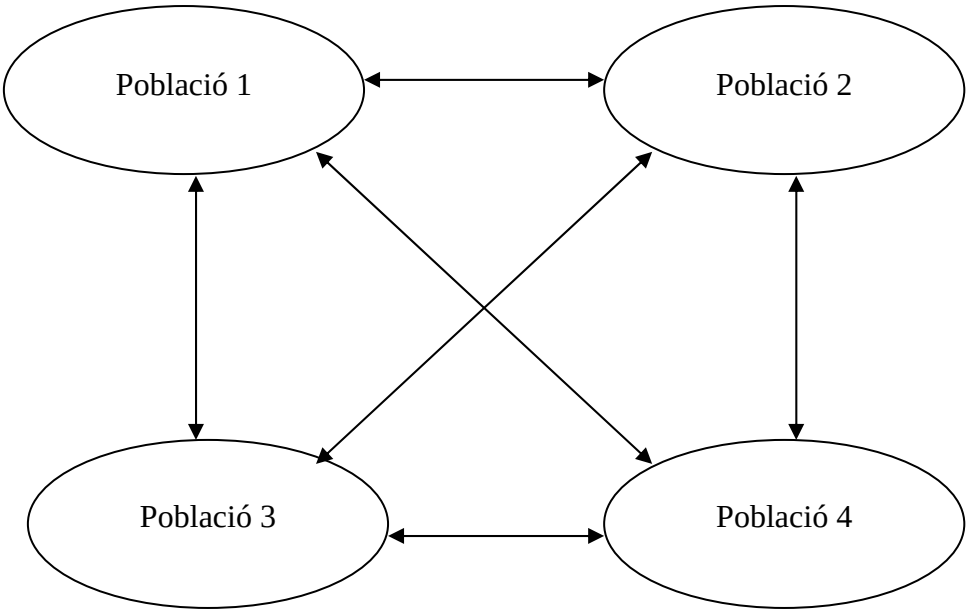


Figura 5.2.3.2: Comunicació de poblacions model xarxa

3. Comunicació en anell: Cada subpoblació envia els seus millors individus a la població veïna, provocant així una migració amb un sol sentit. (fig 5.2.3.3)

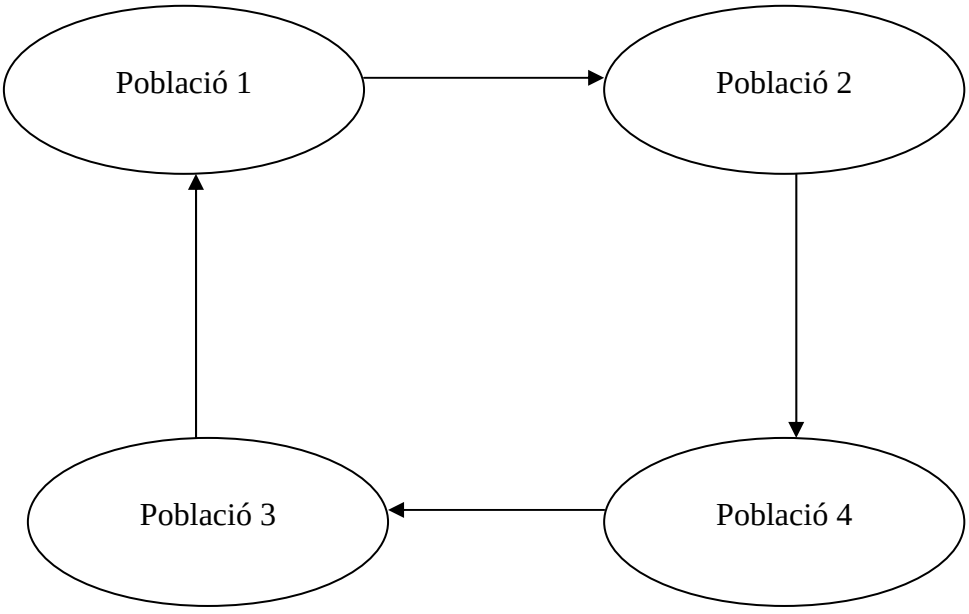


Figura 5.2.3.3: Comunicació de poblacions model anell

3.3. Recerca TABU

El mètode de cerca TABU és un algoritme que pot ser utilitzat per resoldre problemes d'optimització combinatòria.

Aquest sistema es basa en l'exploració dels veïns d'un gen, triant llavors el veí amb un VO més òptim, i explorant llavors el veïns d'aquest. Aquest procés es va repetint fins a un número determinat de vegades o fins que es compleixi una condició de termini.

Els veïns que no han estat escollits són afegits a la llista TABU per no tenir-los en compte ja que el seu VO no ens interessa. En cas que al generar un entorn ens aparegués un gen TABU se li aplica una mutació per tal de no repetir càlculs que no ens serveixen.

3.3.1. Elements de la recerca TABU

Per portar a la pràctica l'esquema anteriorment comentat i concretar-lo en un algoritme, s'ha d'especificar els següents elements:

- Una representació cromosòmica a manera de codi.
- Un gen inicial.
- Una mitjana d'avaluació.
- Un criteri de generació de l'entorn.
- Llista TABU.
- Mutació.
- Una o varies iteracions de la generació de l'entorn.

- Una condició de fi d'algoritme.

1. Codificació

Com passa amb els algoritmes genètics la primera decisió a prendre és com codifiquem la nostra informació (cromosomes) per tal que es pugui representar de manera senzilla i faciliti la implementació d'un algoritme de treball.

La codificació més comuna és a través de cadenes binàries tot i que es puguin utilitzar molts més mètodes com números reals, lletres, arbres..... La única condició important és que es pugui generar fàcilment l'entorn del gen que vulguem estudiar.

2. Gen inicial

A diferència dels algoritmes genètics en la recerca TABU no creem una població inicial sinó que a partir d'un sol i únic gen generem tota la població que estudiarem. Aquest gen és creat aleatòriament.

3. Mitjana d'avaluació

Per poder escollir quins gens són millors que altres necessitem tenir un valor que ens indiqui la qualitat d'aquells individu. A aquest valor se l'anomena valor objectiu, i és el paràmetre que ens interessa que sigui més gran o més petit depenen de la realitat del nostre problema.

4. Criteri de generació de l'entorn

Una vegada ja tenim un gen en curs, es genera aquest entorn. Hi ha moltes maneres de generar aquest entorn. Sí que és important establir com es vol que es generi l'entorn de la solució, i quina mida ha de tenir aquest entorn, doncs depenen el mètode la mida pot ser variable. A continuació citem un parell d'exemples tot i que n'hi ha molts més:

- **Intercanvi parell de cromosomes:** s'agafen parelles de cromosomes i s'intercanvien entre ells, és a dir, la solució actual i el veí generat són idèntics però el cromosoma 1 de la solució ocupa la posició 2 del veí i el cromosoma 2 ocupa la posició 1. Aquest intercanvi permet crear tants veïns com cromosomes tenim -1.

- **Intercanvi total de cromosomes:** s'agafa un cromosoma i s'intercanvia per un altre. És a dir en un gen de 3 cromosomes intercanviar el cromosoma 1 per el 2 i el 3. i el cromosoma 2 per el 3 obtenint així un entorn = factorial número de cromosomes.

5. Llista TABU

Amb l'entorn generat i avaluat es procedeix a organitzar les solucions en dues llistes. A la llista TABU i posarem tots els gens amb un valor objectiu més dolent, perquè així d'aquesta manera recopil·larem quins gens no ens interessa explorar doncs ja els hem avaluat i no ens ha servit el seu VO. El gens amb un VO serà afegit a la llista NO TABU i generarem el seu entorn per a poder repetir el procès.

6. Mutació

L'operador mutació s'aplica quan ens apareix un gen que pertany a la llista TABU. Es defineix com una variació elemental de les informacions contingudes en el codi genètic (habitualment, un canvi d'un cromosoma a l'altre).

Aquest operador permet, per una part, augmentar l'exploració en l'espai de cerca cap a nous entorns ja que produeix un increment de la diversitat poblacional, i, per altre part, evita l'aparició d'algoritmes bloquejats o de poblacions degenerades (poblacions iguals o similars).

Els models més usuals de mutació són els següents:

- Mutació tova: Consisteix en intercanviar, a l'atzar, dos gens consecutius en la seqüència.

- Mutació intercanvi: Consisteix en intercanviar, a l'atzar dos gens no consecutius en la seqüència.

- Mutació inversió: S'escullen a l'atzar dos punts de la seqüència. Els gens compresos entre els dos punts inverteixen el seu ordre.

7. Regeneració de l'entorn i condició de fi.

Després d'avaluar els veïns de la solució actual, i escollir el millor perquè passi a ser la nova solució s'ha de generar l'entorn d'aquest

Aquest nou entorn servirà per escollir la nova solució que adoptarem i per afegir individus a la llista TABU tornant-se a repetir el procés que forma la selecció, implementació llistes TABU i no TABU, mutació (en cas de gens TABU) i regeneració de l'entorn, sempre amb l'objectiu de trobar individus més aptes a cada pas.

El número d'iteracions o regeneracions dependrà principalment de la complexitat del problema. A més número de regeneracions, més gran serà la probabilitat de trobar una solució factible o propera a l'òptim final del procés, però s'ha de tenir en compte que, si aquest número és molt gran, el cost computacional pot ser molt elevat. Per això, en ocasions el número de regeneracions depèn directament del temps d'execució del problema. Una altra possibilitat consisteix en parar l'algoritme quan aquest arribi a l'òptim (sempre que aquest sigui conegut), o bé quan es consideri que la solució en curs és satisfactòria (respecte a l'òptim conegut). O simplement observar el VO obtingut en cada iteració i en cas de no millorar aquest paràmetre respecte un valor absolut durant un número d'iteracions parar el procés.

3.3.2. Problemes de TABU

La recerca TABU tendeix a millorar el valor objectiu però el principal inconvenient és que al fer generacions de l'entorn d'un gen no explora una gran diversitat de cromosomes i cau en un òptim local.

Tot i que la generació de la llista TABU ajuda a no caure en òptims locals és molt difícil amb aquest mètode trobar l'òptim global amb poques iteracions, i recordem que l'objectiu dels mètodes heurístics és la reducció del nombre d'iteracions a realitzar.

4. Implementació dels algoritmes de resolució.

Per començar es podria dir que la nostra aplicació consta de dues parts, o més ben dit de dos mètodes heurístics que són:

- Algoritmes genètics: escollint aquesta opció buscarem una solució òptima utilitzant els algoritmes genètics i podrem escollir quines polítiques intervindran així com el número d'iteracions, la mida de la població i el nombre màxim d'igualtats entre la mitjana d'una població i la mitjana de la seva població descendent.

- Cerca Tabu: Amb aquesta opció podrem analitzar com funciona la cerca tabu i podrem escollir quantes iteracions volem realitzar.

Tot i que aquest dos mètodes són diferents tots dos comparteixen la mateixa idea sobre l'exploració d'entorns. És per aquest motiu que la base de dades que utilitzem per tenir enregistrada la informació sobre els productes, les rutes i les màquines ens serveix per a tots dos sistemes.

A la figura 4.1 mostrem quan fem la selecció en el programa.

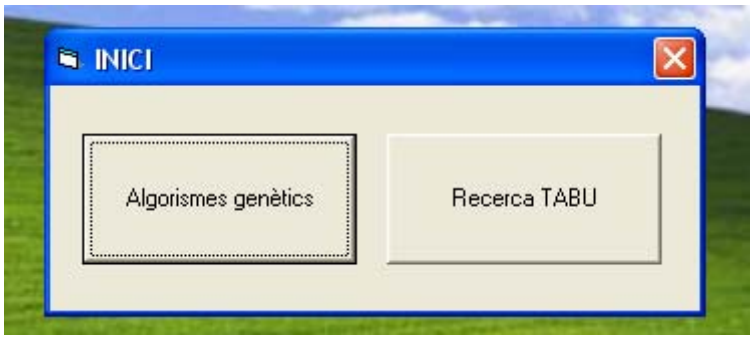


Figura 4.1: Selecció del métode

4.1. Pretractament de dades

Per poder llegir correctament les dades hem decidit utilitzar una base de dades en format access 97 degut a la comptabilitat que presenta amb el visual basic 6 que és la versió que hem utilitzat per fer la nostra eina.

Tenim la base de dades fragmentada en 3 taules tal com indica la figura 4.1.1:

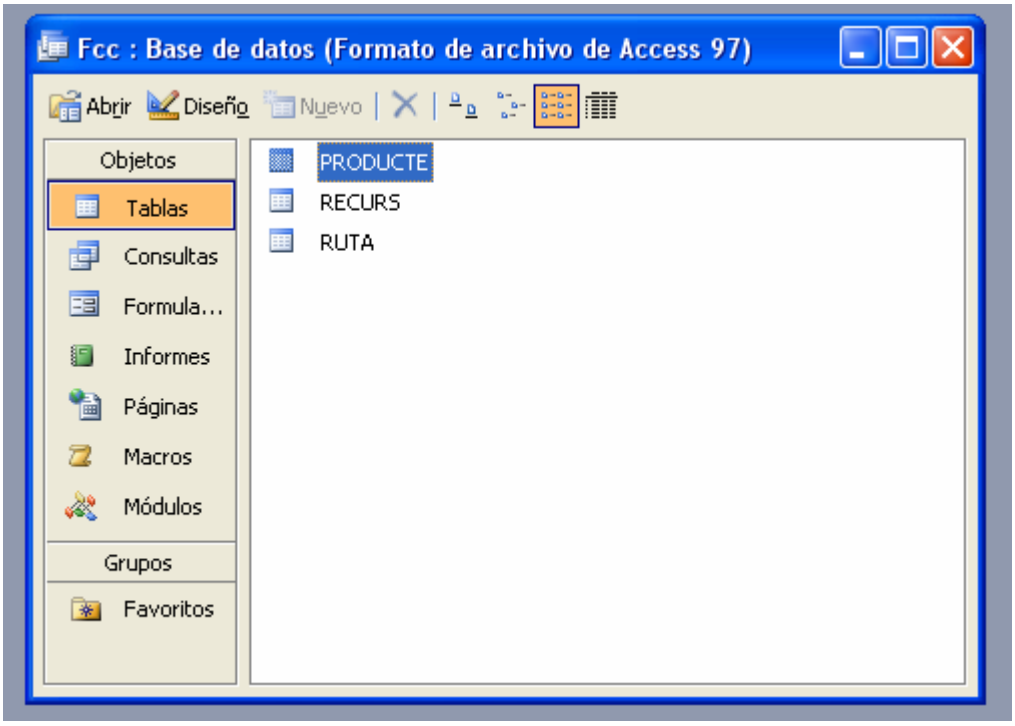


Figura 4.1.1: Base de dades

- Producte: En aquesta taula hi tenim el codi, el nom i la descripció dels productes que podem seqüenciar.(fig 4.1.2)

PRODUCTE : Tabla

	CodiProd	Nom Prod	Descripció Proc
+	1	Brida	Petit
+	2	Anell presoner	Gran
+	3	Suport inferior	Mitjà
+	4	Centrador de m	XS
+	5	Distribuïdor refri	XXS
*	(Autonumérico)		

Figura 4.1.2: Taula producte

- Recurs: En aquesta taula hi tenim el codi, el nom, tipus i la marca de cada màquina que tenim per a poder seqüenciar.(fig 4.1.3)

RECURS : Tabla

	CodiRecurs	Nom Recurs	Tipus Recurs	Marca Recurs
+	1	Fresa-01	Fresa	Kraft-T800
+	2	Taladre-01	Taladre	Weiler-VO75
+	3	Torn-01	Torn	Pinacho-TR7
*	(Autonumérico)			

Figura 4.1.3: Taula recurs

- Ruta: En aquesta taula hi tenim totes les rutes codificades de la següent manera.(fig 4.1.4)

RUTA : Tabla

CodiProducte	CodiRuta	OrdreRuta	CodiRecurs	Temps
1	1	1	1	11
1	1	2	3	11
1	1	3	2	24
1	2	1	1	11
1	2	2	3	10
1	2	3	2	12
1	2	4	3	5
1	2	5	2	16
1	3	1	1	11
1	3	2	2	16
1	3	3	3	11
1	3	4	2	8

Figura 4.1.4: Taula ruta

  - Codi Producte, aquí hi tenim el codi del producte.

- Codi Ruta, aquesta casella ens indica el número de ruta.

- Ordre Ruta, aquesta casella ens indica el número d'operació.

- Codi Recurs, aquesta casella ens indica la màquina que realitza l'operació.

- Temps, aquesta casella ens indica el temps que necessita l'operació en concret.

Amb aquesta codificació obtenim diferents variables que ens serveixen per començar a treballar amb la nostra eina informàtica.

De la taula recursos n'obtenim el número de màquines que tenim al taller, i de la taula ruta n'obtenim el nombre màxim de rutes de cada producte a més a més d'utilitzar-la després per a l'algorisme de càrrega vertical per obtenir el valor fitness de cada gen.

La simplicitat d'aquesta base de dades permet afegir productes i màquines sense complexitat només hem d'anar a la taula ruta i afegir seguint la mateixa codificació tants productes o màquines com vulguem. Les taules estan enllaçades entre elles per tant les altres taules ja s'actualitzen automàticament.

En cas de voler eliminar algun producte i/o recurs hem de tenir en compte que hem de mantenir la seqüenciació iniciada. Per tant si tenim 4 màquines sempre les hem de tenir codificades de l'1 al 4, mai hem de tenir un interval sense màquina assignada.

4.2. Codificació del problema

Una vegada codificada la informació dels productes, les seves rutes i els temps que necessita cada operació d'aquestes rutes, a la base de dades. Hem de recodificar aquesta informació per tal de poder-hi treballar.

Per a poder treballar amb les dades recollides hem creat uns vectors que anomenarem a partir d'ara gens. Aquests gens tindran un nombre definit d'elements que

anomenarem cromosomes. En cada cromosoma del gen hi tenim assignat un producte de la base de dades. El número de cromosomes bé definit de la següent manera: un primer cromosoma hi dipositem un nom propi i un últim cromosoma on hi dipositem el temps que necessita la seqüència, els cromosomes del mig (que van de 2 a 10 doncs és el rang de productes que podem seqüenciar) contenen la ruta que és d'aplicació al producte en qüestió. Per tant la mida dels gens va de 4 cromosomes (2 productes+nom propi+valor objectiu) fins a 12 (10 productes+nom propi+valor objectiu). El nom propi ocupa la posició 0 del cromosoma (per tant tenim de 0 a 11 cromosomes).

El número de gens que crearem serà diferent en cada algoritme, doncs en l'algoritme genètic crearem (mida de la població * el número d'iteracions) gens, i en la recerca TABU crearem (número d'iteracions*(nombre productes-1)) gens. Aquests gens tindran tants cromosomes com productes hi hagi a la seqüència (recordem amb un màxim de 10). A més d'aquests cromosomes que contenen la ruta del producte en qüestió, el gen disposa de dos elements més un on hi calculem el valor objectiu de la seqüencia i un primer on posem un nom propi al gen.

A la figura 4.2.1 podem observar l'estructura d'un gen de 10 productes. Podem observar com la seqüència del gen número 3 on P1/R6, P2/R2, P3/R1, P4/R4, P5/R7, P6/R10, P7/R4, P8/R7, P9/R9, P10/R1 té un valor objectiu de 148 unitats de temps. És a dir la seqüència de fabricació del gen 3 un cop hi apliquem l'algorisme de càrrega vertical obtenim un temps de 148.

Cromosoma 2, producte 5 de la base de dades, ruta1.

Nom	P1	P5	P4	P2	P2	P3	P5	P1	P3	P4	VO
1	7	1	2	1	6	10	3	5	7	1	152
2	2	3	2	6	5	11	1	5	6	2	166
3	6	2	1	4	7	10	4	7	9	1	148

Figura 4.2.1: Exemple codificació d'un gen de 10 cromosomes

A partir d'aquí ja podem començar a aplicar diferents mètodes heurístics, en el nostre cas aplicarem algoritmes genètics i recerca TABU.

4.3. Algoritmes genètics

El nostre algorisme treballa seguint moltes de les regles esmentades en l'apartat 5.3, no obstant algunes les hem hagut d'adaptar a la realitat del nostre problema. Tot seguit expliquem com les hem adaptat de manera que obtinguem una aproximació dels mateixos efectes.

4.3.1. Població inicial

La població inicial és generada a partir d'una funció aleatòria, és a dir cada cromosoma és escollit aleatòriament tenint en compte que hi ha un màxim de rutes que pot suportar cada producte, és a dir cada producte té un número determinat de rutes, per tant si un producte té 4 rutes, el seu cromosoma serà un valor aleatori entre 1 i 4.

En un principi havíem creat una funció que no prenia una població inicial aleatòria sinó que prenia un valor determinat de rutes. Aquesta funció agafava per a cada producte les rutes que tenien més i menys operacions i les que la suma dels temps era més i menys gran.

El problema que presentava aquesta funció és que en la duplicitat de cromosomes (una ruta amb les mateixes característiques que una altra, per exemple mateix nombre de rutes), i amb seqüències llargues de 7 productes o més, la població inicial es disparava i no podíem assegurar una varietat de cromosomes. Per tant teníem un nombre molt elevat de gens i per assegurar una varietat hauríem hagut d'afegir gens aleatoris. Cosa que feia augmentar molt el nombre de gens a estudiar desvirtuant així l'utilització d'un algorisme genètic per evitar l'estudi d'un gran nombre de combinacions.

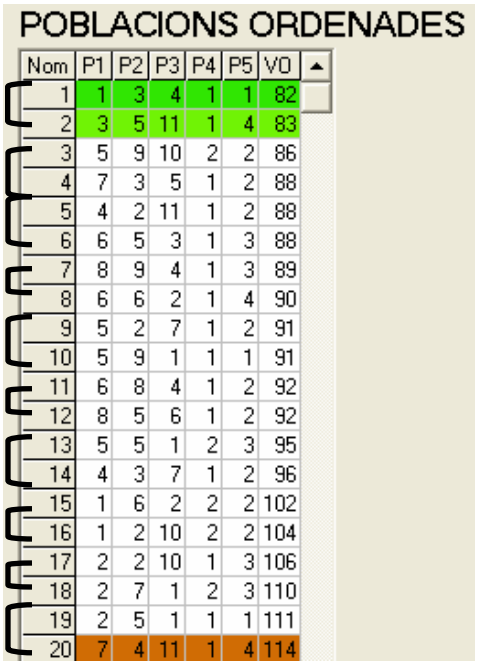
És per aquest motiu que hem decidit crear la població inicial totalment aleatòria. Assegurant així una major diversitat genètica tot i tenir en alguns casos menys gens.

4.3.2. Selecció dels gens

Una vegada tenim la població ordenada per el seu valor fitness hem de decidir com volem que es creuin els gens entre ells. En la nostra aplicació informàtica hi hem incorporat 3 sistemes de selecció de gens. A continuació els descrivim:

1. Selecció FIT-FIT

Amb aquesta opció, sobre la població ordenada pel seu fitness, creuem el primer gen amb el segon, és a dir el més fort amb el segon més fort. I així generem dos descendents, d'aquesta manera podem dir que creuem el primer amb el segon, el tercer amb el quart i així successivament fins al final de la població. La figura 4.3.2.1 ens mostra com seleccionem els progenitors.

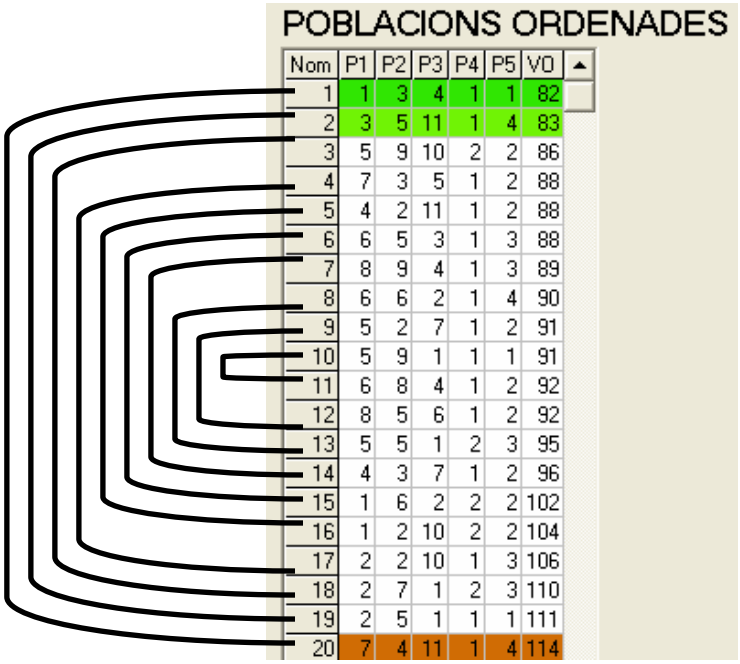


Nom	P1	P2	P3	P4	P5	VO
1	1	3	4	1	1	82
2	3	5	11	1	4	83
3	5	9	10	2	2	86
4	7	3	5	1	2	88
5	4	2	11	1	2	88
6	6	5	3	1	3	88
7	8	9	4	1	3	89
8	6	6	2	1	4	90
9	5	2	7	1	2	91
10	5	9	1	1	1	91
11	6	8	4	1	2	92
12	8	5	6	1	2	92
13	5	5	1	2	3	95
14	4	3	7	1	2	96
15	1	6	2	2	2	102
16	1	2	10	2	2	104
17	2	2	10	1	3	106
18	2	7	1	2	3	110
19	2	5	1	1	1	111
20	7	4	11	1	4	114

Figura 4.3.2.1: Selecció de creuaments amb FIT-FIT

2. Selecció FIT-WEAK

Amb aquesta opció, creuem el primer amb l'últim és a dir el gen més fort amb el més dèbil. Llavors el segon amb el penúltim i així successivament fins a arribar al mig de la població. En cada creuament obtenim dos descendents tal i com mostra figura 4.3.2.2.



POBLACIONS ORDENADES

Nom	P1	P2	P3	P4	P5	VO
1	1	3	4	1	1	82
2	3	5	11	1	4	83
3	5	9	10	2	2	86
4	7	3	5	1	2	88
5	4	2	11	1	2	88
6	6	5	3	1	3	88
7	8	9	4	1	3	89
8	6	6	2	1	4	90
9	5	2	7	1	2	91
10	5	9	1	1	1	91
11	6	8	4	1	2	92
12	8	5	6	1	2	92
13	5	5	1	2	3	95
14	4	3	7	1	2	96
15	1	6	2	2	2	102
16	1	2	10	2	2	104
17	2	2	10	1	3	106
18	2	7	1	2	3	110
19	2	5	1	1	1	111
20	7	4	11	1	4	114

Figura 4.3.2.2: Selecció de creuaments amb FIT-WEAK

3. Selecció Ruleta

Aquesta és una de les funcions que hem adaptat al nostre problema. Aquesta opció creua cada cromosoma de la població amb un dels 5 primers essent el gen més fort el que té més probabilitat de ser escollit. En aquesta funció de cada creuament només n'obtenim un descendent i fem el procés per a tots i cadascun dels gens.

4.3.3. Selecció del sistema de creuament

Una vegada ja tenim seleccionades les parelles de gens a creuar tenim varies maneres de fer-ho. En la nostra eina hi hem implementat tres maneres diferents de fer-ho:

1. Uniforme

Aquesta opció agafa cromosoma per cromosoma dels progenitors i aleatoriament els diposita en els descendents. Si el cromosoma 1 del descendent 1 pren el valor del progenitor 1 el descendent 2 pendrà el valor del progenitor 2, pel cromosoma 2 tornem a fer el mateix procés podent obtenir un gen amb els cromosomes 1, 2 i 5 del progenitor 1 i la resta del progenitor 2. En la figura 4.3.3.1 veiem com funciona.

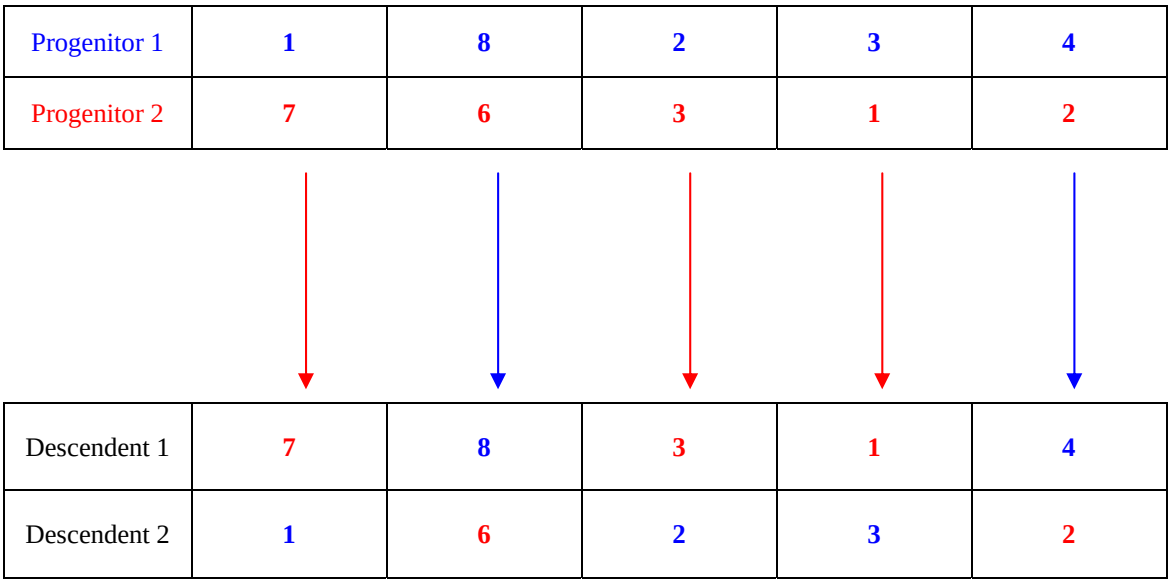


Figura 4.3.3.1:Creuament uniforme

2. Bloc d'un punt creuament

L'opció un punt de creuament parteix els cromosomes per la meitat i cada descendent rep una meitat de cada progenitor. És a dir el descendent rep la primera part d'un progenitor i la segona part d'un altre tal i com mostra la figura 4.3.3.2.

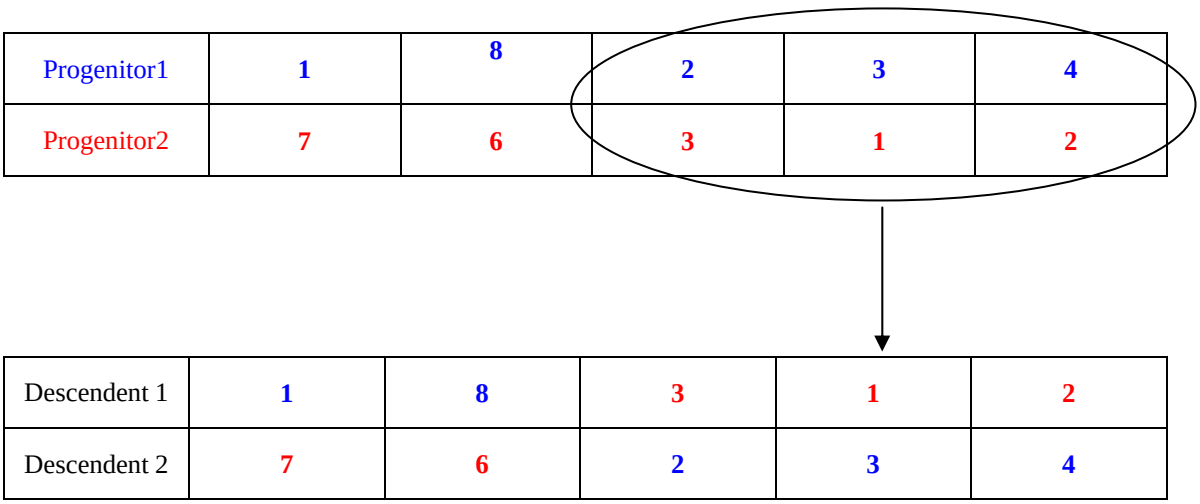
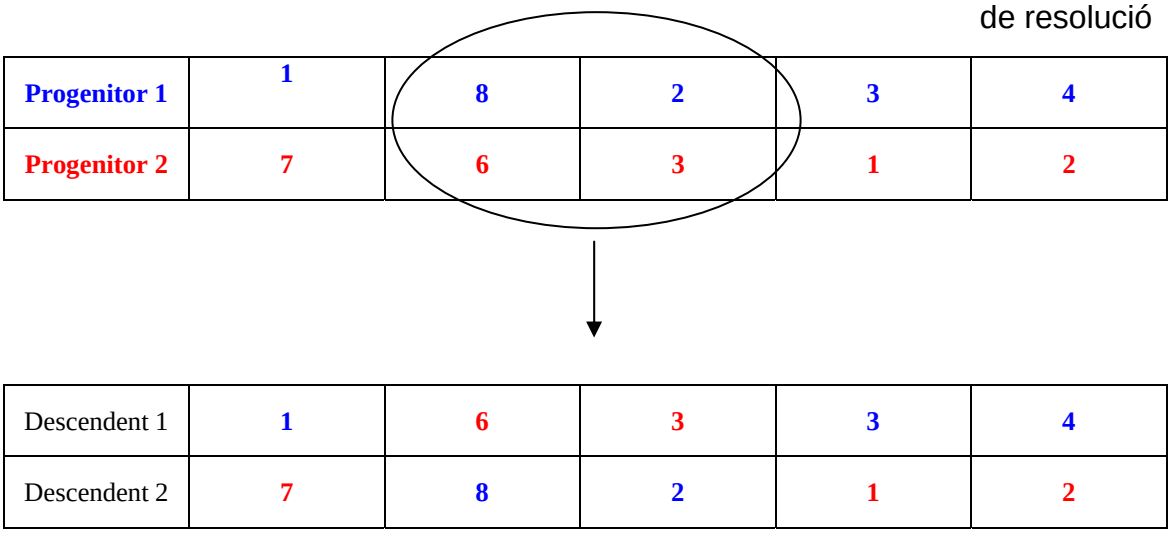


Figura 4.3.3.2: Creuament amb bloc amb un punt de creuament

3. Bloc de dos punts de creuament

L'opció dos punts de creuament parteix els cromosomes amb tres parts i cada descendent rep la primera i tercera part d'un progenitor i la segona de l'altra, tal com mostra la figura 4.3.3.



Progenitor 1	**1**	**8**	**2**	**3**	**4**
Progenitor 2	**7**	**6**	**3**	**1**	**2**
Descendent 1	**1**	**6**	**3**	**3**	**4**
Descendent 2	**7**	**8**	**2**	**1**	**2**

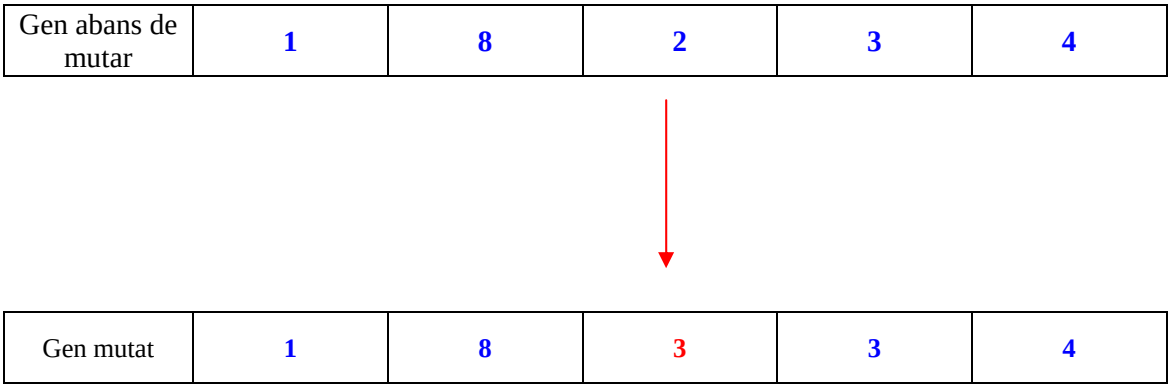
Figura 4.3.3: Creuament amb bloc amb dos punts de creuament

4.3.4. Mutació

La mutació és una característica molt important dins de qualsevol algorisme genètic. En la nostra aplicació també ho és i en aquest cas tenim tres paràmetres que ens indiquen com fem la mutació a més a més de poder escollir fins a quin punt volem que els gens mutin o no. Tenim 3 opcions a escollir pel que fa a la mutació:

1. *Només un element*

Aquesta opció fa que només pugui mutar un cromosoma de tot el gen. És una bona opció si el que pretenem és estudiar com es comporta l'algorisme controlant la mutació tal com mostra la figura 4.3.4.1



Gen abans de mutar	**1**	**8**	**2**	**3**	**4**
Gen mutat	**1**	**8**	**3**	**3**	**4**

Figura 4.3.4.1: Mutació d'un gen només per un element

2. *Mutació per bloc*

Aquesta opció pren un bloc dins el gen i el fa mutar tot ell. És a dir en un gen format per 7 cromosomes hi poden mutar els cromosomes 2,3,4. Aquesta opció fa que en cas de produir-se la mutació el gen canvia molt significativament, també es caracteritza perquè la posició dels gens que muten és consecutiva, tal i com mostra la figura 4.3.4.2

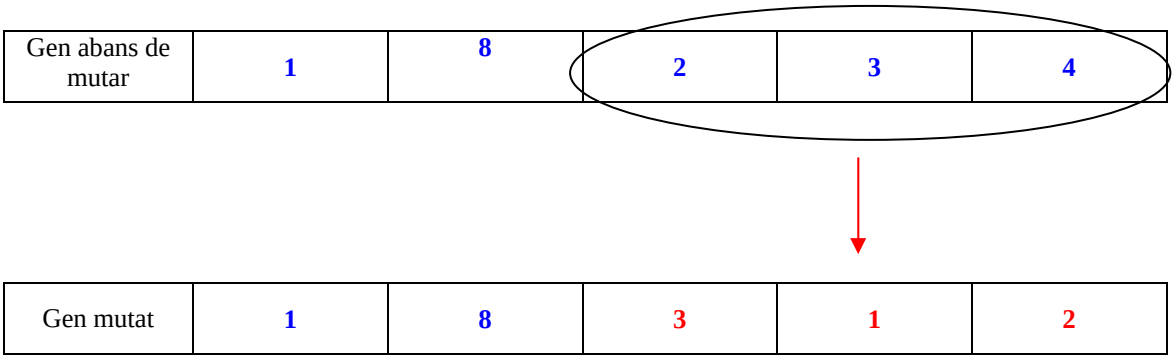


Figura 4.3.4.2: Mutació d'un gen amb bloc

3. *Mutació uniforme*

Aquesta opció igual que en el creuament uniforme estudia cromosoma per cromosoma i aleatòriament el cromosoma en qüestió muta. D'aquesta manera podem tenir gens on muten els cromosomes 1,4,5, és a dir la posició dels cromosomes mutats no té perquè ser consecutiva. Com més alta sigui la probabilitat de mutació més cromosomes mutaran. La figura 4.3.4.3 n'és un exemple.

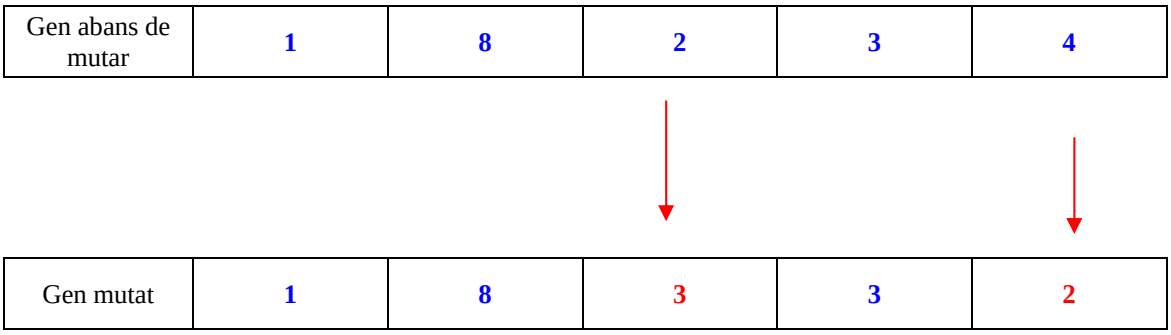


Figura 4.3.4.3: Mutació d'un gen uniformement

4.3.5. Nivell de mutació

Finalment també podem escollir amb quina probabilitat volem que la condició de mutar esdevingui certa. En aquest cas tenim 3 opcions: alta, moderada i baixa. Si escollim alta els gens mutaran molt i el resultat serà que en cada població perdem informació de l'anterior. Una mutació moderada farà que els gens no tots mutin cosa que farà que en els creuaments tinguem en compte la informació genètica de les anteriors poblacions i la mutació ens farà explorar nous entorns. I finalment una mutació baixa farà que ràpidament localitzem un òptim local tot i que potser no serà gaire bo.

4.3.6. Funcionament general de l'algoritme genètic

En primer lloc explicarem les dues pantalles del programa explicant les diferents opcions que podem escollir.

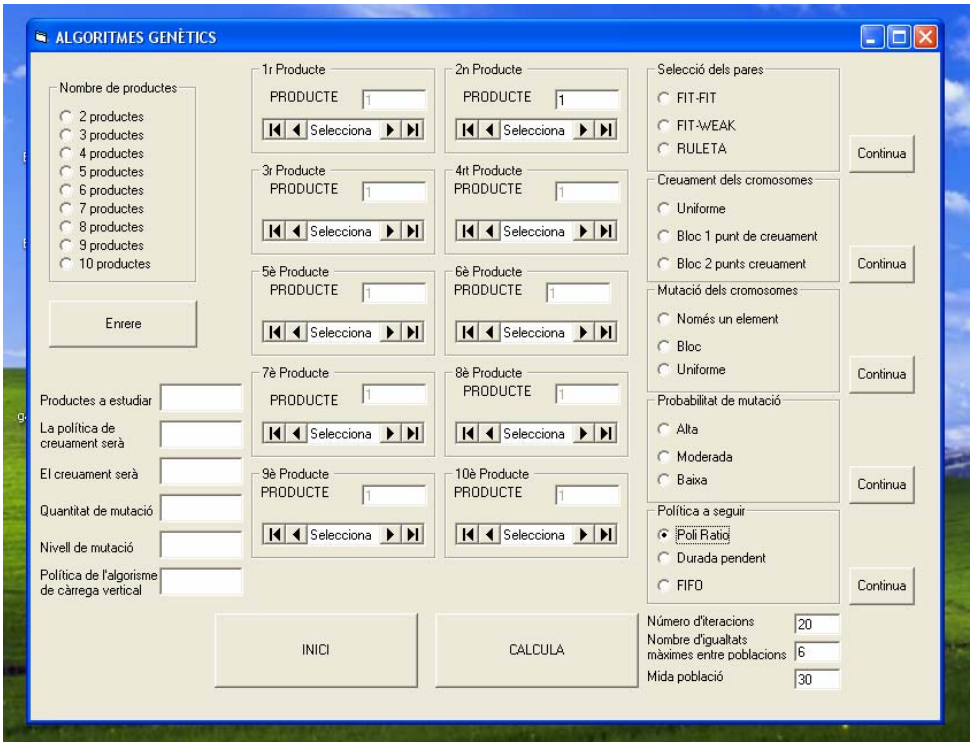


Figura 4.3.6.1:pantalla selecció algoritmes genètics

Aquesta és la primera pantalla (fig. 4.3.6.1) que ens apareix quan triem l'opció d'algoritmes genètics. A la part de dalt a l'esquerra hem de triar la quantitat de productes a seqüenciar i en els diferents requadres triarem quins productes seran. Les opcions de la dreta fan referència a les diferents normes de selecció, creuament i mutació dels gens així com també la política que seguirà algoritme de càrrega vertical.

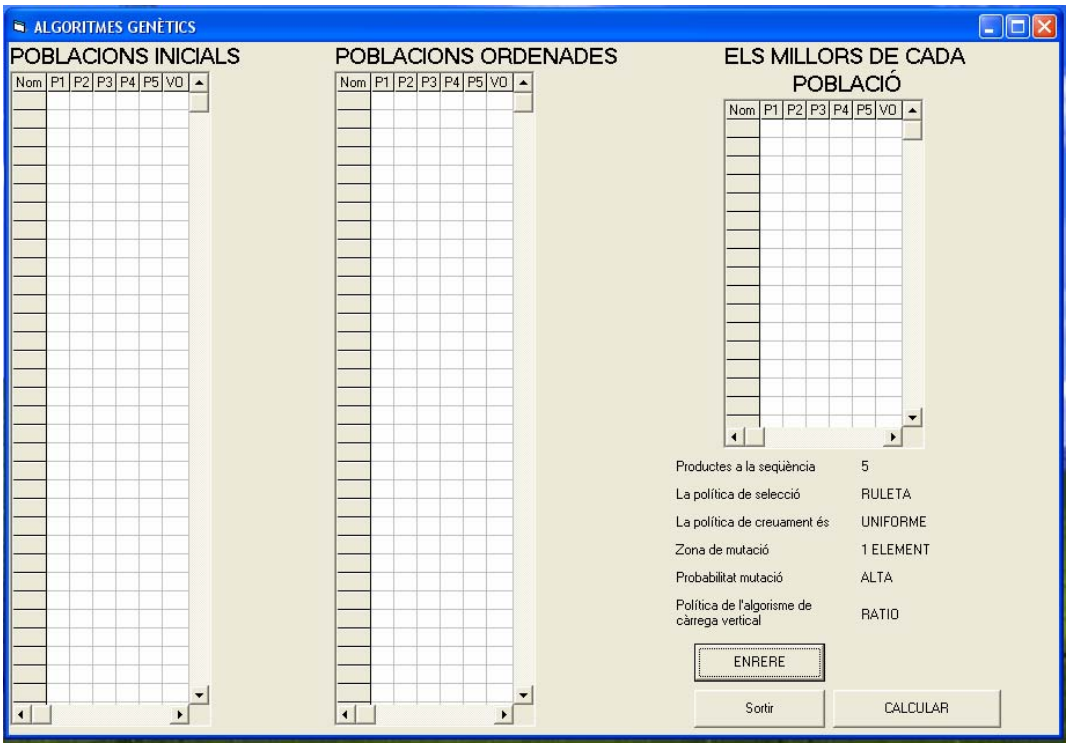
A baix a la dreta tenim 3 caselles amb un valor predeterminat però tot i així el podem modificar. Aquests valors es refereixen al número d'iteracions que farà el programa, la mida de les poblacions i finalment un fre que el que fa és comparar les mitjanes de les poblacions i si es compleix que en sis vegades tenen el mateix valor atura l'algorisme per evitar operacions innecessàries doncs haurem caigut en un òptim local.

A continuació expliquem de manera senzilla com funciona la nostra eina informàtica, i com queda reflexat amb la segona pantalla.

A la columna de l'esquerra i anirem recopilant els gens sense ordenar, a la columna del mig i col·locarem la població inicial ordenada, i finalment a la columna de la dreta i dipositem el gen amb un millor fitness (en el nostre cas un fitness més baix). A sota la

columna de la dreta tenim un conjunt de dades que ens indica quina combinació hem triat.

Una vegada tenim la població ordenada pel seu fitness utilitzem les regles per creuar i mutar la població hi ho dipositem a continuació en la primera columna obtenint així una altra població inicial desordenada que l'ordenarem i la col·locarem a la segona columna n'extreurem el millor i tornarem a creuar i mutar, així tantes vegades com iteracions haguem escollit o fins que la condició de fre no es compleixi. A la figura 4.3.6.2 podem veure la pantalla abans de començar el procés.



4.3.6.2: Pantalla resultats algoritmes genètics

A continuació expliquem esquemàticament com funciona:

1. Generem una població inicial.

2. Busquem el valor objectiu (fitness) de cada gen d'aquesta població.

3. Ordenem la població inicial segons el valor objectiu (de més petit a més gran).

4. Seleccionem els gens a creuar de la població utilitzant una de les diferents polítiques de creuament.

5. Creuem els gens seleccionats amb un dels tres mètodes de creuament, i així obtenim dos fills per a cada parella de gens.

6. Mutem els descendents de la població inicial segons els diferents paràmetres que haguem escollit.

7. Amb la descendència de la població, una vegada mutada, tornem a ordenar-la i a repetir tot el procès, tantes vegades com així ho haguem indicat. La fig 4.3.6.3 ens mostra el resultat obtingut.

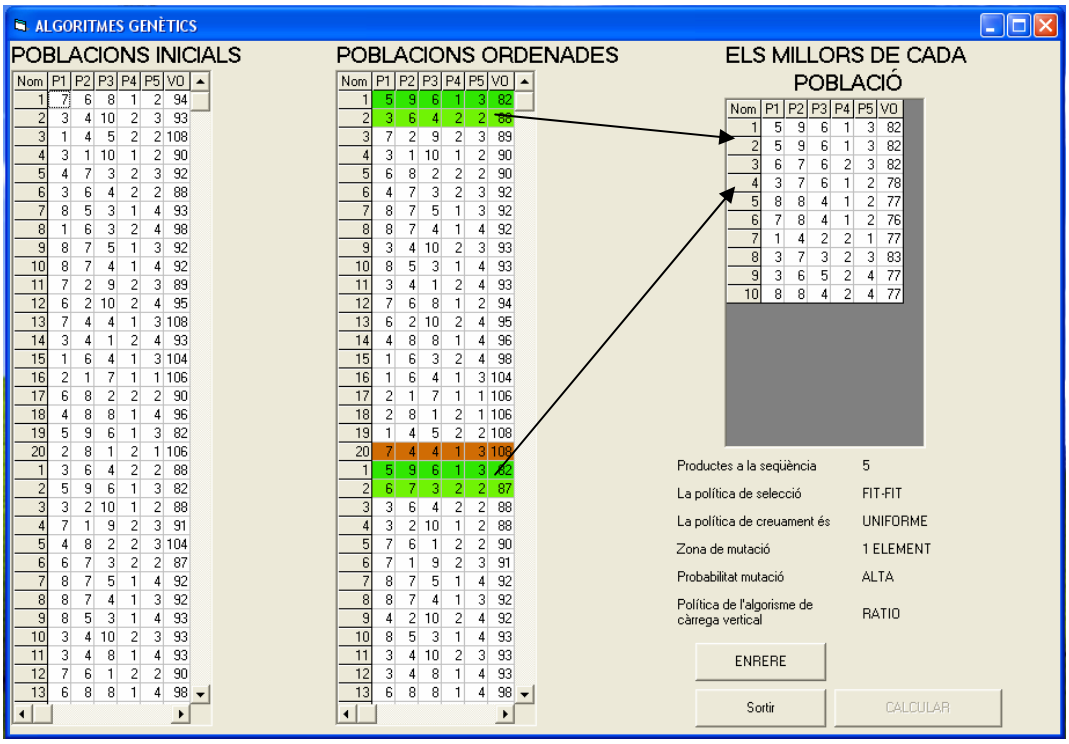


Figura 4.3.6.3: Pantalla resultats algoritmes genètics amb resultats

4.4. Cerca TABU

La cerca Tabu es diferencia especialment dels algoritmes genètics perquè no disposen d'una població inicial d'on n'extreuen el mateix nombre de descendents o menys,aprofitant la informació genètica, sinó que d'un gen aleatori va reproduint el seu entorn i va obtenint una població on es va quedant amb el millor fitness.

4.4.1. Generar entorn

La cerca TABU no disposa de creuaments l'únic que fem és a partir d'un gen crear-ne tants com productes tenim menys un. En el cas que un cromosoma hagi d'ocupar una posició on el producte no pugui suportar el número de ruta aquest s'elimina i es genera una ruta aleatòria dins el rang. És a dir si segons la funció genera_entorn al cromosoma 3 hi hagués d'anar la ruta 8 però el producte que ocupa la posició 3 només suporta 4 rutes llavors generarem una ruta entre 1 i 4.

La figura 4.4.1 es mostra com generem la població.

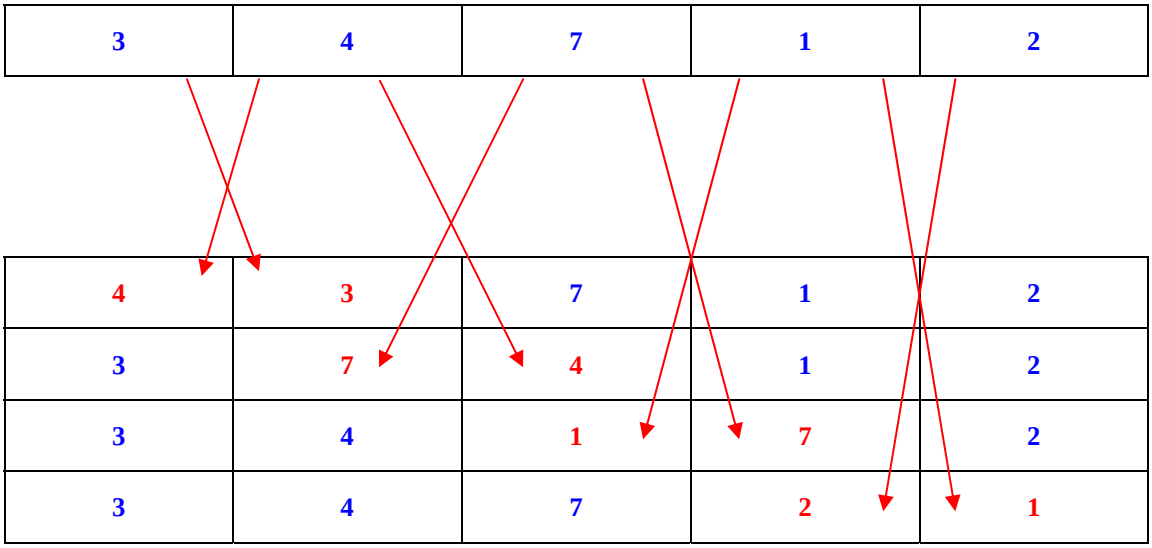


Figura 4.4.1: Generació de l'entorn a partir d'un gen

4.4.2. Mutació en TABU

En el mètode TABU la mutació realment dóna un salt qualitatiu al fitness de la població. Si no disposéssim d'aquesta funció en la majoria d'ocasions cauríem en una convergència on en cada iteració estudiaríem les mateixes combinacions o si més no molt semblant.

A continuació procedirem a explicar les dues pantalles de què disposa la nostra eina informàtica per desenvolupar la cerca Tabu, i les funcions que hi treballen.

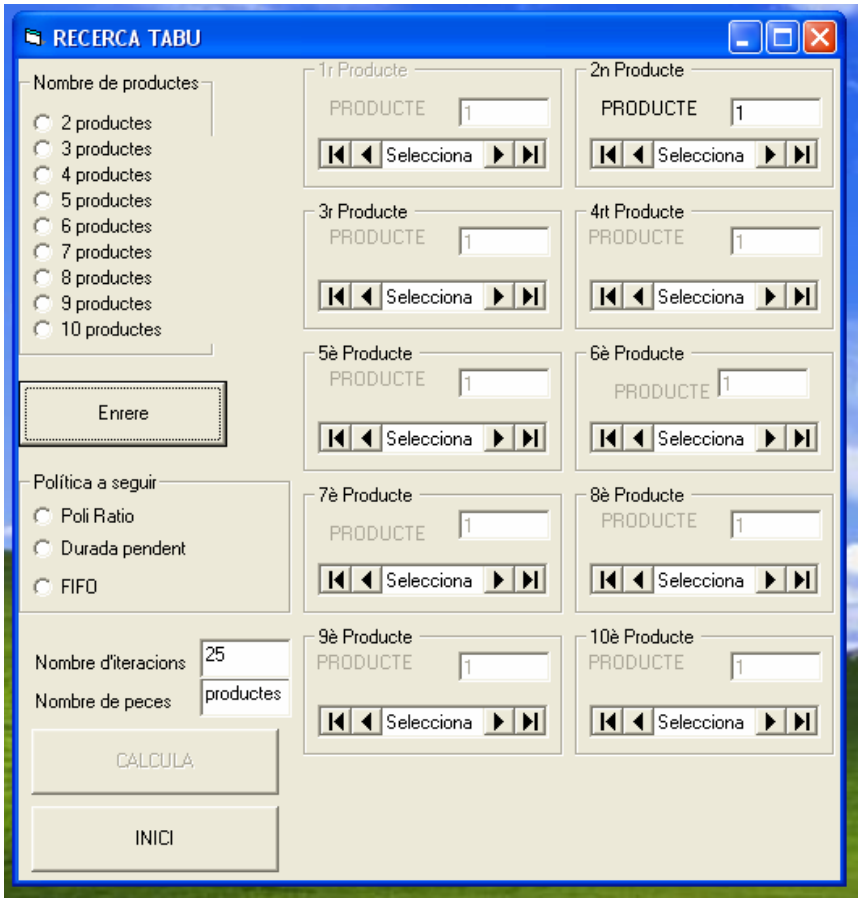


Figura 4.4.2.1: Pantalla selecció en TABU

A la primera pantalla (fig. 4.4.2.1) podem escollir la quantitat de productes a seqüenciar i quins productes han de ser. A més a més també podem escollir el número d'iteracions que farà el programa. Per defecte el número d'iteracions serà 25 tot i que el podem modificar al nostre gust. També de la mateixa manera que amb l'algoritme genètic també podem escollir la política de càrrega (FIFO, durada pendent i Ratio). Una vegada ho tinguem tot a punt el botó calcula estarà disponible per començar.

La segona pantalla és bastant semblant a la dels algoritmes genètics, també disposa de 3 columnes on a cada columna hi posem una sèrie de gens per tal d'anar observant com funciona el procés. Un botó per començar el càlcul, un altre per retrocedir i finalment un per tancar el programa.

El gen subratllat en verd és el gen que estudiem, a partir d'aquest gen creem una petita població (en aquest cas 1,2,3 i 4), avaluem els seus fitness i llavors procedim a

estudiar el millor de la població i els altres a considerar-los estats TABU doncs ja han estat avaluats i hem trobat resultats millors.

Per tant tornem a generar una població partint de la millor solució anterior, però en aquesta ocasió si en la creació de l'entorn del gen en qüestio trobem un gen que ja ha estat estudiat llavors aquest gen rep una mutació de 3 dels seus cromosomes, el primer, l'últim i el del mig. Utilitzant aquesta mutació en cas d'obtenir un gen amb bon fitness al generar l'entorn enriquirem molt la diversitat. Si es dona el cas que un gen rep la mutació aquest (sense mutar) és afegit a la tercera llista per així poder observar i estudiar tot el procés. A la figura 4.4.2.2 veiem com comença el procés.

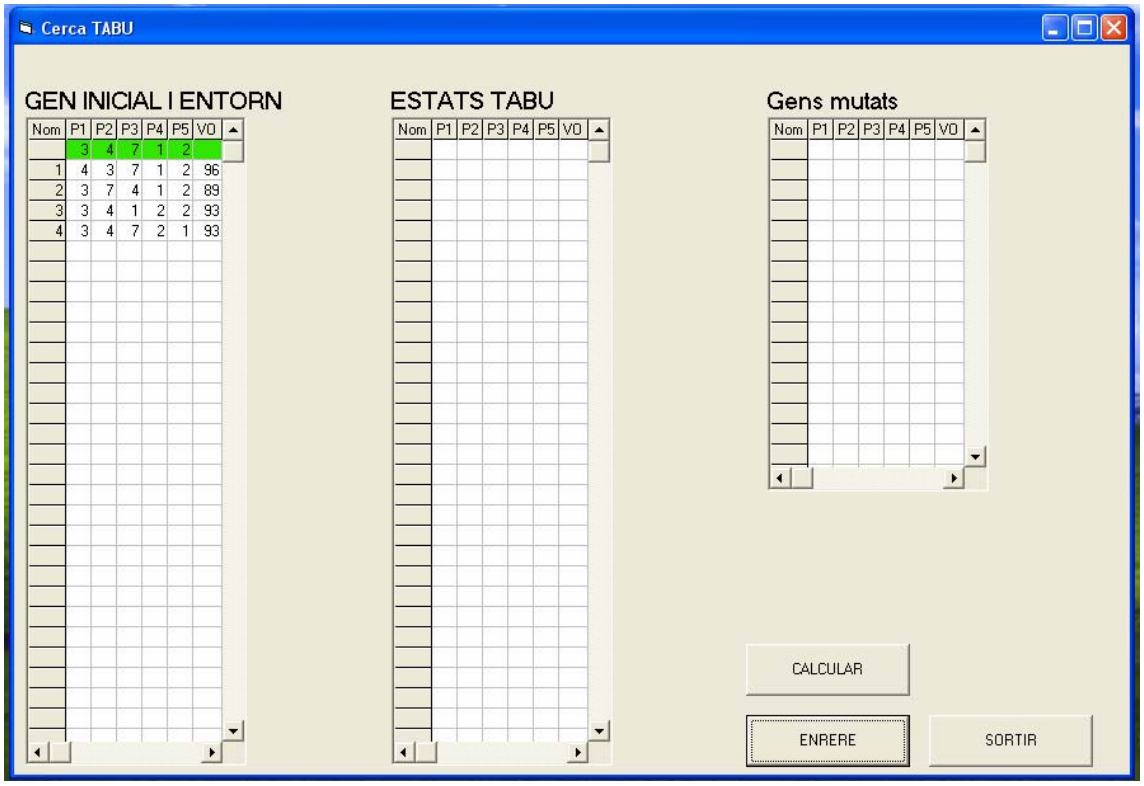


Figura 4.4.2.2: Pantalla resultats en TABU

Tot seguit explicarem esquemàticament com funciona la cerca TABU

1. Generem aleatòriament un gen.

2. D'aquest gen en n'extraiem una població.

3. D'aquesta població estudiem el descendent amb més fitness i la resta els afegim a la llista d'estats TABU que són els que ja hem explorat i no ens serveixen.

4. Amb el gen amb millor fitness repetim el procés i obtenim una nova població.

5. En el cas que un gen de la població tingui els mateixos cromosomes que un gen de l'estat TABU, mutem aquest gen i l'afegim a la tercera llista.

6. Repetim el procés tantes vegades com iteracions haguem seleccionat.(fig. 4.4.2.3)

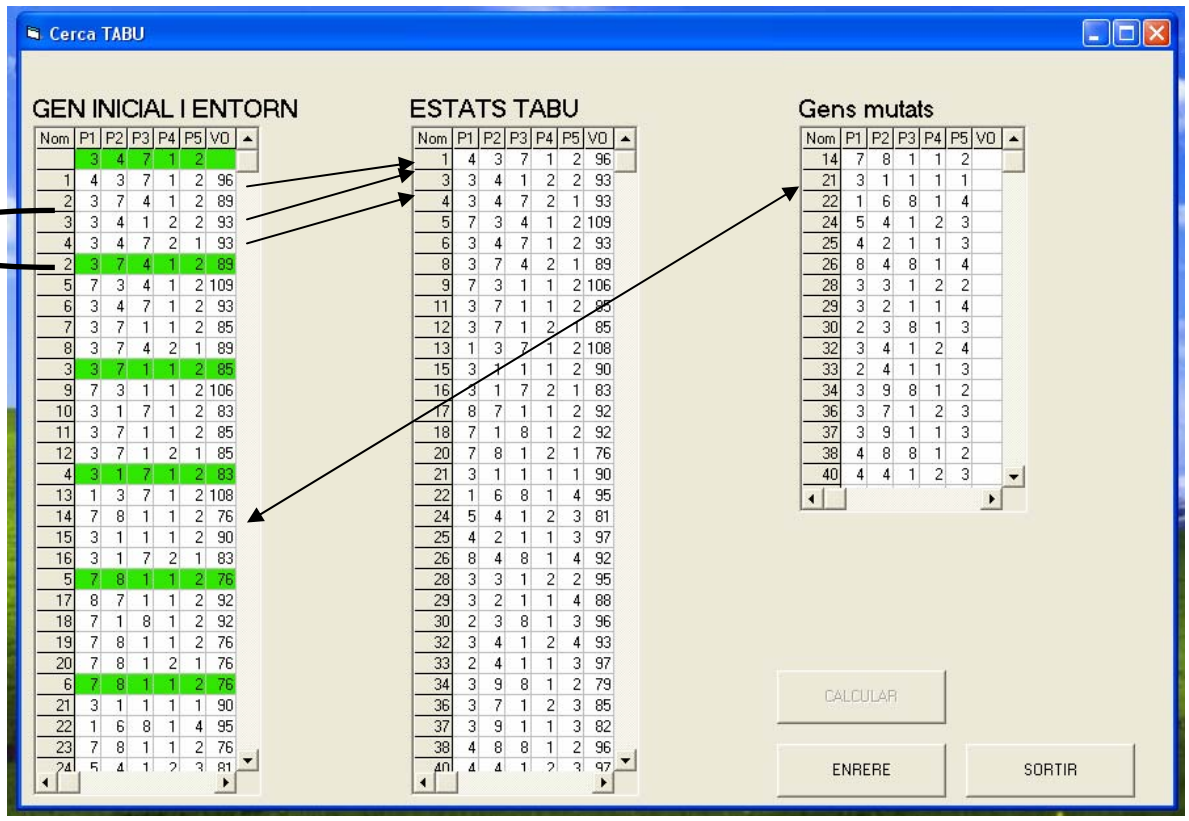


Figura 4.4.2.3: Pantalla resultats TABU amb resultats

5. Funcions i variables del programa

Degut a que les dues opcions (algorismes genètics i cerca Tabu) no funcionen alhora, tenim tot una sèrie de variables que les podem utilitzar en un cas i en l'altre tot i que no facin exactament la mateixa funció.

A continuació no comentarem totes les funcions i variables doncs tenim petites funcions que no són de rellevància per a tot el procès i algunes variables senzillament fan de contador per les diferents iteracions. Tot i així a l'annex B disposem de tot el codi informàtic de l'aplicació.

5.1. Variables de tipus taula

En aquest algorisme utilitzem varies variables de tipus taula la majoria són per guardar informació sobre cada producte que tenim dins la seqüència.

5.1.1. Variable w()

Aquestes variables guarden el codi de cada peça relacionant així la posició del cromosoma amb el producte que conté. Prenen el valor dels diferents formularis.

5.1.2. Variable max()

En aquestes variables guardem el número màxim de rutes que pot suportar cada producte, aquesta informació l'obtenim utilitzant la funció minmax.

5.1.3. Variable g()

Aquestes variables són utilitzades per generar números aleatòris.

5.1.4. Variable cromo()

Aquesta variable l'utilitzem sempre per guardar els gens sense ordenar. En l'algoritme genètic i guardem les poblacions inicials i en la cerca TABU i guardem la població d'entorn. La codificació d'aquesta variable és la següent: en primer lloc disposa d'un nom propi, després disposa de tants cromosomes com peces tenim a seqüenciar i a la posició 11 (recordem que el màxim de la nostra aplicació és 10) i tenim el valor fitness que ens servirà per ordenar llavors la població.

5.1.5. Variable cromor()

Aquesta variable funciona igual que la cromo amb l'única diferència que sempre està ordenada pel que fa als algorismes genètics i pel que fa a la cerca TABU i guardem els gens que després ens generen població.

5.1.6. Variable cromom()

L'última de les taules de gens, en aquesta taula i guardem els gens que muten en la cerca TABU.

6. Resultats i validació del programa.

Degut a la impossibilitat de saber amb exactitud sí la solució que ens retorna el programa és la millor. S'ha decidit utilitzar el programa actual que disposa el Grep desenvolupat en un treball fi de carrera de l'any 2003 per validar la solució ja que aquest programa calcula totes les combinacions possibles de seqüenciació. S'ha triat la combinació formada pels productes 1,2,3,4 i 5 respectivament per tal de determinar fins a quin punt és viable la nostra eina informàtica.

Utilitzant el programa del grep obtenim que aquesta combinació de productes té 6336 combinacions de rutes de fabricació possibles, de totes aquestes seqüències el valor objectiu mínim és de 75 ut. D'aquests 6336 gens només 15 contenen el valor objectiu de 75, per tant estem parlant d'un 0.23 %.

A partir d'aquí hem fet una validació per els algoritmes genètics i per la recerca TABU.

6.1. Validació algoritmes genètics.

Per fer la validació hem realitzat seqüències de 10 (doncs el nostre algoritme no sempre arriba a la mateixa solució o valor objectiu). Aquestes seqüències les hem realitzat modificant en cada una, una sèrie de paràmetres. Seguidament adjuntem les taules resum, i a l'annex C hi ha les taules que recullen tots els resultats de les diferents seqüències.

En la comprovació no s'han tingut en compte les opcions que es refereixen a blocs, (creuament i mutació) doncs aquestes opcions són més o menys efectives depenen l'ordre dels productes dins el gen. És a dir el primer cromosoma sol rebre una discriminació del bloc per tant sí aquest cromosoma pot contenir molta informació (suposem 13 rutes) és més difícil que vagi variant que un cromosoma intermig del gen.

Les variables que hem tingut en compte són:

- Sistema de selecció: Hem comparat les tres regles de selecció que tenim FIT-FIT, FIT-WEAK i RULETA.

- Població: hem estudiat els casos on tenim la població de 10, 20 i 30.

- Iteracions: hem estudiat els casos on tenim de 10, 20 i 30 iteracions, hem de recordar que la mutació es realitza amb de cada iteració.

- Mutació: un sol element o uniforme conjuntament amb una probabilitat alta i/o baixa per determinar fins a quin punt és important la mutació.

- No hem posat límit al número de vegades que repetim la mitjana de la població actual amb l'anterior així sabrem fins a quin punt es repeteixen les mitjanes.

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
FIT-FIT	UNIFORME	NOMÉS UN CROMOSOMA	BAIXA	10	10	77,6	1,4
					20	76	2,5
					30	75,8	3,1
				20	10	77,1	3,8
					20	76,2	5,3
					30	75,4	9,6
				30	10	76,1	6,2
					20	75,4	7,8
					30	75	8,8
			ALTA	10	10	77,3	1,7
					20	76,4	2,7
					30	75,7	2,6
				20	10	75,5	4,2
					20	75,4	3,8
					30	75,3	6,2
				30	10	75,5	5,6
					20	75,3	6,3
					30	75	9
		UNIFORME	BAIXA	10	10	76,3	1,6
					20	75,7	1,9
					30	75,5	1,7
				20	10	76,1	3
					20	76,1	4,5
					30	75,2	4,9
				30	10	75,7	5,4
					20	75,1	5,9
					30	75	6
			ALTA	10	10	76,7	1,3
					20	76,3	2,3
					30	75	2,5
				20	10	76,1	3,7
					20	75,6	5,3
					30	75,4	6
				30	10	75,7	2,8
					20	75	3,7
					30	75	4

Taula 6.1.1: Taula global de resultats FIT FIT

Tal i com podem veure a la taula 6.1.1 en els casos més extrems que són els de 10 iteracions, podem observar com la mutació no afecta gaire on inclús amb un nivell de mutació baixa obtenim millors resultats. En canvi amb poblacions una mica més altes, una mutació alta ens condueix a l'òptim global.

Aquest fet té una explicació molt senzilla. Mentre que el procés realitza les seves iteracions es va aproximant a un òptim local. És a dir creuant la població inicial indefinidament arribaríem a un òptim local que podria ser 77, però aplicant una mutació alta afegim informació genètica a la població cosa que ens fa "marxar" d'aquesta tendència i aconseguim trobar altres òptims locals. És per aquest motiu que en poblacions petites i poques iteracions no es nota, ja que disposem de pocs gens, però en poblacions una mica més considerables obtenim un bon resultat.

Tot i això s'ha d'esmentar que una mutació alta desvirtua el mètode provocant una multitud de pics que no segueixen la tendència natural del procés. Aquest fet es deu a que tot i aplicar la regla de creuament FIT-FIT amb una mutació alta molts gens canvien significativament i això provoca que generem una població nova amb molts cromosomes que no apareixien a la població inicial.

Recordem que el FIT-FIT és un mètode que explora una població creuant els individus més fort entre ells d'aquesta manera es dirigeix cap a un òptim local però si comprovem el VO de cada iteració veurem que no segueix un ordre on el VO vagi millorant sinó que van sorgint diferents pics. A la figura 6.1.1 veiem un recull del millor VO per a cada iteració en un cas de FIT-FIT, creuament uniforme on només mutem un element amb probabilitat baixa de mutar.

Nom	P1	P2	P3	P4	P5	VO
1	6	7	5	2	2	82
2	1	4	2	1	1	77
3	6	9	3	2	1	84
4	6	3	3	2	1	83
5	3	7	6	1	3	78
6	3	7	6	2	3	78
7	3	6	6	2	3	77
8	3	7	6	2	3	78
9	3	9	6	2	1	75
10	3	9	6	1	1	75
11	3	7	6	1	3	78
12	3	9	8	2	2	79
13	3	6	9	2	4	81
14	3	6	9	2	3	81
15	6	9	6	2	2	79
16	1	3	2	2	3	79

Figura 6.1.1 Resultats obtinguts en un procés FIT-FIT

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
FIT-WEAK	UNIFORME	NOMÉS UN CROMOSOMA	BAIXA	10	10	77,5	2,3
					20	75,8	3,1
					30	75,8	2,6
				20	10	77,4	5,8
					20	76,1	5,6
					30	75,4	8
				30	10	76,3	6,4
					20	75,6	9,3
					30	75,5	9,5
			ALTA	10	10	76,8	2,2
					20	76,2	2,8
					30	76,1	2,2
				20	10	76,6	3,6
					20	76,2	5,1
					30	76	4,5
				30	10	75,5	9
					20	75,4	9,2
					30	75,5	9
		UNIFORME	BAIXA	10	10	76,8	2
					20	76,2	2,9
					30	75,8	1,6
				20	10	76,8	3,2
					20	76	4,5
					30	75,3	6,4
				30	10	76,1	5,6
					20	75,4	6,9
					30	75,2	9,2
			ALTA	10	10	76	0,9
					20	75,9	1,2
					30	75,3	1,8
				20	10	75,8	1,9
					20	75,3	2,6
					30	75,3	3,9
				30	10	75,6	3,7
					20	75,1	5,2
					30	75	7

Taula 6.1.2: Taula global de resultats FIT-WEAK

Tal i com podem veure a la taula 6.1.2 en els casos més extrems que són els de 10 iteracions, en el cas d'un cromosoma la diferència és mínima però en canvi quan la probabilitat és per a cada cromosoma es pot apreciar un lleugera millora en el VO.

A diferència del FIT-FIT el FIT-WEAK combina el millor amb el pitjor aquest fet fa que la diversitat dels gens progenitors sigui més elevada i per tant en les poblacions generades apareix més diversitat. Aquest fet sumat amb la mutació ens dóna un bon nivell d'exploració de possibles entorns.

El sistema FIT-WEAK per sistema ja té una varietat de pics doncs el nostre millor gen es combina amb el pitjor i la descendència pot resultar mediocre. Tot i que també pot passar al revés que dos gens midiocres (en una població de 20 parlariem del 7 i el 14) formin una bona descendència.

A la taula 6.1.2 tenim un exemple dels resultats obtinguts amb FIT-WEAK, amb creuament uniforme, mutació uniforme i una mutació elevada.

Nom	P1	P2	P3	P4	P5	VO
1	3	7	6	2	3	78
2	3	9	6	1	4	75
3	3	2	7	1	2	81
4	6	9	6	2	2	79
5	5	4	2	1	2	81
6	7	8	3	1	1	76
7	7	8	2	1	1	76
8	7	8	2	1	1	76
9	7	8	4	1	1	76
10	7	8	6	2	3	78
11	8	8	2	2	3	77
12	7	8	5	1	4	82
13	3	6	5	2	1	77
14	5	4	3	2	3	81
15	7	8	6	1	3	78
16	7	8	2	2	4	76

Figura 6.1.2: Resultats obtinguts en un procés en FIT-WEAK

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
RULETA	UNIFORME	NOMÉS UN CROMOSOMA	BAIXA	10	10	77,3	1,2
					20	76,6	1,8
					30	76,3	2,1
				20	10	76,6	6,6
					20	76,8	6
					30	75,8	6,6
				30	10	76,6	9,5
					20	75,5	10,4
					30	75,1	9,6
			ALTA	10	10	77	1,1
					20	75,7	1,4
					30	76,2	1,4
				20	10	75,9	3,1
					20	75,3	4,7
					30	75,4	4,9
				30	10	76	4,6
					20	75,8	4,1
					30	75,2	6,2
		UNIFORME	BAIXA	10	10	76,3	1,3
					20	75,4	1,4
					30	75,3	1,5
				20	10	75,5	4
					20	75,4	3,4
					30	75,3	3,3
				30	10	75,3	5,1
					20	75,5	4,9
					30	75,4	5,9
			ALTA	10	10	77,6	0,9
					20	75,1	1,5
					30	75	2,3
				20	10	76	1,4
					20	75,1	2,8
					30	75	3,7
				30	10	75,6	4
					20	75	4,7
					30	75	5

Taula 6.1.3: Taula global de resultats amb RULETA

Tal i com podem observar a la figura 6.1.3 aquest mètode per poblacions i iteracions superiors a 10 ens retorna valors molt pròxims a l'òptim global.

El mètode ruleta és el que amb una mutació baixa o moderada, els resultats segueixen més una tendència on cada solució trobada és millor que l'anterior fins arribar a un òptim local. Tal i com ens mostren les dades sí apliquem una probabilitat de mutació alta, en aquest cas sense importar el número d'elements, pràcticament ja obtenim un valor objectiu de 76 per tant sí apliquem un nombre d'iteracions superior a 10 o la població superior a 10 també ja obtenim pràcticament el valor objectiu de tota la població.

El fet d'aplicar una mutació alta fa que ens allunyem de l'òptim local però en canvi que explorem nous entorn on l'òptim local és millor que en l'entorn on ens trobavem.

Per tal de poder entendre aquesta afirmació les següents figures ens mostren els resultats del mètode ruleta. La figura 6.1.3 amb una mutació alta i la 6.1.4 amb una mutació baixa.

Nom	P1	P2	P3	P4	P5	VO
1	1	3	9	1	2	81
2	3	6	5	1	2	77
3	3	9	5	2	4	75
4	3	9	7	2	4	75
5	3	9	7	1	2	75
6	3	2	11	2	3	81
7	3	9	5	1	4	75
8	3	9	8	1	2	79
9	3	7	5	1	3	78
10	3	9	7	1	3	75
11	3	6	7	1	3	77
12	3	6	9	1	2	81
13	3	9	5	1	3	75
14	3	6	7	2	3	77
15	7	8	4	1	1	76
16	7	8	4	1	4	76

Figura 6.1.3: RULETA amb mutació alta

Nom	P1	P2	P3	P4	P5	VO
1	6	7	5	2	2	82
2	6	7	5	2	2	82
3	5	9	5	2	2	82
4	3	9	5	2	1	75
5	3	9	5	1	3	75
6	3	9	5	1	2	75
7	3	9	5	1	3	75
8	3	9	5	1	3	75
9	3	9	5	1	3	75
10	3	9	5	1	3	75
11	3	9	5	1	3	75
12	3	9	5	1	3	75
13	3	9	5	1	3	75
14	3	9	5	1	3	75
15	3	9	5	1	3	75
16	3	9	5	1	3	75

Figura 6.1.4: RULETA amb mutació baixa

Observant les dades en global, podem fer diverses afirmacions:

- En general la mutació ajuda a no caure en òptims locals, tot i que si s'estudien les iteracions es comprovarà com es perd la tendència a millorar, és a dir amb una mutació alta la desviació augmentarà i per tant apareixeran més pics.

- A més nivell de població i iteracions, lògicament el nombre de vegades on igualem les mitjanes de la població actual amb l'anterior augmenta.

- Tot i estudiar poques combinacions (el cas més desfavorable 10 x 10) els resultats obtinguts són més que acceptables. Doncs l'estalvi d'operacions és molt més significatiu que la possible pèrdua de temps en la seqüència.

- En el cas de ruleta sí que ens podem estalviar realitzar multitud d'iteracions utilitzant el fre de mitjanes i així reduir més el temps que gasta el programa.

6.2. Validació TABU.

Per comprovar la fiabilitat del mètode TABU hem realitzat varis anàlisis 10 vegades i en aquest cas només podem modificar el nombre d'iteracions a realitzar. Els anàlisis comprovaran l'algoritme amb 5,10,20,40 i 60 iteracions. Tal i com es mostra a les taules de la figura 8.1.5

nº iteracions	VO
5	79
	84
	77
	76
	81
	76
	77
	87
	79
	83
	79,9

Nº iteracions	VO
10	77
	77
	76
	79
	79
	76
	77
	76
	76
	82
	77,5

nº iteracions	VO
20	79
	76
	81
	77
	76
	77
	79
	76
	76
	77
	77,4

nº iteracions	VO
40	76
	76
	76
	77
	75
	76
	75
	76
	76
	75
	75,8

Nº iteracions	VO
60	76
	76
	76
	76
	76
	76
	75
	76
	76
	76
	75,9

Figura 6.2.5: Taules de resultat amb TABU

Tal i com podem observar el VO que ens retorna la recerca TABU no és tant acurada com el de l'algoritme genètic. Però en canvi és molt més ràpid doncs amb 40 iteracions realitzem 40 x 4=160 combinacions a calcular. El valor obtingut en aquest cas ja és satisfactori doncs recordem que l'òptim global (la millor solució) és 75 i en alguns casos ja l'obtenim.

A més a més de reduir cost computacional a costa de les iteracions, en la recerca tabu el cost de la creació de població i creuament és molt més inferior doncs només tenim en compte una llista que no hem de repetir, mentre que els algoritmes genètics operaven sobre la població.

7. Conclusions

Els objectius del present projecte s'han complert:

Hem estudiat els algoritmes genètics juntament la recerca tabu i hem desenvolupat una eina informàtica que ens permet fer seqüències més llargues que les que podíem realitzar fins avui. L'eina informàtica que s'ha desenvolupat també permet seguir pas per pas com ha anat combinant els gens i com els ha anat mutant, es poden seleccionar les polítiques de selecció dels progenitors així com la manera que s'han de creuar. També podem escollir quines parts del gen poden mutar i amb quina probabilitat.

Tot i això aquest projecte deixa moltes línies obertes per continuar:

- Flexibilitzar la codificació dels gens per tal que la posició on estigui situat el cromosoma no sigui significatiu, provocant així posicions millors que altres.

- Evitar la repetició de gens, tant en la població inicial com en la població descendent augmentant així l'entorn a explorar.

- Evitar que l'algoritme caigui en òptims locals combinant quan calgui els paràmetres de selecció, creuament i mutació.

- Implementar el temps de preparació de les màquines per obtenir un resultat més acurat a la realitat.

- Automatització de l'algoritme, quan entrés una seqüència a calcular l'algoritme determinés els paràmetres a seguir.

- Paral·lelisme d'algoritmes: que es generessin vàries poblacions que iteressin paral·lelament i intercanviessin els seus cromosomes.

8. Relació de documents

El present projecte està format per els següents documents:

DOCUMENT NÚMERO 1:Memòria.

DOCUMENT NÚMERO 2: Pressupost.

9. Bibliografia.

Dominguez Machuca, J. A. *Dirección de Operaciones*. Editorial McGraw Hill. 1995.

Morales, Eduardo. *Búsqueda, Optimización y Aprendizaje*. Març 1999

Petroutsos, Evangelos. *Visual Basic 6.* Madrid:Anaya multimedia. Madrid 1999.

Bordera Martín, Josep. Disseny d'una aplicació informàtica per a la seqüenciació de peces en un taller mecànic. Projecte/Treball Fi de Carrera. Enginyeria Industrial. Escola Politècnica Superior. Universitat de Girona. Desembre 2003.

Casadesús,M. De Castro, R. Ferrer, I. *Organització de la producció: Presentacions.* Girona 2005

Fernández-Baños Marín, Ignacio. *Programación de la secuencia de fabricación en una màquina, con tiempos de preparación variables, mediante la aplicación de algoritmos genéticos*. Projecte/Treball Fi de Carrera. Enginyeria Industrial. Escola Tècnica superior d'enginyeria Industrial de Barcelona. Desembre 2003.

Martí, Rafael. *Meta Heurísticas en Optimización Combinatoria*: *Algoritmos Genéticos*. 2000.

Joo Park, Byung i Rim Choi,Hyung. *A Genetic Algorithm for Integration of Process Planning and Scheduling in a Job Shop*. Dong-A University, depatment of management information system, Korea 2006.

Rocha, J. Ramos, C. Vale,Z. *PROCESS PLANNING USING A GENETIC ALGORITHM APPROACH*. Instituto Superior de Engenharia do Porto. Porto 1999.

Xing, Y. Chen, Z. Sung J. Hu L. *An Improved Adapatative Genetic Algorithm for Job-Shop Scheduling Problem*. Key laboratory for Precision and Non-traditional Machining Technology of Ministry of Education Dalian Universtiy of Technology. Xina 2007.

A. Manual d'usuari

En el present annex s'explica quins passos previs s'han de fer per utilitzar el programa i com s'ha d'utilitzar el programa.

A 1. Passos previs a l'execució.

A 1.1. Instal·lació del programa

Per tal de poder operar amb el programa de mètodes heurístics no cal fer cap instal·lació només cal executar l'arxiu PFC-Fcc i tenir la base de dades Fcc dins la carpeta C:\TEMP.

A 1.2. Modificació de la base de dades

Per a poder afegir productes i/o rutes a la base de dades només s'ha d'anar a la taula ruta i afegir les dades conforme la codificació d'aquesta.

CodiProducte	CodiRuta	OrdreRuta	CodiRecurs	Temps
3	9	1	3	4
3	9	2	1	6
3	9	3	2	7
3	9	4	1	4
3	9	5	3	3
3	9	6	2	2
3	10	1	1	6
3	10	2	2	4
3	10	3	3	3
3	10	4	2	2
3	10	5	1	4
3	10	6	2	3
3	10	7	1	4

Figura 1: taula rutes

A la figura 1 veiem les rutes 9 i 10 del producte 3. En cas de voler afegir una ruta nova només l'hem d'insertar amb la mateixa codificació. En cas de voler eliminar una ruta s'ha de tenir present que els codis de la ruta han de ser correlatius i no pot haver-hi un salt.
En cas de voler afegir una màquina s'ha d'anar a la taula recursos i afegir-la. Igualment com passa amb les rutes els codis han de ser correlatius.

	CodiRecurs	Nom Recurs	Tipus Recurs
+	1	Fresa-01	Fresa
+	2	Taladre-01	Taladre
+	3	Torn-01	Torn

Figura 2: taula de màquines

A.2. El programa de càlcul.

A 2.1. Formulari inicial.

El primer quadre que ens sortirà només té dues opcions: Algoritmes genètics i cerca TABU. Cada opció ens portarà cap a l'opció corresponent de càlcul.

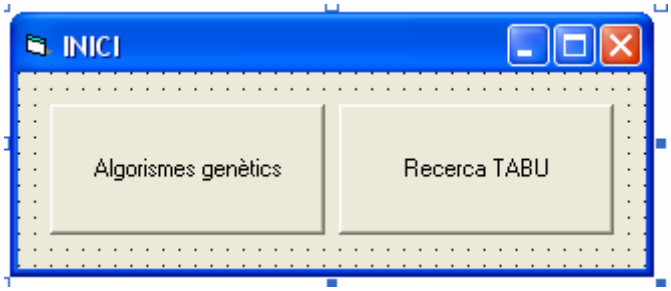


Figura 3: Selecció del mètode a realitzar.

A 2.2 Algoritmes genètics

A continuació es pot observar la pantalla de selecció dels paràmetres per a realitzar l'algoritme genètic. A continuació definirem totes les interaccions que podem realitzar-hi.

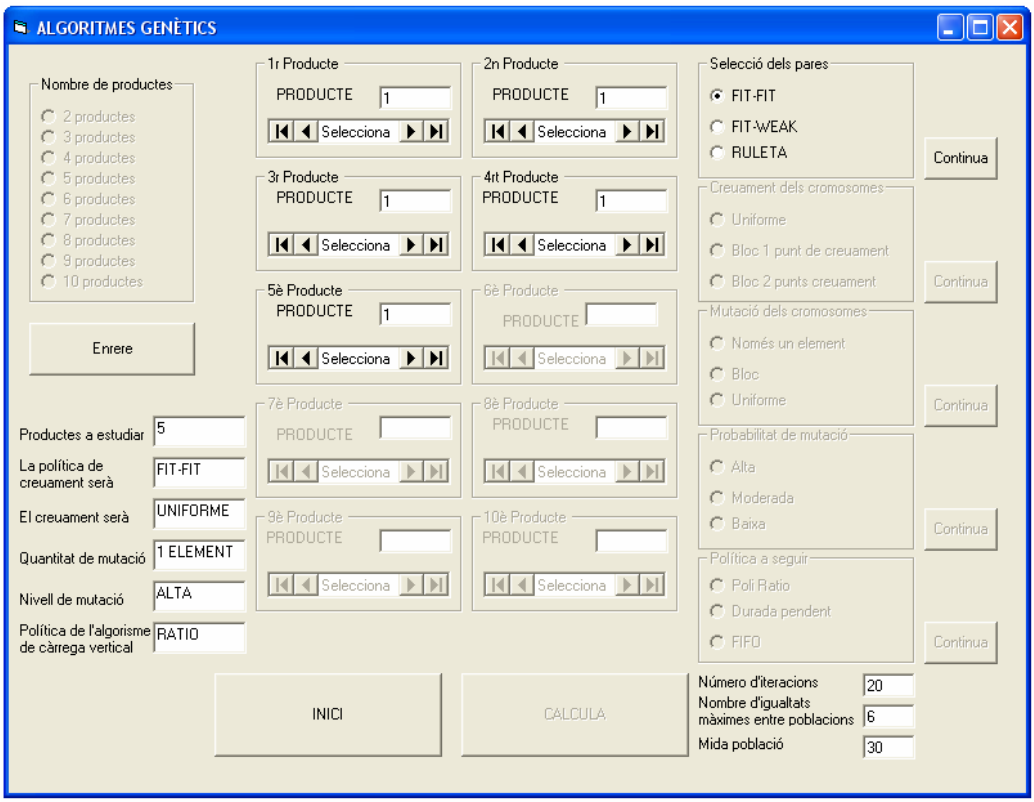


Figura 4: Pantalla algoritmes genètics

La primera opció que podem modificar és la del nombre de productes que volem seqüenciar figura 3. Una vegada haguem escollit aquesta opció llavors podrem escollir la següent opció que serà quins productes volem seqüenciar.

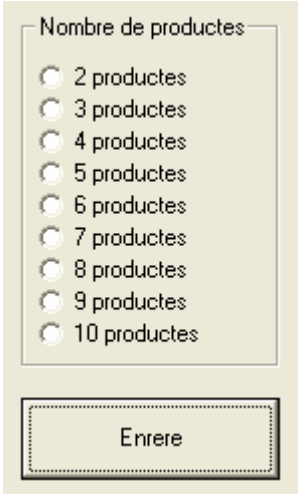


Figura 5: Selecció nombre productes

L'opció ENRERE torna la selecció a l'estat inicial i per tant haurem de tornar a escollir el nombre de productes i reiniciar tota la selecció.
Una vegada escollit el nombre de productes s'activaran els quadres de selecció del producte en qüestió i també s'activarà el primer quadre de la dreta on podrem escollir la selecció dels progenitors. La figura 4 és l'exemple per escollir els productes en un cas de 4 productes. La resta de quadres quedaran deshabilitats. La figura 5 és el quadre on hem de triar la selecció dels progenitors una vegada haguem decidit la selecció hem de prémer el botó continuar.

Figura 6: Selecció de productes

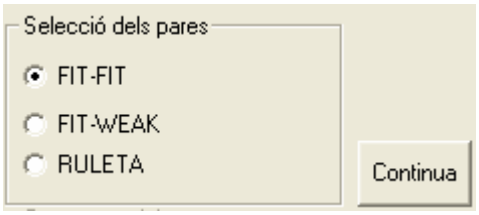


Figura 7: Selecció dels progenitors

Una vegada seleccionat l'opció de selecció s'ha d'escollir el mètode de creuament i altre cop prémer el botó continua corresponent (figura 6).

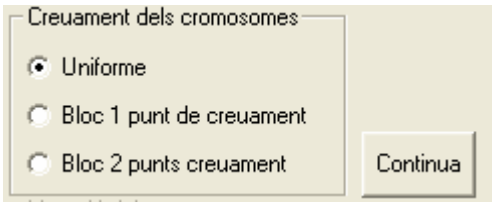


Figura 8: Selecció del creuament entre gens

Seleccionat el mètode de creuament s'activarà l'opció d'escollir mutació dels cromosomes, aquí altra vegada una vegada seleccionada l'opció desitjada hem de prèmer el boto continua corresponent (figura 7).

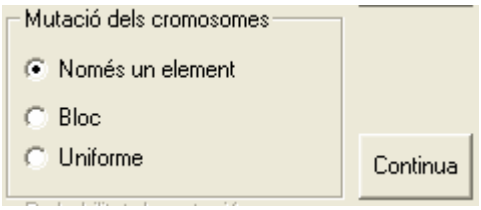


Figura 9: Selecció de la mutació dels gens

Finalment una vegada haguem escollit totes les opcions anteriors només ens faltarà escollir la probabilitat de mutació i una vegada escollida hem de tornar a prémer el botó continua (figura 8).

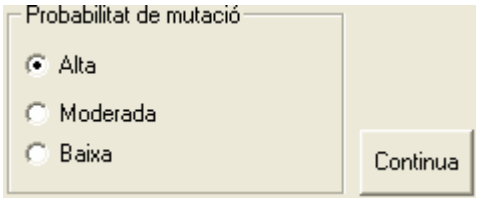


Figura 10: Selecció de la probabilitat de mutació

L'últim quadre habilitat és el de la figura 9 on en aquest escollim la política que seguirà l'algoritme de càrrega vertical.

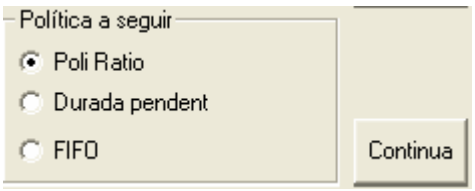


Figura 11: Selecció política algoritme càrrega vertical

Amb la selecció de la política a seguir s'habilitarà el botó CALCULA que començarà el procès dels algoritmes genètics no obstant encara podem canviar 3 opcions més, que són el nombre d'iteracions, la mida de la població i la quantitat màxima d'igualtats entre les mitjanes de les poblacions. Per defecte aquestes opcions ja tenen un valor tot i que es pot canviar per qualsevol altre. Només s'ha de tenir en compte que la mida de la població ha de ser parell per als mètodes FIT-FIT i FIT-WEAK.

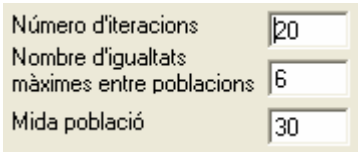


Figura 12: Selecció de nombre d'iteracions, igualtats i població

L'opció **INICI** ens retorna a la selecció dels mètodes (algoritmes genètics i cerca TABU).
L'opció **CALCULA** inicia l'algoritme genètic utilitzant tots els paràmetres escollits.

Figura 13: Botons INICI i CALCULA

Tots els paràmetres que haguem escollit queden recollits tal i com podem observar a la figura 14. En aquests quadres de text no s’hi ha d’escriure res.

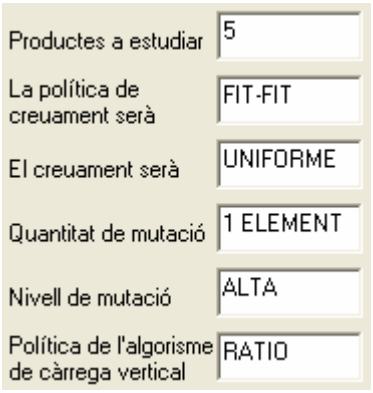


Figura 14: Recull de paràmetres

A 2.3. Pantalla resultats algoritmes genètics.

La pantalla de resultats dels algoritmes genètics està dividida en tres parts. A la primera graella s'hi dipositen els gens de la població inicial, a la segona la població inicial degudament ordenada i a la tercera el millor gen de cada població. A la figura 15 es pot observar com és la pantalla de resultats.

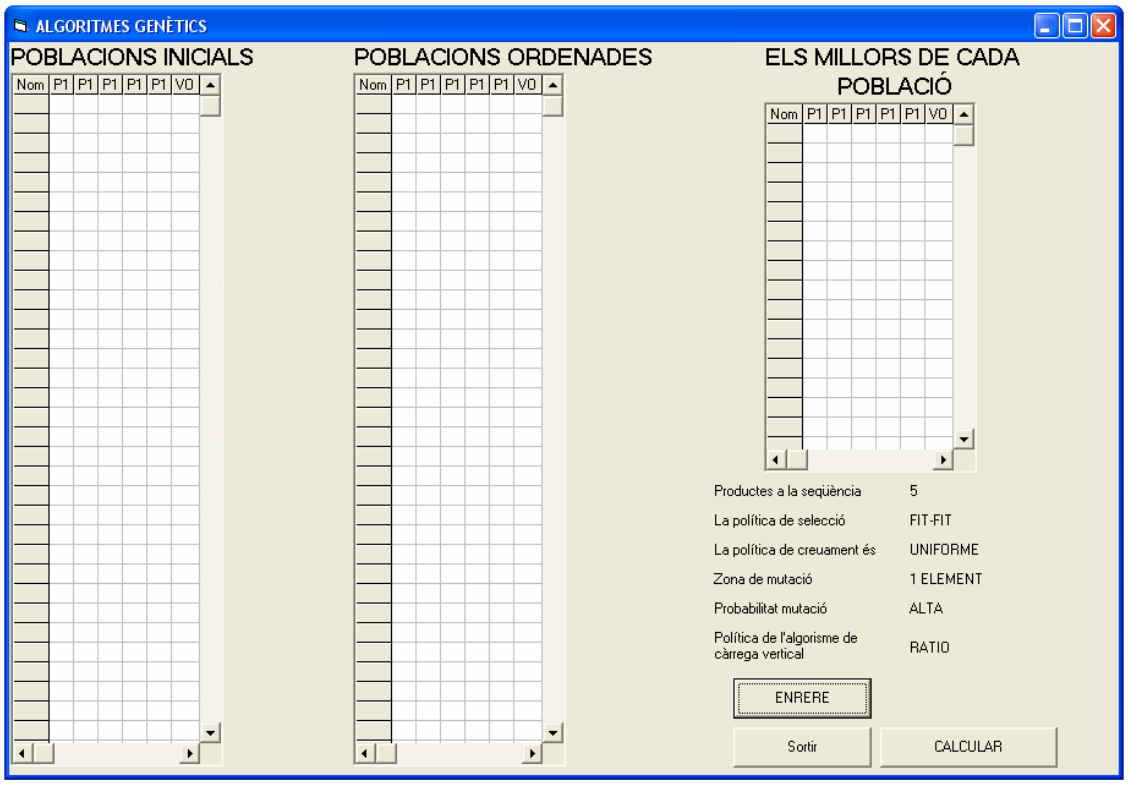


Figura 15: Pantalla resultats

En aquesta pantalla només tenim 3 interaccions que fer:
Botó **ENRERE:** tornem a la selecció de paràmetres de l'algoritme genètic.
Botó **Sortir:** Sortim de tota l'aplicació.
Botó **Calcular:** Comença el càlcul.
A sota la tercera graella hi ha un recull del paràmetres que s'han tingut en compte.

A 2.4. Cerca TABU.

La pantalla de selecció per el mètode TABU és molt més simple que en els algoritmes genètics doncs aquí només hem de seleccionar el nombre de productes, les peces a seqüenciar i la política de l'algoritme de càrrega vertical. A més a més també podem escollir el nombre d'iteracions. A la figura 16 podem observar l'estructura.

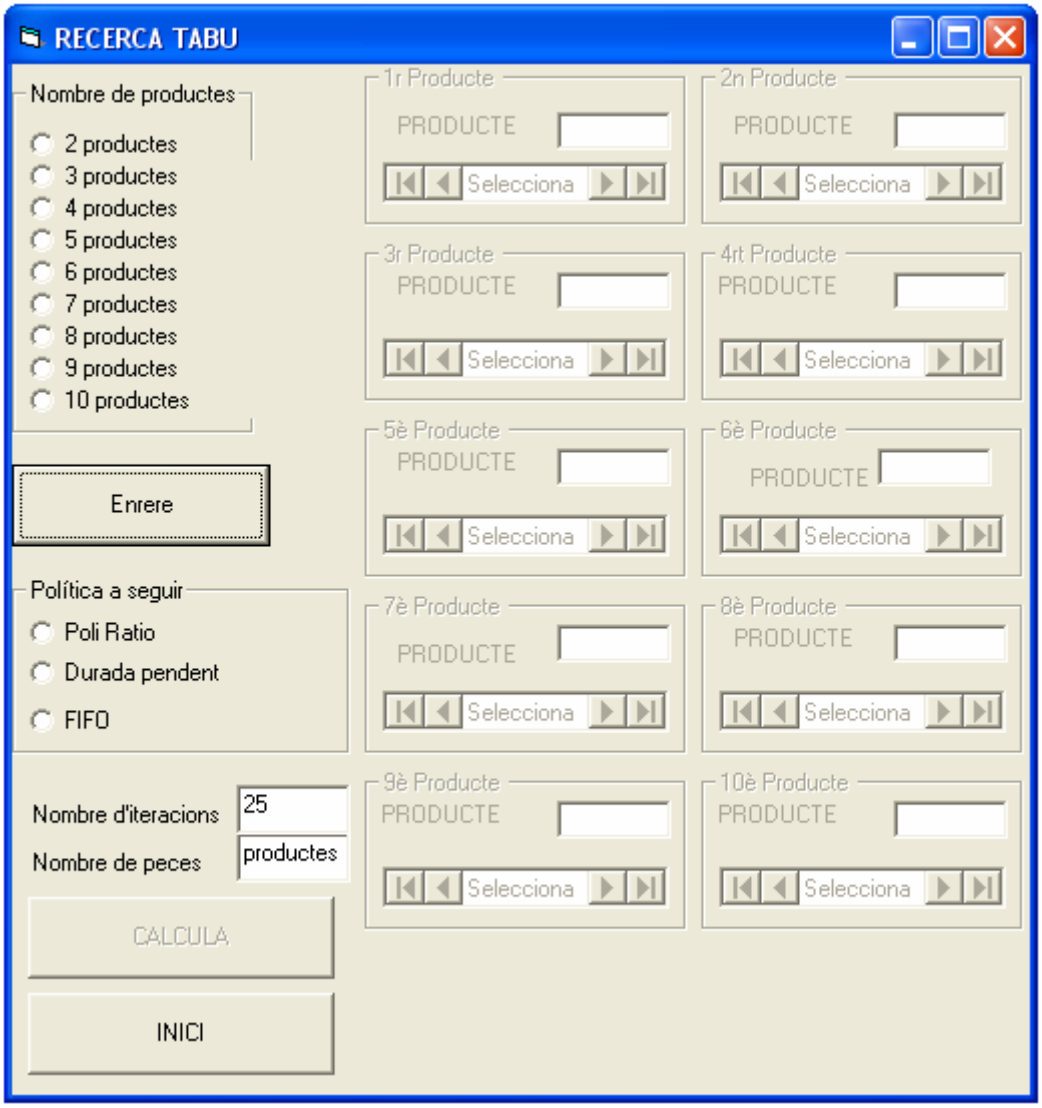


Figura 16: Selecció paràmetres TABU

De la mateixa manera que amb els algoritmes genètics el nombre d'iteracions ve definit a 25 però el podem canviar per el valor que més ens interessi.

A 2.5. El programa de càlcul. Resultats TABU

La pantalla de resultats en TABU també la tenim dividida en tres parts. La primera graella conté el gen que estem estudiant i l'entorn que hem generat per a poder distingir-los el gen que estudiem ens apareix subratllat en verd. A la segona graella apareixeran els gens que són TABU i que per tant no es tornaran a estudiar i finalment a la tercera graella apareixeran els gens que han estat mutats degut que estaven a la llista TABU i que per tant no hem de tornar a estudiar.

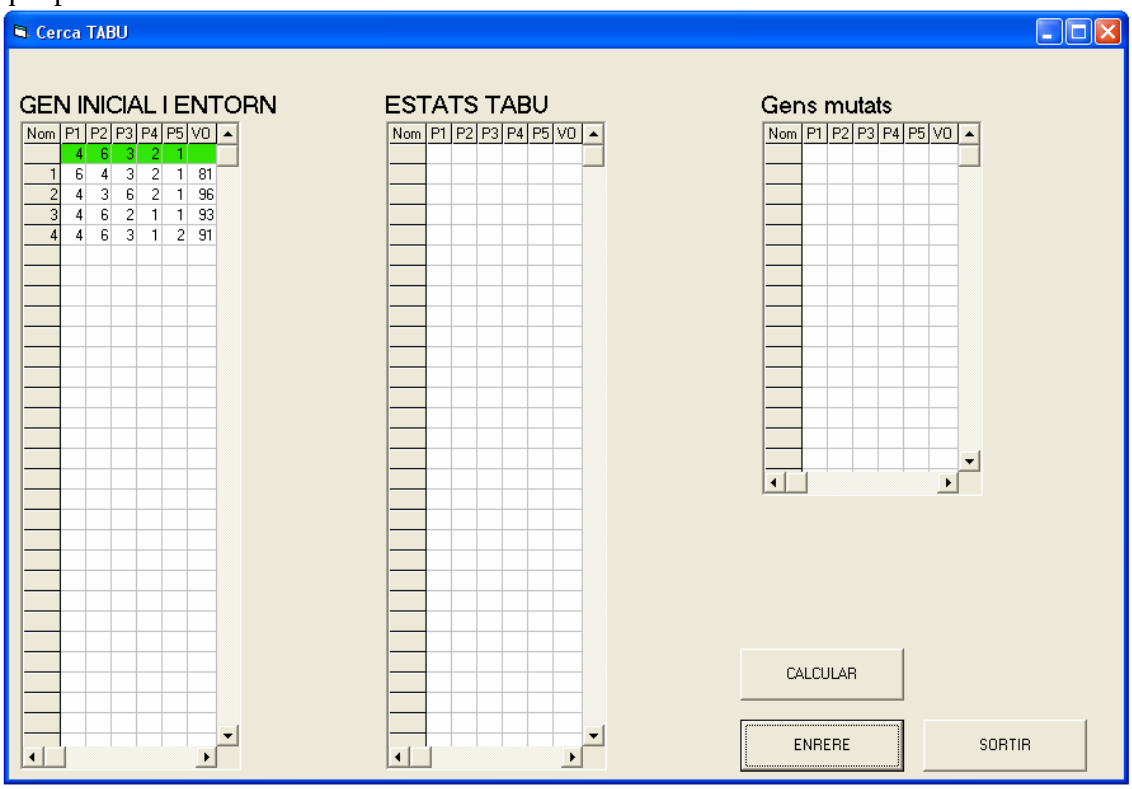


Figura 17: Pantalla resultats TABU

En aquesta pantalla només tenim 3 interaccions que fer:
Botó **ENRERE:** tornem a la selecció de paràmetres de la cerca TABU.
Botó **Sortir:** Sortim de tota l'aplicació.
Botó **Calcular:** Comença el càlcul.

B. Codi informàtic.

En primer lloc es mostraran totes les interfícies gràfiques que hi ha en el programa per tal de poder veure quines parts té cadascuna, i llavors es dona tot el codi informàtic separat per interfícies més un primer mòdul on es defineixen totes les variables.

B.1. Interfície gràfica

A continuació adjuntem totes les pantalles que apareixen al programa per poder-les identificar amb la seva part de codi.

B.1.1. Inici

A la figura B.1. tenim la primera pantalla que ens apareix en el programa.

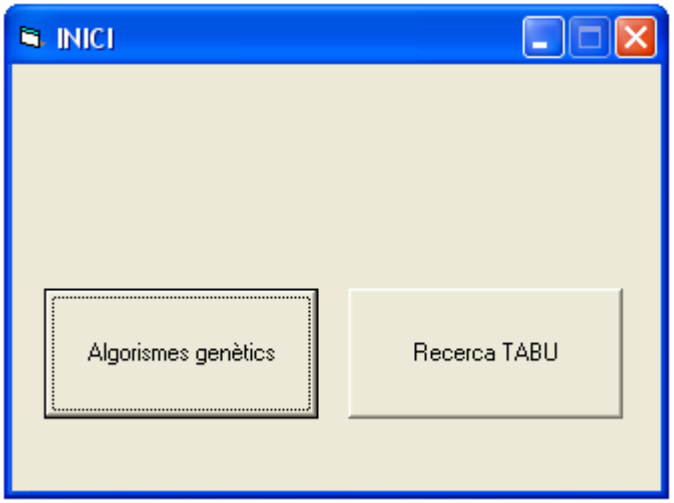


figura B.1. Selecció del mètode

B.1.2. AG

A la figura B.2. hi figura la pantalla de selecció de paràmetres en l'algoritme genètics.

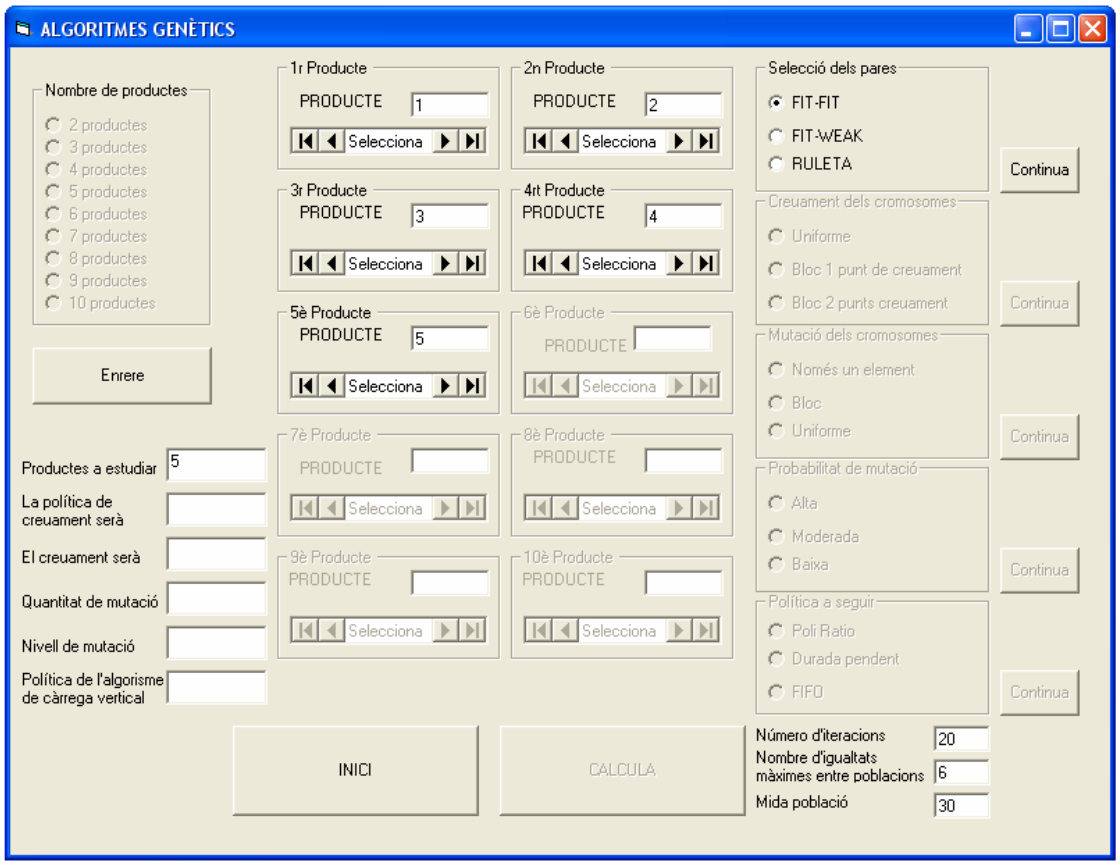


Figura B.2. Selecció de paràmetres algoritmes genètics

B.1.3. AG2

A la figura B.3. tenim la pantalla de resultats dels algoritmes genètics

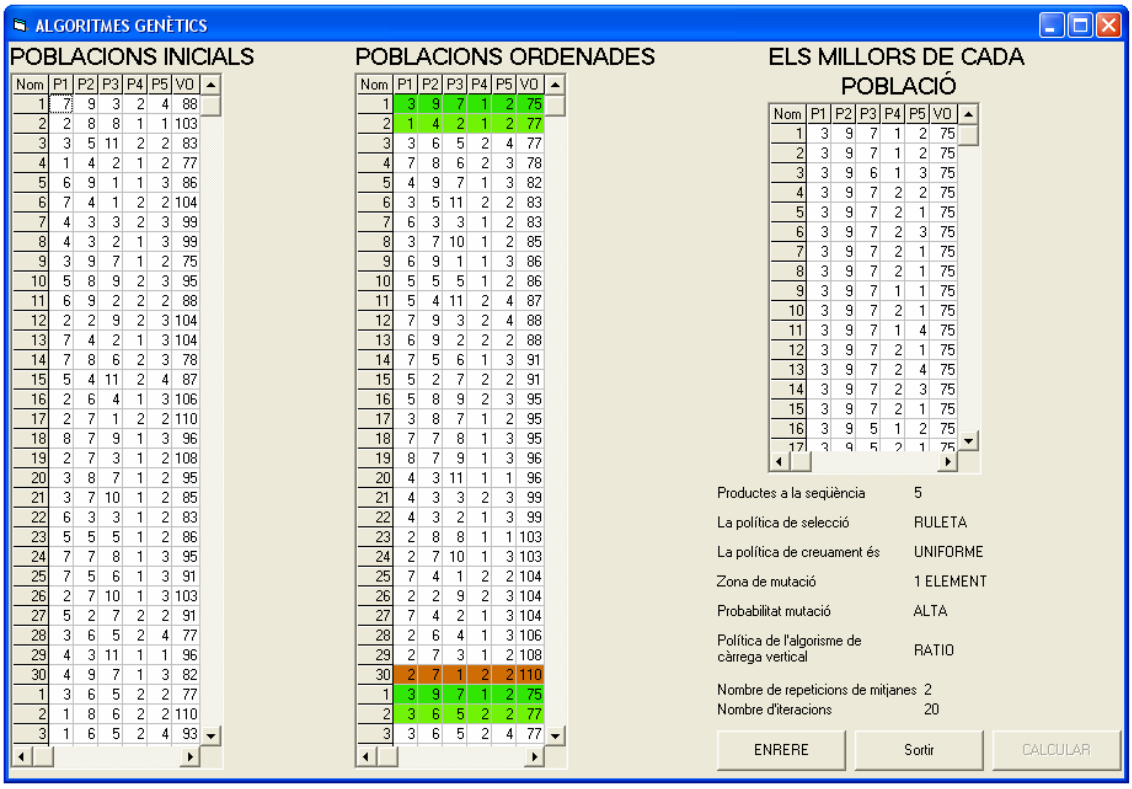


Figura B.3.

B.1.4. TABU

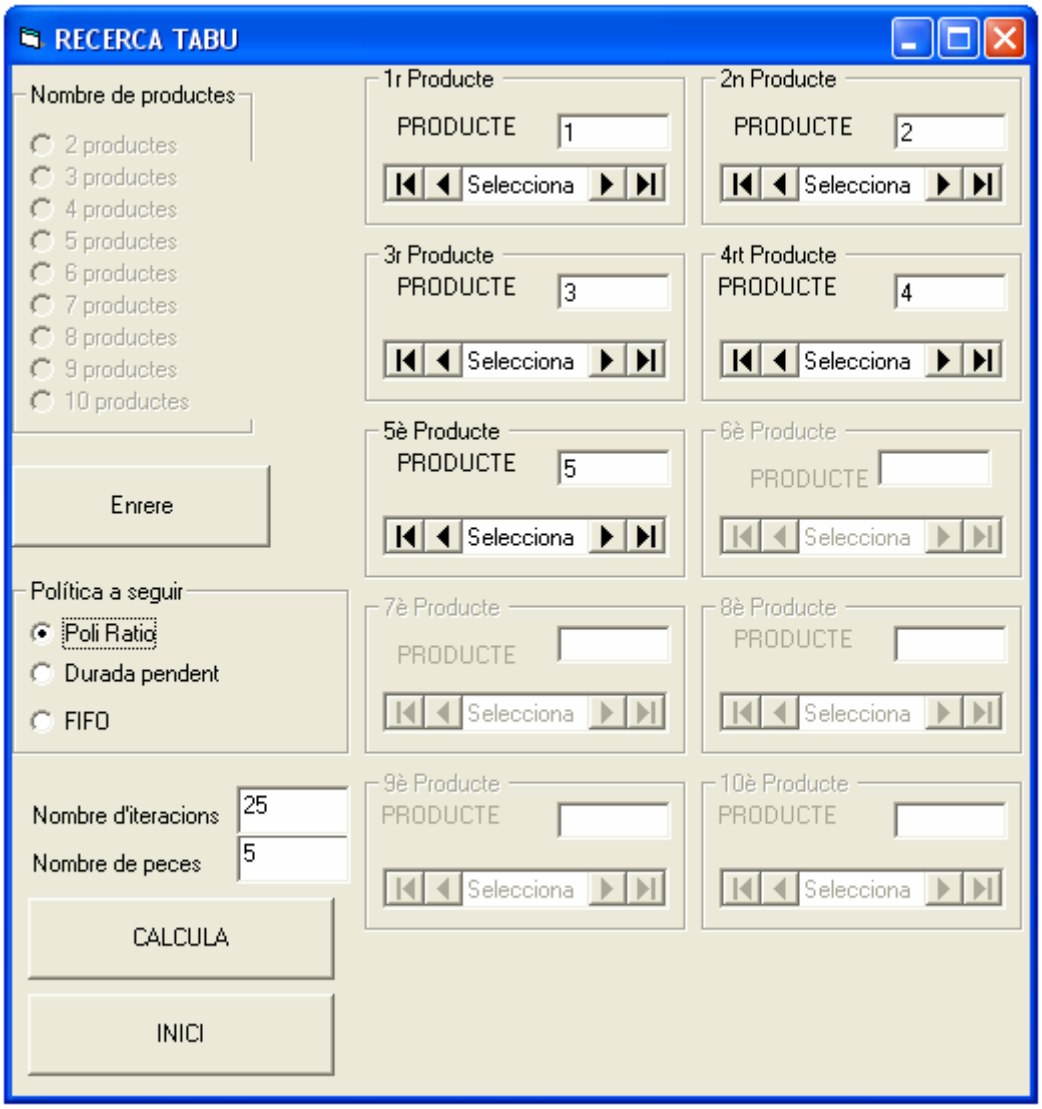


Figura B.4. Selecció de paràmetres en TABU

B.1.5. TABU 2

A la figura B.5. tenim la pantalla de resultats del mètode TABU.

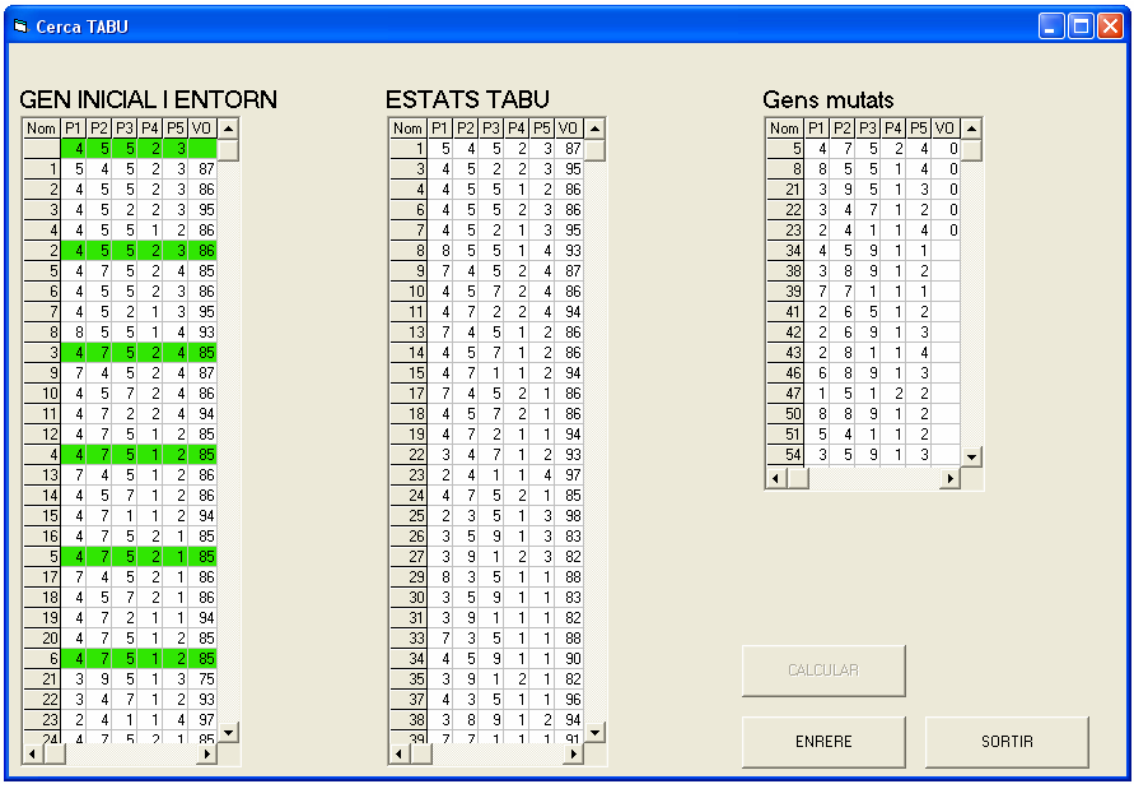


Figura B.5. Resultats amb mètode TABU.

B.2.CODI INFORMÀTIC D'INTERFÍCIES.

A continuació presentem tot el codi informàtic de la nostra aplicació dividit en les interfícies gràfiques corresponent més un mòdul on hi declarem totes les variables.

B.2.1. Mòdul de variables

Aquí declarem totes les variables.

```
'VARIABLES TIPUS TAULA
Global w(1 To 10) As Single 'codi peces
Global max(1 To 10) As Single 'max rutes per producte
Global c(1 To 10) As Single '2 rutes amb el mateix valor
Global g(1 To 10) As Single 'aleatorietat per canvis
Global p(1 To 10) As Integer 'control de ultima combinació en AG control de cromosomes en TABU

'VARIABLES CROMO (on guardem els gens)
Global cromo(1 To 5000, 0 To 11) As String 'llista desordenada de gens
Global cromor(1 To 5000, 0 To 11) As String 'llista ordenada de gens
Global cromom(1 To 5000, 0 To 11) As String  'llista gens tabu

'Variables d'ús freqüent i contadors
Global totalproductes As Single 'aqui guardarem la quantitat de peces
Global n As Integer 'número de gens
Global d As Integer 'número d'iteracions
Global d2 As Integer 'número màxim d'igualtats entre les mitjanes de la població
Global d3 As Integer 'mida població
Global u As Integer 'columnes
Global l As Integer 'files
Global l11 As Integer 'contador files 1
Global l12 As Integer 'contador files 2
Global l13 As Integer 'contador files 3
Global cr As Boolean 'canvi ruta
Global cp As Boolean 'canvi peça
Global mit As Integer 'mitjana actual de la població
Global mit2 As Integer 'mitjana anterior de la població
Global x As Integer 'contador d'iteracions
Global i As Single 'contador varis
Global i2 As Single 'contador varis
Global i3 As Single 'contador varis
Global j As Single 'contador varis
Global j2 As Integer 'contador de mitjanes
```

Public t As Integer 'més ràpida
Public t2 As String 'segona igual de ràpida
Public m As Single 'posicions cromo i cromor
Public m2 As String 'posicions cromo i cromor
Public m3 As Integer 'contador per cromom
Public s As Single 'Contador operacions
Public b1 As Integer

‘Variables de selecció
Global selpares As Single 'seleccionarem el tipus de selecció pels creuaments
Global selcreua As Single 'seleccionem el tipus de creuament
Global selmuta As Single 'seleccionem el tipus de mutació
Global selnivell As Single 'seleccionem el nivell de mutació
Global selpoli As Single 'seleccinem la politica a seguir per l'algorisme de carrega vertical

‘Variables generació blocs
Global pbloc As Integer 'posició inicial bloc
Global lbloc As Integer 'llargada del bloc

‘Variables base de dades
Global Basedades As Database
Global taulaproductes As Recordset
Global taularecursos As Recordset
Global taularutes As Recordset

'VARIABLES ALGORISME CÀRREGA VERTICAL
'Variables Clau:
Global TotalMaquines As Integer
Global Estat As String

'Variables de tipus Taula:
Global Peza() As Integer
Global Maq() As Integer
Global Maxratio() As Integer
Global Ruta() As Integer

'Variables Específiques o d'Us freqüent:
Global At As Integer, Carrega As Integer, Demanda As Integer, DemandaCambiada As Integer
Global Comptador As Integer
Global ComptaMaq As Integer, Fila As Integer, Columna As Integer
Global Penalitzacio As Integer
Global Alt As Integer, Ample As Integer

B.2.2. INICI

```
Private Sub Command1_Click()
INICI.Hide
AG.Show
End Sub

Private Sub Command2_Click()
INICI.Hide
TABU.Show
End Sub
```

B.2.3. AG

```
Private Sub Form_Load() 'Iniciem el programa i carreguem la base de dades
Set Basedades = OpenDatabase("c:\Temp\Fcc")
Set taulaproductes = Basedades.OpenRecordset("PRODUCTE", dbOpenTable)
Set taularecursos = Basedades.OpenRecordset("RECURS", dbOpenTable)
Set taularutes = Basedades.OpenRecordset("RUTA", dbOpenTable)
amaga
End Sub

Function amaga() 'Anul·lem totes les opcions per fer que l'usuari vagi pas a pas
Frame1.Enabled = False
Text1.Enabled = False
Data1.Enabled = False
Label1.Enabled = False
Frame2.Enabled = False
Text2.Enabled = False
Data2.Enabled = False
Label2.Enabled = False
Frame3.Enabled = False
Text3.Enabled = False
Data3.Enabled = False
Label3.Enabled = False
Frame4.Enabled = False
Text4.Enabled = False
Data4.Enabled = False
Label4.Enabled = False
Frame5.Enabled = False
 Text5.Enabled = False
Data5.Enabled = False
Label5.Enabled = False
Frame6.Enabled = False
Text6.Enabled = False
Data6.Enabled = False
Label6.Enabled = False
Frame7.Enabled = False
Text7.Enabled = False
Data7.Enabled = False
Label7.Enabled = False
Frame8.Enabled = False
Text8.Enabled = False
Data8.Enabled = False
Label8.Enabled = False
Frame9.Enabled = False
Text9.Enabled = False
Data9.Enabled = False
Label9.Enabled = False
```

```
Frame10.Enabled = False
Text10.Enabled = False
Data10.Enabled = False
Label10.Enabled = False
Quadrepolis.Enabled = False
PoliRatio.Enabled = False
PoliDP.Enabled = False
PoliFIFO.Enabled = False
CALCULA.Enabled = False
Quadrepares.Enabled = False
optifitfit.Enabled = False
optifitweak.Enabled = False
optiruleta.Enabled = False
Quadrecromosomes.Enabled = False
optiuniforme.Enabled = False
optibloc1.Enabled = False
optibloc2.Enabled = False
Quadremuta.Enabled = False
optimuta1.Enabled = False
optimutabloc.Enabled = False
optimutauniforme.Enabled = False
Quadrenivell.Enabled = False
optimutalta.Enabled = False
optimutamoderada.Enabled = False
optimutabaixa.Enabled = False
a1.Enabled = False
A2.Enabled = False
A3.Enabled = False
A4.Enabled = False
A5.Enabled = False
End Function

Private Sub Option1_Click() 'Activem l'opció de seqüenciar 2 productes
dosproductes
bloqueja
End Sub

Private Sub Option2_Click() 'Activem l'opció de seqüenciar 3 productes
tresproductes
bloqueja
End Sub

Private Sub Option3_Click() 'Activem l'opció de seqüenciar 4 productes
quatreproductes
bloqueja
End Sub

Private Sub Option4_Click() 'Activem l'opció de seqüenciar 5 productes
cincproductes
bloqueja
```

End Sub

Private Sub Option5_Click() 'Activem l'opció de seqüenciar 6 productes
sisproductes
bloqueja
End Sub

Private Sub Option6_Click() 'Activem l'opció de seqüenciar 7 productes
setproductes
bloqueja
End Sub

Private Sub Option7_Click() 'Activem l'opció de seqüenciar 8 productes
vuitproductes
bloqueja
End Sub

Private Sub Option8_Click() 'Activem l'opció de seqüenciar 9 productes
nouproductes
bloqueja
End Sub

Private Sub Option9_Click() 'Activem l'opció de seqüenciar 10 productes
deuproductes
bloqueja
End Sub

Function bloqueja() 'tanquem la possibilitat de tornar a triar el número de productes
Option1.Enabled = False
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
Option5.Enabled = False
Option6.Enabled = False
Option7.Enabled = False
Option8.Enabled = False
Option9.Enabled = False
Option1 = False
Option2 = False
Option3 = False
Option4 = False
Option5 = False
Option6 = False
Option7 = False
Option8 = False
Option9 = False
End Function
Function dosproductes() 'selecció de dos productes i activem el control per
Frame1.Enabled = True 'seleccionar els dos productes que vulguem de la base

```
Text1.Enabled = True
Data1.Enabled = True
Label1.Enabled = True
Frame2.Enabled = True
Text2.Enabled = True
Data2.Enabled = True
Label2.Enabled = True
PRODUCTES.Text = "2"
Data1.Refresh
Data2.Refresh
espares 'habilitem la selecció de la regla de selecció dels progenitors.
End Function
Function tresproductes() 'per poder escollir els tres productes
dosproductes
Frame3.Enabled = True
Text3.Enabled = True
Data3.Enabled = True
Label3.Enabled = True
PRODUCTES.Text = "3"
Data3.Refresh
End Function
Function quatreproductes() 'per poder escollir els quatre productes
tresproductes
Frame4.Enabled = True
Text4.Enabled = True
Data4.Enabled = True
Label4.Enabled = True
PRODUCTES.Text = "4"
Data4.Refresh
End Function
Function cincproductes() 'per poder escollir els cinc productes
quatreproductes
Frame5.Enabled = True
Text5.Enabled = True
Data5.Enabled = True
Label5.Enabled = True
PRODUCTES.Text = "5"
Data5.Refresh
End Function
Function sisproductes() 'per poder escollir els 6 productes
cincproductes
Frame6.Enabled = True
Text6.Enabled = True
Data6.Enabled = True
Label6.Enabled = True
PRODUCTES.Text = "6"
Data6.Refresh
End Function
Function setproductes() 'per poder escollir els 7 productes
sisproductes
```

```
Frame7.Enabled = True
Text7.Enabled = True
Data7.Enabled = True
Label7.Enabled = True
PRODUCTES.Text = "7"
Data7.Refresh
End Function
Function vuitproductes() 'per poder escollir els 8 productes
setproductes
Frame8.Enabled = True
Text8.Enabled = True
Data8.Enabled = True
Label8.Enabled = True
PRODUCTES.Text = "8"
Data8.Refresh
End Function

Function nouproductes() 'per poder escollir els 9 productes
vuitproductes
Frame9.Enabled = True
Text9.Enabled = True
Data9.Enabled = True
Label9.Enabled = True
PRODUCTES.Text = "9"
Data9.Refresh
End Function

Function deuproductes() 'per poder escollir els 10 productes
nouproductes
Frame10.Enabled = True
Text10.Enabled = True
Data10.Enabled = True
Label10.Enabled = True
PRODUCTES.Text = "10"
Data10.Refresh
End Function
Function espares() 'habilitem la selecció de la selecció dels progenitors.
Quadrepares.Enabled = True
optifitfit = True
optifitfit.Enabled = True
optifitweak.Enabled = True
optiruleta.Enabled = True
a1.Enabled = True
End Function

Private Sub a1_Click() 'seleccionem el tipus de selecció dels progenitors
If optifitfit = True Then 'i habilitem la selecció del tipus de creuament
   pares.Text = "FIT-FIT"
   selpares = 1
   End If
```

```
If optifitweak = True Then
   pares.Text = "FIT-WEAK"
   selpares = 2
   End If
If optiruleta = True Then
   pares.Text = "RULETA"
   selpares = 3
   End If
Quadrepares.Enabled = False
optifitfit.Enabled = False
optifitweak.Enabled = False
optiruleta.Enabled = False
Quadrecromosomes.Enabled = True
optiuniforme = True
optiuniforme.Enabled = True
optibloc1.Enabled = True
optibloc2.Enabled = True
optibloc2 = False
a1.Enabled = False
A2.Enabled = True
End Sub

Private Sub A2_Click() 'seleccionem el tipus de creuament i habilitem la mutació
If optiuniforme = True Then
   creuament.Text = "UNIFORME"
   selcreua = 1
   End If
If optibloc1 = True Then
   creuament.Text = "BLOC"
   selcreua = 2
   End If
If optibloc2 = True Then
   creuament.Text = "BLOC 2"
   selcreua = 3
   End If
Quadrecromosomes.Enabled = False
optiuniforme.Enabled = False
optibloc1.Enabled = False
optibloc2.Enabled = False
Quadremuta.Enabled = True
optimuta1 = True
optimuta1.Enabled = True
optimutabloc.Enabled = True
optimutauniforme.Enabled = True
A2.Enabled = False
A3.Enabled = True
End Sub

Private Sub A3_Click() 'seleccionem la mutació i habilitem
If optimuta1 = True Then 'la selecció de la probabilitat de mutació
```

```
   mutació.Text = "1 ELEMENT"
   selmuta = 1
   End If
If optimutabloc = True Then
   mutació.Text = "BLOC"
   selmuta = 2
   End If
If optimutauniforme = True Then
   mutació.Text = "uniforme"
   selmuta = 3
   End If
Quadremuta.Enabled = False
optimutauniforme.Enabled = False
optimutabloc.Enabled = False
optimuta1.Enabled = False
Quadrenivell.Enabled = True
optimutalta = True
optimutalta.Enabled = True
optimutamoderada.Enabled = True
optimutabaixa.Enabled = True
A3.Enabled = False
A4.Enabled = True
End Sub

Private Sub A4_Click() 'seleccionem la possibilitat de mutació i habilitem la
política de ACV
If optimutalta = True Then
   nivellmuta.Text = "ALTA"
   selnivell = 3
   End If
If optimutamoderada = True Then
   nivellmuta.Text = "MODERADA"
   selnivell = 5
   End If
If optimutabaixa = True Then
   nivellmuta.Text = "BAIXA"
   selnivell = 8
   End If
Quadrenivell.Enabled = False
optimutalta = False
optimutalta.Enabled = False
optimutamoderada.Enabled = False
optimutabaixa.Enabled = False
A4.Enabled = False
A5.Enabled = True
politica
End Sub
Function politica() ' habilitem la selecció de la política de
Quadrepolis.Enabled = True 'Algorisme de Carrega Vertical
PoliRatio.Enabled = True
```

```
PoliDP.Enabled = True
PoliFIFO.Enabled = True
PoliRatio = True
End Function

Private Sub A5_Click() 'Una vegada seleccionada la política de l'ACV habilitem el botó calcular
If PoliRatio = True Then
   poliav.Text = "RATIO"
   selpoli = 1
   End If
If PoliDP = True Then
   poliav.Text = "DP"
   selpoli = 2
   End If
If PoliFIFO = True Then
   poliav.Text = "FIFO"
   selpoli = 3
   End If
Quadrepolis.Enabled = False
PoliRatio.Enabled = False
PoliDP.Enabled = False
PoliFIFO.Enabled = False
A5.Enabled = False
CALCULA.Enabled = True
End Sub

Private Sub CALCULA_Click() 'assignem per a cada cromosoma el codi del seu producte
totalproductes = PRODUCTES
d = Text11
d2 = Text12
d3 = Text13
For i = 1 To 10
   w(i) = 0
Next
If PRODUCTES = 10 Then
   w(1) = Text1.Text
   w(2) = Text2.Text
   w(3) = Text3.Text
   w(4) = Text4.Text
   w(5) = Text5.Text
   w(6) = Text6.Text
   w(7) = Text7.Text
   w(8) = Text8.Text
   w(9) = Text9.Text
   w(10) = Text10.Text
   AG.Hide
   AG2.Show
ElseIf PRODUCTES = 9 Then
```

```
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    w(8) = Text8.Text
    w(9) = Text9.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 8 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    w(8) = Text8.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 7 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 6 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 5 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 4 Then
```

```
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 3 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    AG.Hide
    AG2.Show
ElseIf PRODUCTES = 2 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    AG.Hide
    AG2.Show
End If
End Sub

Private Sub SORTIR_Click() 'Tornem a l'inici
AG.Hide
INICI.Show
End Sub

Private Sub ENRERE_Click() 'Per poder tornar a escollir el número de peces a seqüenciar
amaga
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True
Option5.Enabled = True
Option6.Enabled = True
Option7.Enabled = True
Option8.Enabled = True
Option9.Enabled = True
PRODUCTES.Text = ""
End Sub
```

B.2.5. G2

```
Private Sub Form_Load() 'Carreguem les bases de dades
Set Basedades = OpenDatabase("c:\temp\Fcc")
Set taulaproductes = Basedades.OpenRecordset("PRODUCTE", dbOpenTable)
Set taularecursos = Basedades.OpenRecordset("RECURS", dbOpenTable)
Set taularutes = Basedades.OpenRecordset("RUTA", dbOpenTable)
productes2 = totalproductes 'A partir d'aquí ens sortirà per pantalla les
If selpares = 1 Then        ' opcions que hem escollit
   pares2 = ("FIT-FIT")
ElseIf selpares = 2 Then
   pares2 = ("FIT-WEAK")
ElseIf selpares = 3 Then
   pares2 = ("RULETA")
End If
If selcreua = 1 Then
   creuament2 = "UNIFORME"
ElseIf selcreua = 2 Then
   creuament2 = "BLOC"
ElseIf selcreua = 3 Then
   creuament2 = "BLOC 2"
End If
If selmuta = 1 Then
   mutació2 = "1 ELEMENT"
ElseIf selmuta = 2 Then
   mutació2 = "BLOC"
ElseIf selmuta = 3 Then
   mutació2 = "uniforme"
End If
If selnivell = 3 Then
   nivellmuta2 = "ALTA"
ElseIf selnivell = 5 Then
   nivellmuta2 = "MODERADA"
ElseIf selnivell = 8 Then
   nivellmuta2 = "BAIXA"
   selnivell = 10
   End If
If selpoli = 1 Then
   poliav2 = "RATIO"
   ElseIf selpoli = 2 Then
   poliav2 = "DP"
   ElseIf selpoli = 3 Then
   poliav2 = "FIFO"
   End If
Graella
minmax
```

```
crearpoblacióini
End Sub

Function Graella() 'dimensionem i nombrem les graelles
Graella1.TextMatrix(0, 0) = "Nom"
Graella1.Rows = ((d3 * d) + 1)
Graella1.Cols = (totalproductes + 2)
Graella1.TextMatrix(0, totalproductes + 1) = ("VO")
Graella1.ColWidth(0) = 450
Graella1.ColWidth(totalproductes + 1) = 350
Graella1.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella1.ColWidth(u) = 300
   Graella1.TextMatrix(0, u) = ("P" & w(u))
Next
Graella2.TextMatrix(0, 0) = "Nom"
Graella2.Cols = (totalproductes + 2)
Graella2.Rows = ((d3 * d) + 1)
Graella2.ColWidth(0) = 450
Graella2.ColWidth(totalproductes + 1) = 350
Graella2.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella2.ColWidth(u) = 300
   Graella2.TextMatrix(0, u) = ("P" & w(u))
Next
Graella2.TextMatrix(0, totalproductes + 1) = ("VO")
Graella3.TextMatrix(0, 0) = "Nom"
Graella3.Cols = (totalproductes + 2)
Graella3.Rows = d + 1
Graella3.ColWidth(0) = 450
Graella3.ColWidth(totalproductes + 1) = 350
Graella3.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella3.ColWidth(u) = 300
   Graella3.TextMatrix(0, u) = ("P" & w(u))
Next
Graella3.TextMatrix(0, totalproductes + 1) = ("VO")
l11 = 1
l12 = 1
l13 = 1
End Function

Function minmax() ' Busquem el número màxim de rutes de cada producte
taularutes.MoveLast
p(1) = taularutes.Fields("CodiProducte")
p(2) = taularutes.Fields("CodiRuta")
p(3) = taularutes.Fields("OrdreRuta")
For i = 1 To totalproductes 'número de peces
   j = 0 'codi ruta
   s = 0 'nombre operacions
```

```
    cp = False
    Do While cp = False 'condició per canviar de peça
       j = j + 1
       s = 1
       taularutes.Index = "PrimaryKey"
       taularutes.Seek "=", w(i), j, s
       cr = False
          Do While cr = False
             If w(i) = p(1) And j = p(2) And s = p(3) Then
                cr = True
                cp = True
             Else
                taularutes.MoveNext ' ens adelentem a la taula per saber si la línia que
estem mirant
                i2 = taularutes.Fields("CodiRuta") ' és l'última de la ruta i/o la peça
                taularutes.MovePrevious
             End If
                If i2 <> j Then
                   If i2 = 0 Then
                      cr = True
                      cp = True
                      End If
                   cr = True
                   s = 0
                   y = 0
                End If
          taularutes.MoveNext
          s = s + 1
          Loop
       max(i) = j
       If i2 <> j And i2 = 1 Then
          cp = True
           End If
    Loop
Next
End Function
Function crearpoblacióini() 'creem la població inicial totalment aleatòria
n = 0
Do While n < d3
 n = n + 1
 cromo(n, 0) = n
 For i = 1 To totalproductes
    Randomize
    g(i) = Rnd
    cromo(n, i) = CInt((1 - max(i)) * g(i) + max(i))
  Next
Loop
For i = 1 To n 'per evitar problemes de codi donem valor 0 a la part del
    For j = totalproductes + 1 To 11 'cromosoma que no fem servir
       cromo(i, j) = 0
```

```
   Next
Next
End Function

Private Sub Command1_Click() 'Acció principal aquí comencem a iterar
x = 0
mit = 1000
mit2 = 500
j2 = 0
Text1 = n
Do While j2 < d2 And x < d
x = x + 1
mit2 = mit
For b1 = 1 To d3
AccioACV
Next
mostra
ordena
selecció
mutació
i2 = 0
For t = 1 To d3
   i2 = cromor(t, 11) + i2
Next
mit = i2 / d3
If mit = mit2 Then
   j2 = j2 + 1
End If
Loop
Command1.Enabled = False
Label5 = j2
Label7 = x
End Sub

Function mostra() 'Aquí donem per pantalla tots els cromosomes sense ordenar.
For l = 1 To d3
   For u = 0 To totalproductes
      Graella1.TextMatrix(l11, u) = cromo(l, u)
      Graella1.TextMatrix(l11, totalproductes + 1) = cromo(l, 11)
   Next
l11 = l11 + 1
Next
End Function

Function ordena() 'Trec els cromosomes de la taula per posar-los en una altra però
aquest cop ordenats per el seu fitness
m = 1
t2 = 1
For i2 = 1 To n
   t = 500
```

```
   For i = 1 To d3 'detecto el cromosoma amb més fitness
      If cromo(i, 11) < t And cromo(i, 11) <> 0 Then
         t = cromo(i, 11)
         t2 = i
      End If
   Next
If i2 = 1 Then
   Graella3.TextMatrix(l13, 0) = l13
   For u = 1 To totalproductes
      Graella3.TextMatrix(l13, u) = cromo(t2, u)
   Next
      Graella3.TextMatrix(l13, totalproductes + 1) = cromo(t2, 11)
l13 = l13 + 1
End If
For u = 1 To totalproductes
   Graella2.ColWidth(u) = 300
   Graella2.TextMatrix(0, u) = ("P" & w(u))
Next
Graella2.TextMatrix(0, totalproductes + 1) = ("VO")
   For j = 1 To 11 'creo el cromosoma més fort en la primera posició de la taula i així
succesivament
      cromor(m, 0) = m
      cromor(m, j) = cromo(t2, j)
      cromo(t2, j) = 0 ' destrueixo el cromosoma desordenat de la taula desordenada
   Next
m = m + 1
Next
For l = 1 To n
   If l = 1 Then
      For u = 1 To totalproductes + 1
         Graella2.Row = l12
         Graella2.Col = u
         Graella2.CellBackColor = (190000)
      Next
   ElseIf l = 2 Then
      For u = 1 To totalproductes + 1
         Graella2.Row = l12
         Graella2.Col = u
         Graella2.CellBackColor = (390000)
      Next
   ElseIf l = d3 Then
      For u = 1 To totalproductes + 1
         Graella2.Row = l12
         Graella2.Col = u
         Graella2.CellBackColor = (290000)
      Next
   End If
   For u = 0 To totalproductes
      Graella2.TextMatrix(l12, u) = cromor(l, u)
      Graella2.TextMatrix(l12, totalproductes + 1) = cromor(l, 11)
```

```
   Next
l12 = l12 + 1
Next
End Function

Function selecció() 'selecció de progenitors.
If selpares = 1 Then
selFitFit
ElseIf selpares = 2 Then
selFitWeak
ElseIf selpares = 3 Then
selruleta
End If
End Function

Function selFitFit() 'Fit-Fit
m = 0
i = 1
m2 = 2
i2 = 2
For m = 1 To n
      creuament
i = i + 2
i2 = i2 + 2
m2 = m2 + 2
m = m + 1
Next
End Function

Function selFitWeak() 'Fit-Weak
m = 0
i = 1
m2 = 2
i2 = d3
For m = 1 To n
      creuament
i = i + 1
m2 = m2 + 2
m = m + 1
i2 = i2 - 1
Next
End Function

Function selruleta() 'Ruleta
m = 0
i = 1
m2 = 0
For m = 1 To n
   Randomize
   g6 = CInt(Rnd * 10)
```

```
   cromo(m, 0) = m
   If g6 < 3 Then
         m2 = 1
      ElseIf g6 < 6 And g6 > 2 Then
         m2 = 2
      ElseIf g6 < 8 And g6 > 5 Then
         m2 = 3
      ElseIf g6 < 10 And g6 > 7 Then
         m2 = 4
      ElseIf g6 = 10 Then
         m2 = 5
      End If
   If selcreua = 1 Then 'uniforme
      For t = 1 To totalproductes
         g(t) = CInt(Rnd * 10)

            If g(t) < 5 Then
                  cromo(m, t) = cromor(m, t)
               Else
                  cromo(m, t) = cromor(m2, t)
               End If
      Next
   End If
   If selcreua = 2 Then 'bloc1
      lbloc = CInt((totalproductes + 2) / 2)
         For t = 1 To lbloc
            cromo(m, t) = cromor(m, t)
         Next
         For t = lbloc To totalproductes
            cromo(m, t) = cromor(m2, t)
         Next
   End If
   If selcreua = 3 Then 'bloc2
      pbloc = CInt(totalproductes / 3) + 1
      lbloc = pbloc + CInt(totalproductes / 3)
         For t = 1 To pbloc
            cromo(m, t) = cromor(m, t)
         Next
         For t = pbloc To lbloc
            cromo(m, t) = cromor(m2, t)
         Next
         For t = lbloc To totalproductes
            cromo(m, t) = cromor(m, t)
         Next
   End If
Next
End Function

Function creuament() 'Selecció del tipus de creuament
cromo(m, 0) = m
```

```
cromo(i, 0) = i
If selcreua = 1 Then
   uniforme
ElseIf selcreua = 2 Then
   bloc1
ElseIf selcreua = 3 Then
   bloc2
End If
End Function

Function uniforme() 'Creuament uniforme
For t = 1 To totalproductes
         Randomize
         g(t) = CInt(Rnd * 10)
         If g(t) < 5 Then
            cromo(m, t) = cromor(i, t)
            cromo(m2, t) = cromor(i2, t)
         Else
            cromo(m, t) = cromor(i2, t)
            cromo(m2, t) = cromor(i, t)
         End If
      Next
End Function

Function bloc1() 'Creuament per bloc amb 1 punt de creuament.
lbloc = CInt((totalproductes + 2) / 2)
For t = 1 To lbloc
   cromo(m, t) = cromor(i, t)
   cromo(m2, t) = cromor(i2, t)
Next
For t = lbloc To totalproductes
   cromo(m, t) = cromor(i2, t)
   cromo(m2, t) = cromor(i, t)
Next
End Function

Function bloc2() 'Creuament per bloc amb 2 punts de creuament
pbloc = CInt(totalproductes / 3) + 1
lbloc = pbloc + CInt(totalproductes / 3)
For t = 1 To pbloc
   cromo(m, t) = cromor(i2, t)
   cromo(m2, t) = cromor(i, t)
Next
For t = pbloc To lbloc
   cromo(m, t) = cromor(i, t)
   cromo(m2, t) = cromor(i2, t)
Next
For t = lbloc To totalproductes
   cromo(m, t) = cromor(i2, t)
   cromo(m2, t) = cromor(i, t)
```

```
Next
End Function

Function mutació() 'Selecció de mutació a nivell de cromosomes
If selmuta = 1 Then
   mutaun
ElseIf selmuta = 2 Then
   mutabloc
ElseIf selmuta = 3 Then
   Mutauniforme
End If
End Function

Function mutaun() 'Mutació d'un únic cromosoma
For i = 1 To n
   Randomize
   g(1) = CInt(Rnd * 10)
   If g(1) > selnivell Then
      g(2) = CInt((1 - totalproductes) * Rnd + totalproductes)
      cromo(i, g(2)) = CInt((1 - max(g(2))) * Rnd + max(g(2)))
   End If
Next
End Function

Function mutabloc() 'Mutació per tot un bloc
pbloc = CInt(totalproductes / 3) + 1
lbloc = pbloc + CInt(totalproductes / 3)
For i = 1 To n
   For t = pbloc To lbloc
      Randomize
      g(1) = CInt(Rnd * 10)
         If g(1) > selnivell Then
            cromo(i, t) = CInt((1 - max(t)) * Rnd + max(t))
         End If
   Next
Next
End Function

Function Mutauniforme() 'Mutació uniforme
For i = 1 To n
   For t = 1 To totalproductes
      Randomize
      g(t) = CInt(Rnd * 10)
      If g(t) > selnivell Then
         cromo(i, t) = CInt((1 - max(t)) * Rnd + max(t))
      End If
   Next
Next
End Function
```

Function AccioACV()

```
'Càlcul del valor fitness del cromosoma.
   ReDim Peza(8, 1 To totalproductes)
   ReDim Ruta(1 To totalproductes)
   TotalMaquines = taularecursos.RecordCount
   ReDim Maq(3, 1 To TotalMaquines)
   ReDim Maxratio(1, 1 To TotalMaquines)
   Ample = 0
   taularutes.Index = "PrimaryKey"
   For i = 1 To totalproductes
      Ruta(i) = cromo(b1, i)
   Next
   t = 0
   Demanda = 0
   i = 1   'Compta Maquines
   Do While i <= TotalMaquines 'Netejar Maquines
      Maq(0, i) = 0
      Maq(1, i) = 0
      i = i + 1
   Loop
   Estat = "PENSANT"
   AgafarDades 'INSTRUCCIO PER UNA OPCIO (NOM)
   Do While Estat <> "ACABAT"
      ActualitzarDades
      Carrega = 0
      ComprovarTemps
      Buidar
      ComprovarCues
      'Carregar
      ComptaMaq = 1   'Compta maquines
      Do While ComptaMaq <= TotalMaquines And Carrega = 0  'Nomes carrego una
maquina en un instant de temps
         If Maq(2, ComptaMaq) > 0 Then
            Carregar    'MODIFICAT
            Carrega = 1
            End If
         ComptaMaq = ComptaMaq + 1
      Loop
      'Acabar
      ComprovarDemanda
      If Demanda = 0 Then
         Estat = "ACABAT"
      Else
         Estat = "PENSANT"
      End If
        cromo(b1, 11) = Ample
   Loop
```

End Function

Function AgafarDades()

```
'Instrucció per obtenir les dades que es necessitaran en el transcurs de
'l'execució del programa.
'NOM
For i = 1 To totalproductes
   Peza(0, i) = w(i)
Next
i = 1 'Compta Peces
Do While i <= totalproductes
   'Ultima Op Feta (UOF)
   Peza(1, i) = 0
   'INSTANT DISPONIBILITAT
   Peza(2, i) = 0
   'RUTA
   Peza(3, i) = Ruta(i)
   'RATIO
   Peza(5, i) = -1
   j = 1   'Compta OP
   taularutes.Index = "PrimaryKey"
   taularutes.Seek "=", Peza(0, i), Peza(3, i), j
   Peza(4, i) = 0
   Do While taularutes.NoMatch = False
      Peza(4, i) = taularutes.Fields("Temps") + Peza(4, i) 'Duració Pendent (DP)
      Peza(6, i) = j      'Op d'aquesta Ruta
      j = j + 1
      taularutes.Index = "PrimaryKey"
      taularutes.Seek "=", Peza(0, i), Peza(3, i), j
   Loop
   i = i + 1
Loop
End Function

Function ActualitzarDades()
'Instrucció per obtenir el temps de l'operació següent de cada peça
'(si existeix) i la màquina que farà aquella operació
j = 1 'Compta peces
Do While j <= totalproductes
   taularutes.Seek "=", Peza(0, j), Ruta(j), Peza(1, j) + 1
   If taularutes.NoMatch = False Then
      Peza(7, j) = taularutes.Fields("CodiRecurs") 'Màquina que la farà
      Peza(8, j) = taularutes.Fields("Temps")      'Durada de la següent OP
   Else
      Peza(7, j) = 0  'Si no hi ha següent OP, la màquina serà la 0
      Peza(8, j) = 1  'Temps per evitar errors de càlcul de DP,...
   End If
   j = j + 1
Loop
End Function

Function ComprovarTemps()
'Funció per comprovar el temps en què està el sistema.
```

```
    ComprovarDemanda
    If Demanda = 0 Then 'Instant inicial pq encara no s'ha fet cua
        t = 0
    Else
        t = 32000   'Temps exagerat i tira endarrera per tenir en compte nomes les maq que
cal
        i = 1   'Compta maquines
        Do While i <= TotalMaquines
            If t > Maq(1, i) And (Maq(3, i) = 1 Or Maq(2, i) >= 1) Then
                t = Maq(1, i)
            End If
        i = i + 1
        Loop
        i = 1 'Compta maquines
        Do While i <= TotalMaquines 'Garanteixo que el temps no torna enrera
            If t > Maq(1, i) Then
             Maq(1, i) = t
            End If
            i = i + 1
        Loop
    End If
End Function
Function Buidar()
'Descarrega les màquines que acaben en aquest instant de temps
i = 1 'Compta maquines
Do While i <= TotalMaquines
    If t = Maq(1, i) And Maq(3, i) = 1 And Maq(0, i) <= t Then
        Maq(3, i) = 0   'Queda Lliure
    End If
    i = i + 1
Loop
End Function

Function ComprovarCues()
'Aquesta funció fa quatre tasques alhora:
'1) Neteja les cues de les màquines per recalcular-les.
'2) Comprova màquina a màquina les peces que hi van,
'3) Mentre comprova els peces, calcula el ratio de cada una.
'4) Mentre calcula el ratio, emmagatzema el màxim de cada màquina.
i = 1   'Compta maquines
Do While i <= TotalMaquines 'Neteja Cues
    If Maq(3, i) = 0 Then   'Nomes d les maq lliures
        Maq(2, i) = 0
    End If
    i = i + 1
Loop
i = 1 'Compta maq
Do While i <= TotalMaquines
    If Maq(3, i) = 0 Then
```

```
        Maxratio(1, i) = 0
         j = 1   'Compta Peces
        Do While j <= totalproductes
           If i = Peza(7, j) And Peza(2, j) <= t And Maq(0, i) <= t Then
              Maq(2, i) = Maq(2, i) + 1   'Augmenta la CUA de la maq i
              If selpoli = 1 Then  'Politica Ratio
                 If Maq(0, i) <= Peza(2, j) Then 'Calcul Penalitzacio
                    Penalitzacio = Peza(2, j) - Maq(0, i)
                 Else
                    Penalitzacio = 0
                 End If
                 Peza(5, j) = (Peza(4, j) * 100 / (Peza(8, j) + Penalitzacio)) 'RATIO
              ElseIf selpoli = 2 Then
                 Peza(5, j) = 30000 / (Peza(4, j) + 1) 'Menys DP=> + Ratio
              ElseIf selpoli = 3 Then
                 Peza(5, j) = 30000 / (Peza(2, j) + 1) 'Abans Disponible=> + Ratio
              End If
              If Maxratio(1, i) <= Peza(5, j) Then
                 Maxratio(1, i) = Peza(5, j)
                 Maxratio(0, i) = j
              End If
           Else
              Peza(5, j) = -1
           End If
           j = j + 1
        Loop
     Else
        Maq(2, i) = 0  'La cua de les ocupades 0 garanteix que no les carregui de nou
     End If
     i = i + 1
  Loop
```

End Function

Function Carregar()

```
'Funció que representa el carregar, és a dir, ocupar la màquina, la peça,
'incrementar les peces fetes, el temps, ...
'Com que estem en el formulari de Una Opció, memoritza un històric dels
'moviments per poder-los representar després amb una taula i un gráfic.

    taularutes.Index = "PrimaryKey"
    taularutes.Seek "=", Peza(0, Maxratio(0, ComptaMaq)), Peza(3, Maxratio(0,
ComptaMaq)), Peza(1, Maxratio(0, ComptaMaq)) + 1
    At = taularutes.Fields("Temps")
    Peza(1, Maxratio(0, ComptaMaq)) = Peza(1, Maxratio(0, ComptaMaq)) + 1 'Nova
UOF
    Peza(2, Maxratio(0, ComptaMaq)) = t + At 'Disponibilitat de la PeZa
    Peza(4, Maxratio(0, ComptaMaq)) = Peza(4, Maxratio(0, ComptaMaq)) - At    'Nova
DP
    Maq(0, ComptaMaq) = t + At   'Instant de disponibilitat de la maq
    Maq(1, ComptaMaq) = t + At   'Instant de control de la maq
```

```
   Maq(3, ComptaMaq) = 1    'Busy (ocupada)
   t = t + At  'Actualitza temps
   If t > Ample Then
      Ample = t   'Aqui AMPLE guarda el temps max (útil per calcular l'amplada
després)
   End If
End Function

Function ComprovarDemanda()
'Funció per saber la demanda general. Serveix per saber si falta fer alguna
'peça (si n'hi ha alguna que s'espera) o si ja estan totes fetes
   i = 1
   DemandaCambiada = 0
   Do While i <= TotalMaquines And DemandaCambiada = 0
      If Maq(2, i) = 0 And Maq(3, i) = 0 Then
         Demanda = 0
      Else
         Demanda = 1
         DemandaCambiada = 1
      End If
   i = i + 1
   Loop
End Function

Private Sub ENRERE_Click()
Unload AG2
AG.Show
End Sub


Private Sub SORTIR_Click()
End
End Sub
```

B.2.5. TABU

```
Private Sub Form_Load()
Command1.Enabled = False
amaga
Option1 = False
Option2 = False
Option3 = False
Option4 = False
Option5 = False
Option6 = False
Option7 = False
Option8 = False
Option9 = False
End Sub

Function amaga()
Frame1.Enabled = False
Text1.Enabled = False
Data1.Enabled = False
Label1.Enabled = False
Frame2.Enabled = False
Text2.Enabled = False
Data2.Enabled = False
Label2.Enabled = False
Frame3.Enabled = False
Text3.Enabled = False
Data3.Enabled = False
Label3.Enabled = False
Frame4.Enabled = False
Text4.Enabled = False
Data4.Enabled = False
Label4.Enabled = False
Frame5.Enabled = False
 Text5.Enabled = False
Data5.Enabled = False
Label5.Enabled = False
Frame6.Enabled = False
Text6.Enabled = False
Data6.Enabled = False
Label6.Enabled = False
Frame7.Enabled = False
Text7.Enabled = False
Data7.Enabled = False
Label7.Enabled = False
Frame8.Enabled = False
Text8.Enabled = False
```

```
Data8.Enabled = False
Label8.Enabled = False
Frame9.Enabled = False
Text9.Enabled = False
Data9.Enabled = False
Label9.Enabled = False
Frame10.Enabled = False
Text10.Enabled = False
Data10.Enabled = False
Label10.Enabled = False
End Function

Function bloqueja()
Option1.Enabled = False
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
Option5.Enabled = False
Option6.Enabled = False
Option7.Enabled = False
Option8.Enabled = False
Option9.Enabled = False
Option1 = False
Option2 = False
Option3 = False
Option4 = False
Option5 = False
Option6 = False
Option7 = False
Option8 = False
Option9 = False
End Function

Private Sub Option1_Click()
dosproductes
bloqueja
End Sub

Private Sub Option2_Click()
tresproductes
bloqueja
End Sub

Private Sub Option3_Click()
quatreproductes
bloqueja
End Sub

Private Sub Option4_Click()
cincproductes
```

```
bloqueja
End Sub

Private Sub Option5_Click()
sisproductes
bloqueja
End Sub

Private Sub Option6_Click()
setproductes
bloqueja
End Sub

Private Sub Option7_Click()
vuitproductes
bloqueja
End Sub

Private Sub Option8_Click()
nouproductes
bloqueja
End Sub

Private Sub Option9_Click()
deuproductes
bloqueja
End Sub

Function dosproductes()
Frame1.Enabled = True
Text1.Enabled = True
Data1.Enabled = True
Label1.Enabled = True
Frame2.Enabled = True
Text2.Enabled = True
Data2.Enabled = True
Label2.Enabled = True
PRODUCTES.Text = "2"
Data1.Refresh
Data2.Refresh
Command1.Enabled = True
End Function

Function tresproductes()
dosproductes
Frame3.Enabled = True
Text3.Enabled = True
Data3.Enabled = True
Label3.Enabled = True
PRODUCTES.Text = "3"
```

```
Data3.Refresh
End Function

Function quatreproductes()
tresproductes
Frame4.Enabled = True
Text4.Enabled = True
Data4.Enabled = True
Label4.Enabled = True
PRODUCTES.Text = "4"
Data4.Refresh
End Function

Function cincproductes()
quatreproductes
Frame5.Enabled = True
Text5.Enabled = True
Data5.Enabled = True
Label5.Enabled = True
PRODUCTES.Text = "5"
Data5.Refresh
End Function

Function sisproductes()
cincproductes
Frame6.Enabled = True
Text6.Enabled = True
Data6.Enabled = True
Label6.Enabled = True
PRODUCTES.Text = "6"
Data6.Refresh
End Function

Function setproductes()
sisproductes
Frame7.Enabled = True
Text7.Enabled = True
Data7.Enabled = True
Label7.Enabled = True
PRODUCTES.Text = "7"
Data7.Refresh
End Function

Function vuitproductes()
setproductes
Frame8.Enabled = True
Text8.Enabled = True
Data8.Enabled = True
Label8.Enabled = True
PRODUCTES.Text = "8"
```

```
Data8.Refresh
End Function

Function nouproductes()
vuitproductes
Frame9.Enabled = True
Text9.Enabled = True
Data9.Enabled = True
Label9.Enabled = True
PRODUCTES.Text = "9"
Data9.Refresh
End Function

Function deuproductes()
nouproductes
Frame10.Enabled = True
Text10.Enabled = True
Data10.Enabled = True
Label10.Enabled = True
PRODUCTES.Text = "10"
Data10.Refresh
End Function

Private Sub Command1_Click()
totalproductes = PRODUCTES
For i = 1 To 10
   w(i) = 0
Next
d = Text11.Text
If PoliRatio = False And PoliDP = False And PoliFIFO = False Or d = 0 Then
Else
If PoliRatio = True Then
   selpoli = 1
   End If
If PoliDP = True Then
   selpoli = 2
   End If
If PoliFIFO = True Then
   selpoli = 3
   End If
If PRODUCTES = 10 Then
   w(1) = Text1.Text
   w(2) = Text2.Text
   w(3) = Text3.Text
   w(4) = Text4.Text
   w(5) = Text5.Text
   w(6) = Text6.Text
   w(7) = Text7.Text
   w(8) = Text8.Text
   w(9) = Text9.Text
```

```
    w(10) = Text10.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 9 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    w(8) = Text8.Text
    w(9) = Text9.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 8 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    w(8) = Text8.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 7 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    w(7) = Text7.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 6 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    w(5) = Text5.Text
    w(6) = Text6.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 5 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
```

```
    w(5) = Text5.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 4 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    w(4) = Text4.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 3 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    w(3) = Text3.Text
    TABU.Hide
    TABU2.Show
ElseIf PRODUCTES = 2 Then
    w(1) = Text1.Text
    w(2) = Text2.Text
    TABU.Hide
    TABU2.Show
End If
End If
End Sub

Private Sub Command2_Click()
TABU.Hide
Unload TABU
INICI.Show
End Sub
Private Sub ENRERE_Click()
amaga
Option1.Enabled = True
Option2.Enabled = True
Option3.Enabled = True
Option4.Enabled = True
Option5.Enabled = True
Option6.Enabled = True
Option7.Enabled = True
Option8.Enabled = True
Option9.Enabled = True
PRODUCTES.Text = ""
End Sub
```

B.2.6. TABU 2

```
Private Sub Form_Load() 'Carreguem la base de dades i fem una primera iteració del procés
Set Basedades = OpenDatabase("c:\Temp\Fcc")
Set taulaproductes = Basedades.OpenRecordset("PRODUCTE", dbOpenTable)
Set taularecursos = Basedades.OpenRecordset("RECURS", dbOpenTable)
Set taularutes = Basedades.OpenRecordset("RUTA", dbOpenTable)
Graella
m = 0
m2 = 1
n = 1
i3 = 1
m3 = 0
minmax
elprimer
generaentorn
For b1 = m2 To m
   AccioACV
Next
For i = 1 To m
Graella1.TextMatrix(l11, 0) = cromo(i, 0)
Graella1.TextMatrix(l11, totalproductes + 1) = cromo(i, 11)
For u = 1 To totalproductes
Graella1.TextMatrix(l11, u) = cromo(i, u)
Next
l11 = l11 + 1
Next
Text1 = n
Text2 = m
Text3 = m2
End Sub

Function Graella() 'Generem les graelles
Graella1.TextMatrix(0, 0) = "Nom"
Graella1.Rows = (((d + 1) * totalproductes) + 1)
Graella1.Cols = (totalproductes + 2)
Graella1.TextMatrix(0, totalproductes + 1) = ("VO")
Graella1.ColWidth(0) = 450
Graella1.ColWidth(totalproductes + 1) = 350
Graella1.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella1.ColWidth(u) = 300
   Graella1.TextMatrix(0, u) = ("P" & w(u))
Next
Graella2.TextMatrix(0, 0) = "Nom"
```

```
Graella2.Rows = ((d * (totalproductes - 2)) + 1)
Graella2.Cols = (totalproductes + 2)
Graella2.TextMatrix(0, totalproductes + 1) = ("VO")
Graella2.ColWidth(0) = 450
Graella2.ColWidth(totalproductes + 1) = 350
Graella2.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella2.ColWidth(u) = 300
   Graella2.TextMatrix(0, u) = ("P" & w(u))
Next
Graella3.TextMatrix(0, 0) = "Nom"
Graella3.Rows = ((d * (totalproductes - 2)) + 1)
Graella3.Cols = (totalproductes + 2)
Graella3.TextMatrix(0, totalproductes + 1) = ("VO")
Graella3.ColWidth(0) = 450
Graella3.ColWidth(totalproductes + 1) = 350
Graella3.Width = (totalproductes * 300) + 1135
For u = 1 To totalproductes
   Graella3.ColWidth(u) = 300
   Graella3.TextMatrix(0, u) = ("P" & w(u))
Next
l11 = 1
l12 = 1
l13 = 1
End Function
Private Sub Command1_Click()
End
End Sub
Private Sub ENRERE_Click()
TABU2.Hide
Unload TABU2
TABU.Show
End Sub
```

Function minmax() 'obtenim el max de rutes per producte

```
taularutes.MoveLast
p(1) = taularutes.Fields("CodiProducte")
p(2) = taularutes.Fields("CodiRuta")
p(3) = taularutes.Fields("OrdreRuta")
For i = 1 To totalproductes 'número de peces
   j = 0 'codi ruta
   s = 0 'nombre operacions
   cp = False
   Do While cp = False 'condició per canviar de peça
      j = j + 1
      s = 1
      taularutes.Index = "PrimaryKey"
      taularutes.Seek "=", w(i), j, s
      cr = False
         Do While cr = False
```

```
            If w(i) = p(1) And j = p(2) And s = p(3) Then
               cr = True
               cp = True
            Else
               taularutes.MoveNext
               i2 = taularutes.Fields("CodiRuta")
               taularutes.MovePrevious
            End If
               If i2 <> j Then
                  If i2 = 0 Then
                     cr = True
                     cp = True
                     End If
                  cr = True
                  s = 0
                  y = 0
               End If
         taularutes.MoveNext
         s = s + 1
         Loop
      max(i) = j
      If i2 <> j And i2 = 1 Then
         cp = True
          End If
   Loop
Next
End Function


Function elprimer() 'generem el primer gen a estudiar
For i = 1 To totalproductes
   Randomize
   g(i) = Rnd
   cromor(1, 0) = 1
   cromor(1, i) = CInt((1 - max(i)) * g(i) + max(i))
Next
cromor(1, 0) = 1
For u = 1 To totalproductes + 1
         Graella1.Row = l11
         Graella1.Col = u
         Graella1.CellBackColor = (190000)
      Next
For u = 1 To totalproductes
   Graella1.TextMatrix(l11, u) = cromor(1, u)
Next
l11 = l11 + 1
End Function

Function generaentorn() 'generem l'entorn del gen
If m = 0 Then
```

```
m2 = 1
ElseIf m <> 0 Then
m2 = (m + 1)
End If
For i2 = 1 To (totalproductes - 1)
   Randomize
   m = m + 1
   cromo(m, 0) = m
   For i = 1 To totalproductes
      If i <> i2 Then
         cromo(m, i) = cromor(n, i)
      ElseIf i = i2 Then
         If cromor(n, i + 1) < max(i) Or cromor(n, i + 1) = max(i) Then
            cromo(m, i) = cromor(n, i + 1)
         Else
            cromo(m, i) = CInt((1 - max(i)) * Rnd + max(i))
         End If

         If cromor(n, i) < max(i + 1) Or cromor(n, i + 1) = max(i) Then
            cromo(m, i + 1) = cromor(n, i)
         Else
            cromo(m, i + 1) = CInt((1 - max(i + 1)) * Rnd + max(i + 1))
         End If
      i = i + 1
      End If
   Next
Next
For i2 = m2 To m
   comprovartabu
Next
End Function

Function comprovartabu() 'Comprovem que l'entorn generat no estigui dins la
llista TABU
Randomize
For i3 = 1 To m3
   s = 0
   For j = 1 To totalproductes
      If cromom(i3, j) = cromo(i2, j) Then
         s = s + 1
      End If
   Next
   If s = totalproductes Then
      cromo(i2, totalproductes) = CInt((1 - max(totalproductes)) * Rnd +
max(totalproductes))
      cromo(i2, CInt(totalproductes / 2)) = CInt((1 - max(CInt(totalproductes / 2))) *
Rnd + max(CInt(totalproductes / 2)))
      cromo(i2, 1) = CInt((1 - max(1)) * Rnd + max(1))
      For u = 1 To totalproductes
         Graella3.TextMatrix(l13, u) = cromo(i2, u)
```

```
      Next
      Graella3.TextMatrix(l13, 0) = (cromo(i2, 0))
      Graella3.TextMatrix(l13, totalproductes + 1) = cromo(i2, 11)
      l13 = l13 + 1
   End If

Next
End Function

Private Sub generar_entorn_Click() 'comencem a calcular
For j2 = 1 To d
   t = cromo(m2, 11)
   i2 = m2
   n = n + 1
   For i = m2 To m
      If cromo(i, 11) < t Then
         t = cromo(i, 11)
         i2 = i
      End If
   Next
   For i = m2 To m
      If i <> i2 Then
         m3 = m3 + 1
         For j = 1 To totalproductes
            cromom(m3, 0) = i
            cromom(m3, j) = cromo(i, j)
         Next
         For u = 1 To totalproductes
            Graella2.TextMatrix(l12, u) = cromo(i, u)
            Graella2.TextMatrix(l12, 0) = i
            Graella2.TextMatrix(l12, totalproductes + 1) = cromo(i, 11)
         Next
         l12 = l12 + 1
      End If
   Next
   For i = 1 To totalproductes
      cromor(n, 0) = n
      cromor(n, i) = cromo(i2, i)
      cromor(n, 11) = cromo(i2, 11)
   Next
   Graella1.TextMatrix(l11, 0) = cromor(n, 0)
   Graella1.TextMatrix(l11, totalproductes + 1) = cromor(n, 11)
   For u = 1 To totalproductes
      Graella1.TextMatrix(l11, u) = cromor(n, u)
   Next
   For u = 1 To totalproductes + 1
      Graella1.Row = l11
      Graella1.Col = u
      Graella1.CellBackColor = (190000)
   Next
```

```
    l11 = l11 + 1
    generaentorn
    For b1 = m2 To m
        AccioACV
    Next
    For i = m2 To m
        For u = 1 To totalproductes
            Graella1.TextMatrix(l11, u) = cromo(i, u)
        Next
        Graella1.TextMatrix(l11, totalproductes + 1) = cromo(i, 11)
        Graella1.TextMatrix(l11, 0) = cromo(i, 0)
        l11 = l11 + 1
    Next
Next
generar_entorn.Enabled = False
Graella3.Rows = l13
End Sub
```

Function AccioACV() 'Càlcul del valor fitness del cromosoma.

```
    ReDim Peza(8, 1 To 10)
    ReDim Ruta(1 To 10)
    TotalMaquines = taularecursos.RecordCount
    ReDim Maq(3, 1 To TotalMaquines)
    ReDim Maxratio(1, 1 To TotalMaquines)
    'ReDim Historic(50, 5, TotalMaquines)  'Prou amb 50 operacions per maq ATENCIO
amb els panells disponibles
    Ample = 0
    taularutes.Index = "PrimaryKey"
    For i = 1 To totalproductes
        Ruta(i) = cromo(b1, i)
    Next
    t = 0
    Demanda = 0
    i = 1   'Compta Maquines
    Do While i <= TotalMaquines 'Netejar Maquines
        Maq(0, i) = 0
        Maq(1, i) = 0
        i = i + 1
    Loop
    Estat = "PENSANT"
    AgafarDades 'INSTRUCCIO PER UNA OPCIO (NOM)
    Do While Estat <> "ACABAT"
        ActualitzarDades
        Carrega = 0
        ComprovarTemps
        Buidar
        ComprovarCues
        'Carregar
        ComptaMaq = 1   'Compta maquines
```

```
      Do While ComptaMaq <= TotalMaquines And Carrega = 0  'Nomes carrego una
maquina en un instant de temps
         If Maq(2, ComptaMaq) > 0 Then
            Carregar    'MODIFICAT
            Carrega = 1
            End If
         ComptaMaq = ComptaMaq + 1
      Loop
      'Acabar
      ComprovarDemanda
      If Demanda = 0 Then
         Estat = "ACABAT"
      Else
         Estat = "PENSANT"
      End If
       cromo(b1, 11) = Ample
   Loop
End Function

Function AgafarDades()
'Instrucció per obtenir les dades que es necessitaran en el transcurs de
'l'execució del programa.
'NOM
For i = 1 To totalproductes
   Peza(0, i) = w(i)
Next
i = 1 'Compta Peces
Do While i <= totalproductes
   'Ultima Op Feta (UOF)
   Peza(1, i) = 0
   'INSTANT DISPONIBILITAT
   Peza(2, i) = 0
   'RUTA
   Peza(3, i) = Ruta(i)
   'RATIO
   Peza(5, i) = -1
   j = 1   'Compta OP
   taularutes.Index = "PrimaryKey"
   taularutes.Seek "=", Peza(0, i), Peza(3, i), j
   Peza(4, i) = 0
   Do While taularutes.NoMatch = False
      Peza(4, i) = taularutes.Fields("Temps") + Peza(4, i) 'Duració Pendent (DP)
      Peza(6, i) = j      'Op d'aquesta Ruta
      j = j + 1
      taularutes.Index = "PrimaryKey"
      taularutes.Seek "=", Peza(0, i), Peza(3, i), j
   Loop
   i = i + 1
Loop
End Function
```

Function ActualitzarDades()

```
'Instrucció per obtenir el temps de l'operació següent de cada peça
'(si existeix) i la màquina que farà aquella operació
j = 1 'Compta peces
Do While j <= totalproductes
   taularutes.Seek "=", Peza(0, j), Ruta(j), Peza(1, j) + 1
   If taularutes.NoMatch = False Then
      Peza(7, j) = taularutes.Fields("CodiRecurs") 'Màquina que la farà
      Peza(8, j) = taularutes.Fields("Temps")      'Durada de la següent OP
   Else
      Peza(7, j) = 0  'Si no hi ha següent OP, la màquina serà la 0
      Peza(8, j) = 1  'Temps per evitar errors de càlcul de DP,...
   End If
   j = j + 1
Loop
```

End Function

Function ComprovarTemps() Funció per comprovar el temps en què està el sistema.

```
   ComprovarDemanda
   If Demanda = 0 Then 'Instant inicial pq encara no s'ha fet cua
      t = 0
   Else
      t = 32000   'Temps exagerat i tira endarrera per tenir en compte nomes les maq que
cal
      i = 1   'Compta maquines
      Do While i <= TotalMaquines
         If t > Maq(1, i) And (Maq(3, i) = 1 Or Maq(2, i) >= 1) Then
            t = Maq(1, i)
         End If
      i = i + 1
      Loop
      i = 1 'Compta maquines
      Do While i <= TotalMaquines 'Garanteixo que el temps no torna enrera
         If t > Maq(1, i) Then
          Maq(1, i) = t
         End If
         i = i + 1
      Loop
   End If
```

End Function

Function Buidar()

```
'Descarrega les màquines que acaben en aquest instant de temps
i = 1 'Compta maquines
Do While i <= TotalMaquines
   If t = Maq(1, i) And Maq(3, i) = 1 And Maq(0, i) <= t Then
      Maq(3, i) = 0   'Queda Lliure
```

```
   End If
   i = i + 1
Loop
End Function

Function ComprovarCues()
'Aquesta funció fa quatre tasques alhora:
'1) Neteja les cues de les màquines per recalcular-les.
'2) Comprova màquina a màquina les peces que hi van,
'3) Mentre comprova els peces, calcula el ratio de cada una.
'4) Mentre calcula el ratio, emmagatzema el màxim de cada màquina.
i = 1   'Compta maquines
Do While i <= TotalMaquines 'Neteja Cues
   If Maq(3, i) = 0 Then   'Nomes d les maq lliures
      Maq(2, i) = 0
   End If
   i = i + 1
Loop
i = 1 'Compta maq
Do While i <= TotalMaquines
   If Maq(3, i) = 0 Then
      Maxratio(1, i) = 0
       j = 1   'Compta Peces
      Do While j <= totalproductes
         If i = Peza(7, j) And Peza(2, j) <= t And Maq(0, i) <= t Then
            Maq(2, i) = Maq(2, i) + 1   'Augmenta la CUA de la maq i
            If selpoli = 1 Then  'Politica Ratio
               If Maq(0, i) <= Peza(2, j) Then 'Calcul Penalitzacio
                  Penalitzacio = Peza(2, j) - Maq(0, i)
               Else
                  Penalitzacio = 0
               End If
               Peza(5, j) = (Peza(4, j) * 100 / (Peza(8, j) + Penalitzacio)) 'RATIO
            ElseIf selpoli = 2 Then
               Peza(5, j) = 30000 / (Peza(4, j) + 1) 'Menys DP=> + Ratio
            ElseIf selpoli = 3 Then
               Peza(5, j) = 30000 / (Peza(2, j) + 1) 'Abans Disponible=> + Ratio
            End If
            If Maxratio(1, i) <= Peza(5, j) Then
               Maxratio(1, i) = Peza(5, j)
               Maxratio(0, i) = j
            End If
         Else
            Peza(5, j) = -1
         End If
         j = j + 1
      Loop
   Else
      Maq(2, i) = 0  'La cua de les ocupades 0 garanteix que no les carregui de nou
   End If
```

```
   i = i + 1
Loop
```
End Function

Function Carregar() Funció que representa el carregar, és a dir, ocupar la màquina, la peça, incrementar les peces fetes, el temps, ... Com que estem en el formulari de Una Opció, memoritza un històric dels moviments per poder-los representar després amb una taula i un gráfic.

```
   taularutes.Index = "PrimaryKey"
   taularutes.Seek "=", Peza(0, Maxratio(0, ComptaMaq)), Peza(3, Maxratio(0,
ComptaMaq)), Peza(1, Maxratio(0, ComptaMaq)) + 1
   At = taularutes.Fields("Temps")
   Peza(1, Maxratio(0, ComptaMaq)) = Peza(1, Maxratio(0, ComptaMaq)) + 1 'Nova
UOF
   Peza(2, Maxratio(0, ComptaMaq)) = t + At 'Disponibilitat de la PeZa
   Peza(4, Maxratio(0, ComptaMaq)) = Peza(4, Maxratio(0, ComptaMaq)) - At    'Nova
DP
   Maq(0, ComptaMaq) = t + At   'Instant de disponibilitat de la maq
   Maq(1, ComptaMaq) = t + At   'Instant de control de la maq
   Maq(3, ComptaMaq) = 1    'Busy (ocupada)
   t = t + At  'Actualitza temps
   If t > Ample Then
      Ample = t   'Aqui AMPLE guarda el temps max (útil per calcular l'amplada
després)
   End If
```
End Function

Function ComprovarDemanda() Funció per saber la demanda general. Serveix per saber si falta fer alguna peça (si n'hi ha alguna que s'espera) o si ja estan totes fetes

```
   i = 1
   DemandaCambiada = 0
   Do While i <= TotalMaquines And DemandaCambiada = 0
      If Maq(2, i) = 0 And Maq(3, i) = 0 Then
         Demanda = 0
      Else
         Demanda = 1
         DemandaCambiada = 1
      End If
   i = i + 1
   Loop
```
End Function

C. Taules de validació.

A continuació adjuntem les taules que hem fet servir per obtenir les taules globals de resultats per als tres métodes. En aquestes taules s'han realitzat els analisis corresponents 10 vegades i se n'ha extret una mitjana que és la que hem posat a la taula de resultats globals. Per a cada mètode s'han realitzat un total de 36 taules de 10 iteracions cadascuna. Després de les taules individuals adjuntem la taula global i llavors cambiem de métode.

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-FIT	Uniforme	Només un element	Baixa	76	0
						81	1
						77	1
						82	2
						77	1
						76	1
						79	4
						75	2
						77	1
						76	1
					MITJANA	77,6	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-FIT	Uniforme	Només un element	Baixa	75	4
						76	1
						75	0
						77	6
						77	5
						78	3
						75	0
						76	1
						76	2
						75	3
					MITJANA	76	2,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-FIT	Uniforme	Només un element	Baixa	77	3
						75	4
						76	3
						76	2
						75	1
						76	3
						75	2
						77	5
						75	6
						76	2
					MITJANA	75,8	3,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-FIT	Uniforme	Només un element	Baixa	76	4
						78	4
						76	2
						78	1
						79	4
						76	6
						78	3
						77	6
						77	4
						76	4
					MITJANA	77,1	3,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-FIT	Uniforme	Només un element	Baixa	76	5
						77	7
						77	8
						76	2
						75	5
						77	3
						77	2
						75	7
						75	9
						77	5
					MITJANA	76,2	5,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-FIT	Uniforme	Només un element	Baixa	76	6
						76	9
						75	5
						75	13
						75	11
						75	8
						76	9
						75	11
						75	11
						76	13
					MITJANA	75,4	9,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-FIT	Uniforme	Només un element	Baixa	76	4
						75	4
						77	10
						77	7
						77	7
						76	5
						76	11
						76	4
						75	8
						76	2
					MITJANA	76,1	6,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-FIT	Uniforme	Només un element	Baixa	75	10
						75	4
						75	7
						75	9
						77	7
						75	4
						76	6
						75	9
						75	12
						76	10
					MITJANA	75,4	7,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-FIT	Uniforme	Només un element	Baixa	75	10
						75	14
						75	5
						75	6
						75	14
						75	6
						75	8
						75	9
						75	7
						75	9
					MITJANA	75	8,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-FIT	Uniforme	Només un element	Alta	77	1
						78	2
						78	1
						75	1
						77	2
						77	2
						77	1
						77	1
						79	4
						78	2
					MITJANA	77,3	1,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-FIT	Uniforme	Només un element	Alta	76	2
						78	4
						77	1
						77	4
						76	3
						77	4
						77	2
						76	2
						75	3
						75	2
					MITJANA	76,4	2,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-FIT	Uniforme	Només un element	Alta	76	5
						75	2
						76	3
						75	3
						76	2
						75	2
						77	1
						76	2
						76	3
						75	3
					MITJANA	75,7	2,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-FIT	Uniforme	Només un element	Alta	75	1
						75	4
						75	4
						76	4
						75	4
						76	6
						75	3
						77	4
						75	9
						76	3
					MITJANA	75,5	4,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-FIT	Uniforme	Només un element	Alta	75	2
						75	10
						75	3
						76	3
						76	4
						76	2
						75	5
						75	2
						75	3
						76	4
					MITJANA	75,4	3,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-FIT	Uniforme	Només un element	Alta	76	3
						75	6
						75	3
						75	11
						75	6
						75	7
						76	5
						75	12
						75	5
						76	4
					MITJANA	75,3	6,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-FIT	Uniforme	Només un element	Alta	75	4
						76	4
						76	4
						76	6
						75	7
						75	3
						76	6
						75	9
						76	7
						75	6
					MITJANA	75,5	5,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-FIT	Uniforme	Només un element	Alta	75	6
						77	5
						75	5
						76	7
						75	3
						75	4
						75	9
						75	8
						75	9
						75	7
					MITJANA	75,3	6,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-FIT	Uniforme	Només un element	Alta	75	13
						75	6
						75	8
						75	6
						75	11
						75	14
						75	7
						75	7
						75	10
						75	8
					MITJANA	75	9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-FIT	Uniforme	Uniforme	Baixa	75	3
						77	3
						75	3
						75	0
						76	2
						77	2
						77	1
						76	0
						78	0
						77	2
					MITJANA	76,3	1,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-FIT	Uniforme	Uniforme	Baixa	79	1
						75	0
						76	4
						76	1
						76	2
						75	1
						75	3
						75	4
						75	1
						75	2
					MITJANA	75,7	1,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-FIT	Uniforme	Uniforme	Baixa	76	2
						75	1
						75	3
						75	1
						75	0
						76	2
						75	1
						76	2
						77	3
						75	2
					MITJANA	75,5	1,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-FIT	Uniforme	Uniforme	Baixa	76	2
						76	4
						77	3
						75	1
						75	4
						76	3
						76	2
						75	2
						78	6
						77	3
					MITJANA	76,1	3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-FIT	Uniforme	Uniforme	Baixa	77	2
						77	5
						76	3
						75	7
						76	5
						77	8
						77	3
						76	5
						75	3
						75	4
					MITJANA	76,1	4,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-FIT	Uniforme	Uniforme	Baixa	75	5
						75	4
						75	5
						76	5
						75	5
						76	5
						75	3
						75	6
						75	5
						75	6
					MITJANA	75,2	4,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-FIT	Uniforme	Uniforme	Baixa	76	4
						75	3
						75	8
						75	8
						76	4
						76	5
						76	6
						75	5
						77	7
						76	4
					MITJANA	75,7	5,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-FIT	Uniforme	Uniforme	Baixa	75	4
						75	4
						75	7
						75	9
						75	7
						75	5
						76	7
						75	7
						75	5
						75	4
					MITJANA	75,1	5,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-FIT	Uniforme	Uniforme	Alta	75	1
						79	2
						76	2
						76	3
						77	1
						79	1
						76	0
						78	1
						75	0
						76	2
					MITJANA	76,7	1,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-FIT	Uniforme	Uniforme	Alta	75	2
						75	3
						76	2
						75	4
						79	1
						75	1
						78	6
						77	1
						77	3
						76	0
					MITJANA	76,3	2,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-FIT	Uniforme	Uniforme	Alta	75	3
						75	1
						75	1
						75	4
						75	4
						75	2
						75	1
						75	3
						75	5
						75	1
					MITJANA	75	2,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-FIT	Uniforme	Uniforme	Alta	76	3
						77	6
						76	7
						76	2
						76	3
						77	6
						76	2
						76	3
						75	5
						76	0
					MITJANA	76,1	3,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-FIT	Uniforme	Uniforme	Alta	76	4
						75	7
						75	6
						76	4
						76	4
						75	3
						75	3
						76	7
						77	7
						75	8
					MITJANA	75,6	5,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-FIT	Uniforme	Uniforme	Alta	75	2
						76	3
						75	4
						75	5
						76	5
						76	2
						76	4
						76	1
						75	1
						77	1
					MITJANA	75,7	2,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-FIT	Uniforme	Uniforme	Alta	75	4
						75	5
						75	6
						75	4
						75	3
						75	2
						75	2
						75	3
						75	5
						75	3
					MITJANA	75	3,7

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
FIT-FIT	UNIFORME	NOMÉS UN ELEMENT	BAIXA	10	10	77,6	1,4
					20	76	2,5
					30	75,8	3,1
				20	10	77,1	3,8
					20	76,2	5,3
					30	75,4	9,6
				30	10	76,1	6,2
					20	75,4	7,8
					30	75	8,8
			ALTA	10	10	77,3	1,7
					20	76,4	2,7
					30	75,7	2,6
				20	10	75,5	4,2
					20	75,4	3,8
					30	75,3	6,2
				30	10	75,5	5,6
					20	75,3	6,3
					30	75	9
		UNIFORME	BAIXA	10	10	76,3	1,6
					20	75,7	1,9
					30	75,5	1,7
				20	10	76,1	3
					20	76,1	4,5
					30	75,2	4,9
				30	10	75,7	5,4
					20	75,1	5,9
					30	75	6
			ALTA	10	10	76,7	1,3
					20	76,3	2,3
					30	75	2,5
				20	10	76,1	3,7
					20	75,6	5,3
					30	----	----
				30	10	75,7	2,8
					20	75	3,7
					30	75	4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-WEAK	Uniforme	Només un element	Baixa	75	2
						77	0
						76	1
						82	4
						78	3
						75	3
						80	1
						79	1
						77	6
						76	2
					MITJANA	77,5	2,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-WEAK	Uniforme	Només un element	Baixa	75	5
						76	1
						77	3
						77	3
						76	4
						75	3
						75	1
						75	2
						77	5
						75	4
					MITJANA	75,8	3,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-WEAK	Uniforme	Només un element	Baixa	75	1
						76	5
						75	2
						76	2
						75	0
						77	4
						75	4
						76	3
						76	1
						77	4
					MITJANA	75,8	2,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-WEAK	Uniforme	Només un element	Baixa	79	3
						77	9
						79	9
						78	6
						76	1
						76	6
						75	9
						79	6
						80	4
						75	5
					MITJANA	77,4	5,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-WEAK	Uniforme	Només un element	Baixa	76	3
						77	11
						77	5
						75	7
						76	9
						76	5
						77	5
						76	5
						75	4
						76	2
					MITJANA	76,1	5,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-WEAK	Uniforme	Només un element	Baixa	77	7
						76	11
						75	5
						75	8
						75	10
						75	9
						76	7
						75	9
						75	9
						75	5
					MITJANA	75,4	8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-WEAK	Uniforme	Només un element	Baixa	76	11
						76	8
						75	3
						77	4
						79	9
						76	9
						76	8
						75	3
						76	3
						77	6
					MITJANA	76,3	6,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-WEAK	Uniforme	Només un element	Baixa	76	4
						75	15
						75	5
						76	9
						76	18
						75	9
						75	10
						76	8
						76	8
						76	7
					MITJANA	75,6	9,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-WEAK	Uniforme	Només un element	Baixa	76	8
						76	6
						75	11
						76	10
						75	8
						75	13
						76	8
						76	10
						75	9
						75	12
					MITJANA	75,5	9,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-WEAK	Uniforme	Només un element	Alta	75	2
						77	1
						75	2
						77	6
						77	4
						76	3
						77	2
						79	1
						76	1
						79	0
					MITJANA	76,8	2,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-WEAK	Uniforme	Només un element	Alta	77	3
						77	1
						75	3
						76	3
						75	2
						77	7
						76	3
						76	2
						77	2
						76	2
					MITJANA	76,2	2,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-WEAK	Uniforme	Només un element	Alta	77	3
						75	2
						77	2
						77	0
						75	4
						77	4
						76	1
						75	4
						76	0
						76	2
					MITJANA	76,1	2,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-WEAK	Uniforme	Només un element	Alta	77	4
						76	5
						76	4
						77	1
						77	4
						76	2
						75	4
						77	2
						78	4
						77	6
					MITJANA	76,6	3,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-WEAK	Uniforme	Només un element	Alta	76	3
						75	6
						79	6
						75	4
						76	3
						76	4
						75	6
						76	5
						78	8
						76	6
					MITJANA	76,2	5,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-WEAK	Uniforme	Només un element	Alta	75	7
						76	4
						75	3
						76	6
						77	6
						76	5
						77	3
						76	4
						76	3
						76	4
					MITJANA	76	4,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-WEAK	Uniforme	Només un element	Alta	77	8
						75	5
						79	4
						76	7
						75	6
						75	5
						77	5
						75	5
						75	5
						75	8
					MITJANA	75,5	9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-WEAK	Uniforme	Només un element	Alta	75	5
						75	10
						75	11
						75	10
						75	8
						76	12
						75	10
						76	5
						76	10
						76	11
					MITJANA	75,4	9,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-WEAK	Uniforme	Només un element	Alta	75	8
						75	5
						76	12
						75	10
						76	7
						75	9
						76	8
						75	11
						76	9
						76	11
					MITJANA	75,5	9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-WEAK	Uniforme	Uniforme	Baixa	77	3
						76	2
						77	0
						75	2
						76	1
						75	2
						78	5
						79	0
						76	1
						79	4
					MITJANA	76,8	2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-WEAK	Uniforme	Uniforme	Baixa	78	3
						75	3
						75	3
						76	3
						75	3
						76	3
						77	4
						79	1
						76	1
						75	5
					MITJANA	76,2	2,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-WEAK	Uniforme	Uniforme	Baixa	75	1
						76	2
						75	2
						76	1
						76	2
						76	2
						76	1
						76	0
						75	2
						77	3
					MITJANA	75,8	1,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-WEAK	Uniforme	Uniforme	Baixa	76	4
						77	3
						77	6
						77	4
						75	2
						77	3
						76	2
						77	2
						79	2
						77	4
					MITJANA	76,8	3,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-WEAK	Uniforme	Uniforme	Baixa	76	2
						76	8
						76	6
						76	6
						76	3
						76	5
						75	4
						76	3
						76	3
						77	5
					MITJANA	76,0	4,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-WEAK	Uniforme	Uniforme	Baixa	75	6
						75	10
						75	6
						75	6
						76	7
						75	7
						76	7
						75	5
						76	7
						75	3
					MITJANA	75,3	6,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-WEAK	Uniforme	Uniforme	Baixa	76	7
						77	7
						76	3
						77	7
						76	5
						75	7
						76	5
						76	5
						77	6
						75	4
					MITJANA	76,1	5,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-WEAK	Uniforme	Uniforme	Baixa	76	8
						76	13
						75	5
						75	4
						75	9
						76	3
						75	6
						75	9
						76	5
						75	7
					MITJANA	75,4	6,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-WEAK	Uniforme	Uniforme	Baixa	75	10
						76	11
						75	10
						76	10
						75	2
						75	10
						75	11
						75	11
						75	8
						75	9
					MITJANA	75,2	9,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	FIT-WEAK	Uniforme	Uniforme	Alta	76	1
						79	1
						77	2
						75	1
						76	1
						76	0
						75	0
						75	3
						75	0
						76	0
					MITJANA	76	0,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	FIT-WEAK	Uniforme	Uniforme	Alta	75	0
						75	1
						75	1
						77	2
						75	0
						75	0
						77	1
						77	1
						78	4
						75	2
					MITJANA	75,9	1,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	FIT-WEAK	Uniforme	Uniforme	Alta	75	2
						75	3
						75	1
						75	2
						77	0
						75	1
						76	4
						75	0
						75	3
						75	2
					MITJANA	75,3	1,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	FIT-WEAK	Uniforme	Uniforme	Alta	75	1
						75	1
						79	1
						77	2
						75	3
						75	2
						76	0
						75	2
						76	5
						75	2
					MITJANA	75,8	1,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	FIT-WEAK	Uniforme	Uniforme	Alta	76	2
						75	1
						75	2
						75	3
						75	2
						76	4
						75	2
						75	3
						76	2
						75	5
					MITJANA	75,3	2,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	FIT-WEAK	Uniforme	Uniforme	Alta	75	6
						75	7
						75	2
						75	6
						76	2
						76	4
						75	4
						76	1
						75	5
						75	2
					MITJANA	75,3	3,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	FIT-WEAK	Uniforme	Uniforme	Alta	77	4
						75	4
						78	2
						76	4
						75	1
						75	3
						75	2
						75	3
						75	7
						75	7
					MITJANA	75,6	3,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	FIT-WEAK	Uniforme	Uniforme	Alta	75	8
						75	4
						75	3
						76	5
						75	6
						75	7
						75	3
						75	6
						75	5
						75	5
					MITJANA	75,1	5,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	FIT-WEAK	Uniforme	Uniforme	Alta	75	8
						75	9
						75	7
						75	5
						75	9
						75	8
						75	7
						75	6
						75	7
						75	4
					MITJANA	75	7

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
FIT-WEAK	UNIFORME	NOMÉS UN ELEMENT	BAIXA	10	10	77,5	2,3
					20	75,8	3,1
					30	75,8	2,6
				20	10	77,4	5,8
					20	76,1	5,6
					30	75,4	8
				30	10	76,3	6,4
					20	75,6	9,3
					30	75,5	9,5
			ALTA	10	10	76,8	2,2
					20	76,2	2,8
					30	76,1	2,2
				20	10	76,6	3,6
					20	76,2	5,1
					30	76	4,5
				30	10	75,5	9
					20	75,4	9,2
					30	75,5	9
		UNIFORME	BAIXA	10	10	76,8	2
					20	76,2	2,9
					30	75,8	1,6
				20	10	76,8	3,2
					20	76	4,5
					30	75,3	6,4
				30	10	76,1	5,6
					20	75,4	6,9
					30	75,2	9,2
			ALTA	10	10	76	0,9
					20	75,9	1,2
					30	75,3	1,8
				20	10	75,8	1,9
					20	75,3	2,6
					30	75,3	3,9
				30	10	75,6	3,7
					20	75,1	5,2
					30	75	7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	RULETA	Uniforme	Només un element	Baixa	75	1
						75	0
						81	1
						75	0
						81	0
						79	4
						75	1
						79	3
						76	1
						77	1
					MITJANA	77,3	1,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	RULETA	Uniforme	Només un element	Baixa	81	2
						76	0
						75	3
						77	1
						75	3
						81	3
						76	0
						75	1
						75	1
						75	4
					MITJANA	76,6	1,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	RULETA	Uniforme	Només un element	Baixa	75	1
						76	2
						81	3
						77	2
						75	1
						75	2
						76	3
						76	2
						75	2
						77	3
					MITJANA	76,3	2,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	RULETA	Uniforme	Només un element	Baixa	75	9
						75	8
						77	9
						79	7
						77	8
						79	2
						77	5
						75	5
						75	5
						77	8
					MITJANA	76,6	6,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	RULETA	Uniforme	Només un element	Baixa	75	4
						77	6
						81	7
						75	8
						76	4
						79	5
						75	6
						78	5
						77	7
						75	8
					MITJANA	76,8	6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	RULETA	Uniforme	Només un element	Baixa	75	5
						77	4
						75	7
						77	5
						76	5
						75	6
						75	9
						77	8
						76	6
						75	11
					MITJANA	75,8	6,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	RULETA	Uniforme	Només un element	Baixa	77	9
						77	10
						82	14
						76	8
						75	8
						75	10
						75	9
						77	8
						77	10
						75	9
					MITJANA	76,6	9,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	RULETA	Uniforme	Només un element	Baixa	75	11
						76	9
						76	15
						75	14
						77	5
						75	10
						75	10
						75	12
						75	9
						76	9
					MITJANA	75,5	10,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	RULETA	Uniforme	Només un element	Baixa	75	10
						75	9
						75	10
						76	6
						75	12
						75	9
						75	4
						75	10
						75	9
						75	17
					MITJANA	75,1	9,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	RULETA	Uniforme	Només un element	Alta	76	1
						76	0
						77	2
						76	1
						77	0
						75	1
						81	3
						79	0
						76	3
						77	3
					MITJANA	77	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	RULETA	Uniforme	Només un element	Alta	75	2
						77	1
						75	0
						77	1
						76	0
						76	4
						75	2
						75	2
						75	2
						76	0
					MITJANA	75,7	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	RULETA	Uniforme	Només un element	Alta	77	0
						77	3
						76	0
						75	1
						77	2
						77	2
						75	2
						76	1
						75	1
						77	2
					MITJANA	76,2	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	RULETA	Uniforme	Només un element	Alta	75	2
						75	2
						76	5
						77	3
						77	4
						75	5
						75	2
						78	3
						76	3
						75	2
					MITJANA	75,9	3,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	RULETA	Uniforme	Només un element	Alta	75	0
						76	8
						75	4
						75	4
						75	9
						75	4
						76	1
						75	6
						75	7
						76	4
					MITJANA	75,3	4,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	RULETA	Uniforme	Només un element	Alta	75	6
						75	6
						75	5
						75	5
						76	3
						75	10
						76	3
						76	6
						76	3
						75	2
					MITJANA	75,4	4,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	RULETA	Uniforme	Només un element	Alta	77	4
						77	7
						75	5
						75	3
						75	2
						79	5
						75	8
						76	2
						75	7
						76	3
					MITJANA	76	4,6

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	RULETA	Uniforme	Només un element	Alta	75	7
						75	4
						76	3
						76	9
						77	1
						76	5
						76	2
						77	2
						75	5
						75	3
					MITJANA	75,8	4,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	RULETA	Uniforme	Només un element	Alta	75	3
						76	3
						75	7
						76	5
						75	8
						75	8
						75	6
						75	6
						75	10
						75	6
					MITJANA	75,2	6,2

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	RULETA	Uniforme	Uniforme	Baixa	77	1
						78	0
						75	1
						75	2
						75	2
						75	0
						77	0
						77	4
						77	1
						77	2
					MITJANA	76,3	1,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	RULETA	Uniforme	Uniforme	Baixa	75	2
						75	1
						75	2
						76	1
						75	2
						75	0
						75	2
						75	1
						77	1
						76	2
					MITJANA	75,4	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	RULETA	Uniforme	Uniforme	Baixa	75	1
						75	2
						76	2
						77	0
						75	0
						75	2
						75	2
						75	3
						75	2
						75	1
					MITJANA	75,3	1,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	RULETA	Uniforme	Uniforme	Baixa	77	3
						75	4
						77	3
						75	5
						75	3
						75	4
						75	5
						75	3
						76	4
						75	6
					MITJANA	75,5	4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	RULETA	Uniforme	Uniforme	Baixa	75	2
						77	5
						76	2
						75	3
						76	4
						75	2
						75	3
						75	6
						75	4
						75	3
					MITJANA	75,4	3,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	RULETA	Uniforme	Uniforme	Baixa	75	3
						75	6
						75	2
						75	6
						76	1
						75	4
						75	6
						76	2
						76	0
						75	3
					MITJANA	75,3	3,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	RULETA	Uniforme	Uniforme	Baixa	75	4
						76	3
						76	4
						75	5
						75	6
						75	5
						75	6
						75	8
						76	5
						75	5
					MITJANA	75,3	5,1

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	RULETA	Uniforme	Uniforme	Baixa	75	2
						75	2
						77	2
						77	3
						75	7
						75	7
						75	7
						76	7
						75	5
						75	7
					MITJANA	75,5	4,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	30	RULETA	Uniforme	Uniforme	Baixa	75	10
						76	6
						76	5
						75	3
						76	4
						75	4
						75	8
						75	9
						76	7
						75	3
					MITJANA	75,4	5,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	10	RULETA	Uniforme	Uniforme	Alta	79	1
						79	0
						77	1
						75	1
						75	3
						81	0
						78	0
						78	1
						79	1
						75	1
					MITJANA	77,6	0,9

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	10	RULETA	Uniforme	Uniforme	Alta	75	0
						75	0
						75	0
						75	1
						75	2
						75	3
						75	2
						76	3
						75	2
						75	2
					MITJANA	75,1	1,5

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	10	RULETA	Uniforme	Uniforme	Alta	75	2
						75	2
						75	1
						75	3
						75	2
						75	3
						75	3
						75	2
						75	1
						75	4
					MITJANA	75	2,3

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	20	RULETA	Uniforme	Uniforme	Alta	76	4
						76	1
						75	2
						76	0
						76	2
						77	1
						77	2
						77	0
						75	0
						75	2
					MITJANA	76	1,4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	20	RULETA	Uniforme	Uniforme	Alta	75	2
						75	4
						75	2
						75	5
						75	2
						75	1
						75	1
						75	5
						75	3
						76	3
					MITJANA	75,1	2,8

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
30	20	RULETA	Uniforme	Uniforme	Alta	75	8
						75	4
						75	5
						75	3
						75	2
						75	5
						75	2
						75	2
						75	2
						75	4
					MITJANA	75	3,7

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
10	30	RULETA	Uniforme	Uniforme	Alta	75	6
						76	3
						75	3
						75	3
						75	7
						76	5
						76	0
						76	4
						75	5
						77	4
					MITJANA	75,6	4

Població	Iteracions	Sel·lecció progenitors	Creuament	Mutació	Probabilitat de mutació	VO	Nombre de mitjanes =
20	30	RULETA	Uniforme	Uniforme	Alta	75	1
						75	3
						75	5
						75	5
						75	6
						75	8
						75	6
						75	4
						75	4
						75	5
					MITJANA	75	4,7

Sel.	Creu.	Mutació	Probabilitat de mutació	Iteracions	Població	VO	Nombre de mitjanes =
RULETA	UNIFORME	NOMÉS UN ELEMENT	BAIXA	10	10	77,3	1,2
					20	76,6	1,8
					30	76,3	2,1
				20	10	76,6	6,6
					20	76,8	6
					30	75,8	6,6
				30	10	76,6	9,5
					20	75,5	10,4
					30	75,1	9,6
			ALTA	10	10	77	1,1
					20	75,7	1,4
					30	76,2	1,4
				20	10	75,9	3,1
					20	75,3	4,7
					30	75,4	4,9
				30	10	76	4,6
					20	75,8	4,1
					30	75,2	6,2
		UNIFORME	BAIXA	10	10	76,3	1,3
					20	75,4	1,4
					30	75,3	1,5
				20	10	75,5	4
					20	75,4	3,4
					30	75,3	3,3
				30	10	75,3	5,1
					20	75,5	4,9
					30	75,4	5,9
			ALTA	10	10	77,6	0,9
					20	75,1	1,5
					30	75	2,3
				20	10	76	1,4
					20	75,1	2,8
					30	75	3,7
				30	10	75,6	4
					20	75	4,7
					30	75	5