Resumen

El trabajo presenta una introducción a los conceptos básicos de la estrategia de producción al nivel operativo y propone una aplicación al resolver mediante una meta heurística de tipo evolutivo, los algoritmos genéticos (AGs), para un problema de taller mecánico compuesto de maquinas idénticas en paralelo recibiendo un conjunto de piezas a procesar. El objetivo que se propone resolver es minimizar a la vez el retraso medio de las piezas Tmed respecto a una fecha de entrega y el tiempo medio de permanencia en el sistema productivo Fmed.

A partir de este objetivo multicriterio, el algoritmo propone una programación virtual del taller que se encuentre en la frontera de Pareto de este sistema bicriterio (Fmed, Tmed). Se ha implementado en un programa informático que tiene el nombre de "Parallel Machine Management", en el entorno de trabajo del software *Microsoft Visual Basic 6.0*.

Los resultados computacionales que se obtienen facilitan al usuario la programación requerida con un relativo ahorro de tiempo explorando el espacio de búsqueda de manera eficiente. Se proponen unos valores orientativos de los parámetros del AG.

Palabras clave: Optimización bicriterio, máquinas idénticas paralelas, algoritmos genéticos, frontera de Pareto.

Sumario

RESUMEN 1

SUMARIO 3

GLOSARIO 6

PREFACIO 7

Origen del proyecto 7

Motivación y contexto actual en España 8

1. INTRODUCCIÓN 9

1.1. Objetivos del proyecto y alcance 9

1.2. Organización del trabajo 10

2. GESTIÓN DE LA PRODUCCIÓN: ASPECTOS ESTRATÉGICOS Y OPERATIVOS 11

2.1. Los sistemas de Producción 11

2.2. Clasificación de los problemas de Programación 13

2.2.1. Implantación de los procesos de producción. Modelos básicos de disposición. 13

2.2.2. Problemas de programación de taller mecánico. Definición. 14

3. ESTADO DEL ARTE SOBRE LOS TRABAJOS DE OPTIMIZACIÓN Y META HEURÍSTICAS. 19

3.1. Los métodos de resolución combinatorios. 19

3.2. Los métodos exactos 19

3.3. Las Meta Heuristicas 20

3.3.1. El enfoque constructivo 21

3.3.2. El enfoque iterativo (Búsqueda Local) 22

3.3.3. El enfoque evolutivo 23

3.4. Los Algoritmos Genéticos 23

3.4.1. Representación de una solución 24

3.4.2. La población inicial 24

3.4.3. El Fitness 25

3.4.4. La selección de los progenitores 25

3.4.5. El cruce 25

3.4.6. La mutación 25

3.5. Estado del arte sobre los problemas multicriterios 27

3.5.1. Transformación de un problema multi objetivos en uniobjetivo. 27

3.5.2. Enfoque no Pareto 28

3.5.3. Enfoque de Pareto 29

4. ALGORITMO GENÉTICO PARA EL PROBLEMA FMED-TMED CON MÁQUINAS IDÉNTICAS EN PARALELO. 32
4.1. Planteo, datos de partida del problema 32
4.2. Codificación de la solución 34
4.3. Población Inicial 40
4.3.1. Tamaño de la población inicial 40
4.3.2. Composición y construcción de la población inicial. 40
4.4. Medida de evaluación o Fitness 44
4.4.1. Cálculo de las cotas 46
4.4.2. Obtención del fitness de una solución 51
4.5. Población de una nueva generación 55
4.5.1. Número de generaciones 55
4.5.2. Elección de los padres en los cruces 56
4.5.3. El cruce 57
4.5.4. Mutaciones 63
4.6. Composición de la población generada 66
4.7. Resultados gráficos, dominio de Pareto. 68
4.7.1. Construcción del dominio de Pareto 68
4.7.2. Problemas de nichos en la frontera de Pareto. 70
5. EXPERIENCIA COMPUTACIONAL 72
5.1. Análisis de los ejemplares 72
5.2. Estudio del Tipo de Fitness 74
5.3. Test de parámetros del algoritmo genético 76
5.3.1. Tamaño de la población inicial y de las diferentes generaciones. 78
5.3.2. Número de generaciones 78
5.3.3. Combinación de los operadores de cruce y mutación 79
5.3.4. Estudio del comportamiento estacionario de las soluciones 84
6. PRESUPUESTO 85
7. ESTUDIO DEL IMPACTO MEDIO AMBIENTAL 86
7.1. Impacto del posible retraso 86
7.2. Impacto de los costes operativos 86
8. CONCLUSIONES 88
AGRADECIMIENTOS 90
BIBLIOGRAFÍA 91
Referencias bibliográficas 91
Bibliografía complementaria 93

ANEXO A: CORRESPONDENCIA CON LOS DATOS Y LOS NOMBRES UTILIZADOS EN EL PROGRAMA. 5

ANEXO B: MEJORES RESULTADOS PARA EL ESTUDIO DEL MEJOR TIPO DE FITNESS. 10

B.1 RESULTADO DE LA ÚLTIMA POBLACIÓN CON EL FITNESS CONTINÚO: 10

B.2 RESULTADO DE LA ÚLTIMA POBLACIÓN CON EL FITNESS DIAGONAL: 11

B.3 RESULTADO DE LA ÚLTIMA POBLACIÓN CON EL FITNESS PERPENDICULAR: 13

ANEXO C MEJORES RESULTADOS PARA EL ESTUDIO DE LOS PARÁMETROS DEL ALGORITMO GENÉTICO. 15

C.1 MEJORES RESLUTADOS PARA DETERMINAR EL MEJOR VALOR DEL NÚMERO DE GENRACIONES. 15

C.2 MEJORES RESULTADOS POR EJEMPLAR PARA LA DETERMINACIÓN DE LA COMBINACIÓN MÁS ADECUADA. 26

ANEXO D: MANUAL DE EMPLEO DEL PROGRAMA "PARALLEL MACHINE MANAGEMENT" 88

D.1 ARRANQUE DEL PROGRAMA 88

D.2 OPCIÓN NUEVO PROYECTO 89

D.3 OPCIÓN ABRIR EJEMPLAR 89

D.4 OPCIÓN ABRIR COLECCIÓN 90

D.5 INTRODUCCIÓN DE LOS DATOS 91

D.5.1 LEER NUEVO 91

D.5.2. LEER EJEMPLAR Y LEER COLECCIÓN 94

D.6 OPCIONES 95

D.7 EJECUTAR 97

ANEXO E: CODIFICACIÓN DEL PROGRAMA EN LENGUAJE BASIC 99

E.1 FORMULARIO MENUPRINCIPAL 99

E.2 FORMULARIO ABRIRNUEVO 99

E.3 FORMULARIO ABRIREJEMPLAR 101

E.4 FORMULARIO ABRIRCOLECCION 104

E.5 MODULO MODULE1 107

E.6 MODULO MODULE2 110

E.7 MODULO MODULE3 131

E.8 MODULO MODULE4 141

E.9 MODULO MODULE5 152

GLOSARIO

Problema de tipo ***Pm/d_i/$s_{h,i}$/,$\overline{F}$, $\overline{T}$*** **[1].**

n : número de piezas (number of jobs) del taller mecánico

m : número de maquinas del taller mecánico (en paralelo y todas idénticas)

r_i : instante de entrada de la pieza **i** en el taller (i=1....n)

d_i : fecha de entrega de la pieza **i** (i=1....n)

p_i : tiempo de proceso de la pieza **i** (i=1....n)

w_i : tiempo total de espera de la pieza **i** (i=1....n)

c_k : instante de salida de la pieza **i** (i=1....n) $c_i = r_i + w_i + p_i$

F_i : tiempo de permanencia de la pieza **i** $F_i = c_i - r_i = w_i + p_i$

L_i : huelgo de la pieza **i** $L_i = c_i - d_i$

T_i : retraso de la pieza **i** $T_i = \max \{L_i, 0\}$

$S_{i,k}$: tiempo de cambio de instalación (setup) entre la pieza i y la pieza k

Nota: Como las máquinas son idénticas no hace falta añadir un índice más para detallar la máquina en la cual se hace el cambio; los tiempos son iguales para todas las máquinas.

Medidas de Eficiencia:

$\overline{F} = \Sigma \{F_i\}/n$

$\overline{T} = \Sigma \{T_i\}/n$

PREFACIO

"It's been a long road, getting from there to here"
"It's been a long time, but my time is finally near"
Diane Warren – Where My Heart Will Take Me (Faith of the Heart)

"El noventa por ciento del éxito se basa simplemente en insistir"
p.19 "La Buena Suerte" Alex Rovira Celma, Fernando Trias de Bes

Origen del proyecto

¿Por qué un proyecto de gestión de sistemas industriales?

El entorno actual de las empresas está caracterizado por mercados sometidos a una fuerte competencia y sobre los cuales las exigencias y las expectativas de los clientes se vuelven cada vez más fuertes en término de calidad, de coste, y de plazo de puesta a disposición.

Esta evolución se acentúa con el desarrollo rápido de nuevas tecnologías de información y de la comunicación que favorecen una relación directa entre las empresas (*Business to Business*) y las empresas con sus clientes (*Business to Customer*).

En este contexto, permanecer siempre eficiente pasa por el fomento de la Innovación y el estudio de su entorno de negocios donde las enseñanzas de Inteligencia Competitiva son imprescindibles.

La Innovación pasa también por la mejora continua de las herramientas usadas en producción. Esta representa una de las mayores preocupaciones para los gestores donde las nociones de *reactividad*, de *costes* y de *calidad* tienen más y más importancia.

En efecto, poder apoyarse en un sistema de producción eficiente y sin fallo a cada instante es vital.

Por lo tanto, estas múltiples restricciones *mercantiles*, *productivas*, y *financieras* tienen incidencias mayores sobre funciones que están muy estrechamente relacionadas con la producción al nivel operacional.

Estos lugares de la producción, con estas funciones vecinas, se manejan a través de la dirección de operaciones, que tiene como misión de *actuar sobre el sistema productivo, con la misión de su gobierno y su control*.

Motivación y contexto actual en España

Fig. 0.1 Estructura y repartición de la Industria en España.

Últimamente, se ha hecho caso en la prensa[1] del estado de las nuevas tendencias en la Industria en España y de la necesidad de las Inteligencias Artificiales (AI) o Tecnologías Inteligentes para conseguir negocios de alto rendimiento.

Hoy en día, se hace hincapié en los vínculos entre velocidad y eficiencia, y como los sistemas modernos de producción tienen que ser flexibles y eficientes para poder acelerar los tiempos de respuesta. Se trata de conseguir una alta diversidad de productos y ciclos de producción más cortos.

En este contexto de globalización y eficiencia y según esta visión, se inició el proyecto que sigue

[1] El País, suplemento elaborado por Siemens, *"De la idea al producto"*, 5 de Mayo 2008.

1. Introducción

1.1. Objetivos del proyecto y alcance

Cuando se pregunta "¿cuál es el objetivo de una empresa?", una posible respuesta sería que, básicamente, el output final de la cadena de valor de cualquiera empresa debe ser un producto o servicio adecuado a las necesidades de los consumidores.

Esta entidad de salida tiene que ser de máxima ***calidad*** perceptible por los mismos, obtenida con el mínimo empleo de recursos, y por tanto con el mínimo ***coste***, lo que se traduce en la máxima ***productividad*** y también con el mínimo ***tiempo***.

Este proyecto demuestra cómo una gestión industrial al nivel operativo puede apoyar el sistema de decisión al nivel estratégico.

El interés creciente para los Sistemas de Información en la gestión empresarial permite hoy en día utilizar los enfoques de Inteligencia Artificial en la gestión de la producción.

Este proyecto tiene el alcance de diseñar un software de gestión que utilice procedimientos inteligentes para una empresa industrial compuesta de una planta de tipo taller mecánico.

En esta planta se encuentran **m máquinas dispuestas en paralelo** recibiendo **n piezas**. Además, estas máquinas son **idénticas**, y tienen la misma velocidad de procesamiento.

Los datos de las entradas (o inputs) introducidos por el usuario o sacados de ficheros especiales serán comprendidos por el sistema productivo, analizados, evaluados, y el software tomará las decisiones adecuadas con las salidas (o outputs) que cumplan con las medidas de eficiencia seleccionadas.

En este caso, los outputs serán la carga por cada máquina así como la secuenciación propuesta por máquina.

Igualmente, la toma de decisión por los responsables de producción en el mundo real es de naturaleza multicriterio. En la práctica, efectuar una elección perfecta en todos puntos de vista es contado. Por tanto, se estudiará el comportamiento y la optimización de este taller de manera bicriterio, y se hallará como minimizar la estancia media de salida de las piezas, **Fmed,** y el retraso medio, **Tmed**.

Un generador de ejemplares permitirá hacer el estudio completo del taller con el fin de optimizar los parámetros de cálculo del software. Y se construirá gráficamente la frontera de

Pareto de cada ejemplar. El estudio se hará mediante la utilización de un algoritmo implementado sobre el programa informático Visual Basic 6.0.

La complejidad del problema debido a las incertidumbres, vinculadas a los datos y objetivos a alcanzar, hace pensar que un enfoque por métodos exactos no permita responder a los objetivos marcados en un tiempo razonable, aunque sea eficiente. Por lo tanto, se ha investigado un procedimiento meta-heurístico para resolver este problema.

1.2. Organización del trabajo

En primer lugar, se aborda sucintamente los sistemas de producción en general, explorando la gestión de la producción por sus aspectos estratégicos y operativos. Se propone una definición del sistema de producción manufacturero dentro de la empresa, así como una breve presentación de la programación de la producción.

En segundo lugar, se hace un pequeño estado del arte de los trabajos ya existentes y relativos a la programación de la producción incluso los métodos meta-heurísticos que se ha decidido utilizar, *los algoritmos genéticos*. Se hará también una breve introducción al concepto de optimización multi-criterio.

A continuación, se aplica el método de resolución a un caso concreto con la ayuda del algoritmo elegido. La programación del software se hace con *Visual Basic 6.0*, con la intención de desempeñar un programa de ayuda a la toma de decisiones y el software diseñado tiene el nombre de "*Parallel Machine Management*".

Más adelante, se hace un análisis de los resultados encontrados como resultado de la experiencia computacional. Se utilizan colecciones de ejemplares con diferentes características para garantizar un estudio completo.

Para finalizar este trabajo se presenta un presupuesto para la utilización industrial del software, así como un breve estudio del impacto ambiental.

La conclusión es el medio de abrir el debate a las expectativas y trabajos futuros en el campo de la gestión de la producción.

2. Gestión de la Producción: aspectos estratégicos y operativos

En el nuevo contexto industrial, las empresas sufren presiones de los clientes. Estos últimos se vuelven más y más exigentes y piden productos y servicios de buena calidad, a menor coste, entregados rápidamente y en el momento propicio, con un servicio post venta que excluya toda competencia. Para satisfacer la demanda en calidad y cantidad respetando a los plazos de entrega y los costes, la empresa-planta debe disponer de una herramienta de producción eficiente. La principal consecuencia del desarrollo industrial es la complejidad creciente de las máquinas y de los equipos de producción. Por tanto, el desarrollo de los talleres mecánicos debe integrar automatización y flexibilidad.

A continuación se repasa en breve los diferentes sistemas de producción:

2.1. Los sistemas de Producción

"*Lo que no añade valor al producto es despilfarro*".
Henri FORD

Un *proceso de producción* está generalmente compuesto de un gran número de *operaciones* o *transformaciones* organizadas en *redes*.

Dichas operaciones aseguran transformaciones de forma (modificación de los productos ellos mismos), en el tiempo (función de almacenamiento) o en el espacio (función de transporte).

Los sistemas de producción difieren por los objetivos que el productor se asigna, los atributos de los objetos transformados y por las características de los procesos de producción.

No son estáticos y evolucionan bajo los efectos del mercado, de las tecnologías, y de las sociedades en las cuales están incluidos.

Se definen por los *outputs* (o *atributos de salida*), coste de producto, calidad de producto, cantidad de producto por unidad de tiempo, plazos medios de entrega de producto o servicio, etc.

Se puede clasificar los sistemas de producción de bienes en dos grandes clases:

- ***Los procesos continuos (enfocados al producto)*** tal como la producción eléctrica, química o la papelería.

- ***Los procesos intermitentes o discretos (enfocados al proceso)*** tal como la mecanización o todas las actividades de ajuste, ensamblaje. Esta visión de los sistemas es muy frecuente en las plantas industriales como la que se estudia en este proyecto.

Así pues, los sistemas de producción en las plantas industriales se caracterizan por su naturaleza discreta.

La teoría de sistemas aplicada al programa que se pretende diseñar aquí, permite descomponer el sistema de programación en tres sub-sistemas: el *sistema físico de producción*, el *sistema de información* y el *sistema de decisión*.

El Programa llamado "*Parallel Machine Management*" está constituido por la parte del sistema de decisión y del sistema de información tratando de las funciones relacionadas directamente con la producción física (Por ejemplo compras, aprovisionamiento, planificación, gestión de recursos, mantenimiento, etc.).

En términos de sistema, el sistema físico (o planta productora) transforma las materias primas en productos acabados (ver Figura 2.1.).

Existe un bucle, ya que en retorno el sistema de decisión recibe las informaciones del seguimiento del sistema físico para poder pilotar este último de la mejora manera posible. Es la parte Inteligente del sistema.

En términos de flujos, un conjunto de flujos regulados recorren el sistema de producción.

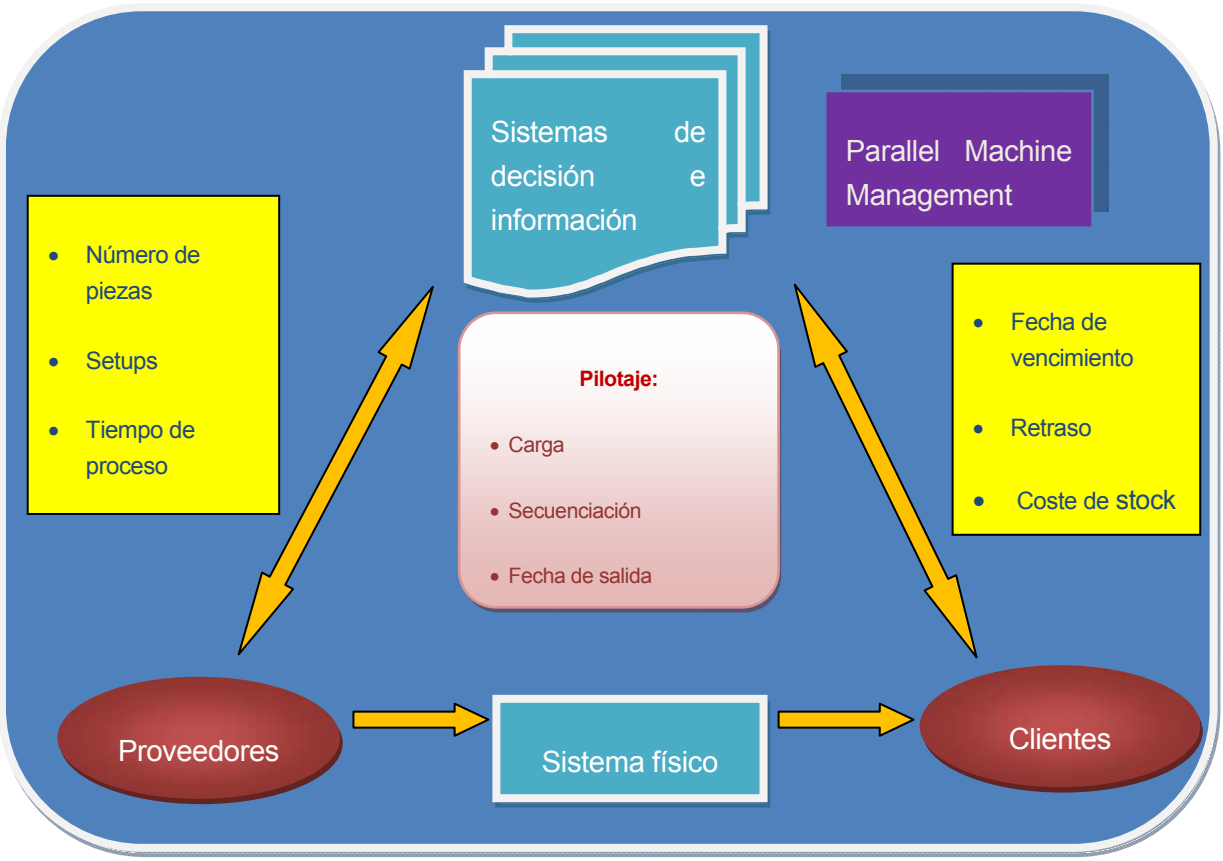


Fig. 2.1. Esquema de la teoría de sistemas de producción.

2.2. Clasificación de los problemas de Programación

2.2.1. Implantación de los procesos de producción. Modelos básicos de disposición.

La disposición física de los procesos en las plantas de producción, sean industriales o de servicios, obedece a dos posibles patrones de concepción: la llamada disposición *orientada al proceso* y *orientada al producto.*

En la ***disposición orientada al proceso***, los puestos de trabajo están agrupados funcionalmente y es el producto que se desplaza hasta donde se hallen. Existen dos tipos de distribución para esta disposición:

- *Distribución funcional- Talleres*: las plantas en que las estaciones de trabajo están agrupadas según el tipo de proceso que efectúan y el producto va de una a otra, por lotes, allá donde se encuentran.

- *Puestos fijos*: cuando el producto es complejo, voluminoso o pesado, son las personas, equipos y materiales los que se desplazan hasta el mismo, allá donde se encuentre.

En la ***orientada al producto***, los puestos están dispuestos de acuerdo con la secuencia de operaciones a seguir por el producto a obtener y por tanto son estos puestos de trabajos los que se "desplazan" hasta donde deben desarrollarse las actividades de producción que debe sufrir el producto. También existen en este caso dos tipos de distribución:

- *Disposición en flujos lineales-cadenas*: Sería el caso de una planta con las estaciones de trabajo situadas en secuencia, unas junto a otras, de forma que el producto avanza rápidamente de una a otra unidad a unidad. Los productos procesados en este tipo de disposición son muy homogéneos (poca variación) y en volumen elevado.

- *Flujos continuos*: tipo de distribución en flujo lineal en la que el producto no puede distinguirse unidad a unidad sino que avanza en flujo continuo.

En una implementación de tipo funcional, el producto avanza por lotes, con lo cual el tiempo en que una unidad de producto debe permanecer en una operación es el tiempo necesario para completar todas las operaciones del lote entero.

El caso estudiado a continuación tiene como alcance el estudio de un taller mecánico. Por tanto, se hace una descripción más precisa del "entorno programación" con los talleres mecánicos.

2.2.2. Problemas de programación de taller mecánico. Definición.

Enunciado del problema de programación:

El prototipo del problema de programación acostumbra a denominarse ***Job shop Problema***.

El enunciado general se puede detallar de la manera siguiente **[2]:**

"*n piezas (lotes de piezas, ordenes de trabajo) se tienen que realizarse en m máquinas (recursos, secciones, puestos de trabajos). Cada pieza se somete a una serie de operaciones prefijadas en un orden establecido.*

Cada operación se asigna a una de las m máquinas, tiene una duración determinada y conocida.

Debe establecerse un programa, es decir, la secuencia de operaciones en cada máquina, que optimice un cierto índice de eficacia (por ejemplo, la ocupación total del taller)."

La programación de operaciones se puede distribuir en 3 sub funciones:

- *La carga asigna a cada operación el recurso en que se va a ejecutar.*
- *La secuenciación define el orden o secuencia en que se ejecutaran las operaciones asignadas al mismo recurso.*
- *La temporización establece las fechas de ejecución.*

Cabe decir que los términos de "**taller mecánico**", "**pieza**" y "**máquina**" se utilizan sistemáticamente en la literatura científica española aunque los problemas puedan ser de diversa naturaleza y pertenecer a sectores diferentes que no tienen relación directa con la mecánica, ni con un taller. Pero la nomenclatura permite el buen entender de todos los gestores en España y Latino América.

No obstante, el idioma más utilizado sigue siendo el Ingles por su alcance internacional. Así pues, en la literatura inglesa se llaman a *"jobs"* para las piezas para no perder el sentido de que son tareas a asignar a cada máquina. Se puede encontrar también los términos de tareas, recursos o partes.

Los entornos máquinas, clasificación

La clasificación de los entornos máquinas depende del tipo de máquinas encontradas y de las restricciones de los trabajos. Así, un taller se define por el número de máquinas que contiene y su tipo.

Los diferentes tipos son los siguientes **[3]:**

- ***Los problemas de una máquina:*** *contiene un recurso disyuntivo que puede ejecutar una sola operación o pieza en un instante dado.*
- ***Los problemas de máquinas en paralelo:*** *que es lo que este proyecto investiga. Consisten en programar un conjunto de piezas en máquinas en paralelo, una*

pieza necesitando una sola máquina para su ejecución. Hay que resolver un doble problema: asignar las piezas a las máquinas y secuenciar las piezas asignadas a una misma máquina. Esta clase de problemas puede ser dividida en tres sub-clases:

Máquinas idénticas. Todas las operaciones tienen el mismo tiempo de proceso en todas las máquinas. (Las máquinas tienen la misma velocidad de trabajo)

Máquinas uniformes. Las máquinas tienen velocidades diferentes, pero estas velocidades son constantes. (No dependen de la operación a realizar). Si la duración de referencia de una operación es P_i, pues el tiempo pedido en la máquina j es P_i/C_j donde C_j es una constante asociada a la máquina j.

Máquinas independientes, la velocidad de cada maquina depende de la operación a realizar.

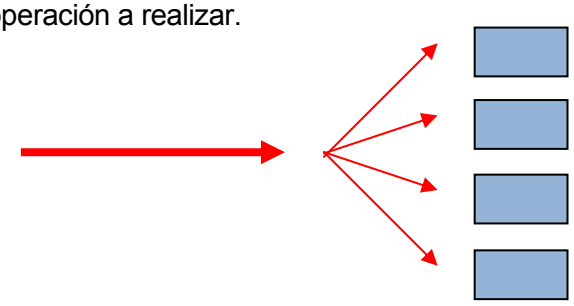


Fig. 2.2. Esquema general de un problema de máquinas en paralelo.

- ***Los problemas de talleres multi máquinas:*** *En los problemas de programación de talleres, los recursos son máquinas que pueden realizar sólo una tarea (u operación) a la vez. Por otra parte, cada trabajo se refiere a una entidad física indivisible, llamada producto, o lote cuando se agrupan varias unidades de productos idénticas.*

Una pieza no puede encontrarse simultáneamente en dos máquinas distintas, y las operaciones de una pieza deben realizarse una después de la otra.

Se consideran 3 tipos de problemas de talleres multi-máquinas:

(i)Taller flowshop.

Las piezas tienen siempre la misma secuencia de paso en las máquinas. Si todos los tiempos de proceso son no nulos, el taller es de tipo flowshop puro.

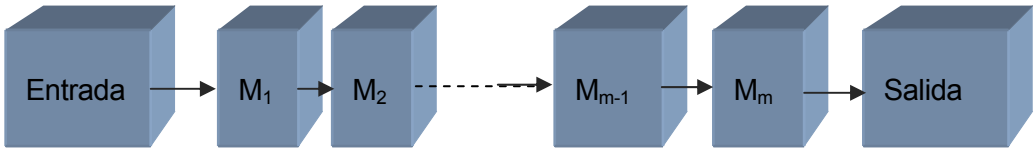


Fig. 2.3. Esquema general de un problema flowshop.

(ii) Taller a tareas (job shop).

En un taller jobshop, ninguna secuencia de operaciones se queda fija, y una parte puede circular varias veces sobre el mismo procesador para fomentar tareas diferentes. Las máquinas no están ordenadas y un gran número de transportes es posible. El transporte de una pieza puede empezar y terminarse sobre cualquier máquina del taller.

El problema del jobshop se diferencia del flowshop por un elemento esencial, la marcha de las piezas no es unidireccional como es para una cadena de montaje por ejemplo.

Esta diferencia se traduce también en un suplemento de datos necesarios. En efecto, además del tiempo de ejecución sobre las distintas máquinas, es necesario conocer el orden de visita de cada recurso en el cual estas operaciones deben efectuarse.

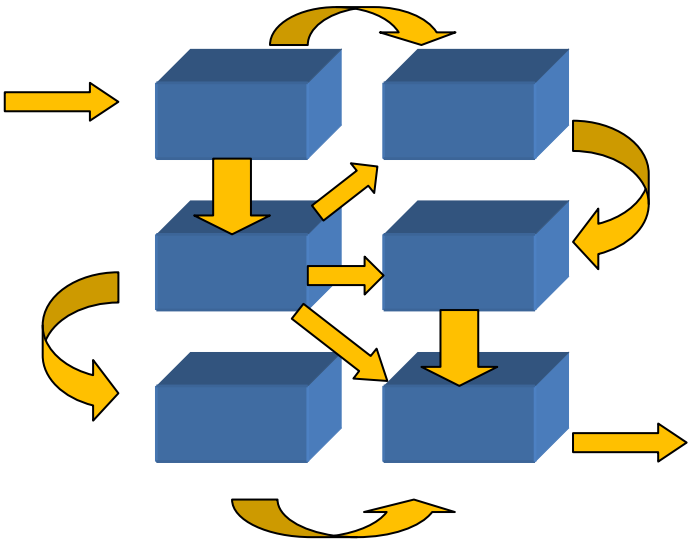


Fig. 2.4. Esquema general de un problema jobshop.

(iii) Taller de montaje (assembly taller).

Una máquina M_k puede efectuar un montaje de dos piezas diferentes, una procedente de una máquina M_i i y otro de una máquina M_j. Generalmente, la simetría entre los productos ensamblados debe respetarse. De la misma forma, se introduce también un taller de desensamblaje.

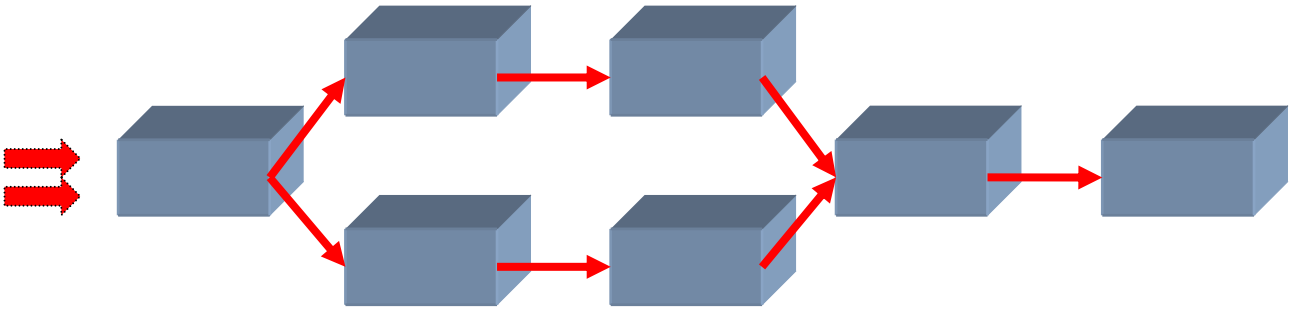


Fig. 2.5. Esquema general de un taller de ensamblaje/des ensamblaje.

3. Estado del arte sobre los trabajos de Optimización y Meta heurísticas.

3.1. Los métodos de resolución combinatorios.

Los problemas de programación pueden considerarse de optimización combinatoria, en el marco de la cual se encuentra un conjunto de métodos. El recurso a métodos exactos permite beneficiarse de fuertes resultados teóricos que acompañan los conceptos de convergencia y optimización global. Por lo tanto, varios problemas clásicos (de tamaño razonable) pueden ser resueltos de manera exacta en un tiempo razonable.

Los métodos exactos consisten generalmente en enumerar el conjunto de las soluciones del espacio de investigación. Para mejorar la enumeración de las soluciones, disponen de técnicas, para detectar cuanto antes los fracasos (cálculo de cotas), y de heurísticos específicos para orientar las distintas elecciones. Sin embargo, el tiempo de cálculo de estos métodos es exponencial, lo cual son sólo utilizables para problemas de pequeño tamaño.

Desde el final de los años 80, los métodos heurísticos, y en particular las meta-heurísticas, han suscitado un interés creciente por su capacidad para proporcionar soluciones de excelente calidad en un tiempo de cálculo reducido. La pérdida del carácter óptimo se ve compensada con la disminución del tiempo de cálculo y en consecuencia con un aumento de la capacidad de reacción. Las meta-heurísticas se benefician también de otras ventajas significativas como su rapidez de adaptación a algunas modificaciones estructurales del problema.

3.2. Los métodos exactos

Estos modelos se utilizan generalmente para resolver los problemas de pequeño tamaño donde el número de combinaciones es suficientemente escaso para poder explorar el espacio de soluciones en un tiempo razonable.

Estos métodos aseguran el óptimo pero tienen un coste computacional alto. El mayor inconveniente lo representa el tiempo que crece de forma exponencial en función del tamaño del problema.

El método de optimización mediante separación y acotación, también llamado Branch and Bound, es una técnica general que incluye una multitud de algoritmos con características comunes. Se trata de explorar el espacio de soluciones posibles usando el principio “divide y

vencerás": es decir, separar la región factible en conjuntos más pequeños, sin eliminar ninguna solución factible, y resolver el problema en cada conjunto.

Se determina un indicador de calidad de la mejor solución para decidir cual separar (sustituir el conjunto "padre" por los "hijos"): el de mejor cota.

Cabe calcular una solución inicial preferible con su cota K, luego se compara esta cota inicial con las cotas halladas K_i en cada conjunto, y se sigue explorando en la dirección de las cotas mejores que K.

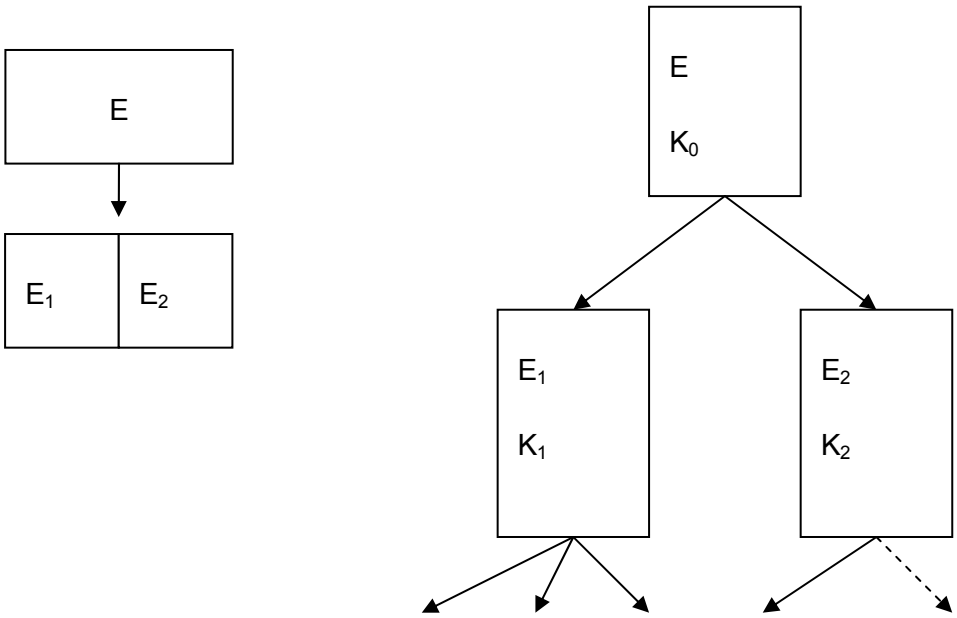


Fig. 3.1. Esquema general del método Branch and Bound.

3.3. Las Meta Heuristicas

Las meta heurísticas forman una familia de algoritmos de optimización destinados a solucionar problemas difíciles para los cuales no se conocen métodos clásicos más eficientes. Estos métodos son suficientemente generales para poder aplicarse a varios problemas de optimización combinatoria.

Esta sección se concentra en la descripción de las tres grandes clases de meta heurísticas: ***constructiva***, de ***búsqueda local o iterativa*** y ***evolutivas*****.**

Las meta heurísticas se apoyan en un equilibrio entre la *intensificación* de la búsqueda y su *diversificación*.

Por un lado, la intensificación permite encontrar unas soluciones de más calidad apoyándose en soluciones ya encontradas, y por otro lado, la diversificación procura estrategias que permiten explorar un espacio más grande de soluciones y escaparse de los mínimos locales.

En efecto, la falta de este equilibrio hace converger demasiado rápidamente hacia los mínimos locales (*falta de diversificación*) o a una exploración demasiado larga (*falta de intensificación*).

3.3.1. El enfoque constructivo

El objetivo de estos métodos consiste en construir progresivamente una programación, solución del problema, tomando a cada etapa una decisión para la colocación de una nueva tarea. La idea consiste en disminuir el tamaño del problema a cada etapa, lo que equivale a limitar el espacio de las soluciones admisibles X a un subconjunto de soluciones más pequeño X_k a cada vez. Un método constructivo encuentra una solución óptima cuando cada uno de los subconjuntos en cuestión contiene al menos una solución optima $s^* \in X$.

Desgraciadamente, raros son los casos donde se cumple tal condición con certeza. La mayoría de los métodos constructivos son de tipo *greedy* (*glotón)*.

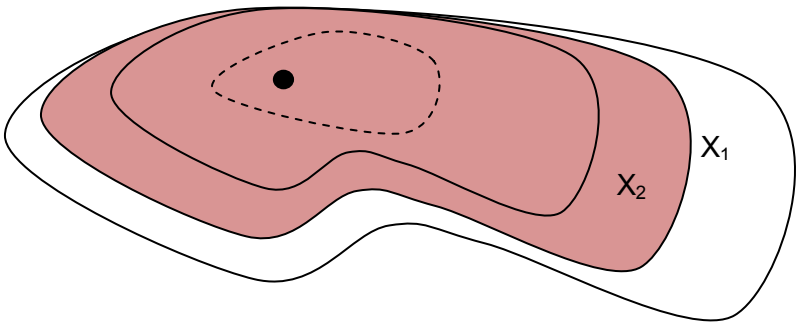


Fig. 3.2. Exploración de X con un enfoque constructivo.

Los métodos constructivos se distinguen por su rapidez y su gran simplicidad. En efecto, se obtiene muy rápidamente una solución admisible para un problema otorgado sin recurrir a técnicas altamente sofisticadas. El inconveniente principal de estos métodos reside en la posible mala calidad de las soluciones obtenidas, debiéndose a la incertidumbre de la mejor elección posible a cada etapa.

Varias heurísticas constructivas se dedican al flowshop tales como GUPTA **[2]**, etc.

3.3.2. El enfoque iterativo (Búsqueda Local)

Los métodos de exploración por vecindario son algoritmos iterativos que exploran el espacio **X** de las soluciones desplazándose paso a paso de una solución a otra. Un método de este tipo comienza a partir de una solución $\mathbf{s_o} \in \mathbf{X}$ elegida arbitrariamente u obtenida por medio de un método constructivo. El paso de una solución admisible a otra se hace sobre la base de un conjunto de modificaciones elementales que se trata de definir de caso en caso.

El vecindario N (s) de una solución $\mathbf{s} \in \mathbf{X}$ se define como el conjunto de las soluciones admisibles alcanzables desde **s** efectuando modificaciones elementales. Los pasos sucesivos de una solución a una solución vecina definen un camino a través del espacio de las soluciones admisibles. El proceso de exploración se para cuando se satisface uno o más criterios de paro.

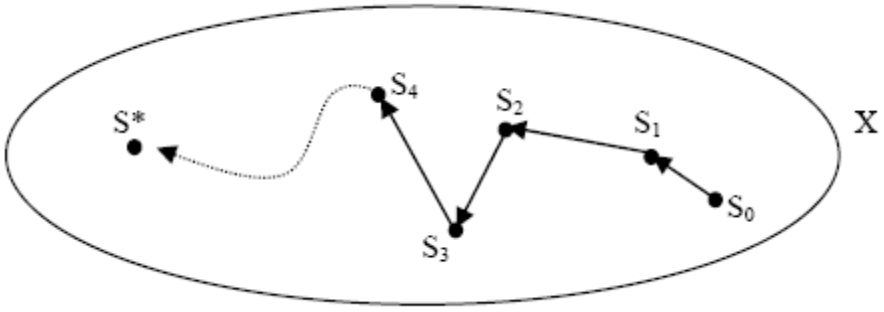


Fig. 3.3. Exploración de X con un enfoque secuencial.

Los métodos iterativos o de investigación local difieren esencialmente por el sistema de vecindario utilizado y la estrategia de curso del sistema de vecindario.

Algunos métodos han sido bastante estudiados y se han comprobado eficientes como el *recocido simulado* (Simulated Annealing) **[4]** y la *búsqueda tabú* (Taboo Research) **[5]**.

3.3.3. El enfoque evolutivo

Contrariamente a los métodos constructivos e iterativos que hacen intervenir una única solución (parcial o no), los métodos evolutivos dirigen a un grupo de soluciones admisibles a cada una de las etapas del proceso de investigación.

La idea central consiste en utilizar regularmente las propiedades colectivas de un conjunto de soluciones distinguibles, llamado población, con el fin de guiar eficazmente la investigación hacia buenas soluciones en el espacio de las soluciones admisibles factibles X.

Después de haber generado una población inicial de soluciones, aleatoriamente o por medio de un método constructivo, un método evolutivo intenta mejorar la calidad media de la población corriente recurriendo a principios de evolución natural.

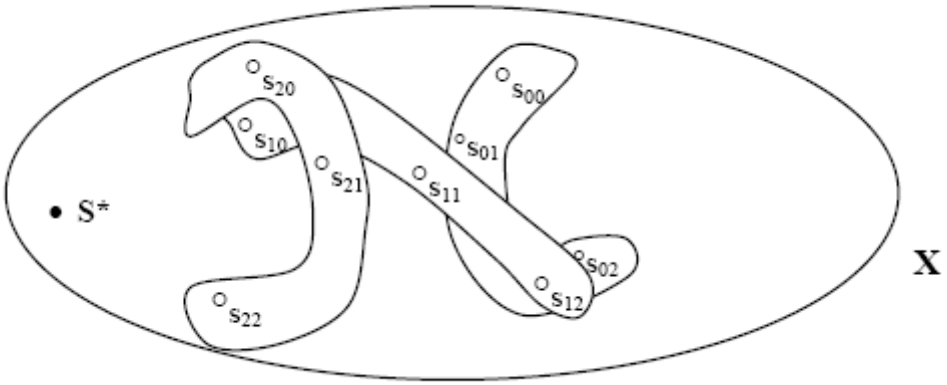


Fig. 3.4. Exploración de X con enfoque evolutivo.

3.4. Los Algoritmos Genéticos

Los algoritmos genéticos (AGs) son métodos evolutivos, que se inspiran por proximidad a los mecanismos biológicos vinculados a los principios de selección y evolución natural de Darwin.

Desarrollados inicialmente por HOLLAND en 1975 **[6]** para responder a necesidades específicas de biología, estos algoritmos se adaptaron, más tarde, para la búsqueda de soluciones a los problemas de programación **[7]**. Esta adaptación se operó desarrollando

una analogía entre un individuo en una población y una solución de un problema en un conjunto de soluciones.

Los algoritmos genéticos utilizan a una población de "individuos" (o soluciones) que evolucionan al mismo tiempo como en la evolución natural en biología. Para cada uno de los individuos, se mide su facultad de adaptación al medio exterior (función de adaptación o en inglés *fitness*). Los algoritmos genéticos se basan entonces en tres funcionalidades que suelen llamarse *operadores.*

3.4.1. Representación de una solución

Es el punto crítico del algoritmo. Se tiene que adaptar de la mejor manera posible al problema y a la evaluación de una solución.

3.4.2. La población inicial

La población inicial será la población base de las sucesivas generaciones; existen varios métodos para generarla. Suele ser concebida aleatoriamente, aunque se pueden utilizar métodos heurísticos para generar soluciones de buena calidad.

De toda manera, es importante garantizar la diversidad estructural de estas soluciones para tener una "representación" de la mayor parte de la población, con más diversidad, con la meta de evitar una convergencia prematura. **[8]**

Una cuestión importante es la relación al tamaño idóneo de la población. En efecto, las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que el hecho de trabajar con poblaciones de gran tamaño puede acarrear problemas relacionados con el exceso de coste computacional.

Goldberg **[7]** demostró que el tamaño óptimo de la población, con codificación binaria, crece exponencialmente con la longitud del cromosoma. Estudios posteriores, como los de Alander **[9]**, coinciden en afirmar que una población inicial no superior a 30 individuos garantiza, en general, dar con una solución factible al problema, aunque cabe decir que los problemas que les sirvieron de base para realizar dicha aseveración eran de complejidad media. En cambio, lo que sí parece seguro es que el tamaño de la población inicial debe estar relacionado con la *longitud de los cromosomas* y la *complejidad del problema*.

3.4.3. El Fitness

El *fitness* (también *función de aptitud* o *función de adaptación*) refleja el nivel de adaptación al problema del individuo representado por el cromosoma. Sirve para determinar qué soluciones tienen mayor o menor probabilidad de sobrevivir.

Esta función no debe crear diferencias muy grandes entre individuos y, por lo tanto, provocar una convergencia prematura, ni diferencias muy pequeñas que puedan producir un estancamiento en el proceso de búsqueda de la solución.

De manera general, se construye un fitness que debe reflejar el valor del individuo de manera real, pero en muchos casos o problemas de optimización combinatoria, donde existen gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no validos.

Para estas situaciones, existen diversas soluciones tales como no considerar aquellos individuos que no verifican las restricciones y seguir iterando hasta obtener individuos validos, asignar a dichos individuos un fitness igual a cero y reconstruir aquellos individuos que no verifican las restricciones por medio de un nuevo operador que suele denominarse *reparador*.

3.4.4. La selección de los progenitores

Permite favorecer los individuos que tienen un mejor *fitness*, aunque se pueda también seleccionar los padres aleatoriamente. De todas formas, se garantiza una diversidad en las soluciones que serán generadas tras los diferentes operadores.

3.4.5. El cruce

Combina dos soluciones padres para formar uno o dos descendientes *(offsprings)* intentando conservar las buenas características de los progenitores.

3.4.6. La mutación

Permite añadir una diversidad a la población mutando ciertas características (*genes*) de una solución mediante unas probabilidades de mutación.

Los parámetros a tener en cuenta para la eficiencia de un algoritmo genético son entonces **el** *tamaño de la población inicial, el número máximo de generaciones* (es decir cuando parar el algoritmo), *la probabilidad de mutación p_m y la probabilidad de cruce p_c ,* así como *los operadores de cruce y mutación*.

A continuación se detalla el algoritmo genérico de un AG:

Algoritmo Método genérico de los AG

Variables locales.;

-Inicio

i ← 0

P_0 : Población inicial de n individuos

-Mientras "*el criterio de parada no ha sido alcanzado*"

-Hacer i ← i + 1

Sea P_i una población vacía,

-**Mientras** P_i no contiene n individuos

-**Hacer** Elegir con un método elegido I_1 e I_2 dos individuos de P_{i-1}.

Aplicar el ***operador de cruce*** con la probabilidad p_c sobre I_1 e I_2 para obtener los hijos E_1 e E_2

Añadir E_1 e E_2 a P_i.

-**Fin mientras**

Aplicar el ***operador de mutación*** con la probabilidad p_m para ciertos individuos de Pi

P_i ← $P_i \cap P_{i-1}$ *(selección entre ambas poblaciones de la resultante)*

Seleccionar p individuos con un proceso de selección elegido de P_i et suprimir los otros n.

-Fin mientras

-Fin.

Fig. 3.5. Algoritmo genérico de un AG.

Se puede mejorar aún más la calidad de la solución hallada con una *búsqueda local* tras la mutación.

3.5. Estado del arte sobre los problemas multicriterios

En los casos reales vinculados a la gestión de producción en un taller, se puede encontrar problemas multi-criterio. El conjunto de soluciones interesantes sirven como soporte a la toma de decisión para la gestión de talleres.

En este caso, se consideran simultáneamente dos criterios importantes para el responsable de producción. Por un lado, conservando la minimización del tiempo total de permanencia (cuyos costes relacionados con el tiempo de espera pueden ser considerados como costes de stock), y por el otro lado, minimizar los retrasos medios.

La optimización multi-criterio es actualmente un dominio en expansión, tanto para la investigación teórica como para las aplicaciones.

En efecto, la mayoría de los problemas que se pueden encontrar en el mundo real son de naturaleza multicriterio. En la práctica, efectuar una elección perfecta en todos puntos de vista es raro.

Existen varias posibilidades de construcción de una solución en una optimización multi-criterio.

3.5.1. Transformación de un problema multi objetivos en uniobjetivo.

Se trata de llevar las restricciones a una ecuación con unos operadores de Lagrange. Se conocen tres métodos básicos.

- Método de agregación
- Método de ϵ-restricción
- Programación por metas

Al final del siglo XIX, el economista Vilfredo Pareto formula el concepto de óptimo de Pareto, con lo cual constituye los fundamentos de la investigación multi-criterio.

Sea F el espacio de las soluciones factibles.

Se considera que un punto x ϵ F es **Óptimo de Pareto** si no existe ningún vector factible x ϵ F que haga disminuir un criterio sin aumentar al mismo tiempo otro criterio.

Sin embargo, en la mayoría de los casos, el óptimo de Pareto no está constituido por una sola solución, sino que se encuentra en un conjunto de soluciones que se llaman **soluciones no dominadas** en el sentido de Pareto.

3.5.2. Enfoque no Pareto

Trata de manera separada los diferentes objetivos, como se observa en la figura 3.6.

Una vez se dispone de las diversas sub-poblaciones, se realizan las mutaciones y cruces habituales que acaban por determinar la frontera o población de Pareto.

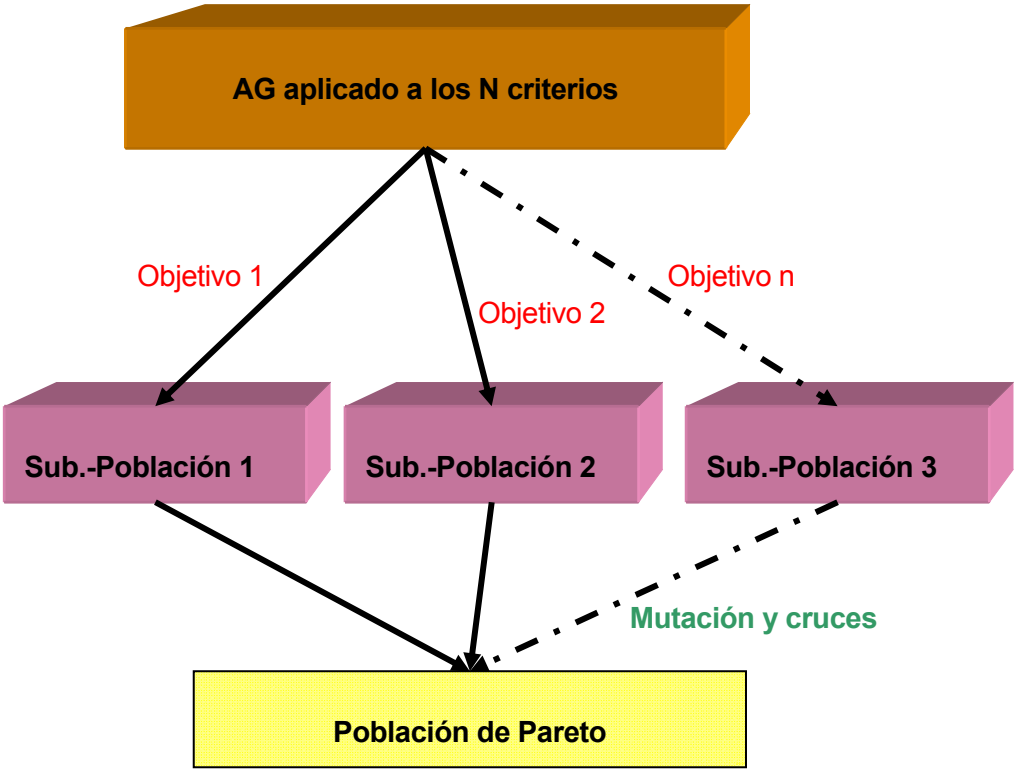


Fig. 3.6. Procedimiento genérico con enfoque no Pareto.

Ejemplo:

VEGA (*Vector Evaluated Genetic Algorithm*): El método presentado por Schaffer **[10]**, es uno de los primeros métodos evolutivos. En una población, los mejores individuos están seleccionados para cada criterio de manera separada. La selección se hace según cada peso proporcional de los seleccionados, y se crea una nueva población con los individuos retenidos (con los operadores clásicos de cruce y mutación).

Se observa una convergencia prematura alrededor de los puntos óptimos mono criterios, lo cual se puede perder en la optimización bi-criterio.

3.5.3. Enfoque de Pareto

Utiliza las nociones de dominancia en la selección de las soluciones.

Tiene el doble objetivo de alcanzar una convergencia con el método de ranking (orden entre los individuos) manteniendo una diversificación.

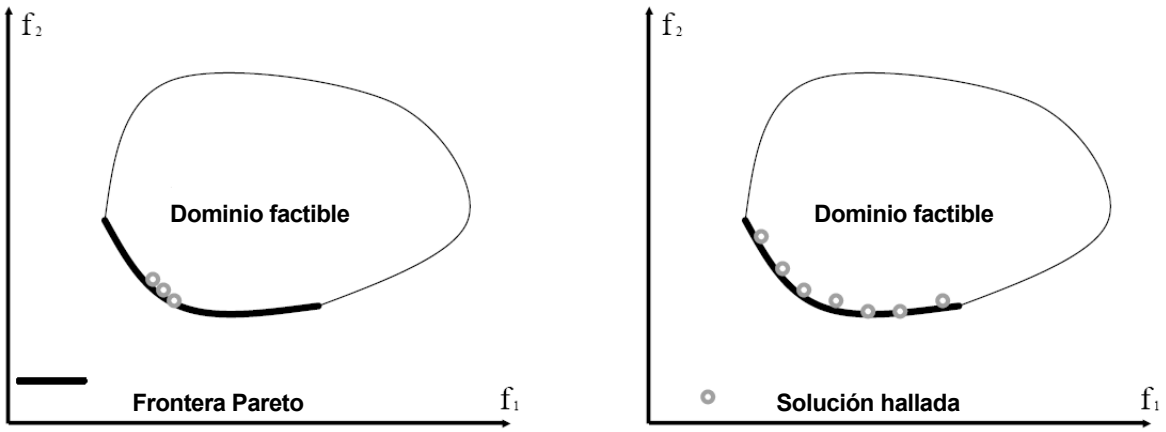


Fig. 3.7. Fronteras de Pareto con enfoque de Pareto.

Ejemplo:

NPGA (*Niched Pareto Genetic Algorithm*): Para evitar una convergencia prematura hacia un óptimo mono criterio, Horn **[11]** propone una evaluación.

Para preservar la diversidad de los individuos, se considera el concepto de “nichos”, lo cual es una sub-población estable de individuos muy cercanos entre ellos.

Pues se utiliza una función basada en la idea que los individuos de un mismo “nicho” deben compartir los mismos recursos.

Así, la fuerza de un individuo será más degradada si se encuentra con más individuos en su vecindario.

Luego se utiliza el concepto de dominación de Pareto. Dos individuos son elegidos al azar, y son comparados a una sub-población escogida al azar igualmente. Si cualquier de los dos es “no dominado”, este mismo es seleccionado para generar la población siguiente. Si ambos son dominados o no dominados en la sub-población, se seleccione el individuo encontrándose en el más pequeño nicho.

El nicho de un punto se determina contando el número de puntos en el interior de un radio dado.

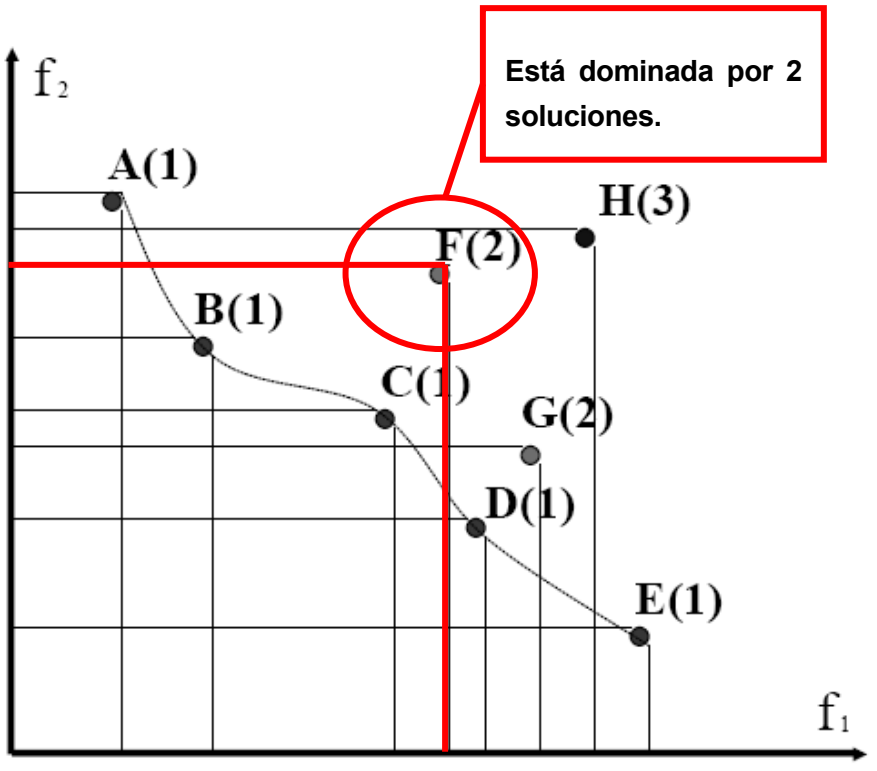


Fig. 3.8. Esquema del método de ranking para un enfoque Pareto. **[12]**

La elección de un AG como técnica de optimización multi-criterio es interesante ya que se brinda un conjunto de soluciones de compromiso entre ambos criterios de optimización. De esta manera, se puede obtener una aproximación de frontera de Pareto en una sola ejecución.

4. Algoritmo genético para el problema Fmed-Tmed con máquinas idénticas en paralelo.

4.1. Planteo, datos de partida del problema

A continuación se va a detallar con más precisión el método que se ha utilizado para resolver la programación multicriterios que se propone estudiar.

El problema que se intenta investigar es un problema ***n/m/Pm***, con la nomenclatura descrita más adelante. Se trata de minimizar al mismo tiempo el tiempo medio de finalización **Fmed** así como el retraso medio **Tmed** de las piezas.

Las m máquinas se hallan en paralelo y son todas idénticas, es decir tienen el mismo tiempo de preparación y misma tasa de producción (o tiempo de proceso).

La puesta en marcha del algoritmo genético se hace mediante un conjunto de parámetros que cabe recordar y explicitar en el caso estudiado.

El objetivo del trabajo es hallar y encontrar los valores de los parámetros más adecuados para conseguir una solución óptima o una solución que cumple la función objetivo con los mejores resultados.

El programa desarrollado permite al usuario introducir los datos que van a servir para crear una solución apropiada a la elaboración de la solución.

Los datos a introducir (o que se pueden leer en un fichero adecuado) son:

- *el número de piezas n*
- *el número de máquinas m*
- *el tiempo de preparación máquina para cada pieza* st_0 *(i) (inicial setup) para la pieza "i"*
- *el tiempo de cambio de pieza por cada máquina st(i, i') (setup); como las máquinas son idénticas, el tiempo de cambio es igual en cada máquina. st(i, i') es el tiempo de preparación que necesita una máquina para pasar de la pieza " i " a la pieza " i' "*

- *el tiempo de proceso p(i) (process time) de la pieza "i" en cualquiera máquina (ya que son idénticas)*
- *la fecha de vencimiento d(i) (due date) de la pieza "i" tras lo cual la pieza tendrá un cierto retraso*

Para ilustrar cada parámetro se ha decidido poner un ejemplo.

Ejemplo numérico:

Se toma un ejemplo numérico de 5 piezas y 2 máquinas para ilustrar los diferentes pasos.

i	1	2	3	4	5
Sto(i)	2	3	2	1	2

Tabla 4.1. Tiempos iníciales de preparación para las máquinas (ejemplo)

i \ i'	1	2	3	4	5
1	0	3	5	4	1
2	5	0	3	6	3
3	2	5	0	3	5
4	4	7	2	0	3
5	5	5	3	4	0

Tabla.4.2. Tiempos preparación para las máquinas (ejemplo).

i	1	2	3	4	5
p(i)	18	8	4	13	3

Tabla.4.3. Tiempos de proceso para las piezas (ejemplo).

i	1	2	3	4	5
d(i)	6	10	14	19	20

Tabla.4.4. Fechas de vencimiento para las piezas (ejemplo).

A continuación, se va a explicar en detalle como se han generado los valores de los parámetros necesarios para la aplicación del algoritmo tomando este ejemplo para cada paso.

4.2. Codificación de la solución

Se ha decidido crear un tipo ***"pieza"*** que contiene una serie de datos de tipo "enteros" respecto a cada pieza.

Fig. 4.1. Modelización de la lista para una piezas i ($1 \leq i \leq n$).

El dato ***código*** permite asignar un "nombre" a cada pieza para facilitar la llamada de una pieza en particular.

El ***setup mínimo*** de una pieza "i" retorna el valor mínimo de tiempo de cambio de pieza entre dicha pieza "i" y las otras "n-1" piezas del sistema.

El ***setup máximo*** de una pieza "i" retorna el valor máximo de tiempo de cambio de pieza entre dicha pieza "i" y las otras "n-1" piezas del sistema.

La ***diferencia de setup*** de una pieza "i" retorna la diferencia entre su valor máximo de tiempo de cambio de pieza y su valor mínimo.

La información ***fecha de termino*** retorna el instante de salida de la pieza "i", llamada también **"ci"** (completion time) en la literatura científica.

La información ***huelgo*** retorna la diferencia entre el instante de salida y la fecha de vencimiento de la pieza "i". Equivale a **Li** en la literatura**.**

La información ***retraso*** retorna la diferencia entre el instante de salida y la fecha de vencimiento de la pieza "i" si es un número positivo. Equivale a **Ti.**

Ejemplo numérico:

Piezas i	**1**	**2**	**3**	**4**	**5**
código	1	2	3	4	5
p(i)	18	8	4	13	3
d(i)	6	10	14	19	20
setup Mínimo	1	3	2	2	3
setup Máximo	5	6	5	7	5
Diferencia de setup	4	3	3	5	2

Tabla.4.5. Resultados del ejemplo.

Se ha decidido crear un nuevo tipo ***máquina,*** conteniendo informaciones sobre las piezas que una máquina va a procesar. Se crea así una ***lista*** de módulos de tipo ***maquinas j***

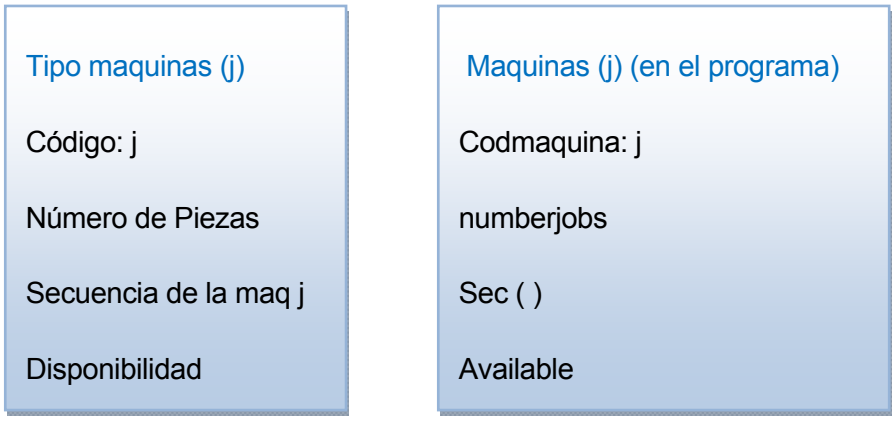


Fig. 4.2. Modelización del tipo "maquina j (1 ≤ j ≤ m).

Los datos ***código***, ***disponibilidad*** y ***número de piezas*** son de tipo *entero*, mientras la ***secuencia de la máquina*** es una ***lista*** de tipo ***pieza*** como se ha explicado anteriormente, conteniendo la secuencia que va recibir la máquina en cuestión.

La información ***disponibilidad*** retorna el instante en que la máquina "j" queda disponible

Número de piezas es el número de piezas asignadas a la maquina "j".

Ejemplo numérico:

Maquinas	**Máquina 1**	**Máquina 2**
Código	1	2
Número de piezas	3	2
Secuencia de la máquina	1-2-3	4-5
disponibilidad	2+18+3+8+3+4=38	1+13+3+3=20

Tabla.4.6. Resultado del ejemplo.

En el programa, maquina(2).sec(3).codigo retorna el número de la pieza que pasa en tercera posición en la máquina 2.

maquina(1).sec(2).huelgo retorna el huelgo de la pieza que pasa en segunda posición en la máquina 1.

A continuación se crea un tipo ***solución*** que va a representar un ***individuo*** de la población (inicial o generada).

Una solución del problema será entonces una lista de máquinas, cada una conteniendo una secuencia de piezas.

Cada solución contendrá la información necesaria a la optimización multicriterio, es decir, los valores Tmed y Fmed que cabe minimizar.

El tamaño de la población será fijado por el usuario, y tiene el nombre ***cant*.**

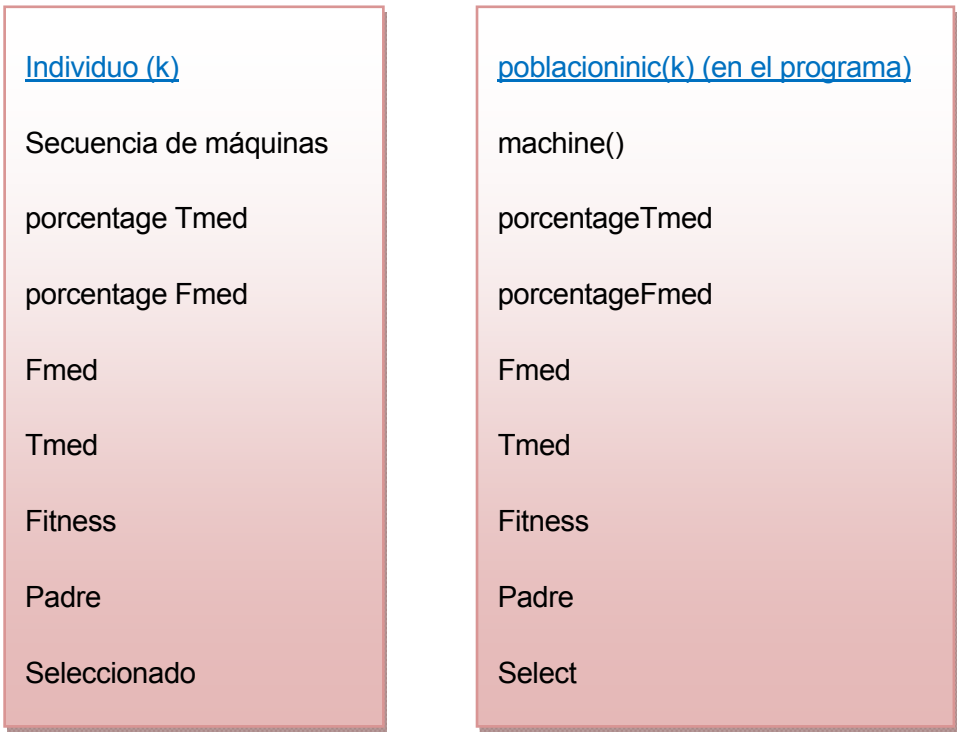


Fig. 4.3. Modelización de un individuo k (1 ≤ k ≤ cant).

La población inicial es una lista de soluciones potenciales conteniendo una pareja de datos (**Tmed**, **Fmed)** por individuo, que incluye una lista de máquinas con sus secuencias propias de piezas.

Los diferentes fitness serán utilizados para la selección de individuos (esta parte será explicada en el apartado "elección de los padres").

Las informaciones ***Padre*** y ***seleccionado*** son de tipo *"binario"*. Padre por ejemplo es "verdad" cuando el individuo en cuestión proviene de una *población "padre"*, y falso si proviene de una *población "hijos"*.

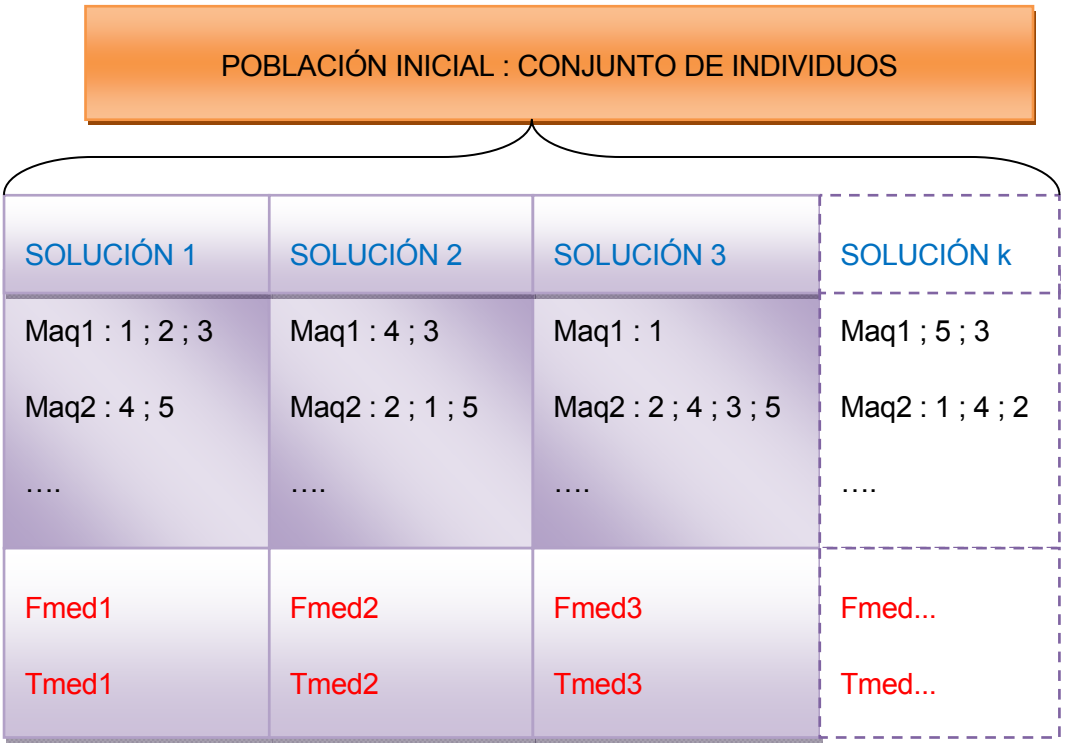


Fig. 4.4. Modelización computacional para el ejemplo a 2 máquinas y 5 piezas.

*Ej: poblacioninic(5).machine(2).sec(1).retraso retorna el retraso de la **primera** pieza que pasa por la **segunda** máquina de la solución **número 5** (o del individuo 5) de la población inicial.*

Para aplicar los diferentes operadores de cruce y mutación, se efectúa previamente una conversión de la codificación de un individuo detallado hasta aquí. Se construye un cromosoma como vector de enteros donde los indicadores de máquina se representan por un “0” en la casilla anterior a la primera pieza que recibe la máquina.

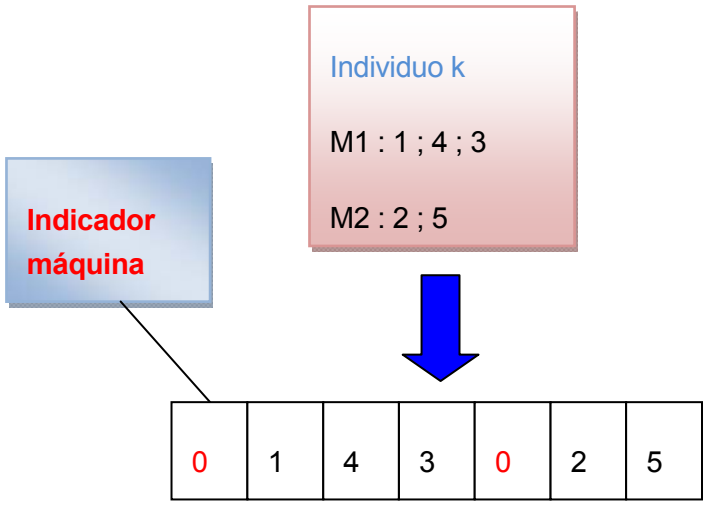


Fig. 4.5. Modelización numérica de la conversión entre un individuo k y su cromosoma.

4.3. Población Inicial

4.3.1. Tamaño de la población inicial

Se ha previsto en el menú “opciones” del software que el usuario pueda decidir el tamaño de la población inicial y de las diferentes generaciones (***cant***).

En un algoritmo genético, se puede estudiar el número más adecuado de individuos para la dicha población que de mejores resultados.

Se ha decidido fijar el tamaño de las poblaciones generadas al de la población inical.

4.3.2. Composición y construcción de la población inicial.

La composición de la población inicial ha sido estudiada para garantizar una cierta diversidad de la población inicial.

Se ha decidido separar la población inicial en dos partes.

Los primeros 50% individuos de la población parten del orden **EDD “Earliest Due Date”**. Se van construyendo a partir de una solución obtenida al ordenar crecientemente las piezas según sus fechas de vencimiento, pero se introducirá la aleatoriedad para generar individuos diferentes (como se explica a continuación).

Los últimos 50% individuos de la población parten de un orden **Setup**, y también se introducirá la aleatoriedad para generar individuos diferentes.

Se justifica aquí que en los tipos “piezas”, se disponga de los valores setup mínimo y máximo.

Los pasos de la creación de la primera parte de la población se indican a continuación.

A continuación, se explica en detalle los pasos del método constructivo utilizado para la construcción de la primera parte de la población inicial.

Se toma como ejemplo numérico el caso a 2 máquinas y 5 piezas utilizado previamente.

- **Paso 1**: se ordenan las piezas de menor a mayor fecha de vencimiento.

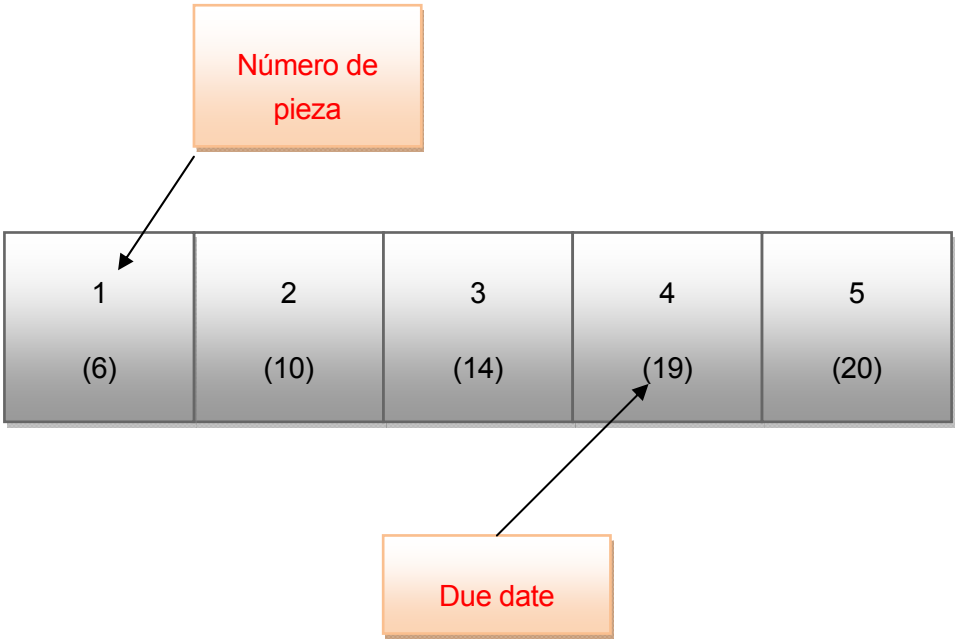


Fig. 4.6. Modelización de la lista de piezas en el orden pre definido (orden EDD).

- **Paso 2**: se crea una lista dinámica de tres enteros, rellenando esta lista poco a poco con las piezas en el orden establecido en el paso 1.

Estos tres enteros tienen cada uno una probabilidad de 25 % y una cuarta opción del cambio de máquina tiene también un 25%. Se escoge un número aleatorio para completar cada cromosoma uno por uno (como se observa en la figura 4.7).

A cada asignación de una pieza a una máquina, se actualiza la lista dinámica tomando las piezas que aún no han sido asignadas. Al final se obtiene una solución completa con las piezas asignadas a cada máquina.

Esto se repite tantas veces como individuos se requieren para obtener un conjunto de soluciones que represente una primera parte de población. (Es decir, vuelve a realizar *cant/2* veces según el número *cant* fijado por el usuario en el menú de opciones).

La segunda parte de la población se hace de la misma manera, con 3 órdenes de partida diferentes.

En efecto, la segunda parte está compuesta de 3 sub poblaciones que parten de secuencias iníciales diferentes y asegura una diversidad en la población inicial.

- *La primera sub-parte* se ordena según un **orden creciente** de los tiempos de cambio de piezas **(setups)**. Por eso se ha introducido previamente en el tipo “pieza” la información del mínimo setup disponible para una pieza.

En el ejemplo de 10 piezas a procesar, un indicador guarda en memoria en el tipo **pieza** el setup mínimo de la pieza 1 con las 9 otras. Hace lo mismo con la pieza 2 etc.

Se obtiene así una lista de 10 setups (el mínimo para cada pieza) con lo cual se vuelve a empezar los pasos seguidos para la primera parte dela población.

- *La segunda sub-parte* se hace de la misma manera pero tomando el **orden decreciente** de los setup. Aquí también un indicador halla el mayor setup de cada pieza con las otras, y se ordena la lista para seguir los pasos de creación de la población.

- *La tercera sub- parte* se hace ordenando de mayor a menor la diferencia de los setups (MAX-MIN) por pieza a procesar. Y se vuelve a empezar los pasos previos.

Se puede observar en la figura 4.8 un recapitulo de la construcción de la población inicial.

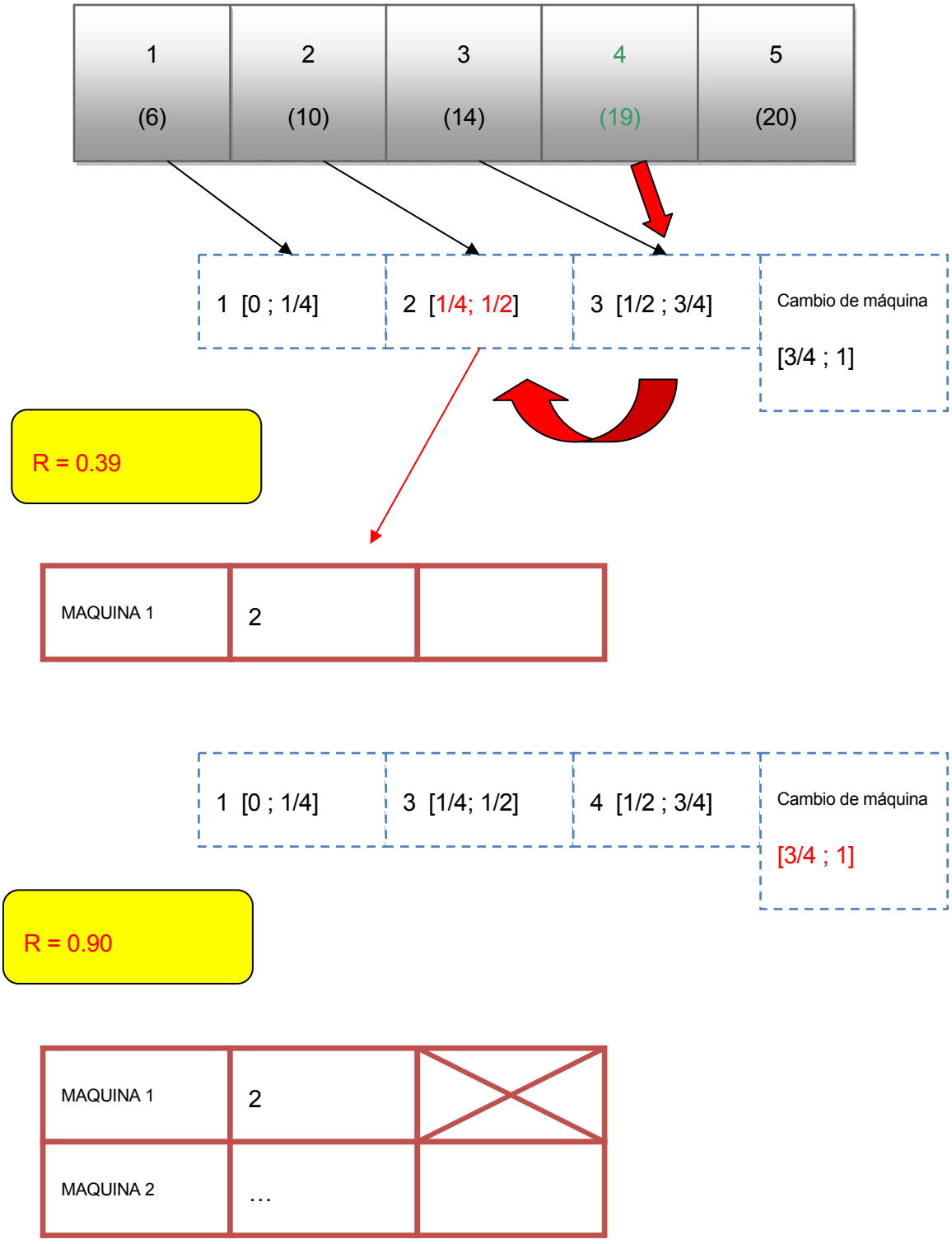


Fig. 4.7. Pasos de la creación de la población inicial.

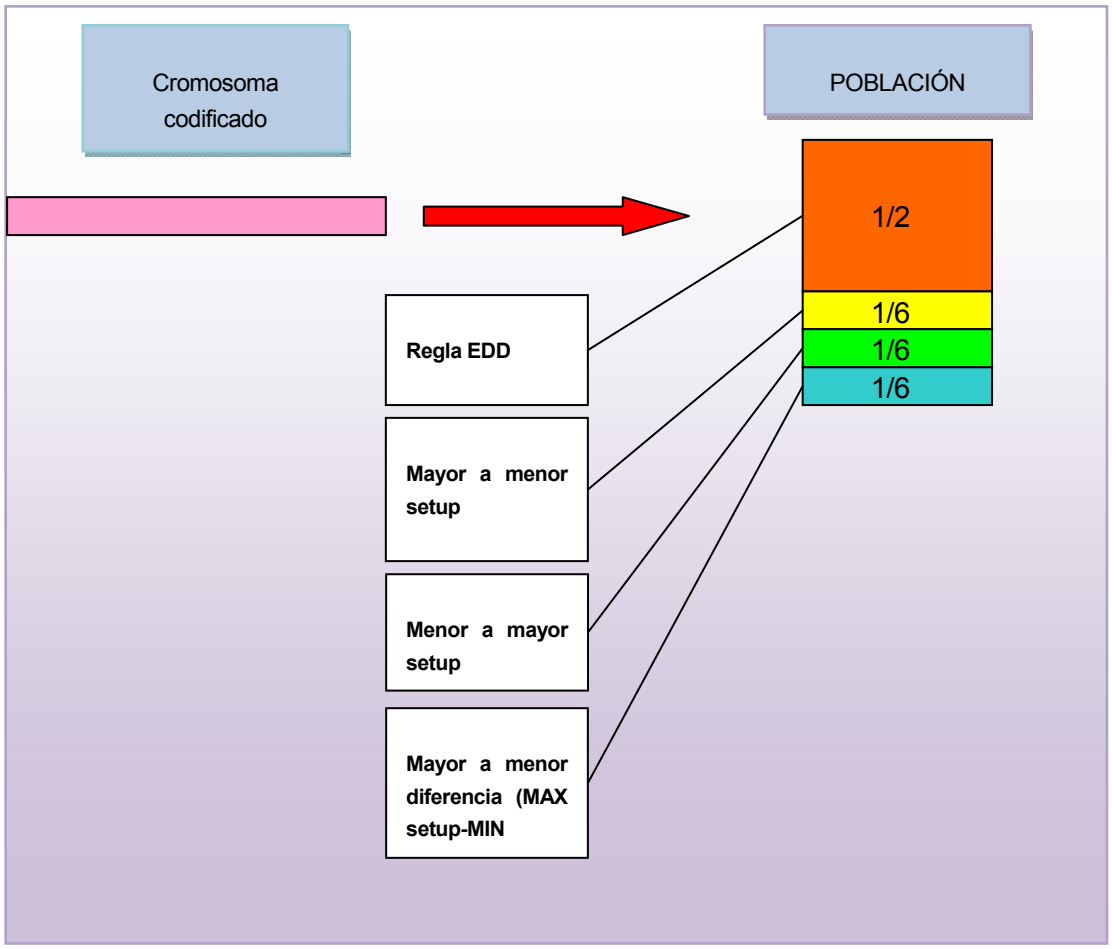


Fig. 4.8. Composiciones de la población inicial.

4.4. Medida de evaluación o Fitness

El fitness está directamente relacionado con los criterios a optimizar. En el caso estudiado, tiene que ver con el ***retraso medio*** así como con la ***fecha de finalización*** de cada pieza.

En el caso estudiado, cabe recordar que en cuanto una máquina se encuentre disponible, está preparada a recibir una nueva pieza. Solo existen unos tiempos iníciales de preparación de máquina.

Además, se considera que las piezas están todas introducidas en el sistema en el instante 0. Por lo tanto, la fecha de finalización corresponde al tiempo de permanencia de la pieza en el sistema.

Se calcula para cada solución de la población en proceso los instantes de salida de cada pieza en cada máquina, y los huelgos así como los retrasos en cuanto haya un retraso observado.

Se hace una media de los retrasos y de los tiempos de permanencia para cada solución.

Se va a crear una función de compartición a partir de los datos calculados para cada solución. Con lo cual se atribuirá un valor de fitness a cada solución respecto a sus propiedades.

Por lo tanto, se utiliza una parrilla (en ingles “grid method”) para asignar a cada solución un valor de fitness según que se encuentre en un área u otro.

El rango de valores se determinará según unas cotas inferiores (notadas Fmin y Tmin) y superiores (Fmax/Tmax) que hay que calcular al introducir los datos en el programa.

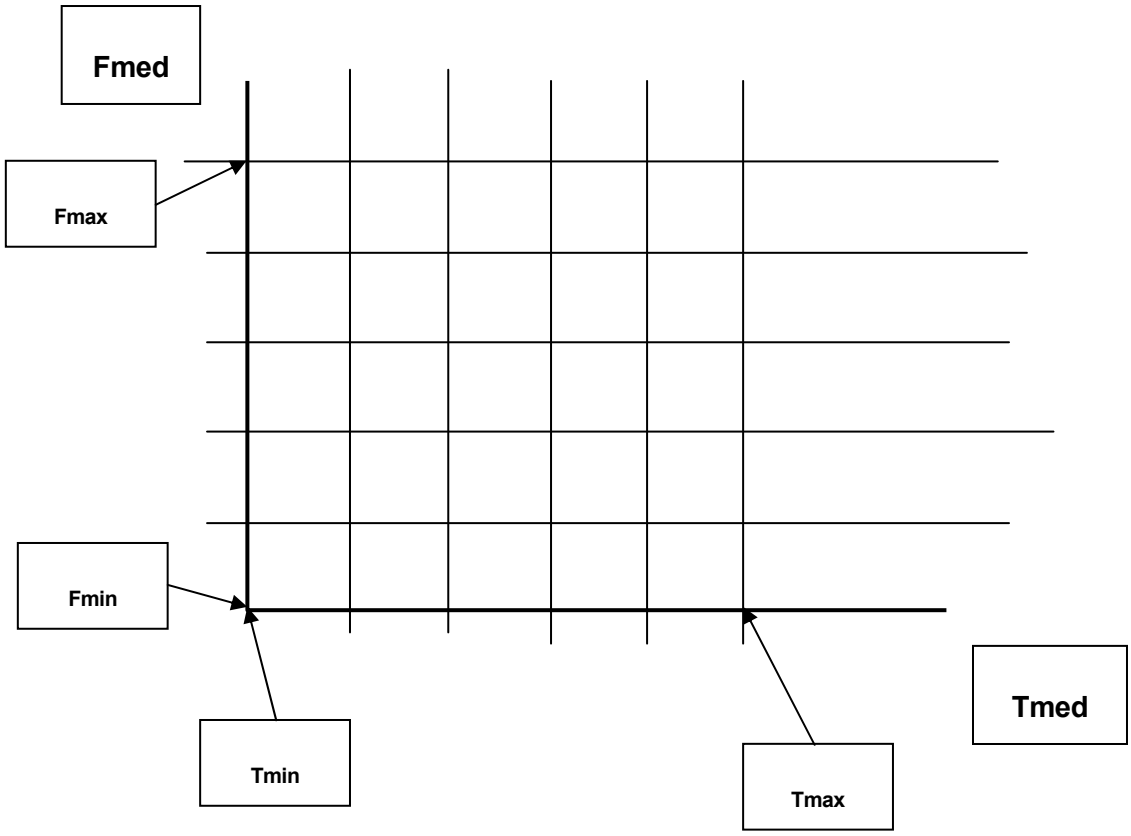


Fig. 4.9. Rango de valores para los parámetros del cuadrado.

4.4.1. Cálculo de las cotas

Para el cálculo de las cotas, se propone un procedimiento de acotación como detallado a continuación:

Se toma el ejemplo ya presentado más adelante con 5 piezas y 2 máquinas para ilustrar los diferentes pasos numéricamente.

a) Cota inferior

Paso 1: Escoger entre los tiempos mínimos de preparación inicial tantos como tantas máquinas hay en el sistema.

En el ejemplo, como sólo hay dos máquinas.

2	3	2	1	2

Tabla.4.7. Elección de los mínimos tiempos de preparación inicial.

Paso 2: Escoger para cada pieza, el tiempo mínimo de preparación de máquina. No importa que el tiempo pertenezca a una pieza que ya había sido asignada.

En la lista obtenida, sustituir los m valores máximos de la lista por los tiempos iníciales escogidos de la primera fila de tiempos iníciales de preparación.

	A	**B**	**C**	**D**	**E**
A	0	3	5	4	1
B	5	0	3	6	3
C	2	5	0	3	5
D	4	7	2	0	3
E	5	5	3	4	0

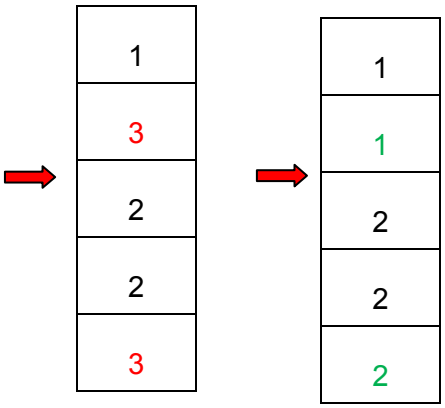


Tabla.4.8. Paso 2 en el caso del ejemplar numérico con 5 piezas y 2 máquinas.

Paso 3: Ordenar la lista obtenida en un orden creciente.

1	1	2	2	2

Tabla.4.9. Paso 3 en el caso del ejemplar.

Paso 4: Ordenar los tiempos de proceso del menor a mayor valor.

3	4	8	13	18

Tabla.4.10. Paso 4 en el caso del ejemplar.

Paso 5: Asignar una pieza a una de las máquinas que quede disponible más temprano y sumar. En caso de empate, utilizar el orden lexicográfico.

	J1	J2	J3	J4	J5
Máquina asignada	M1	M2	M1	M2	M1
$S_i^0 t$	1	1	2	2	2
P_i^0	3	4	8	13	18
C_i^0	**4**	**5**	**10**	**15**	**20**
F_i	**4**	**5**	**14**	**20**	**34**

Tabla.4.11. Paso 5 en el caso del ejemplar.

A partir de los datos, se calcula el Fmed correspondiente, y se redondea la cifra obtenida al entero inferior. Así se obtiene la cota Fmed inferior de:

Fmed= (4+5+14+20+34) / 5 = 15.4 => 15

Paso 6: Ordenar la lista de las fechas de vencimiento de menor a mayor y calcular la diferencia entre los valores de fechas de salida encontradas.

F_i	4	5	14	20	34
d_i	6	10	14	19	20
T_i	**0**	**0**	**0**	**1**	**14**

Tabla.4.12. Tiempos de preparación inicial

Con lo cual se calcula la cota inferior para Tmed y se redondea igualmente la cifra obtenida al entero inferior.

Tmed= (0+0+0+1+14) / 5 = 3 => 3

b) Cota superior:

Para las cotas superiores de Tmed y Fmed, se aplican los mismos pasos que para las cotas inferiores pero cogiendo los valores mayores para los tiempos iniciales de preparación máquina y los tiempos de preparación por máquina. Como cota superior se calcula en el caso de que todas las piezas sean procesadas en la misma máquina.

En el paso 1, se escoge el valor máximo de los tiempos iniciales de preparación ya que se procesan todo en la misma máquina.

En el paso 2, se sustituyen el menor valor escogido de la matriz de tiempos de preparación por el valor inicial (suponemos que todo se realiza en una única máquina).

En los pasos 3 y 4, se ordenan las listas en un orden decreciente.

En los pasos 5 y 6, se redondea al entero superior en caso de valores decimales.

Con el mismo ejemplo de 5 piezas y 2 máquinas, las cotas superiores se calcularán así:

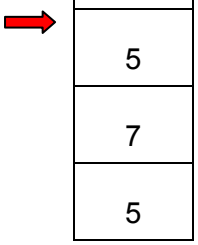
Paso 1:

2	3	2	1	2

Tabla.4.13 Paso 1 en el caso del ejemplar.

Paso 2:

	A	**B**	**C**	**D**	**E**
A	0	3	5	4	1
B	5	0	3	6	3
C	2	5	0	3	5
D	4	7	2	0	3
E	5	5	3	4	0



5
6
5
7
5

3
6
5
7
5

Tabla.4.14. Paso 2 en el caso del ejemplar.

Paso 3:

3	5	5	6	7

Tabla.4.15. Paso 3 en el caso del ejemplar.

Paso 4:

3	4	8	13	18

Tabla.4.16. Paso 4 en el caso del ejemplar.

Paso 5:

	J1	J2	J3	J4	J5
Máquina asignada	M1	M1	M1	M1	M1
$S_i^0 t$	3	5	5	6	7
P_i^0	3	4	8	13	18
C_i^0	**6**	**9**	**13**	**19**	**25**
F_i	**6**	**15**	**28**	**47**	**72**

Tabla.4.17. .Paso 5 en el caso del ejemplar.

Fmed = (6+15+28+47+72) / 5 = [33.6] => 34

Paso 6:

F_i	6	15	28	47	72
d_i	6	10	14	19	20
T_i	**0**	**5**	**14**	**28**	**52**

Tabla.4.18. .Paso 6 en el caso del ejemplar.

Tmed = (0+5+14+28+52) / 5 = [19.8] => 20

Entonces, las cotas halladas para los valores Fmed y Tmed, son

	Cota inferior	Cota superior
(Fmed, Tmed)	(15, 3)	(34, 20)

Tabla.4.19. Ejemplo numérico del cálculo de las cotas.

4.4.2. Obtención del fitness de una solución

Uno de los elementos principales en un algoritmo genético es el fitness con que se evalúa una posible solución.

En la determinación de una solución multicriterio según dos criterios, el objetivo es hallar la frontera de Pareto. En dicho gráfico se construirá una parrilla donde sus límites vendrán determinadas por las cotas calculadas.

Se ha decidido fijar una cuadrícula de 10 filas y 10 columnas. Por lo tanto, se llevan los valores de cotas encontrados de Tmed y Fmed para disponer de una escala de 0 a 10.

Ejemplo (para una población de 6 individuos):

individuos	***1***	***2***	***3***	***4***	***5***	***6***	***Cota inf.***	***Cota sup.***
Fmed	18	20	21	24	28	30	**15**	**31**
Valor a una escala [0;10]	8.125	6.875	6.25	4.375	1.875	0.625	**10**	**0**
Tmed	4	5.5	7	12	16.3	17.5	**3**	**18**
Valor a una escala [0;10]	9.33	8.33	7.33	4	1.133	0.333	**10**	**0**

Tabla.4.20. Determinación de los valores de fitness a pedir de Fmed y Tmed (escala de 0 a 10)

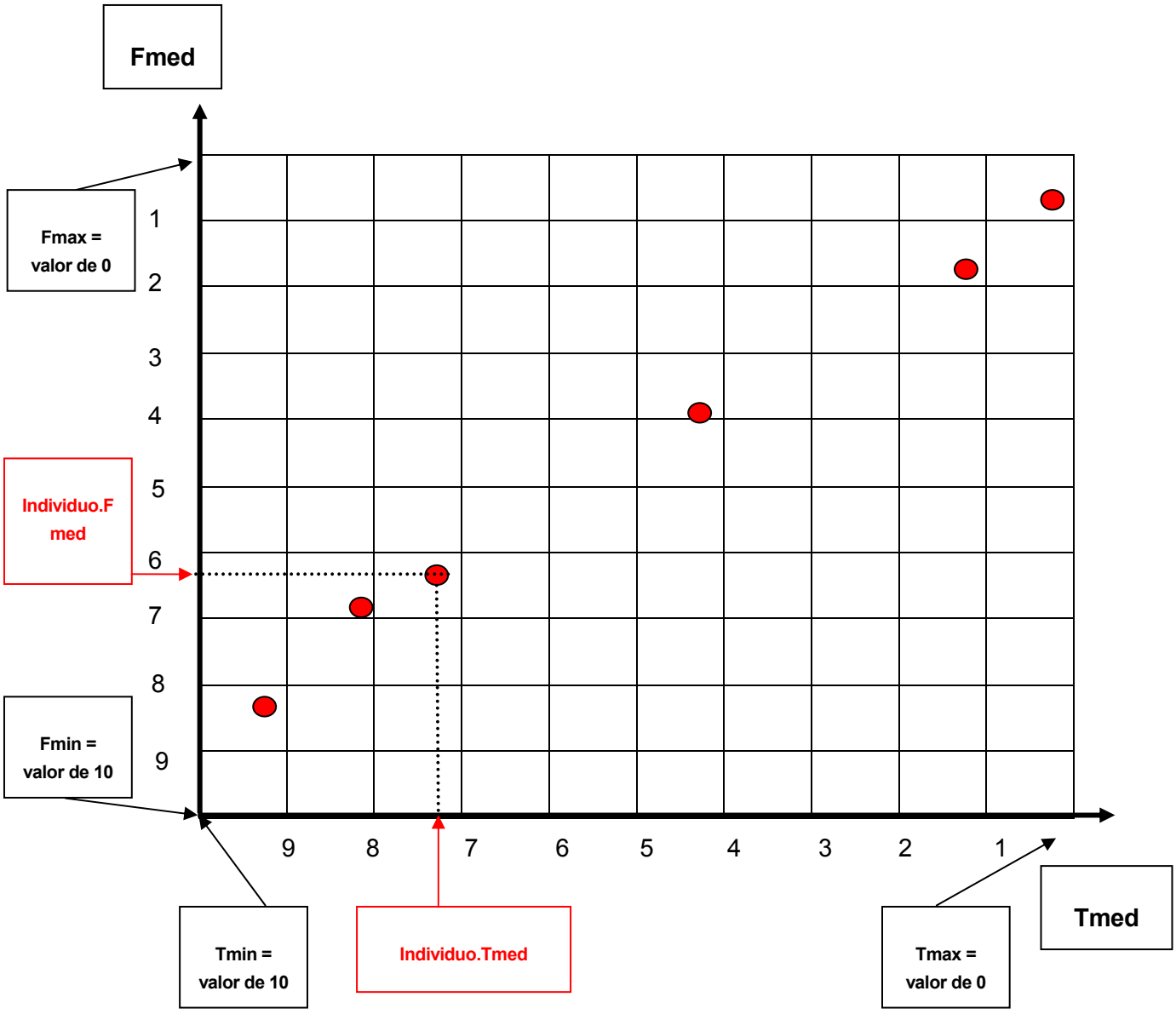


Fig. 4.9. Repartición de los individuos en la cuadrícula (en escala [0; 10]).

A partir de aquí se han estudiado dos tipos de funciones para obtener un valor de fitness:

a) Fitness continúo:

En este caso, se multiplica el valor de la abscisa de un individuo (representado por un punto) con el valor de su ordenada, por lo tanto se obtiene para cada individuo un valor de fitness que se sitúa entre los valores de fitness extremos 10*10= 100 y 0*0 = 0.

individuos	*1*	*2*	*3*	*4*	*5*	*6*	*Cota inf.*	*Cota sup.*
Valor (Fmed)	8.125	6.875	6.25	4.375	1.875	0.625	**10**	**0**
Valor (Tmed)	9.33	8.33	7.33	4	1.133	0.333	**10**	**0**
Fitness	**75.80**	**57.268**	**45.812**	**17.5**	**2.124**	**0.208**	**100**	**0**

Tabla.4.21. Cálculo del fitness del ejemplo con el método continúo.

b) Fitness diagonal:

En este caso, según donde se encuentre en la cuadrícula, un individuo tendrá un valor de fitness asignado. El rango de valores de fitness va de 10+10=20 hasta 0+0 = 0.

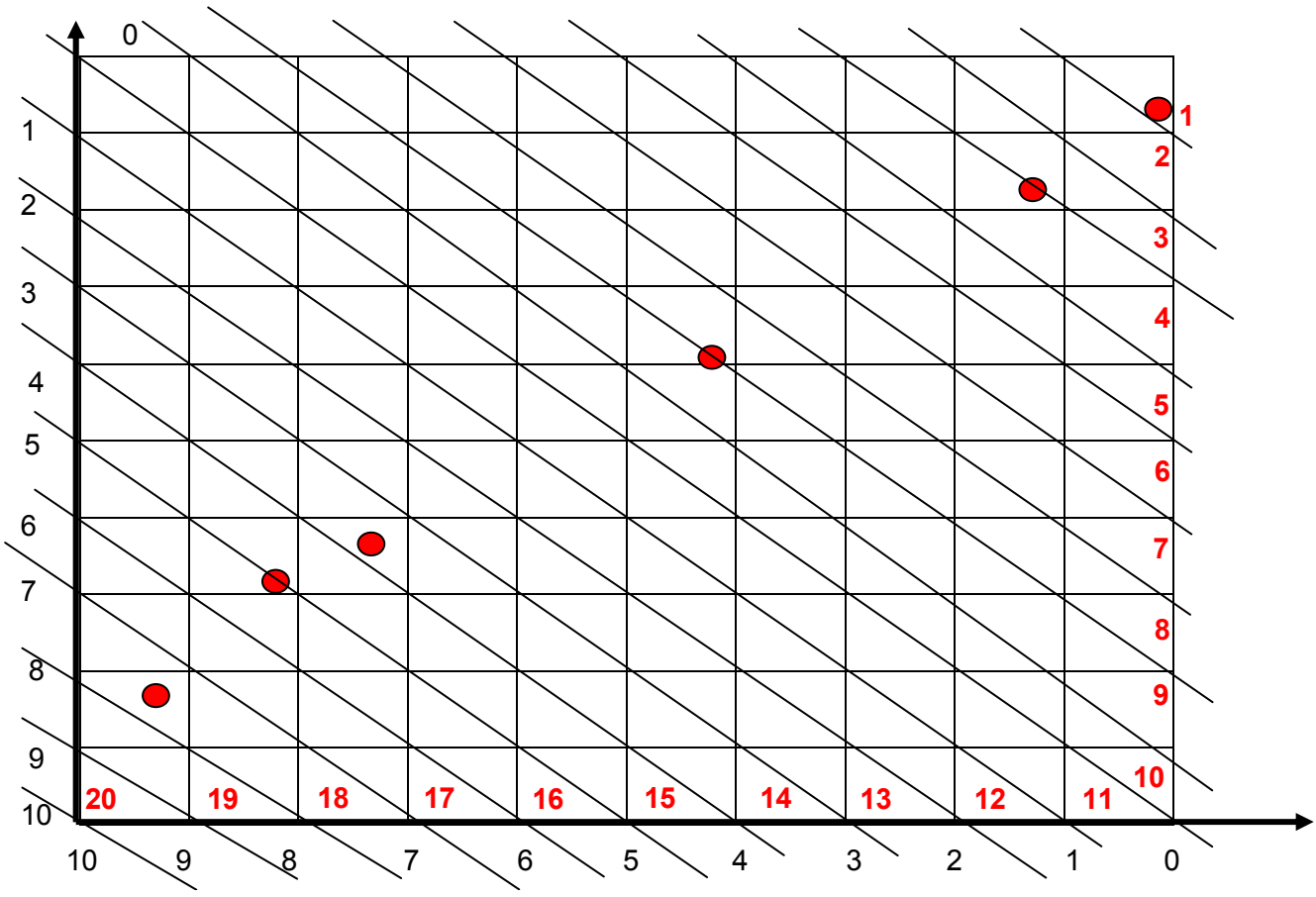


Fig. 4.10. Asignación del fitness según la parrilla con el método diagonal.

individuos	*1*	*2*	*3*	*4*	*5*	*6*	*Cota inf.*	*Cota sup.*
Valor (Fmed)	8.125	6.875	6.25	4.375	1.875	0.625	**10**	**0**
Valor (Tmed)	9.33	8.33	7.33	4	1.133	0.333	**10**	**0**
Fitness	**18**	**16**	**14**	**9**	**4**	**1**	**20**	**1**

Tabla.4.22. Asignación de los valores de fitness al ejemplo con el método diagonal.

c) Fitness perpendicular.

Se asignan los valores de fitness sin tanta precisión y premiando a las soluciones donde uno de sus criterios tiene un valor óptimo o cercano al óptimo. Se valoran igualmente los individuos teniendo un criterio muy bueno en particular independientemente del otro critero.

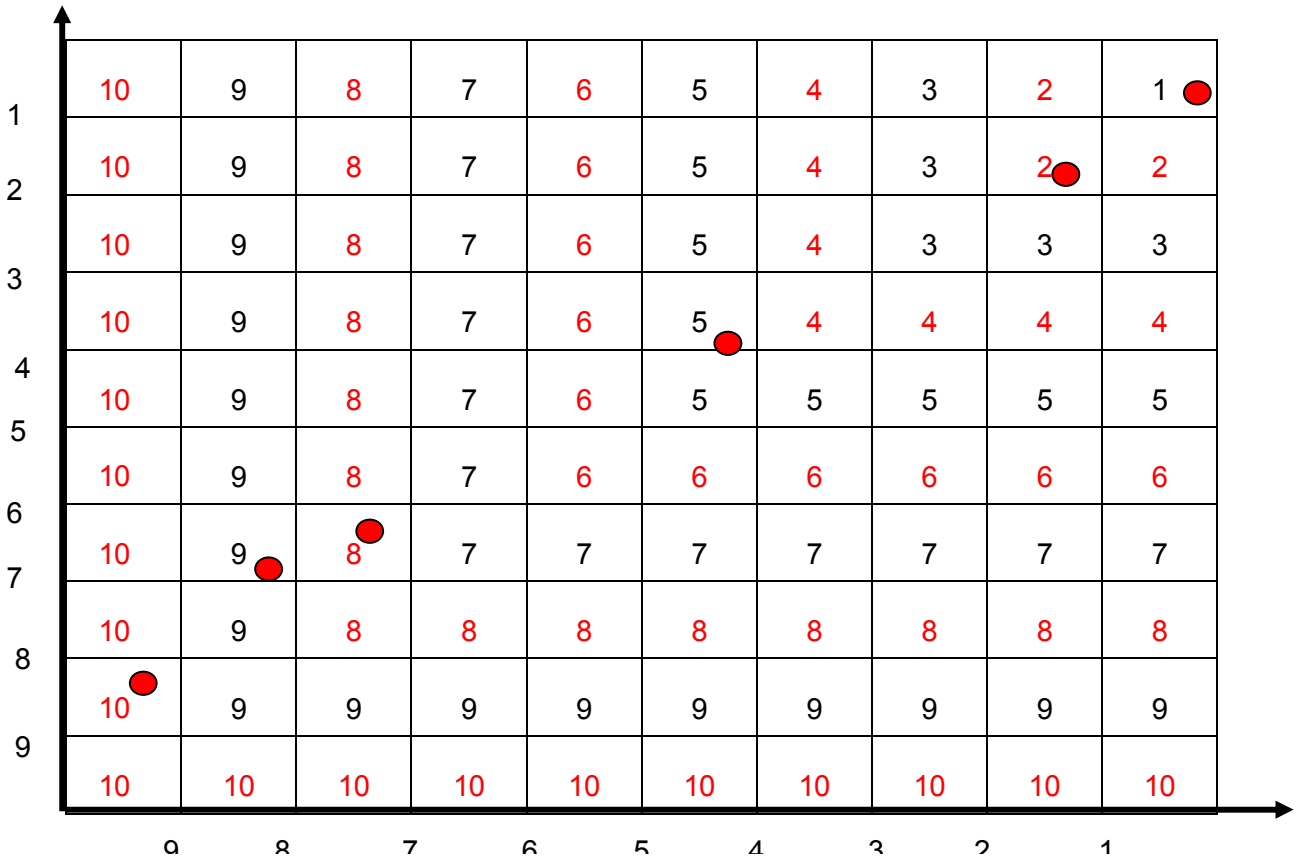


Fig. 4.11. Repartición y asignación de los valores de fitness con el método perpendicular.

individuos	*1*	*2*	*3*	*4*	*5*	*6*	*Cota inf.*	*Cota sup.*
Valor (Fmed)	8.125	6.875	6.25	4.375	1.875	0.625	**10**	**0**
Valor (Tmed)	9.33	8.33	7.33	4	1.133	0.333	**10**	**0**
Fitness	**10**	**9**	**8**	**5**	**2**	**1**	**10**	**0**

Tabla.4.23. Asignación de los valores de fitness al ejemplo con el método perpendicular.

4.5. Población de una nueva generación

La generación de la nueva población introduce algunas nociones como explicadas a continuación:

4.5.1. Número de generaciones

El número de generaciones se puede definir como el número de veces que se necesita cruzar, y mutar una población antes de obtener un conjunto de hijos y padres de suficiente calidad.

Se ha optado por la solución que consiste en efectuar un número fijo de generaciones. Se utiliza un contador que se incrementa a cada generación.

La condición de paro se cumple cuando el número de generaciones es igual al número prefijado por el usuario en el menú “opciones.”

Se ha decidido cruzar tantas veces como se necesita para obtener un número de descendientes igual al número de progenitores. Entonces, si se tiene una población de cant individuos, se obtendrán cant descendientes resultantes de los diferentes cruces.

Además, el operador de cruce aplicado a una pareja de progenitores permite obtener dos descendientes. Por lo tanto, para una población de cant individuos, se hará cant/2 cruces de parejas.

4.5.2. Elección de los padres en los cruces

En los diferentes artículos estudiados **[13]** previamente, se menciona la aleatoriedad en la elección de los padres como que refleja mejor las condiciones naturales. Sin embargo, se ha decidido añadir un factor de elitismo **[14]** en la elección de los padres para conservar las soluciones de buena calidad.

Un 20% de los descendientes serán escogidos a partir individuos de mejor fitness, o sea de la élite. Se ha decidido impedir el cruce entre 100% de elites para favorecer y mantener una cierta diversidad en la población generada, Entonces, cada uno de estos miembros de la élite será cruzado al azar con uno de los individuos restantes (formen parte o no de la élite).

Para el resto de cruces, los progenitores serán elegidos al azar entre ellos como lo sería en la naturaleza, conservando el hecho de que un individuo de mayor fitness tenga más probabilidad de ser elegido y, por tanto, más probabilidad de sobrevivir.

***Ejemplo*:**

Para una población de 30 individuos, la élite representa los 3 mejores individuos de la población, y se requieren 30 descendientes. Para ello, son necesarios 15 cruces, que darán como resultantes 15 parejas de individuos. Una vez se dispone de los 30 individuos, se ordena la lista de individuos de mayor a menor fitness.

Se cruza el individuo 1 de la lista con uno de los 29 restantes individuos que siguen en la lista al azar.

Se cruza el individuo 2 de la lista con uno al azar de los 28 individuos que siguen en la lista.

Se cruza el individuo 3 de la lista con uno al azar de los 27 individuos que siguen en la lista.

Luego, se aplican 12 cruces entre los 26 individuos al azar que no forman parte del 10% de las elites.

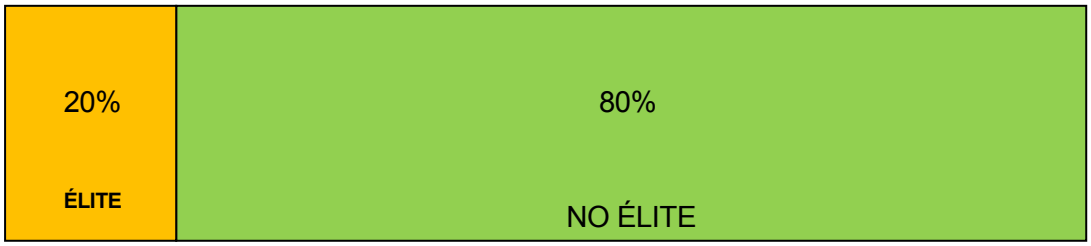


Fig. 4.12. Clasificación de los individuos de la población entre miembros de la élite y no miembros.

4.5.3. El cruce

Como es habitual en los algoritmos genéticos, un cruce de dos progenitores dará lugar al nacimiento a dos descendientes.

El menú propone 3 tipos de cruce que han sido inspirados por varios artículos. **[13]**

Para aplicar con más facilidad los diferentes operadores de cruce previamente se ha transformado la configuración elegida en la codificación de un individuo en un cromosoma valido para el cruce.

El cromosoma es un vector de enteros, donde los indicadores de cambio de máquina son representados por un **"0"** en la casilla anterior a la primera pieza que recibe una máquina.

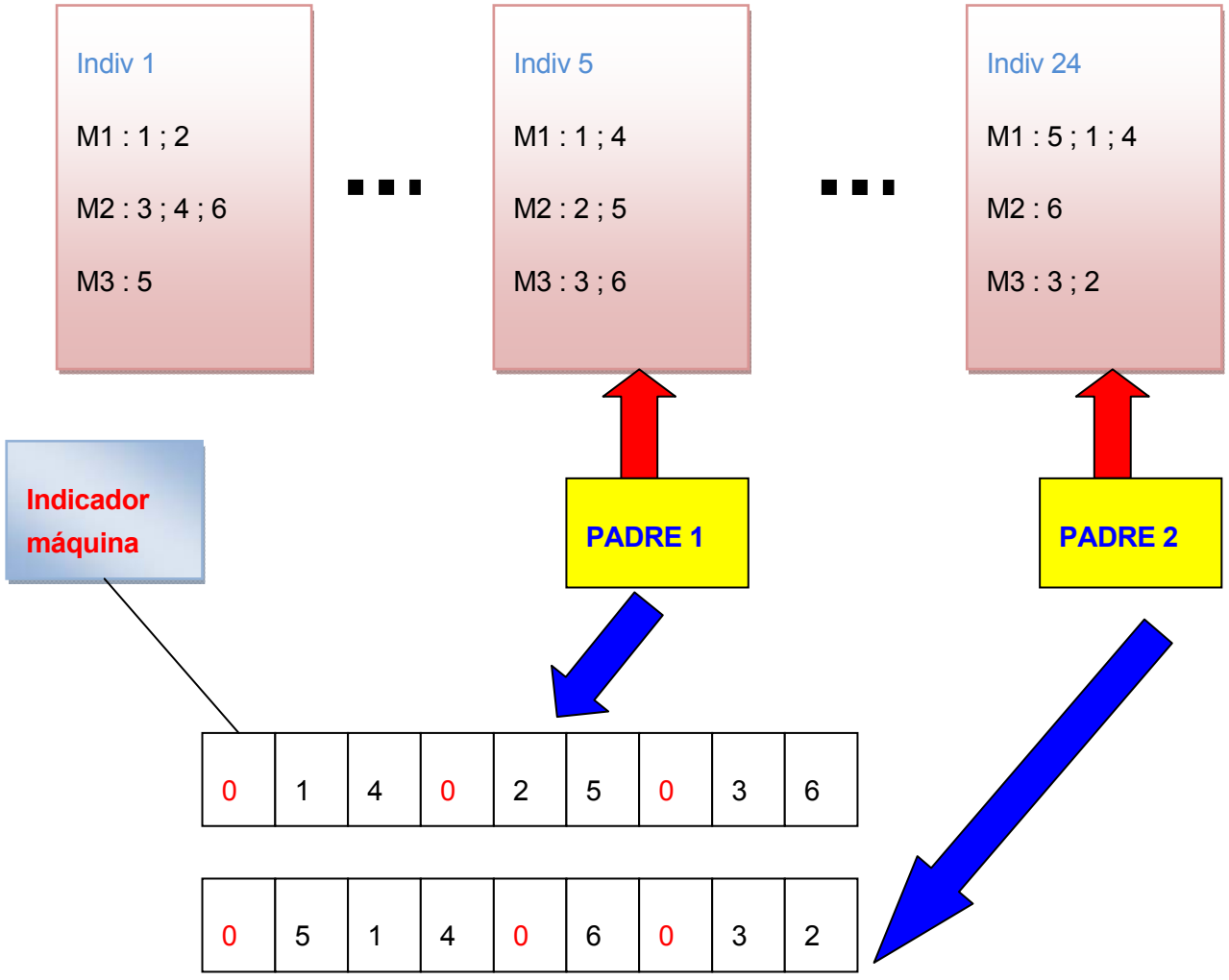


Fig. 4.13. Modelización de dos posibles soluciones como cromosomas aptos para el cruce.

La dificultad está en el hecho de que los indicadores de máquinas no deben ser movidos. Por tanto, provisionalmente se crea una segunda versión del cromosoma para el operador cruce quitando los indicadores de máquinas, y se volverán a colocar tras el cruce entre dos progenitores.

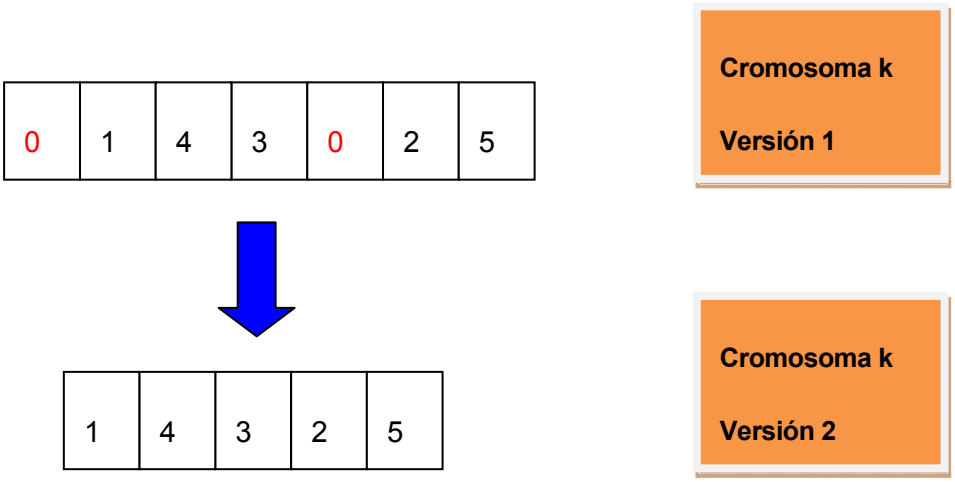


Fig. 4.14. Modelización de una posible solución como cromosoma apto para el cruce.

a) Cruce 1 punto

Se elige al azar un punto de cruce, y se copian los genes de un padre en uno de los hijos hasta el punto de cruce.

El resto de los genes no copiados se completan en el mismo orden en el que están colocados en el otro padre.

Se ha decidido fijar los indicadores de máquinas “0” del primer padre para crear el hijo 1, así como los indicadores de máquinas del segundo padre para el hijo 2.

En la figura 4.15 se muestra el resultado del cruce de los dos padres presentados en la 4.13.

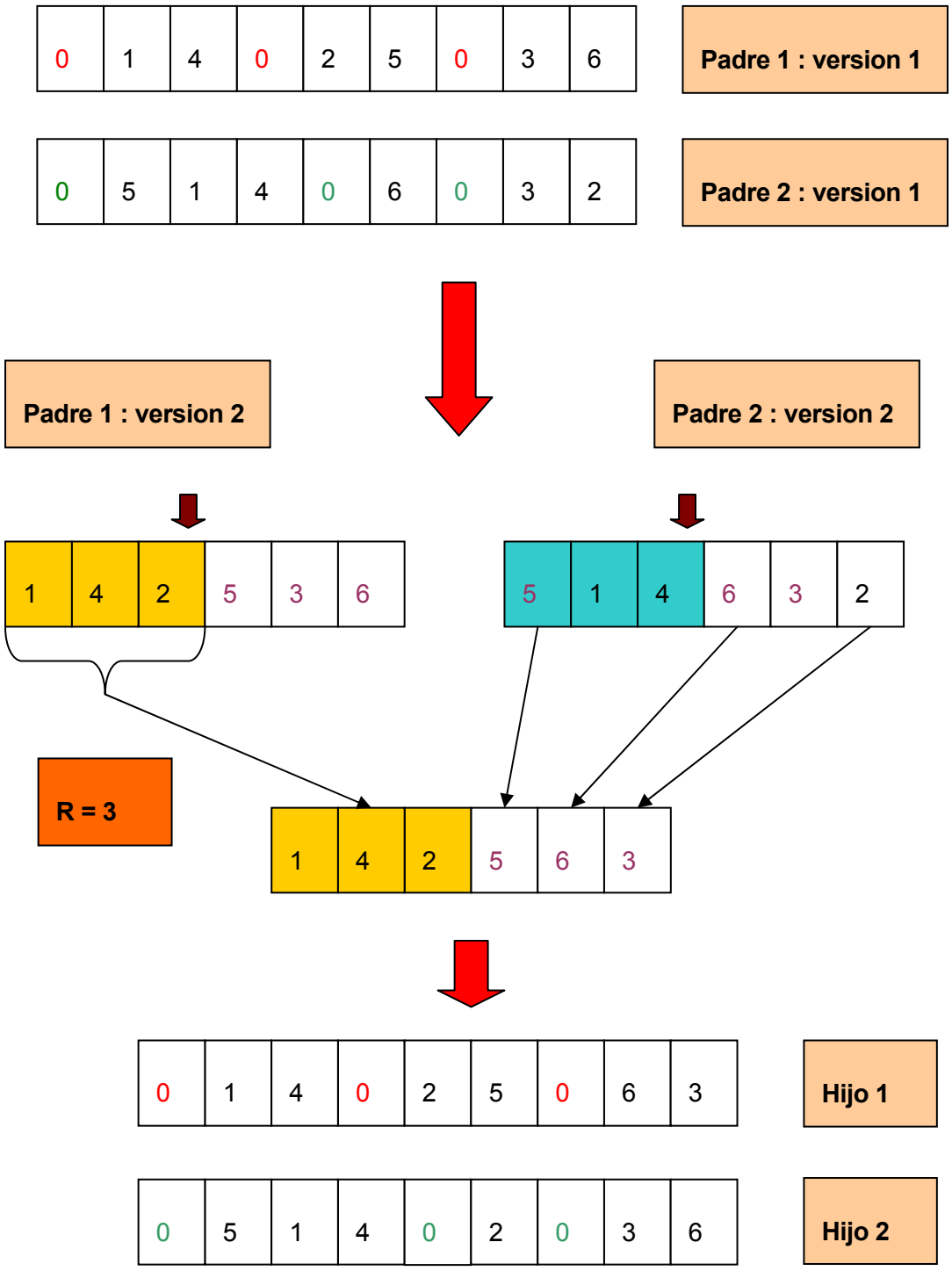


Fig. 4.15. Modelización del cruce 1 punto.

b) El cruce OX (Order Crossover)

En este caso, se procede igual que en el caso anterior con la salvedad que existen dos puntos de cruce.

Como para el cruce 1 punto, se generan aleatoriamente dos puntos de cruce. Los genes de un padre situados en el primer punto de cruce y el segundo se copian en el primer hijo, el resto de los genes no copiados se completan en el orden del segundo padre. Ídem para generar el segundo hijo.

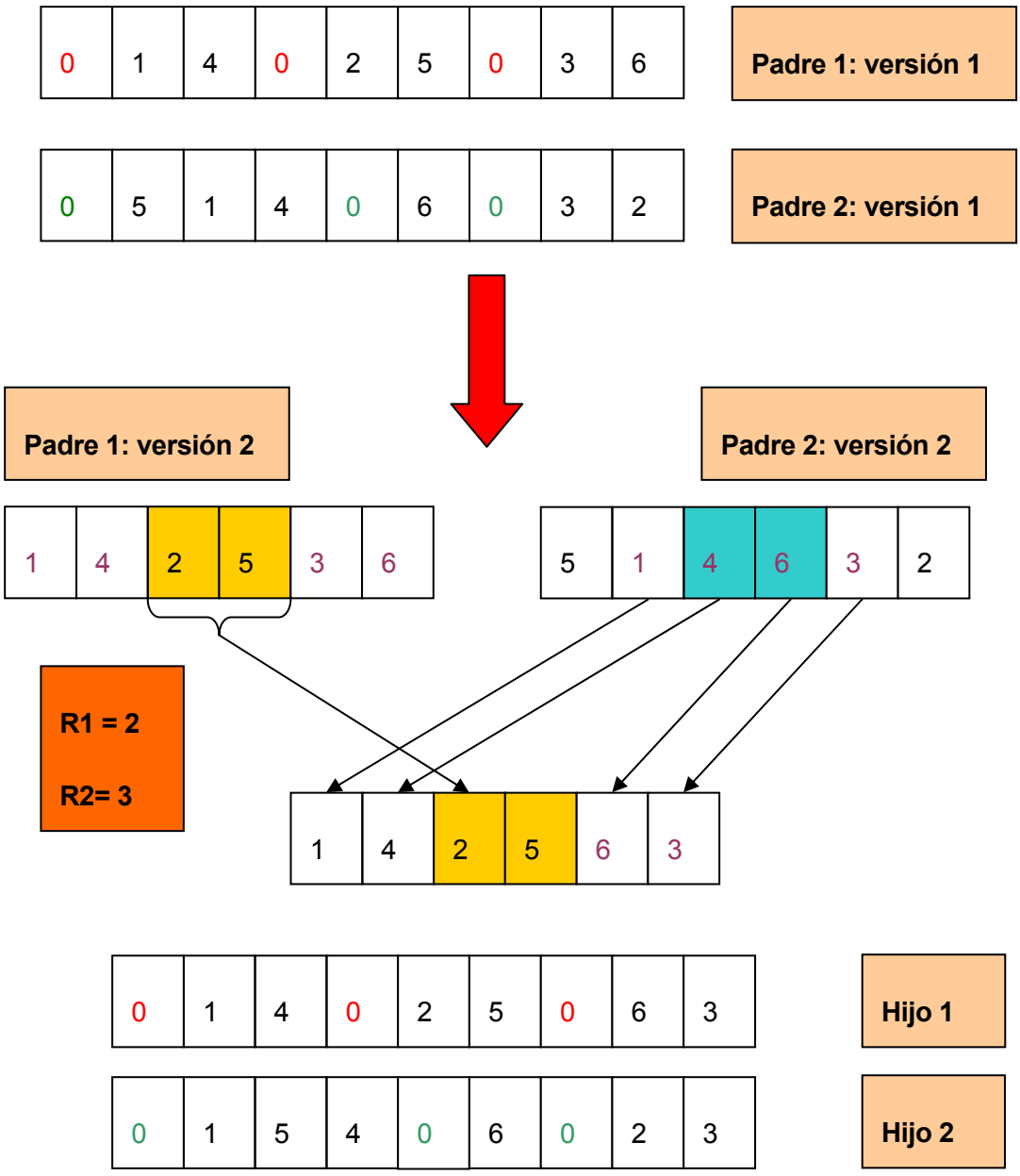


Fig. 4.16. Modelización del cruce OX a dos puntos.

c) El cruce PMX (Partially-Matched Crossover)

Se generan al azar dos puntos de cruce como para el cruce OX, Se copian en un hijo los genes que se sitúan entre el primer punto y el segundo punto de cruce del primer padre. El resto de los genes sigue dos pasos:

1. Los genes que no han sido copiados y que en el otro padre no están situados entre el primer y segundo punto de cruce, se copian en el mismo lugar donde se encuentran en el otro padre.

2. El resto de genes no copiados se completan en el mismo orden en el cual los tiene el otro padre.

Ídem para generar el segundo hijo basado en los genes que se sitúan entre el primer y el segundo puntos de cruce del segundo padre.

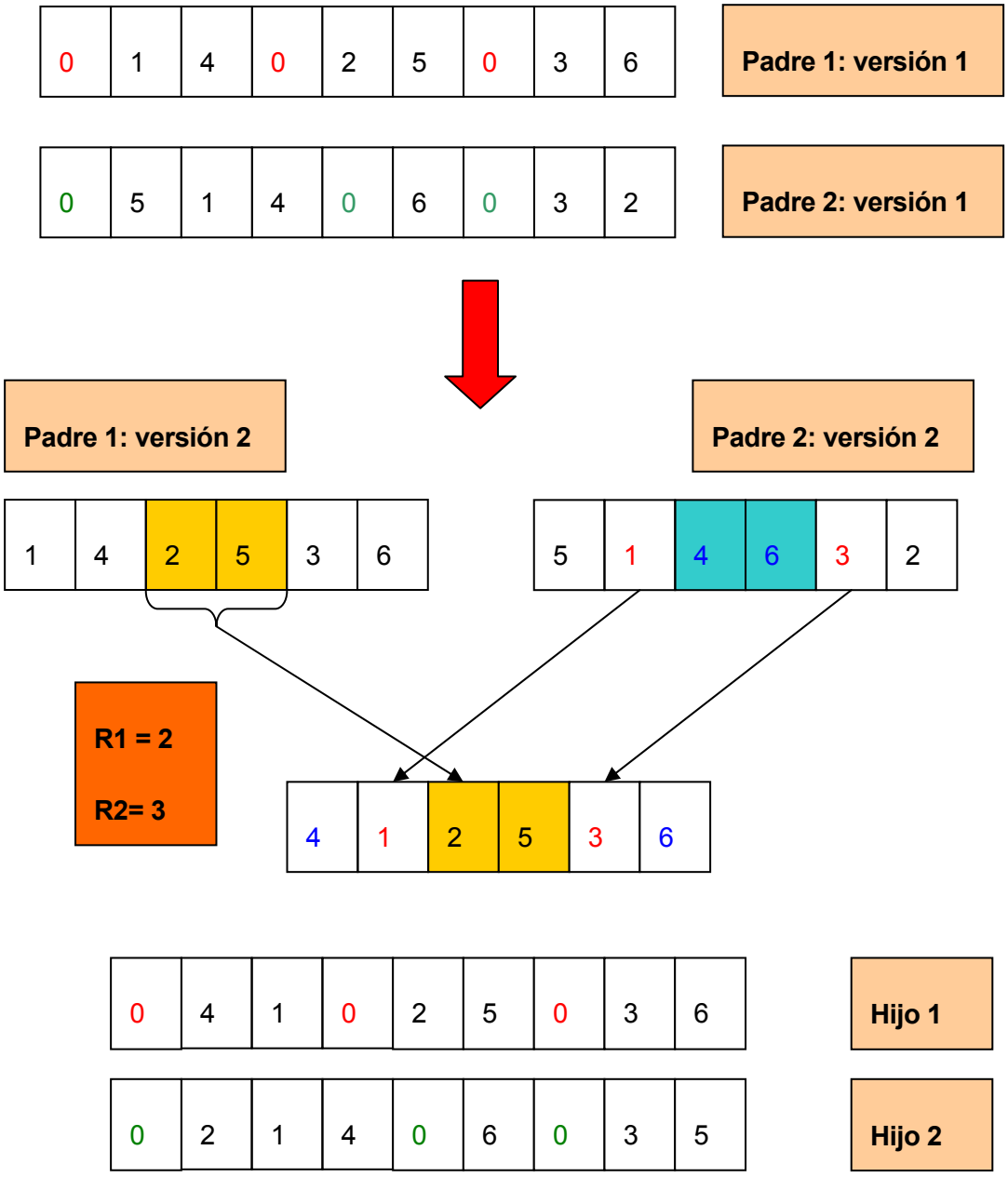


Fig. 4.17. Modelización del cruce PMX de dos puntos.

Se ha visto anteriormente que el menú prevé una parte probabilidad de cruce, que queda a la elección del usuario. Los estudios han demostrado que la probabilidad de cruce debe ser superior al 80% con lo cual el programa se lo advierte al usuario si éste entra un valor inferior a 80%.

Cuando se eligen y se crean los dos cromosomas padres, se genera aleatoriamente una probabilidad de cruce con la función Random.

Si la probabilidad obtenida es menor o igual a la probabilidad seleccionada en el menú opciones, se aplica el operador cruce.

Si no cumple, no se aplica ningún cruce como lo sería en la selección natural, y dos cromosomas hijos se generan iguales a los cromosomas padres elegidos.

4.5.4. Mutaciones

Una vez obtenidos los dos hijos, se somete a estos hijos a un proceso de mutación con el objetivo de originar variaciones elementales en la población.

Análogamente, se configura en el menú opciones una probabilidad de mutación que no debe ser elevada. Se ha fijado un límite superior al 40% lo cual se advierte al usuario del programa en caso de que quiera escoger un valor superior.

Si la probabilidad de mutación hallada aleatoriamente en el momento de la elección de los padres es menor o igual que el valor prefijado en el menú, se aplica el operador mutación sobre los cromosomas hijo resultantes del cruce; si no se cumple, los cromosomas quedan iguales.

Para aplicar el operador mutación se ha decidido trabajar con las primeras versiones de los cromosomas hijos (que incluyen los indicadores máquinas).

En efecto, impidiendo el movimiento del primer gen (es decir el primer indicador de la primera máquina), se conserva la posibilidad de poder mover un gen con entero positivo (una pieza) o un gen con valor nulo (indicador máquina). Esta última acción modifica el número de piezas que procesará cada máquina.

Se han desarrollado dos tipos de mutación: blanda e intercambio **[13]**

a) Mutación blanda

La mutación blanda consiste en intercambiar un gen, seleccionado al azar, con su consecutivo en la secuencia.

Por ejemplo, si el gen seleccionado es el quinto de la cadena, será con el sexto (figura 4.18):

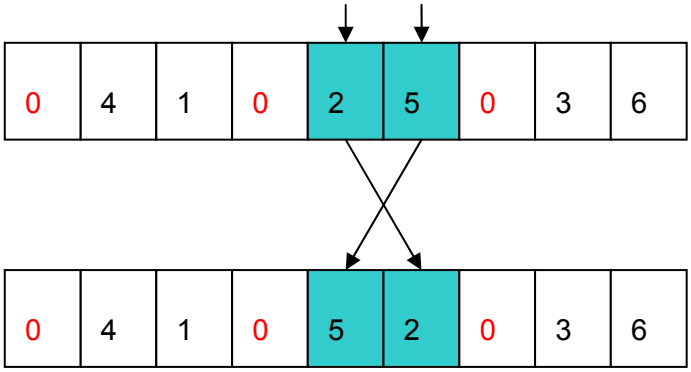


Fig. 4.18. Mutación blanda sobre un cromosoma sin intervención de indicador máquina.

En este ejemplo se mueven dos genes en el interior de las operaciones de una máquina, es decir, se modifica el orden de secuenciación para la máquina 2.

Sin embargo, si el gen a mutar es el número 3, se mueve un gen con un gen de valor ***nulo***, que está representando un indicador de cambio de máquina. En este caso, sería el equivalente de mover una pieza de una máquina a otra. Se modifica el número de piezas que van a procesar las dos máquinas, al mismo tiempo que se modifica su propia secuenciación.

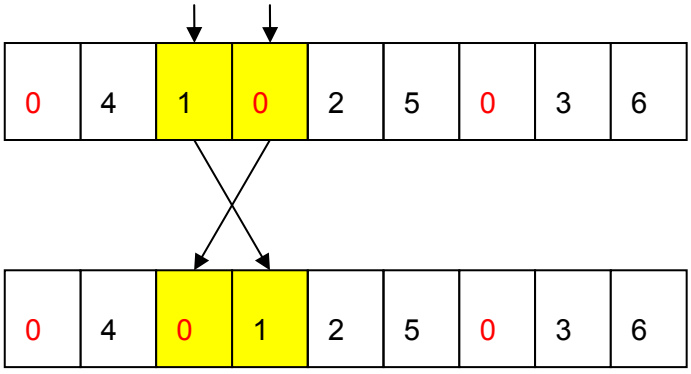


Fig. 4.19. Mutación blanda sobre un cromosoma con intervención de un indicador máquina.

En este caso, la máquina 1 tenía la secuencia de las piezas 4 y 1 mientras que la máquina 2 tenía la secuencia de 2 y 5. Tras la mutación con el gen de valor nulo, la máquina 1 procesa

únicamente la pieza 4 mientras la máquina 2 procesa la secuencia de piezas 1, 2 y 5. Así pues, la máquina 2 ha pasado de procesar dos piezas a tres y la máquina 1 de dos a una.

b) Mutación intercambio

La mutación intercambio consiste en intercambiar dos genes, seleccionados al azar, y no consecutivos en la cadena.

Se pueden mover análogamente como máximo un gen de valor nulo (indicador de maquina) con un gen numerado.

Mover dos indicadores de máquina en este caso se revela inútil, ya que la configuración de ambos casos es idéntica (no cambiaría).

Siguiendo con el ejemplo anterior, la mutación intercambio entre los genes 3 y 6 se muestre en la figura 4.20.

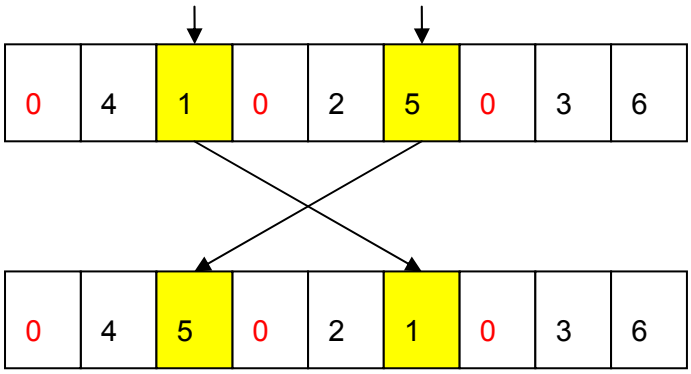


Fig. 4.20. Mutación intercambio sobre un cromosoma sin intervención de un indicador máquina.

La misma mutación sobre los genes 3 y 7 produce el movimiento de un gen de valor nulo (figura 4.21.)

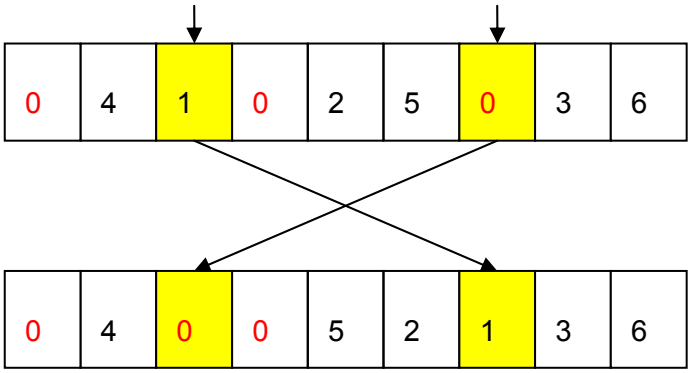


Fig. 4.21. Mutación intercambio sobre un cromosoma con intervención de un indicador máquina.

En este caso, la segunda máquina, tras la mutación, ya no tendrá ninguna pieza a procesar.

La solución obviamente parece peor, pero participa del proceso de diversificación de la población generada. El hecho de empeorar una solución dada permite evitar los óptimos locales y quizá dar en próximas generaciones una solución mucho mejor. Si se releva que es la óptima, se encontrará automáticamente en la siguiente población.

4.6. Composición de la población generada

Como se ha visto en el apartado 4.5., los operadores de cruce y mutación pueden aplicarse a toda la población. Todos los individuos pueden cruzar con otro individuo de la población y sus resultados sufrir alguna mutación.

Del cruce de dos padres resultan dos hijos. Sea ***cant*** el número de individuos en una población, tras los ***cant/2*** cruces, se obtiene un conjunto de soluciones de ***cant*** padres y ***cant*** hijos.

Se crea una lista de soluciones "***hijos + padres***" para calcular sus ***fitness***. Para cada solución de este conjunto, además se guarda la información sobre si la solución era una solución *"padre"* o *"hijo"*.

La lista se ordena enseguida de los mejores a los peores valores de ***fitness***.

La población de la siguiente generación estará constituida por ***cant*** individuos:

- El 30% de individuos de la población será escogido de las soluciones con mejor ***fitness*** en la lista "hijos + padres".
- 20% serán escogidos de la lista "padres" entre mejores padres siempre que no hayan sido ya escogidos..
- Los 50% restantes serán escogidos de la lista "hijos" entre mejores hijos siempre que no hayan sido ya escogidos.

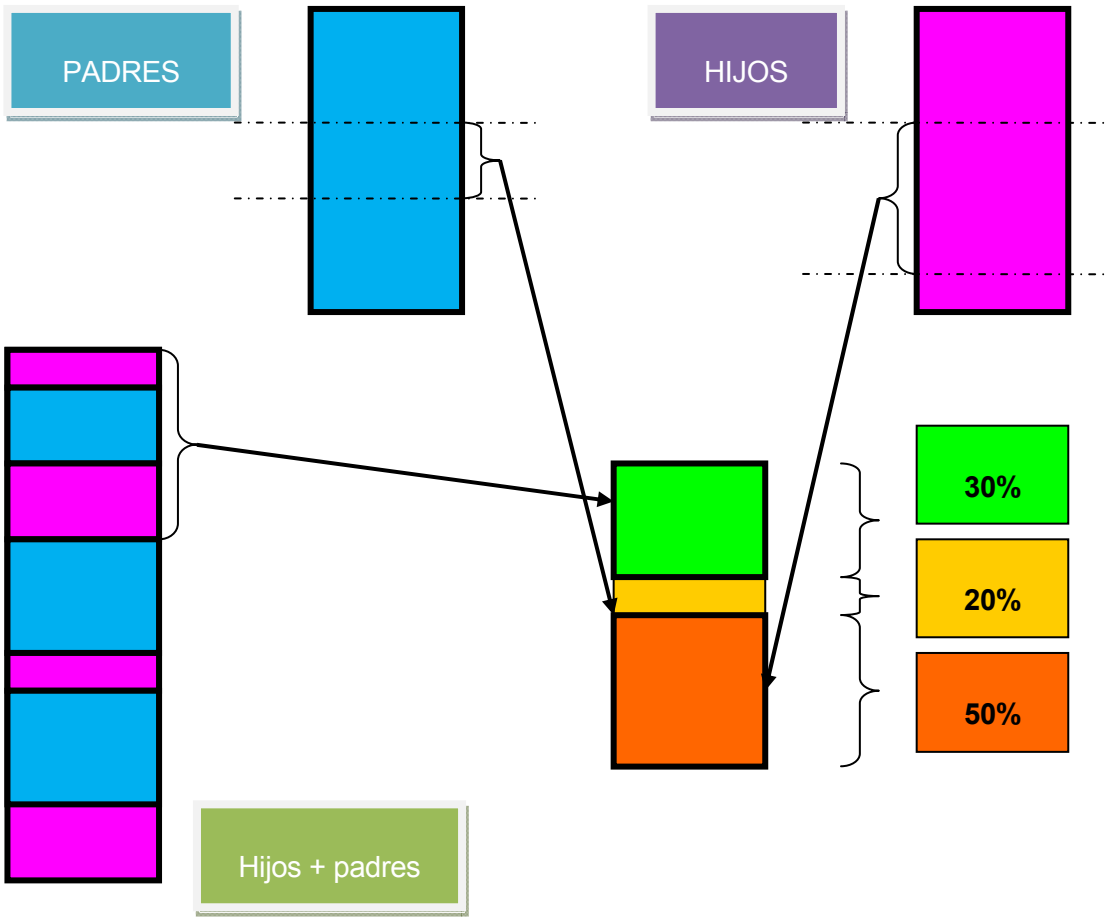


Fig. 4.22. Composición de la población a formar en una nueva generación.

A cada generación, se crea una nueva población. La población generada se convertirá en la población de padres a la siguiente iteración donde se vuelve a empezar el proceso detallado hasta aquí.

La población se regenera hasta alcanzar un número de veces fijado en el menú opciones: ***"num"*** cuyo contador se incrementa a cada nueva población generada.

4.7. Resultados gráficos, dominio de Pareto.

4.7.1. Construcción del dominio de Pareto

Se ha decidido construir el dominio de Pareto y mostrar los resultados en pantalla.

El algoritmo genético permite obtener un conjunto de soluciones a cada nueva generación. Por lo tanto, se construye en paralelo a cada nueva generación creada una lista de soluciones llamada "Solución de Pareto". En esta lista de soluciones, se guardan los mejores individuos que no están dominados en el sentido de Pareto presentado en el apartado 3.5.3.

Además, como se puede ver en la figura 4.23, si un individuo, que ya ha sido seleccionado para la solución de Pareto, se encuentra dominado por otro individuo seleccionado en una otra generación, se borra de la lista. En efecto, en la solución de Pareto, se conservan únicamente las soluciones que no se son dominadas en el sentido de Pareto.

Al cabo de tantas generaciones como fijado por el usuario, se encuentran un conjunto de soluciones que representan el dominio de Pareto del ejemplar en curso.

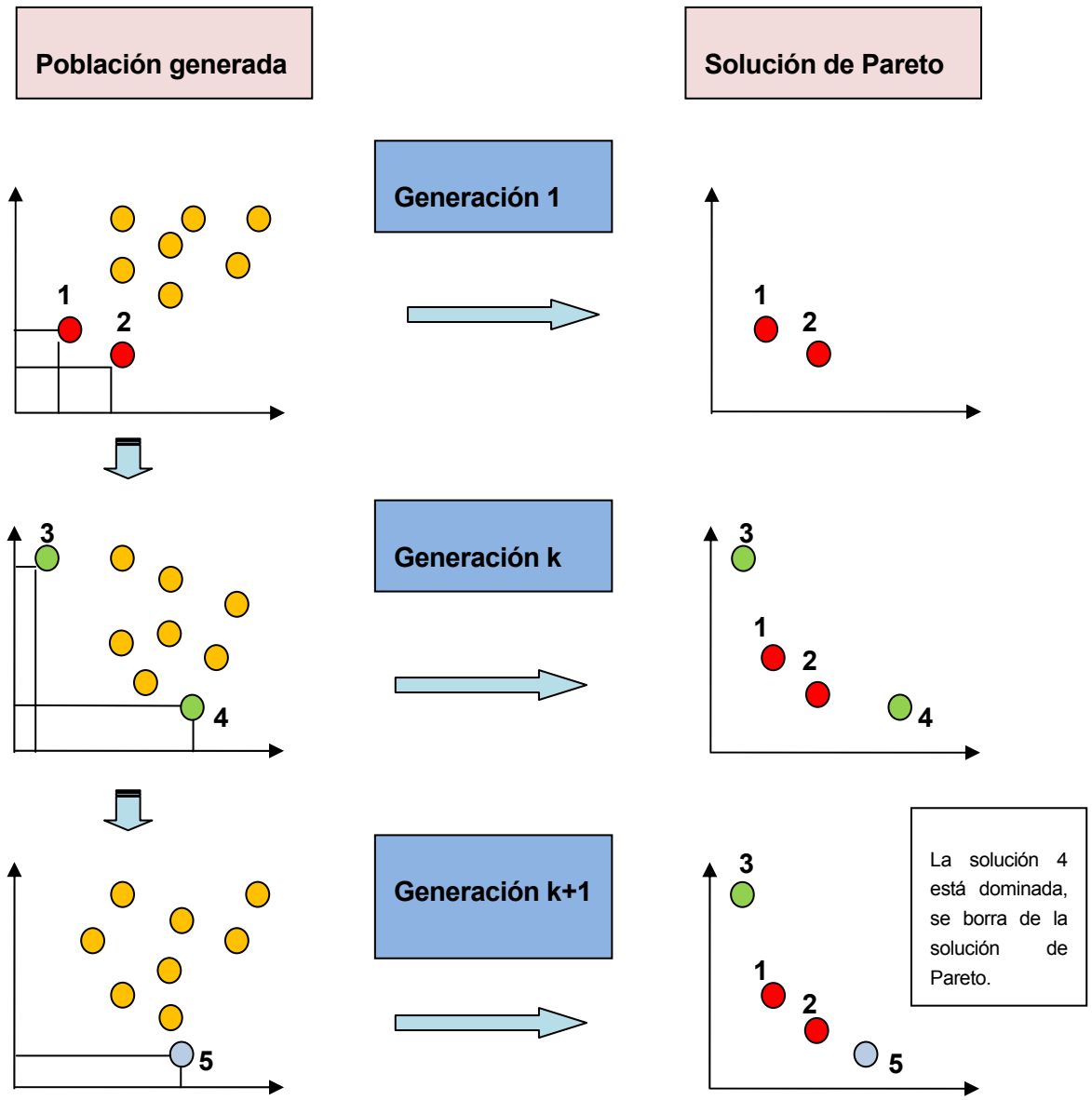


Fig. 4.23. Procedimiento de la construcción del dominio de Pareto para una generación k (1 ≤ k ≤ num).

El programa escribe, en un archivo de texto, los resultados numéricos para cada individuo que forma parte de la solución de Pareto. Si los datos se ponen en una tabla Excel, se pueden representar con un grafico con los valores de Tmed en abscisas y Fmed en ordenadas y correspondientes a la zona de Pareto del ejemplar.

4.7.2. Problemas de nichos en la frontera de Pareto.

Un problema habitual en las ejecuciones de los Algoritmos genéticos surge debido a la velocidad con la que el algoritmo converge. En algunos casos la convergencia es muy rápida, lo que suele denominarse *convergencia prematura*, en la cual el algoritmo converge hacia óptimos locales.

Además, en un problema bi criterios, donde se manipula un conjunto de soluciones (aquí una población), las soluciones pueden converger en una misma área. Por tanto para la construcción del dominio de Pareto, una convergencia prematura puede impedir una búsqueda hacia zonas más extremas (pequeño Tmed y grande Fmed, y su contrario).

La diversidad de los individuos retenidos para formar parte de la población generada es importante, y el estudio de la diversidad se hará según el tipo de asignación de fitness que se escoge.

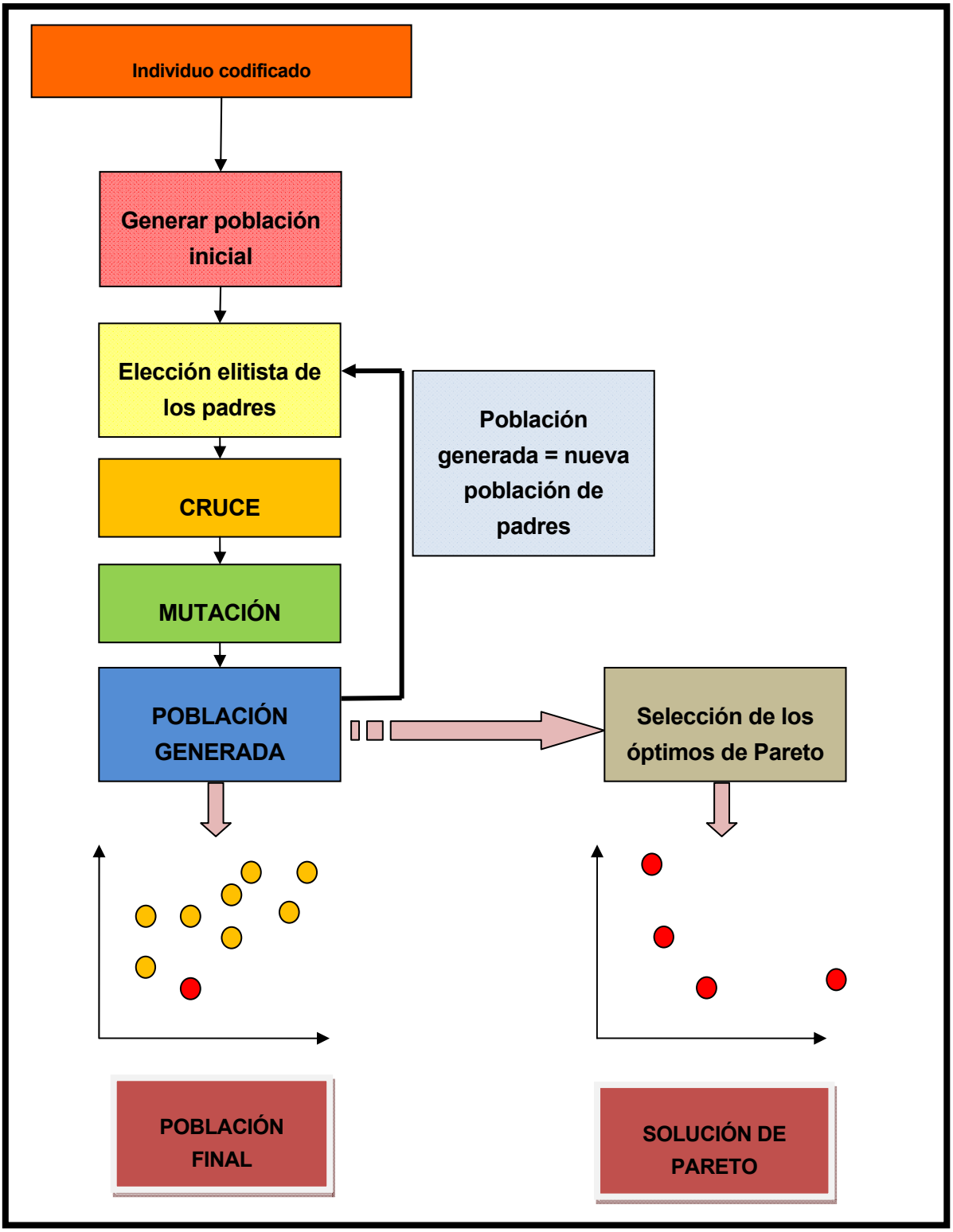


Fig. 4.24. Esquema general del proceso de construcción de la solución de Pareto para un ejemplar.

5. Experiencia Computacional

Se ha llevado a cabo la programación informática del algoritmo planteado en el capítulo anterior. Se ha codificado en lenguaje Basic y ha sido compilado mediante el compilador *Microsoft Visual Basic versión 6.0*.

Aunque quede posible para el decisor de fijar los parámetros del algoritmo genético en el software, se ha propuesto un estudio de los parámetros que proporcionan mejores resultados y que el usuario del software podrá poner como valores por defecto. En este capitulo, se presentan los tipos de ejemplares que han sido utilizados para la experiencia computacional. Luego se hace un estudio del fitness que ha permitido obtener soluciones más heterogéneas, y por fin se hace un estudio de los parámetros propios al algoritmo genético.

5.1. Análisis de los ejemplares

Para comprobar la viabilidad y aptitud del algoritmo propuesto se dispone de tres colecciones de 100 ejemplares, donde n (número de piezas) toma los valores de 15, 20 y 50 en cada una de las colecciones.

Han sido generados por un generador que sigue la filosofía de Potts y Van Wassenhove **[14]**. Se caracterizan todos porque tienen unos tiempos de preparación dependientes de la secuencia. No obstante, estos tiempos son independientes de las máquinas ya que estas son idénticas.

Estos ejemplares están basados en cuatro parámetros **Pmax**, **λ**, **T** y **R** además de **n** (número de piezas) y **m** (número de máquinas).

Los tiempos de proceso se distribuyen uniformemente entre 1 y Pmax y los tiempos de preparación entre 1 y Smax, con **Smax= λ*Pmax.**

Los tiempos de preparación, en las máquinas, se distribuyen uniformemente entre 1 y Smax/2.

Las fechas de vencimiento se distribuyen uniformemente entre Cmax*(1-T-R/2) y Cmax*(1-T+R/2) donde Cmax es el makespan teórico.

Las colecciones generadas están formadas por 10 ejemplares cada una y son combinación de **λ**, **T** y **R,** cuyos valores se muestran en la tabla 5.1.

λ	0.5	0.8	1.2	
T	0.2	0.4	0.6	0.8
R	0.2	0.4	0.6	0.8

Tabla.5.1. Tipos de ejemplares disponibles según valores de λ, T y R.

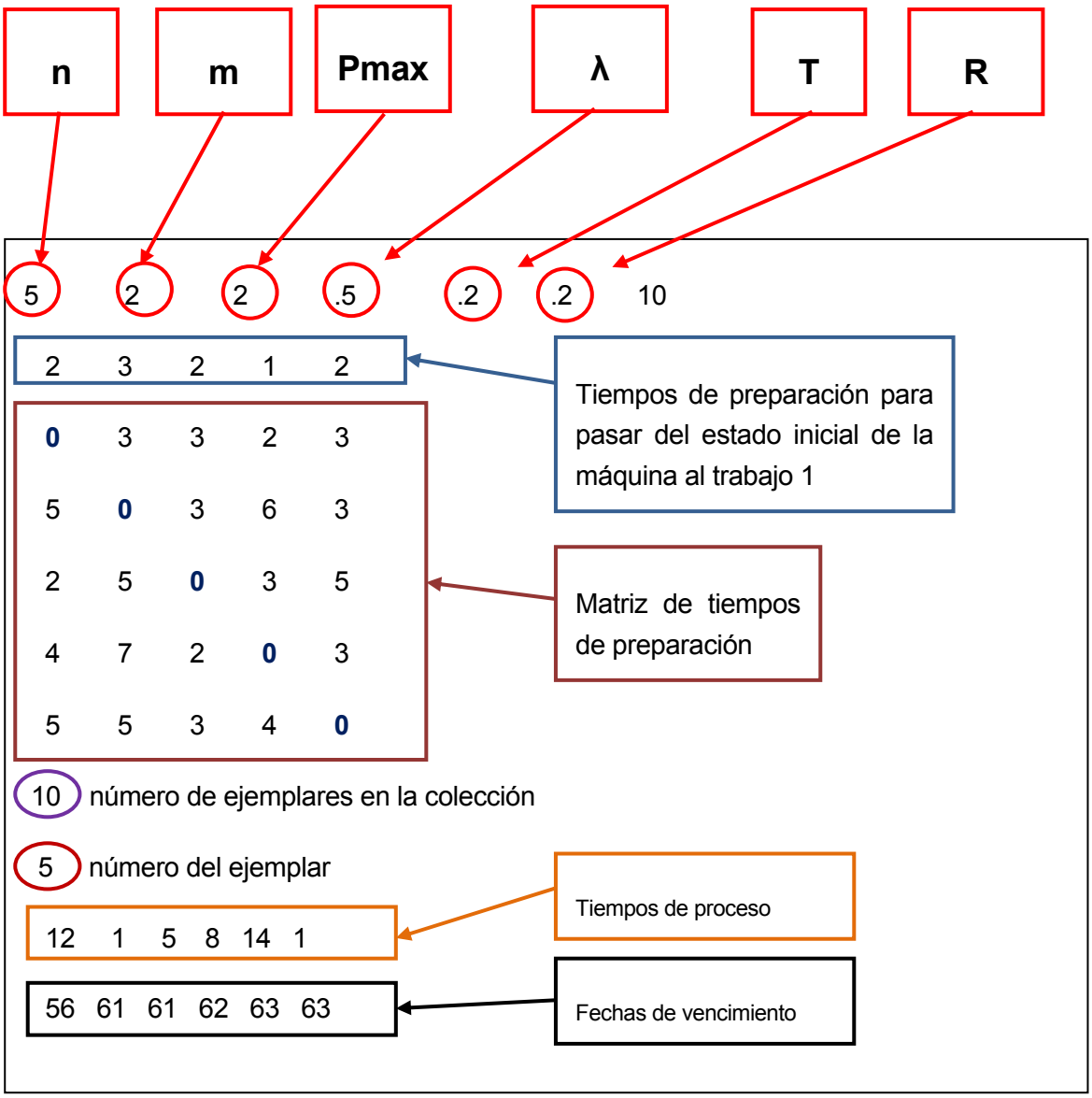


Fig. 5.1. Ejemplo de un archivo de datos para el programa.

En el Laboratori d'Organització Industrial del Departamento de Organización de Empresas, se han generado un total de 48 colecciones (3 valores λ* 4 valores T * 4 valores R) para cada combinación de número de piezas y máquinas.

Núm. de máq.\Piezas	15	20	50
2	X	X	
3		X	X
4			X

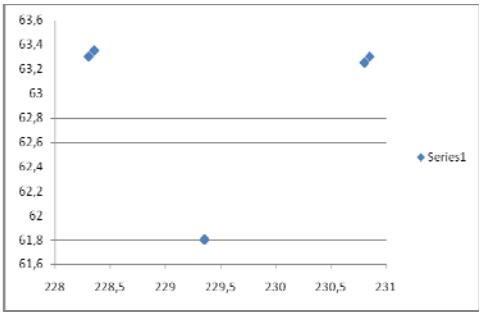
Tabla.5.2. Tipos de colecciones disponibles según n^0 de piezas y de máquinas.

5.2. Estudio del Tipo de Fitness

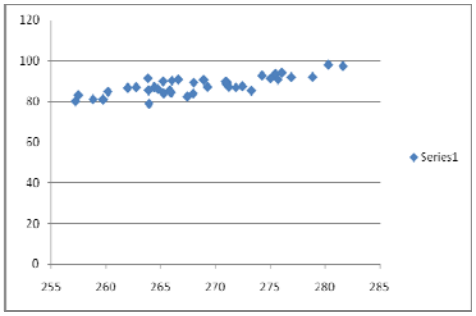
La medida de eficiencia de cada solución generada va a determinar en cuanto una solución se va a conservar a la siguiente generación. Por lo tanto, cuanto más diversa sea la población, se podrá explorar un espacio más amplio de búsqueda, y se podrá encontrar más fácilmente soluciones susceptibles de pertenecer a la frontera de Pareto.

El estudio se ha hecho sobre consideraciones gráficas en la última población generada por el algoritmo.: cuando más dispersos estén los puntos que aparecen en una población, mejor para tener soluciones muy diferentes y mediante cruces y mutaciones alcanzar buenas soluciones de la frontera. Entonces consideraremos de mejor calidad el fitness.

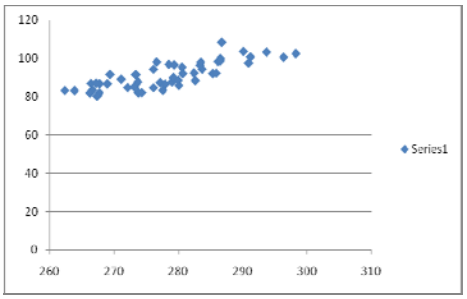
En los gráficos siguientes, se puede ver las soluciones obtenidas para las soluciones que componen la última población generada para los 3 tipos de fitness diferentes. En este ejemplo, el número de individuos de una población es de 60 y el número de generaciones 80: los tipos diagonal y perpendicular proporcionan ambos soluciones más diversas que el tipo continúo.



Tipo continúo



Tipo diagonal



Tipo perpendicular

Fig. 5.2. Poblaciones finales del algoritmo genético con un ejemplo de 20 piezas y 3 máquinas para los 3 diferentes tipos de fitness.

Se han utilizados 4 ejemplares con los mismos parámetros (20 piezas y 3 máquinas) sacado del juego de ejemplares disponibles y se han hecho 4 series de pruebas sobre estos 4 ejemplares cambiando a cada serie el número de generaciones (de 20 hasta 80 generaciones). A cada prueba se aplica el algoritmo con los 3 tipos de fitness: el fitness continuos, diagonal, y perpendicular (ver apartado 4.4.2).

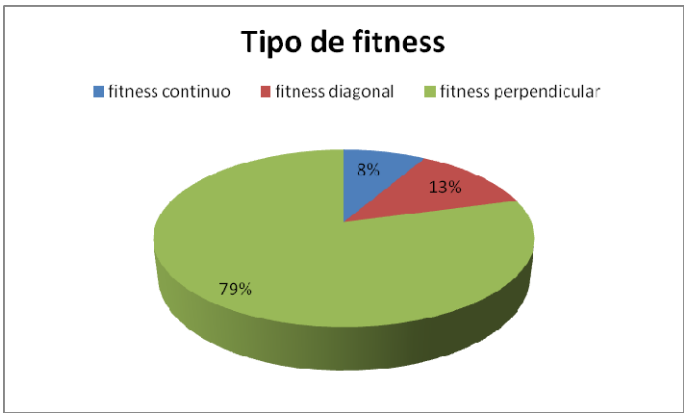


Fig. 5.3. Proporción de fitness encontrado al cabo de 24 pruebas.

Para el fitness continuo, se observa habitualmente una convergencia prematura, debido al elitismo proporcionado; en efecto, rápidamente las soluciones convergen hacia el centro del gráfico y se pierden las soluciones que se encuentran en los extremos del gráfico.

Para el fitness diagonal, las soluciones son más dispersas que para el fitness continuo, y proporcionas soluciones finales en la población de Pareto más heterogéneas, pero se pierden las soluciones que tienen uno de los dos criterios muy bueno y el otro malo.

El tercero tipo de fitness, perpendicular, favorece igualmente las soluciones con los dos criterios buenos, así como las soluciones que se encuentran en zonas donde solo tienen un criterio bueno sobre ambos criterios. En la población se encuentra más diversidad, con individuos que puedan tener los dos criterios buenos o individuos que tengan un solo criterio bueno sobre los dos.

Entonces, se ha fijado el fitness perpendicular para estudiar los parámetros del algoritmo genético.

5.3. Test de parámetros del algoritmo genético

Una vez más, el usuario puede fijar a su voluntad los valores de los diferentes parámetros según los criterios y las restricciones que valore él mismo (tiempo de ejecución, precisión, etc.), pero se ha hecho un estudio de los valores de los parámetros que dan los mejores resultados en la búsqueda de la frontera de Pareto. Y estos parámetros se tomarán como parámetros por defecto por el usuario en caso de que no tenga preferencias a la hora de fijarlos.

El análisis de los resultados del test de los parámetros del algoritmo persigue dos objetivos.

- Analizar la influencia de los diferentes parámetros elegibles (apartado 4.5) en el algoritmo.
- Determinar cuál o cuáles son las mejores combinaciones de los operadores, es decir, aquellas que proporcionan soluciones más competitivas.

Para analizar los valores de los parámetros más adecuados del algoritmo genético, se ha decidido coger un ejemplar de cada colección. Se han renombrado todos los archivos para que puedan ser leídos por el programa. *(Ver en anexo A: correspondencia entre nombres previos de archivos y los nombres utilizados para el test de parámetros*)

Se ha utilizado 24 ejemplares que fueron considerados como las más difíciles de tratar. Se han utilizado ejemplares de 50 piezas y 4 máquinas y con diferentes valores de parámetros diferentes para explorar el máximo de ejemplares con propiedades diferentes. (Ver apartado 5.1)

Entonces, los parámetros cuya cierta combinación pueda dar mejores resultados en la frontera de Pareto son los siguientes:

- Tamaño de la población inicial
- Número de generaciones
- Probabilidad de cruce
- Probabilidad de mutación
- Tipo de cruce a aplicar
- Tipo de mutación a aplicar

Estos parámetros se encuentran todos en el menú opciones del programa elaborado. *(Ver anexo D: manual de utilización del programa)*

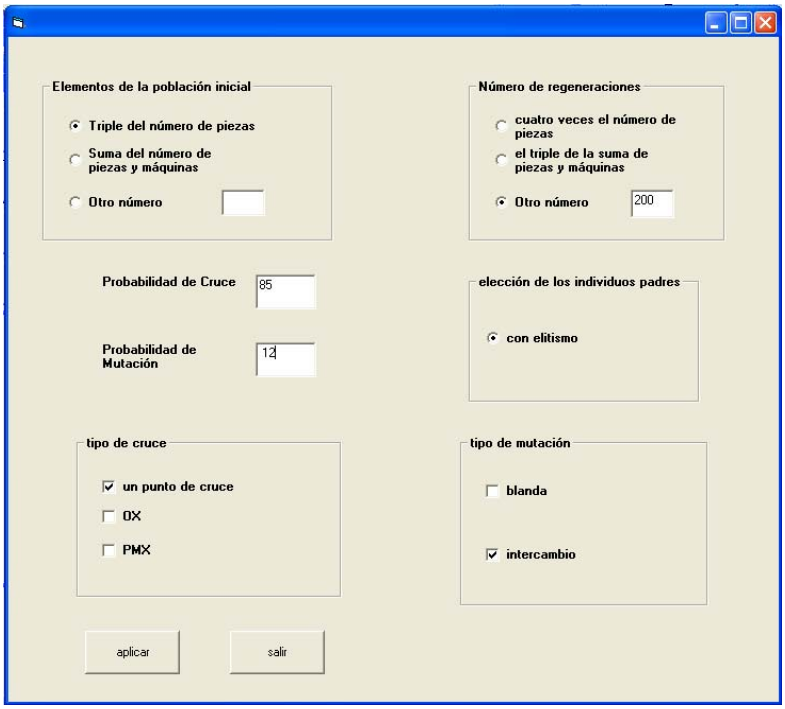


Fig. 5.4. Menú propuesto por el programa “Parallel Machine Management”

Las combinaciones entre ellas son infinitas, por lo que analizarlas todas es imposible.

Sin embargo, el estudio de artículos demuestra que los dos primeros están acotados superiormente ya que a medida que la población y el número de generaciones aumentan, el coste computacional resulta cada vez más elevado hasta un punto que impide toda viabilidad en el proyecto. Por lo tanto, la elección de estos dos operadores supondrá un compromiso entre la aptitud de la solución y el coste computacional.

5.3.1. Tamaño de la población inicial y de las diferentes generaciones.

Utilizando artículos **[13]**, y tras la comprobación con el algoritmo genético de que el rango de soluciones era aceptable en un tiempo de cálculo razonable, se ha decidido fijar el tamaño de la población a 3 veces el número de piezas. Sean ***cant*** el tamaño de la población inicial y ***n*** el número de piezas: ***Cant= 3*n***

5.3.2. Número de generaciones

El estudio se ha hecho sobre una colección de 96 ejemplares con ciertas propiedades. (*Ver anexo C).* Se tiene que encontrar un compromiso entre un número que dé mejores soluciones con un tiempo computacional razonable.

Los resultados están resumidos en la figura 5.5.

A partir del estudio de artículos **[13]**, se ha partido de un número de generaciones igual a cuatro veces el número de piezas, a partir del cual se ha bajado el número de generaciones de 10 en 10 hasta encontrar soluciones de menor calidad.

Luego, se aumenta el número de generaciones para analizar si es posible alcanzar mejores soluciones. En general, se obtiene la misma calidad de soluciones.

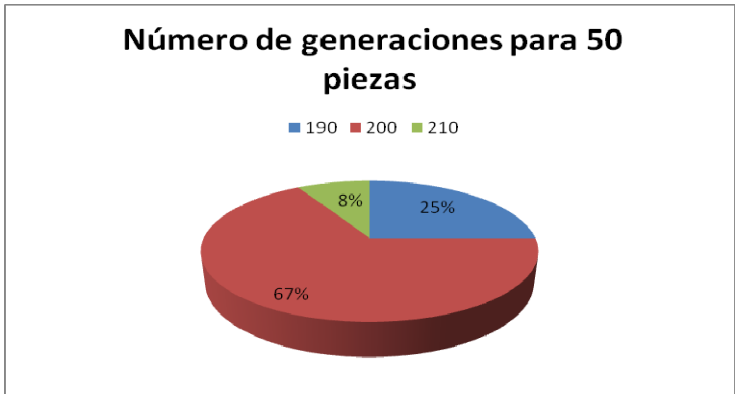


Fig. 5.5. Número de generaciones para ejemplares de 50 piezas y 4 máquinas.

Por lo tanto, los mejores resultados suelen aparecer para un número de generaciones igual a cuatro veces el número de piezas.

Pasado cuatro veces el número de generaciones, se alcanzan casi siempre las mismas soluciones para cada solución final, así como el tiempo computacional aumenta mucho. Así pues, se ha fijado el número de generaciones a cuatro veces el número de piezas para el estudio de las combinaciones del resto de parámetros.

5.3.3. Combinación de los operadores de cruce y mutación

Como se ha visto en el capítulo 4, la probabilidad de cruce estará comprendida entre el 80% y el 100%, así como la probabilidad de mutación entre 0% y 20%. Los posibles tipos de cruces son 3 (1 punto de cruce, OX, PMX); los tipos de mutación son 2 (blanda y intercambio).

El estudio de los operadores se ha hecho en 24 ejemplares escogidos de la colección de 96 ejemplares puestos a disposición. *(Ver anexo D)*

Los diferentes valores de parámetros que han sido estudiados aparecen en la tabla 5.3.:

Probabilidad de cruce	Probabilidad de mutación	Tipo de cruce	Tipo de mutación
90%	10%	1punto	blanda
80%	20%	OX	Intercambio
		PMX	

Tabla.5.3. Valores de parámetros para el estudio de la mejor combinación.

El número de combinaciones que se ha estudiado es de 2*2*3*2 = 24.

Tras haber recogido los datos Fmed y Tmed de las soluciones de Pareto para una selección de 24 ejemplares con 24 combinaciones diferentes (ver anexo D), se tiene a disposición 24*24= 576 juegos de resultados a analizar.

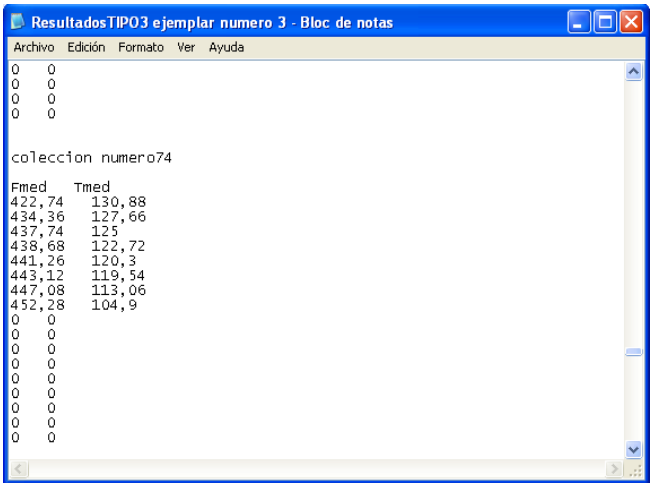


Fig. 5.6. Ejemplo de un fichero resultado del ejemplar número 3 del fichero número 74 del tipo 3 dado por el programa "Parallel Machine Management".

La dificultad reside en el hecho de que se tiene que analizar independientemente las parejas de valores Fmed y Tmed para cada solución, así como que para cada combinación el número de soluciones del dominio de Pareto es diferente, con lo cual un estudio con métodos estadísticos no es muy relevante.

Por tanto, se ha decido construir un método para seleccionar la combinación sobre las 24 posibles que dé los mejores resultados para los 24 ejemplares estudiados.

Se crea un valor de tipo fitness como se explica a continuación:

- Cada juego de valores Fmed y Tmed que forma una solución de la población de Pareto, se pone a una escala de 0 a 10, como se ha hecho para la construcción de los valores de fitness en el apartado 4.4.2.

- Para cada pareja de valores, comprendidos entre [0, 10], se extrae el valor máximo entre los dos y se redondea a dos decimales. Se obtiene así un valor numérico.

- Se hace un promedio de los valores obtenidos en el paso anterior para todas las soluciones de una población de Pareto y se obtiene un valor de fitness "medio" correspondiente a la combinación estudiada para un ejemplar.

Se obtiene tantos valores de fitness como combinaciones hay para los 24 ejemplares estudiados. Se selecciona la combinación que da el mejor fitness para cada ejemplar. En caso de empate entre dos valores de fitness, se mira más precisamente los valores de Fmed

y Tmed de las soluciones que constituyen los dominios de Pareto para las combinaciones donde hay el empate, y se escoge la combinación donde aparecen los valores mínimos de Fmed y Tmed. Un ejemplo de los pasos sucesivos se puede observar en la tabla 5.4. En este ejemplo, se elige la segunda combinación (9,9683>9,9467). En caso de muchos empates con dos decimales, se comparan los valores con más decimales.

80	**10**	**1punto**	**blanda**

Fmed	Tmed	Fmed [0, 10]	Tmed [0,10]	Valor máximo	Fitness "medio"
378,36	7,98	9,0980	9,9329	9,93	
381,38	6,2	9,0799	9,9478	9,95	
385,7	4,36	9,0539	9,9633	9,96	9,9467

80	**10**	**1punto**	**intercambio**

Fmed	Tmed	Fmed [0, 10]	Tmed [0,10]	Valor máximo	Fitness "medio"
345,46	5,42	9,2953	9,9544	9,95	
348,54	5,22	9,2769	9,9561	9,96	
350,76	5,02	9,2635	9,9578	9,96	
353,56	3,38	9,2467	9,9715	9,97	
376,42	2,38	9,1096	9,98	9,98	
384,18	1,34	9,0631	9,9887	9,99	9,9683

Tabla.5.4. Selección de la mejor combinación en un ejemplo entre dos de ellas.

La aleatoriedad juega un papel importante en la construcción del algoritmo. Por lo tanto, se ha decidido coger las tres mejores combinaciones de cada ejemplar de los 24 estudiados. La hoja de calculo está disponible en el CD ajunto al informe.

En la tabla 5.5, se puede observar las combinaciones retenidas.

	1				2				3			
1	80	10	PMX	inter	80	20	OX	inter	90	10	1punto	inter
2	80	10	1punto	inter	90	10	PMX	inter	90	20	PMX	inter
3	80	20	OX	inter	80	20	PMX	inter	90	20	PMX	inter
4	80	10	1punto	inter	80	20	1punto	inter	90	20	1punto	inter
5	90	20	OX	inter	80	10	1punto	blanda	80	10	OX	inter
6	90	10	OX	inter	80	20	1punto	blanda	90	10	PMX	inter
7	80	20	1punto	inter	80	10	1punto	inter	90	10	PMX	inter
8	80	20	1punto	inter	80	20	OX	inter	90	20	OX	inter
9	80	10	1punto	inter	80	10	PMX	inter	80	20	1punto	inter
10	90	10	OX	inter	80	10	1punto	inter	90	10	1punto	blanda
11	90	20	1punto	inter	90	10	PMX	inter	90	20	OX	inter
12	80	20	OX	inter	90	10	1punto	inter	90	10	PMX	blanda
13	80	10	1punto	inter	80	20	OX	inter	90	10	OX	inter
14	90	20	1punto	inter	80	20	OX	blanda	90	20	PMX	inter
15	90	20	1punto	inter	90	20	OX	blanda	90	20	PMX	blanda
16	80	10	PMX	inter	80	20	OX	inter	80	20	PMX	inter
17	90	20	1punto	blanda	80	10	1punto	inter	80	20	1punto	inter
18	80	20	1punto	inter	80	10	1punto	inter	80	10	PMX	inter
19	90	10	PMX	inter	80	20	OX	inter	90	20	OX	inter
20	80	10	1punto	blanda	80	10	OX	inter	90	20	PMX	inter
21	80	20	1punto	inter	80	10	PMX	inter	90	10	OX	inter
22	80	20	1punto	inter	80	20	1punto	blanda	90	10	1punto	inter
23	80	20	1punto	inter	90	10	OX	blanda	90	10	PMX	blanda
24	90	20	1punto	inter	80	10	OX	inter	90	10	PMX	inter

Tabla.5.5. Tabla de las 3 mejores combinaciones retenidas para los 24 ejemplares estudiados.

A partir de las 3*24= 72 combinaciones retenidas, se hace un análisis estadístico. Y se estudia cada parámetro independientemente del otro. Los resultados para cada combinación se exponen en la figura 5.7.

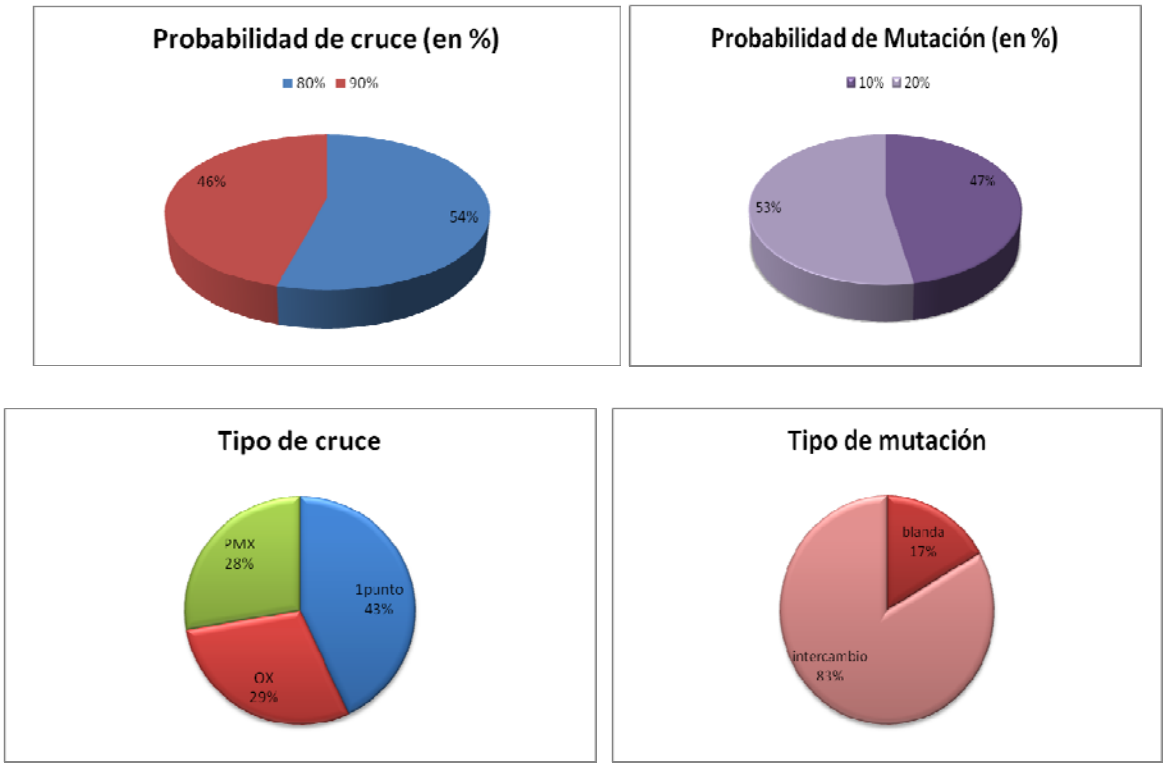


Fig. 5.7. Estudio de los parámetros sobre los 24 ejemplares.

La probabilidad de cruce del 80 % suele dar soluciones de mejor calidad (54%). Aunque no sea radical, se puede comprobar en la primera columna de la figura 5.10 que la probabilidad de cruce de 80% está también en la mayoría de las mejores soluciones (con un 62% de presencia en las mejores soluciones).

De la misma manera, no aparece obvio que la probabilidad de mutación que proporcione soluciones de mejor calidad sea un 20% con solo un 53% de presencia. Sin embargo, suele aparecer en la mayoría de las mejores soluciones como lo demuestra la primera columna de la figura 5.10 con un 58% de presencia. La probabilidad de mutación es el operador con el cual más se puede discutir.

El tipo de cruce que concede soluciones de mejor calidad es el tipo 1 punto que está en más de 1/3 (43%) de las 72 soluciones. Además, está presente en 2/3 (67%) de las mejores soluciones como se puede ver en primera columna de la figura 5.10.

Por fin, el tipo de mutación Intercambio proporciona sin duda las soluciones de mejor calidad con una presencia del 83%.

La combinación de los parámetros que se aconsejan en la elección del usuario y que proporcionan mejores resultados, considerando un límite de tiempo medio de ejecución por ejemplar de un minuto, es la siguiente:

Probabilidad de cruce	Probabilidad de mutación	Tipo de cruce	Tipo de mutación
80%	20%	1 punto	Intercambio

Tabla.5.6. Mejor combinación encontrada tras el estudio de los parámetros.

5.3.4. Estudio del comportamiento estacionario de las soluciones

El algoritmo desarrollado da lugar a mucha aleatoriedad en la parte construcción de la población inicial, así como en la elección de los progenitores para el cruce. Por lo tanto, cabe repetir los cálculos algunas veces para comprobar el comportamiento estacionario del algoritmo.

Los cálculos se han hecho 3 veces sobre todos los ejemplares de cada fichero disponible.

En total son unos 3(repeticiones de cálculos)*10(ejemplares por ficheros)*(2*96(ficheros a 2 o 3 máquinas)+48(ficheros a una máquina)= 7200 poblaciones de Pareto que han sido calculadas.

El decisor, en esta situación, debe valorar cuál de las soluciones de la frontera de Pareto elegirá en función de sus preferencias puntuales en ese momento.

Los resultados están disponibles en el CD ajunto y se encuentran los mismos resultados en mayoría para las 3 repeticiones de cálculos.

6. Presupuesto

A continuación, se lleva a cabo un análisis económico de la implantación de este software de programación ***"Parallel Machine Management"***.

En la fase de desarrollo del proyecto, los gastos iniciales que se estiman se reflejan en la tabla 6.1:

CONCEPTO	HORAS (h)	precio (€/h)	total (€)
1,Definición del Problema			
1.1 Búsqueda de Información	**200**	**30**	**6000**
1.2 Definición de las especificaciones	**50**	**30**	**1500**
2. Diseño			
2.1 diseño de los parámetros	**100**	**50**	**5000**
2.2 Diseño de la interfaz gráfica e implementación	**80**	**50**	**4000**
2.3 Depuración y pruebas	**160**	**50**	**8000**
3. Interpretación y Datos	**150**	**30**	**4500**
4. Estudio de la actividad Industrial	**150**	**30**	**4500**
5. Investigación tecnológica	**150**	**30**	**4500**
6. Actividades Administrativas			**600**
		TOTAL	**38600**

Tabla.6.1. Presupuesto de una aplicación industrial del programa "Parallel Machine Management".

Además, en el caso estudiado en este proyecto, se ha hecho hincapié que en la optimización bicriterio planteada, la gestión de los costes se hace según lo que al decisor le parece más pertinente.

Por un lado, la minimización de las fechas de terminación puede ser eficiente en una política interna de gestión del mantenimiento y del almacenaje con los costes de posesión relacionados.

Por otro lado, una minimización del retraso medio puede permitir el ahorro de todos los costes relacionados con un retraso en el pedido. Estos costes se detallan en el capítulo siguiente.

Si el decisor siempre mostrará una misma predilección por cada criterio, se podría comparar las soluciones sin la ayuda y con la que proporciona el software, y evaluarlo como una inversión.

7. Estudio del impacto medio ambiental

Respecto al impacto ambiental de la investigación posible desempeñada para este proyecto, el ahorro está esencialmente encuadrado en el despilfarro de papel ya que todo se hace con las herramientas informáticas puestas a disposición gracias al software diseñado.

Respecto al ahorro que puede proporcionar el programa desarrollado a una empresa, se puede asociar unos ahorros para cada criterio que se puede optimizar.

Una vez más, se deja al decisor la oportunidad de favorecer uno de los dos criterios, o encontrar un compromiso entre ambos.

7.1. Impacto del posible retraso

En muchas empresas, pueden surgir problemas de logística debidos a los retrasos en la fabricación de las piezas que pueden aparecer. En efecto, unos retrasos en la entrega de un producto pueden acarrear problemas con los medios de transporte para que se sean transportados hasta el almacén central o los centros de distribución.

Los costes relacionados con un viaje suplementario pueden ser ahorrados minimizando estos retrasos. Los nuevos Indicadores Clave de Desempeño (Key Performance Indicators) **[15]** para la cadena de suministro (Supply Chain) son la reducción de emisiones de CO_2 (por el consumo de carburante de los medios de transportes), así como la reducción de la congestión del tráfico o la reducción de costes debidos a los transportes. Esto puede evaluarse fácilmente con la presencia de ecotasas.

Si el decisor opta por una secuenciación de piezas que minimice el retraso medio, puede ahorrar o minimizar los costes mencionados hasta aquí.

7.2. Impacto de los costes operativos

En un proceso más enfocado en los costes directos de la planta industrial, o sea todo lo relacionado con la producción del producto final, el decisor puede jugar con el segundo criterio: el tiempo de fabricación medio de cada pieza.

En término de costes internos, los costes relacionados con los gastos de energía se pueden recortar. Si se asocian unos costes en relación al consumo de energía en kwh (kilovatio-hora), la programación adecuada permite ahorrar en gastos de electricidad para el taller por ejemplo.

Igualmente, para el equipo de producción físico, una programación adecuada permite un uso más eficiente del equipo de producción como el tiempo que las máquinas están en funcionamiento.

Los costes relacionados al mantenimiento se pueden también recortar. Se puede imaginar fácilmente una política de empresa que desee pasar sus costes de mantenimiento de fijos (como departamento interno a la empresa con un mantenimiento regular) a variables (llamando una empresa externa cuando surge un problema en el equipo). A largo plazo, si el decisor opta para una secuenciación de piezas adaptadas a la programación de la producción según el criterio más valorado en cada momento, podría aplicar las acciones correctivas asociadas al mantenimiento y favorecer este tipo de política.

A un nivel más operativo, se pueden asociar unos costes de posesión para cada pieza fabricada como costes de almacenaje. Que sea un producto en curso o semielaborado o bien final, se tiene a disposición del decisor si le conviene más tener unos costes de almacenaje, en caso de que la pieza sea producida con una fecha de terminación mínima, o unos coste de retraso en relación a las entregas a clientes si cumplir con las fechas acordadas se puede relajar.

8. Conclusiones

Este proyecto se ha dedicado al estudio de un problema de secuenciación bicriterio de máquinas idénticas en paralelo sin ponderar a un criterio en particular. Se deja al decisor la posibilidad de elegir la importancia que asigne a cada criterio y, por lo tanto, se han hallado las soluciones que se encuentran en el dominio de Pareto del problema.

Se ha buscado algunos métodos meta heurísticos para llevar a cabo este problema porque son buenos métodos para resolver gran variedad de problemas. Y se ha utilizado un Algoritmo Genético porque se trata de algoritmos que manejan poblaciones y son eficientes para obtener conjuntos de soluciones y dibujar la frontera de Pareto.

Aunque la estructura básica de un Algoritmo Genético esté bien definida en la literatura científica mundial y no varíe en su aplicación a la multitud de problemas que puede abarcar, se tiene que adaptar a cada problema. En este caso, los parámetros más significativos que se han introducidos son:

- Una codificación concreta determinada por las piezas y las máquinas.
- Una población inicial heterogénea que permita empezar el algoritmo con una cierta diversidad en la población.
- Un tipo de selección en los progenitores que van a ser cruzados que incluye la noción de elitismo. Así, se conserva un cierto porcentaje de buenas soluciones a cada generación.
- Tres tipos de cruce, que son el cruce 1punto, el cruce OX y el cruce PMX.
- Dos tipos de mutación, que son la mutación Blanda y la mutación Intercambio.

Además, para llevar a cabo el problema, se ha estudiado diferentes tipos de fitness como medida de eficiencia de una solución: un fitness continuo, un fitness diagonal y un fitness perpendicular. Se ha guardado el fitness perpendicular que proporcionaba conjuntos de soluciones más diversas en cada población generada.

La aplicación de un Algoritmo Genético ha permitido explorar el espacio de búsqueda de forma eficiente. Esta afirmación se basa en la diversidad y la competitividad de los individuos. Dicha diversidad empieza, en el algoritmo propuesto, por la generación heterogénea de la población inicial y continua en la creación de nuevos individuos, en la cual el algoritmo selecciona sólo un porcentaje de individuos más competitivos, y da a los

individuos con menor fitness la posibilidad de ser seleccionados para cruzarse con otros. En este momento tienen la oportunidad de pasar alguna de sus características positivas a uno de sus descendientes.

Se ha llevado a cabo un estudio de los diferentes parámetros que conllevan soluciones de mejor calidad. Estos parámetros son el tamaño de la población inicial, el número de generaciones, las probabilidades de cruce y mutación, así como el tipo de cruce y el tipo de mutación a aplicar.

- Los valores del tamaño de la población inicial y del número de generaciones están condicionados por el compromiso existente entre el coste computacional y la aptitud de las soluciones. Dependiendo de las circunstancias en que se encuentre el usuario, éste optara por unos valores u otros. se propone que el tamaño de la población debe ser igual a tres veces el número de piezas y el número de generaciones igual a cuatro veces el número de piezas.

La variación de los operadores que participan en el desarrollo del Algoritmo Genético propuesto influye en la obtención de unas soluciones en el dominio de Pareto. Por lo tanto se han hecho unas pruebas combinando todos los parámetros con el fin de hallar la combinación más adecuada al problema tratado.

- El mejor conjunto de soluciones se alcanza cuando la probabilidad de cruce tiene un valor de 80% y cuando la probabilidad de mutación adopta un valor del 20%, y, por otro lado, cuando se combinan el tipo de cruce de 1punto con la mutación Intercambio.

No obstante, se deja al usuario-decisor la elección de los valores de los operadores según las restricciones que imponga él.

Finalmente, las futuras líneas de trabajo recomendadas siguen dos caminos:

- Intentar construir una solución de Pareto con otras metas heurísticas, con un recocido simulado, o una búsqueda tabú...

- Aplicar un Algoritmo Memetico (genetic local search) **[16]** donde el operador de mutación sea sustituido por un método de búsqueda local aplicado a cada descendiente generado. Así, cada descendiente nacido de un cruce, se acerca de la frontera de Pareto con el operador de búsqueda local y proporciona soluciones aún mejores.

Agradecimientos

El autor quiere agradecer a su tutor Manel Mateo, del Departamento de Organización de Empresas, que le dio gusto por la dirección de operaciones a través de su curso y le dio las ganas de seguir en este campo para el Proyecto Final de Carrera. Le agradece por haber sido paciente, abierto y también por haber sido estricto y haberle enseñado a ser perfeccionista en la redacción de un informe.

Quiere agradecer a sus colegas y amigos Borja Besalduch, Samer Hawat, Alan Taxonera y Rodolfo Velez, titulados de la ETSEIB, y a Francesca Mainella su "hermana de corazón" por haber estado a su lado desde el principio y haberle acompañado durante todos estos meses, cuando la motivación así como la inspiración bajaba.

Quiere agradecer a la ETSEIB, por haberle hecho conocer el campo de la Ingeniería Industrial y de los negocios en Cataluña y España, por sus múltiples actividades y acciones a nivel internacional en las cuales ha activamente participado, y por su proximidad con las empresas. Le agradece también por haberle permitido de ser más que un mero estudiante extranjero entre sus paredes permitiéndole conseguir una titulación superior española de que está orgulloso.

Y por fin, quiere agradecer a Barcelona, capital catalana, donde ha conocido personas y vivido experiencias que en ningún otro lugar hubiera sido posible. Se siente unido a Barcelona para siempre. Y espera volver a trabajar en ella en un futuro próximo.

Bibliografía

Referencias bibliográficas

[1] GRAHAM R., LENSTRA JK, LAWLER E., KAN AH., Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* pp. 287–326, 1979.

[2] COMPANYS, R.; COROMINAS, A. *Organización de la Producción II, dirección de operaciones. Volumen 4.* 1985.

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

[4] KIRKPATRICK S, GELATT C.D.Jr & VECCHI M.P., *Optimisation by simulated annealing*", Science VOL.220 p. 671, 1983.

[5] GLOVER F., *Future paths for integer programming and links to artificial intelligence*, Computers and Operations Research, VOL. 13, pp.533-549, 1986.

[6] HOLLAND J. *Adaptation in natural and artificial systems. (Cambridge, Mass, MIT press),* 1975.

[7] GOLDBERG D.E., *Genetic algorithms in search, Optimisation and Machine Learning*. Addison-Wesley, Mass., 1989.

[8] MARTIN W.N, LIENIG J. and COHOON J.P, *Island (migration) models: evolutionary algorithms based on punctuated equilibria,* In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, Handbook of Evolutionary Computation, p C6.3:1-C6.3:16, Institute of Physics Publishing, Bristol, UK, 2000.

[9] ALANDER JT., *Dynamic Population Size Research*, 'Computer Systems and Software Engineering' Proceedings. P 65-70, 1992.

[10] SCHAFFER D., *Vector Evaluated Genetic Algorithm,* Proceedings of the 1st International Conference on Genetic Algorithms. P 93-100, 1985.

[11] HORN J., *Niched Pareto Genetic Algorithm*, Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence. Proceedings of the First IEEE Conference on, VOL.1, p. 82-87, 1994.

[12] LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE, *Optimisation multi critères, approche par méta heuristiques*, Trabajo de Investigación, 2004.

[13] FERNÁNDEZ-BAÑOS, I. *Programación de la secuenciación de fabricación en una máquina con tiempos de preparación variables, mediante la aplicación de Algoritmos Genéticos.* Proyecto de fin de carrera de la ETSEIB Barcelona, 2003.

[14] POTTS CN. y VAN WASSENHOVE LN., *A Branch and Bound Algorithm for the Total Weighted Tardiness Problem,* Operations research Vol.33 No.2, Operations Research Society of America, 1982.

[15] CAP GEMINI. Study, Global Commerce Initiative publication, *Future Supply chain 2016 report, www.futuresupplychain.com*, 2008.

[16] LUST T.; TEGHEM J. *Memots: a memetic algorithm integrating tabu search for combinatorial multiobjective optimization*. , RAIRO Operations Research publications on line, p.3, 2008

Bibliografía complementaria

[1] El País, *"De la idea al producto"*, Suplemento elaborado por Siemens, 5 de Mayo 2008

[2] HUNTER, J. C. *La Paradoja*. Ediciones Urano, S.A Barcelona, 1996.

[3] CUATRECASAS, L. *Organización de la producción y dirección de operaciones*, Editorial Ramon Areces, 1999.

[4] DOMINGUEZ MACHUCA, J.A.. *Dirección de operaciones, aspectos estratégicos en la producción y los servicios*, Mac Graw Hill, 1995.

[5] DIAZ A. *Producción, gestión y control*, Editorial Ariel, Barcelona, 1993.

[6] MEREDEITH J.R., SHAFER S.M., *Operation Management for MBAs*, John Wiley & Sons, New York, 1999.

[7] GOLDRATT, E.M, COX J. *La Meta,* Editorial Taular, Madrid, 1986.

[8] ROVIRA CELMA A., TRIAS DE BES F. La buena Suerte, Claves de la Prosperidad. Empresa activa., Barcelona,.2004