**UNIVERSIDAD DE LOS ANDES**

**Facultad de Ingeniería**
**Departamento de Ingeniería Industrial**
**Maestría en el área de Sistemas de Producción.**

# COMBINACIÓN DE MÉTODOS HEURÍSTICOS CON REDES DE PETRI PARA LA PROGRAMACIÓN DE SISTEMAS DE MANUFACTURA FLEXIBLE.

**Ing. Ana Lucía Castro López**

**Asesor: Dr. Gonzalo Mejía Delgadillo**

**Bogotá D.C. Enero de 2006**

**Tabla de Contenido**

# I. Introducción

En el desarrollo de herramientas que ayuden a programar las operaciones de los procesos de producción, asignando recursos a través del tiempo con el objeto de optimizar uno o más objetivos, se han propuesto numerosas técnicas de análisis, entre ellas la combinación de las Redes de Petri (RdP) con técnicas heurísticas, que gracias a su poder de adaptación a casi cualquier tipo de proceso, se han constituido en un instrumento teórico que puede ajustarse a los procesos que se llevan a cabo en sistemas reales de producción.

El propósito de este trabajo es el de desarrollar una aplicación de las Redes de Petri en los sistemas de manufactura flexible – **SMF** – en la industria metalmecánica, específicamente en la producción de frenos para la industria automotriz, utilizando el enfoque de las Redes de Petri (RP) para su modelaje y la aplicación de alguna herramienta heurística para la obtención de programas factibles de los **SMF**. Se quieren representar las situaciones que se manifiestan en el proceso productivo de los frenos, que debido a su complejidad deben tratarse específicamente, pero que gracias a las propiedades de las RP es posible obtener representaciones gráficas y analíticas que puedan ser procesadas y estudiadas por métodos heurísticos para obtener programas de producción que se ajusten a las necesidades y expectativas de las empresas.

El proyecto tiene importancia debido a que la problemática que aborda surge de manera natural en ambientes de producción de bienes cuyos diseños y/o estructuras se originan en combinaciones de las diferentes necesidades o gustos de los clientes, estando estas opciones restringidas por la oferta del productor, que debe ir evolucionando a medida que los clientes van exigiendo que nuevos productos que cumplan con sus expectativas.

De igual manera se debe tener en cuenta que la obtención de soluciones óptimas para este tipo de problemas – de carácter NP – puede ser muy costosa en recursos invertidos (tiempo, software, hardware, especialistas), o inclusive imposibles de resolver con los recursos disponibles; lo anterior valora fuertemente la obtención de una *buena solución cercana a la óptima,* si ésta es obtenida de forma económica en función de los recursos invertidos.

Con esta premisa se hace necesaria la utilización de alguna técnica heurística que permita hallar la programación factible de los trabajos ,que cumpla con el objetivo de minimizar el tiempo de culminación de todos los trabajos a programar, y que permita tener en cuenta los tiempos de setups, que son una porción importante en la programación de trabajos, cuyos tiempos de preparación dependen de la secuencia con la que son ejecutados.

También adquiere importancia hallar soluciones aproximadas mediante herramientas heurísticas debido a que su enfoque de solución no está asociado al uso de reglas de despacho, que son frecuentemente usadas en la industria, y que consideran principalmente las condiciones locales y actuales de la máquina, y no el entorno comprendido por futuros trabajos y las otras máquinas.

# II. Objetivos

- General

  El objetivo del proyecto es estimar la aplicabilidad en una instancia real, del enfoque combinado de RP con heurísticos como los Algoritmos Genéticos, para minimizar el tiempo en el que se completa el último trabajo que está en el sistema ($C_{max}$) , teniendo en cuenta los tiempos de setup dependientes de la secuencia.

- Específicos

  - Encontrar un caso de estudio que pueda ser modelado por las RP y que sea representativo de una planta real en funcionamiento, para que haya un acercamiento entre las herramientas teóricas de análisis y la realidad de los procesos productivos.
  - Implementar un heurístico que permita la obtención de programaciones del **SMF** factibles que sean cercanas a los programas empleados realmente en el sistema de producción a estudiar y que permita mejorar la utilización de los recursos, disminuir costos por fallas en producción y que contribuyan al mejoramiento de la calidad total.
  - Lograr un algoritmo que modele procesos productivos con RdP, que sea flexible en su estructura para ser ajustado a cualquier tipo de configuración que se plantee, capaz de producir buenas soluciones, en tiempo razonable.
  - Lograr una buena sintonización de los parámetros que intervienen en los AG, de tal manera que el algoritmo pueda responder adecuadamente a diferentes tipos de problemas que se le planteen.

# 1. Contexto General

Los Sistemas de Manufactura Flexible –SMF- son sistemas consistentes en varias estaciones de trabajo conectadas por un sistema de alimentación de materiales que es capaz de permitir el flujo de trabajos pasando por diversas rutas a través del sistema[7], permitiendo que varios tipos de productos puedan ser simultáneamente procesados debido a que se tienen las herramientas y la información de procesamiento necesarias para trabajar en múltiples tipos de productos, lo que les brinda, a estos sistemas, una gran flexibilidad y adaptabilidad, que es aprovechada para satisfacer las necesidades de los clientes en cuanto a la variedad de productos que pueden ser elaborados, en un tiempo y costo razonables[2].

En los SMF es necesario introducir una aplicación de prueba y control, de tal manera que se pueda reaccionar a los diferentes eventos que pueden ocurrir y de esta manera facilitar su monitoreo y control. Pero, esta no es una tarea fácil, este tipo de sistemas representan muchos problemas de planeación, programación, monitoreo y verificación, para los cuales se han propuesto varios métodos de solución, pero que debido a su naturaleza NP-hard es difícil encontrar una respuesta óptima.

Los métodos propuestos para solucionar el problema pueden ser clasificados en tres categorías[18]:

1) Los métodos de optimización, que requieren una cantidad considerable de iteraciones y el tiempo computacional crece exponencialmente con el tamaño del problema, lo que para los objetivos de esta tesis, los hace no aplicables debido a la complejidad de las aplicaciones reales.

2) Los métodos heurísticos, que se basan en reglas de despacho que generalmente dan programas satisfactorios, sin embargo, cada sistema requiere configuraciones diferentes, lo que hace que encontrar y programar procedimientos de respuesta para cada problema, sea una tarea que consuma gran cantidad de tiempo. Entre los métodos heurísticos se encuentran los AG que son una técnica de solución común para la solución de problemas que surgen en las operaciones de manufactura. Fueron usados para: determinar la estructura del modelo de sistemas lineales y no-lineales que mejor representen sistemas de entrada y salida de datos [5]; para problemas en el ambiente de planeación de la producción [3] y [4]; se propuso un algoritmo genético modificado adaptaciones del área de búsqueda (mGSA); para optimización multiobjetivo por AG, llamados multi-objective genetic algorithms (MOGAs) [21] y por Enfriamiento Simulado (ES) [22]; aplicaciones de ES para encontrar niveles de óptimos de generación de energía, sujeto a las restricciones de niveles de emisiones, costo del combustible y seguridad en las líneas de transmisión [20].

3) Las técnicas híbridas, tratan de combinar las ventajas de los otros métodos delimitando un espacio de estados con la ayuda de las heurísticas. En esta categoría se encuentran técnicas en las que la SMF es modelada como una RP y se quiere encontrar la secuencia de disparos que minimice el costo de ir desde el marcaje inicial al marcaje final, por medio de un proceso de búsqueda dirigido por una heurística. Es así como se han desarrollado algoritmos de programación que minimizan el *makespan*, el tiempo de finalización del último trabajo, orientados por una heurística basada en las soluciones de ecuaciones de estado, para predecir el costo total desde estado inicial a través del estado actual hasta el marcaje final [18]; minimización de la tardanza ponderada, manipulando tiempos de alistamiento dependientes de la secuencia, estaciones con máquinas en paralelo no necesariamente idénticas y capacidad de almacenamiento temporal limitado en las estaciones por medio de AG [17]; integración de

un sistema de control jerárquico con un esquema de programación basado en las RP para Células de Manufactura Flexible (CMF) [15]; aproximaciones para el modelamiento y programación de SMF por medio de RP e Inteligencia Artificial, utilizando una aproximación del algoritmo stage-search modificado[8] y [9]; aplicaciones de los métodos híbridos de búsqueda: en la programación de pruebas para la fabricación de semiconductores [2], en el diseño óptimo de un generador superconductor [19]; uso de tipos particulares de RP: Controllable-Output nets (Conets) [10], Colored Petri Nets [11]. También se han integrado métodos para la solución de problemas de programación de la producción basados en Constraint Logic Programming (CLP) y AG para ser aplicados en problemas con una sola línea, múltiples productos y tiempos dependientes de la secuencia [25].

De acuerdo con la bibliografía revisada, entre los métodos híbridos se ha hecho cada vez más común el uso de las RdP para resolver problemas de programación, en los que han sido aplicadas para modelar, analizar, simular, planear, programar y controlar los SMF aprovechando sus capacidades para:

- Modelar actividades concurrentes y asincrónicas [2].
- Representar relaciones de precedencia [2].
- Modelar situaciones especiales de los sistemas de manufactura como conflictos y límites de capacidad en los sitios de almacenamiento, manejo de excesos de flujos, entre otros [3].
- Asociar el tiempo con lugares y transiciones, lo que permite describir el sistema cuyo funcionamiento es dependiente del tiempo. [2]

En cuanto a la programación, es difícil encontrar un método efectivo que genere soluciones óptimas a problemas clasificados como NP-Hard. Para dar soluciones

aproximadas se han desarrollado muchos métodos cuyo uso en general ha sido limitado por el número de variables involucradas y la complejidad de las interrelaciones que existen entre ellas. Regularmente sus efectos son desconocidos o demasiado complejos para ser modelados con exactitud. Adicionalmente, hay muchas variables cualitativas que necesitan ser consideradas lo que normalmente representa limitaciones a las técnicas de solución existentes e impiden identificar soluciones óptimas o cercanas al óptimo.[3]

Es por lo anterior, que los métodos heurísticos se revelan como una herramienta que puede tener una amplia aplicación para este tipo de problemas en los que han mostrado ser muy eficientes y confiables pero cuyo principal problema es la dificultad de escapar de la optimalidad local, lo cual ha propiciado que el enfoque de la inteligencia artificial haya revivido como solución de problemas que requieren de la búsqueda heurística. Varias aproximaciones han surgido para el manejo de problemas calificados NP-Hard como son: algoritmos genéticos, redes neuronales, enfriamiento o recocido simulado, búsqueda tabú, análisis de objetivos y búsqueda dispersa, entre otros.

La Búsqueda Tabú (BT) es un procedimiento heurístico utilizado para resolver problemas de optimización de gran escala, dicho procedimiento heurístico está diseñado para guiar a otros métodos (o a procesos componentes) para escapar de la optimalidad local. La filosofía de la BT está basada en manejar y explotar una colección de principios para resolver problemas de manera inteligente y tiene como elemento principal el uso de memoria flexible [16]

El enfriamiento simulado, es un proceso computacional que refleja los pasos establecidos en el proceso físico de tratamiento térmico de materiales. En el recocido, por ejemplo, un metal es llevado a elevados niveles energéticos, hasta que alcanza su punto de fusión. Luego, gradualmente es enfriado hasta alcanzar un estado sólido, de mínima energía, previamente definido. Por su naturaleza, este algoritmo la función objetivo es el nivel energético que permite explorar una

buena parte del espacio de estados, de tal forma que la solución final puede resultar insensible al estado inicial aprovechando el hecho de que las sustancias físicas usualmente se mueven desde configuraciones de alta energía a las de menor energía, así que el descenso al mínimo, ocurre en forma natural, en consecuencia, la probabilidad de quedar atrapado en un minimo local, es baja.

Los *algoritmos genéticos* son métodos sistemáticos para la resolución de problemas de búsqueda y optimización que aplican a estos los mismos métodos de la evolución biológica: selección basada en la población, reproducción sexual y mutación. Tratan de resolver problemas en los que tras parametrizarlos en una serie de variables ($x_i$,...,$x_n$) que se codifican en un cromosoma, el objetivo es **hallar ($x_i$,...,$x_n$) tales que $F(x_i,...,x_n)$ sea máximo**, aplicando al cromosoma los operadores utilizados por el algoritmo.

Hay que tener en cuenta que un algoritmo genético es independiente del problema, lo cual lo hace un algoritmo *robusto*, por ser útil para cualquier problema, pero a la vez *débil*, pues no está especializado en ninguno. Las soluciones codificadas en un cromosoma *compiten* para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El *ambiente,* constituido por las otras soluciones, ejercerá una presión selectiva sobre la población, de forma que sólo los mejor adaptados (aquellos que resuelvan mejor el problema) sobrevivan o leguen su material genético a las siguientes generaciones, igual que en la evolución de las especies.

- Para su aplicación hay que tener en cuenta que:
    - Es una herramienta en la que su espacio de búsqueda (i.e., sus posibles soluciones) debe estar delimitado dentro de un cierto rango.
    - Debe poderse definir una función objetivo que nos indique qué tan buena o mala es una cierta respuesta.
    - Sea capaz de "castigar" a las malas soluciones, y de "premiar" a las buenas, de forma que sean estas últimas las que se propaguen con

mayor rapidez; y finalmente, las soluciones deben codificarse de una forma que resulte relativamente fácil de implementar.

# 2. Red de Petri - RdP

## 2.1. Estructura de una RdP

Una RdP es una forma particular de un grafo dirigido bipartito[33], consistente en:

- Nodos, la RdP tiene dos tipos de nodos conocidos como plazas (p) y transiciones(t), que gráficamente se representan con círculos para las plazas y barras para las transiciones,

  ◯ Plazas $P = \{p_1, p_2, ..., p_m\}$

  ▯ Transiciones $T = \{t_1, t_2, ..., t_n\}$

  $$P \cap T = \varnothing \text{ y } P \cup T \neq \varnothing$$

  Las plazas representan condiciones y las transiciones representan eventos. Las transiciones tienen un número de plazas entrantes y salientes que representan las pre-condiciones y las post-condiciones para la ocurrencia del evento respectivo, ya que solamente se dispara una transición una vez haya los tokens requeridos en todas las plazas de entrada.

- Los arcos entre ellos van dirigidos de plazas a transiciones y/o de transiciones a plazas y cada arco puede tener un peso asociado, un arco con un peso asociado m se interpreta como un grupo de m arcos paralelos, esta propiedad de los arcos en la RdP puede ser asimilada en la programación de SMF como tamaños de lotes que van transitando por la red.

  $F \subseteq (P \times T) \cup (T \times P)$ es el conjunto de arcos

$W: F \rightarrow \{1,2,3,...\}$ es la función de pesos

- El Marcaje M. El marcaje representa el estado de la red en cualquier momento, asignando números no negativos a cada plaza a medida que se va haciendo el recorrido por la red. Cada número asignado demuestra la presencia de tokens en una plaza determinada. Un token en una plaza representa el estado lógico (falso o verdadero) de la condición asociada a esa plaza.

$M_0: P \rightarrow \{0,1,2,3,...\}$ es el Marcaje inicial

La estructura de una RdP entonces es $N = (P, T, F, W, M_0)$

## 2.2. Reglas de disparo

Para poder recorrer la red es necesario disparar cada una de las transiciones a través de las siguientes reglas de disparo[33]:

- Una transición está disponible si el número de tokens en sus plazas de entrada es mayor o igual a $w_i(p_i,t)$, los pesos de los arcos que conectan estas plazas con la transición en consideración.

- Una transición disponible puede o no puede ser disparada dependiendo de que ocurra o no el evento que representa esa transición.

- El disparo de una transición disponible t remueve w(p,t) tokens de cada plaza de entrada p a t y suma w(t,p) tokens a cada plaza de salida p de t.

Cuando hay una transición sin ninguna plaza de entrada, esta es llamada transición fuente y está disponible sin ninguna condición, mientras que una transición sin ninguna plaza de salida es llamada transición sifón, ya que consume todos los tokens de las plazas de entrada y no produce ninguno.

## 2.3. Métodos de Análisis[33]

Los métodos de análisis de una RdP pueden ser clasificados en tres tipos:

### 2.3.1. Método de Árbol

Es un método exhaustivo que consiste en encontrar todos los marcajes posibles disparando cada una de las transiciones disponibles partiendo desde $M_0$. Lo que resulta en una representación de todos los marcajes en forma de árbol. Es un buen método ya que es aplicable a todo tipo de redes pero que se limita a redes pequeñas debido a la explosión de estados.

### 2.3.2. Aproximación por matrices – ecuaciones

- Matriz de incidencia.

  La matriz de incidencia $A = [a_{ij}]$ de una RdP es una matriz de nxm, donde n es el número de transiciones y m el número de plazas en la red, y tiene la información de cuáles plazas entran y salen de cada una de las transiciones y del peso de los arcos que hay entre las plazas y las transiciones, es decir, la cantidad de tokens que entran y salen de una transición, es por esto que puede descomponerse en dos submatrices $a_{ij}^{+} = w(i, j)$ es el peso de los arcos desde la transición i hasta su plaza de salida j y $a_{ij}^{-} = w(j, i)$ es el peso de los arcos desde la plaza de entrada j a la transición i

  $$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

  Una transición se encuentra disponible en el marcaje M si $a_{ij}^{-} \leq \mathrm{M(j)}$, j=1,2,...,m.

- Ecuaciones de estado.

Estas ecuaciones sirven para ver la evolución de los tokens y del tiempo en la red. Se tiene entonces:

- Un vector columna mx1 $M_k$, la posición j-esima en $M_k$ denota el número de tokens en una plaza j inmediatamente después de el k-ésimo disparo en alguna secuencia de disparos.
- Un vector $u_k$ de control, es un vector columna nx1, con n-1 0's y una entrada i diferente de cero, indicando que la transición i fue disparada en el k-ésimo disparo. Mientras que la i-ésima fila de la matriz de incidencia A denota el cambio en el marcaje como consecuencia de que la transición i fue disparada.

$$M_k = M_{k-1} + A^T u_k \qquad k = 1,2,....$$

- Condición de alcance.

Si un marcaje $M_d$ es alcanzable desde $M_0$ a través de una secuencia de disparos $\{u_1, u_2,..., u_d\}$

$$M_d = M_0 + A^T \sum_{k=1}^{d} u_k \quad o \quad A^T x = \Delta M$$

donde $\Delta M = M_d - M_0$ y $x = \sum_{k-1}^{d} u_k$ , $x$ es un vector columna de nx1 de enteros no negativos llamado el *vector contador de disparos*. La i-ésima posición de $x$ muestra la cantidad de veces que la transición i debe ser disparada para transformar $M_0$ en $M_d$. Este sistema tiene solución x si $\Delta M$ es ortogonal a toda solución y del sistema homogéneo

$$Ay = 0$$

La matriz A tienen entonces la siguiente partición:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

donde $A_{12}$ es una matriz cuadrada no singular de *rxr*, donde *r* es el rango de la matriz A.

### 2.3.3. Técnicas de reducción o descomposición

Para facilitar el análisis de grandes sistemas, se utilizan técnicas para reducir sistemas complejos en unos más simples que mantengan las propiedades del sistema que se está analizando, es decir, técnicas que preserven las propiedades de supervivencia, seguridad y límites.

Estas son algunas de las transformaciones más simples que pueden ser usadas para el análisis de las RdP:

- Fusión de plazas en serie (FPS)
- Fusión de transiciones en serie (FTS)
- Fusión de plazas en paralelo (FPP)
- Fusión de transiciones en paralelo (FTP)
- Eliminación de plazas con auto-ciclos (EPA)
- Eliminación de transiciones con auto-ciclos (ETA)

## 2.4. Propiedades de las RdP[33]

Hay dos tipos de propiedades que pueden ser observadas en un modelo con RdP, aquellas que son dependientes del marcaje inicial, llamadas propiedades de comportamiento y aquellas que son independientes del marcaje inicial, llamadas propiedades estructurales.

### 2.4.1. De comportamiento

- **Alcanzabilidad**

Con cada transición que se activa de acuerdo con las reglas de disparo se va formando una secuencia de disparos que va conduciendo a una secuencia de marcajes. Cuando una secuencia de disparos $\sigma$ conduce a la red desde el $M_0$ hasta el marcaje final $M_n$, entonces se dice que $M_n$ es alcanzable desde $M_0$, $\sigma$ transforma a $M_0$ en $M_n$, $M_0[\sigma > M_n$, siendo $\sigma = t_1, t_2, ..., t_n$. El conjunto de todos los marcajes a los que se puede llegar desde $M_0$ en una red $(N, M_0)$ se denota $R(N,M_0)$ o $R(M_0)$. El conjunto de todas las posibles secuencias de disparo desde $M_0$ en una red $(N, M_0)$ se denota $L(N,M_0)$ o $L(M_0)$. El problema de alcance en una RdP es encontrar si $M_n \in R(M_0)$ para un marcaje dado.

- **Acotamiento**

Una RdP $(N,M_0)$ es limitada si el número de tokens en cada uno de las plazas no excede un número finito k para cualquier marcaje alcanzable desde $M_0$ $M(p) \leq k$ para cada plaza p y cada marcaje $M \in R(M_0)$.

- **Liveness**

Se refiere a la ausencia total de puntos muertos en los que la red se pueda estancar, es decir, siempre se llega a un marcaje de destino, no importa cual, desde el $M_0$ con una secuencia de disparos de todas las transiciones de la red.

### 2.4.2. Estructurales

Son aquellas que dependen de la estructura topológica de las RdP, son independientes del $M_0$, en el sentido de que estas propiedades se

mantienen para cualquier marcaje inicial pero se refieren a la existencia de cierta secuencia de disparos. Entonces, estas propiedades pueden expresarse en términos de la matriz de incidencia A y en sus ecuaciones. Acá solamente se dan las definiciones generales de cada una de las propiedades, si se desea ver mayor información remitirse a [33].

- **Liveness estructural**

Una RdP N se dice estructuralmente viva si existe un marcaje inicial para N.

- **Controlabilidad**

Una RdP N se dice completamente controlable si cualquier marcaje es alcanzable desde cualquier otro marcaje.

- **Acotamiento estructural**

Una RdP N se dice completamente limitada si es limitada para cualquier marcaje inicial $M_0$ finito.

- **Conservación**

Una RdP N se dice que es (parcialmente) conservativa si existe un entero positivo y(p) para cualquier plaza p tal que la suma de los pesos de los tokens, $M^T y = M_0^T y = a$ constante, para (algún) todo $M \in R(M_0)$ y para cualquier $M_0$.

- **Repetitividad**

Una RdP N se dice ser (parcialmente) repetitiva si existe un $M_0$ y una secuencia de disparos $\sigma$ desde $M_0$ tal que (alguna) toda transición ocurre con frecuencia infinita en $\sigma$.

- **Consistencia**

Una RdP N se dice ser (parcialmente) consistente si existe un $M_0$ y una secuencia de disparos $\sigma$ desde $M_0$ que vuelva a $M_0$ tal que (alguna) toda transición ocurra al menos una vez en $\sigma$.

## 2.5. Tiempo en las RdP

Las RdP temporizadas han sido ampliamente usadas para el modelamiento y evaluación del desempeño de los Sistemas Flexibles de Manufactura [34], ya que admiten el tiempo de proceso de cada operación como parte fundamental de su estructura. El factor tiempo puede ser asociado en una RdP a las transiciones o a las plazas dependiendo de la configuración de la red. Si se definieron las transiciones como las operaciones desarrolladas en una máquina, entonces las transiciones se toman el tiempo de proceso de dicha operación para ser disparadas, en este caso todas las plazas son definidas como buffers [31]. Si por el contrario, las transiciones se refieren a eventos en la red y las plazas se asocian a las operaciones de los trabajos a desarrollar, entonces las transiciones se disparan, con duración cero, solamente cuando todos los tokens que las habilitan se encuentran disponibles, es decir, durante el intervalo de tiempo que dura la operación en ejecución asociado a las plazas, los tokens no se encuentran disponibles [2].

Un sistema de marcaje es utilizado para indicar la distribución de los tokens en la red, dado que al momento de disparar una transición se retiran los tokens disponibles de cada plaza de entrada y se depositan en cada plaza de salida. Los tokens depositados en $p_i$ permanecen en estado no disponible en el intervalo de tiempo $(t, t+d_i)$, donde $t$ es el tiempo en el que los tokens llegaron a la plaza $i$ y $d_i$ es el tiempo de proceso asociado a dicha plaza. El tiempo de proceso remanente muestra cuánto hace falta para que una operación sea completada. Las

ecuaciones de estado relacionadas con el sistema de marcaje y el tiempo remanente se muestran a continuación:

*Marcaje* $\quad \overline{M^{p}}(k+1) = \overline{M^{p}}(k) + \overline{A} \bullet \overline{u}(k)$

*Tiempo proceso remanente* $\quad \overline{M^{r}}(k+1) = \overline{M^{r}}(k) - \tau(k) \bullet \overline{P} \bullet \overline{M^{p}}(k) + \overline{W} \bullet \overline{A}^{+} \bullet \overline{u}(k)$

El estado general del sistema puede ser expresado de la siguiente manera:

$$\overline{X}(k+1) = \begin{bmatrix} \overline{M}^{p}(k+1) \\ \overline{M}^{r}(k+1) \end{bmatrix} = \begin{bmatrix} \overline{I} & \overline{0} \\ -\tau(k) \bullet \overline{P} & \overline{I} \end{bmatrix} \bullet \begin{bmatrix} \overline{M}^{p}(k) \\ \overline{M}^{r}(k) \end{bmatrix} + \begin{bmatrix} \overline{A} \\ \overline{W} \bullet \overline{A^{+}} \end{bmatrix} \bullet \overline{u}(k)$$

donde $\overline{X}(k+1)$ es el vector de variables de estado *(2nx1)* en el estado *k+1*, es decir, después de *k+1* transiciones disparadas, que consiste de un vector (nx1) de marcaje $\overline{M}^{p}(k+1)$ y un vector *(nx1)* de tiempo remanente $\overline{M}^{r}(k+1)$ donde *n* es el número de plazas en la red y m el número de transiciones.

$\overline{I}$ : Matriz identidad de *nxn*

$\overline{0}$ : Matriz de ceros de *nxn*

$\tau(k)$ : Tiempo transcurrido entre dos disparos consecutivos

$\overline{P}$ : Matriz diagonal *(nxn)* que sirve para distinguir las plazas operación de las plazas recurso. $\overline{P}_{ij}$ = 1 *si i=j* $\wedge$ *i* es una plaza operación, $\overline{P}_{ij}$ = 0 en el resto

$\overline{u}(k)$ : *(mx1)* es un vector de control que determina cuál transición se dispara después de *k* disparos. Es un vector de ceros y unos, $\overline{u}_{j}(k)$ = 1 indica que la transición *j* es disparada, 0 de lo contrario.

$\overline{A}$ : *(nxm)* es la matriz de incidencia

$\overline{W}$ : Es una matriz diagonal de *(nxn),* donde $w_{ij}$ = tiempo de proceso asociado a la plaza *i* cuando *i = j*, 0 de lo contrario.

Cuando una acción $\bar{u}(k)$ es tomada, el tiempo de proceso remanente asociado con los tokens en una plaza operación son actualizados con dos términos dinámicos: 1. una reducción universal causada por el tiempo transcurrido entre dos estados consecutivos, que se representa con $-\tau(k) \bullet \overline{P} \bullet \overline{M}^{p}(k)$, esto es, cuando la *j-ésima* transición es disparada después de $\tau(k)$ unidades de tiempo, donde $\tau(k)$ es el máximo tiempo de proceso remanente de las plazas de entrada a la *j-ésima* transición, que debe ser sustraído del tiempo de proceso remanente y los tokens correspondientes son movidos desde las plazas de entrada a las plazas de salida. y 2. el tiempo de proceso adicionado por un nuevo disparo $\overline{W} \bullet \overline{A^{+}} \bullet \bar{u}(k)$.

# 3. Algoritmos Genéticos - AG

La primera vez que se habló de Algoritmos Genéticos – AG, fue en los estudios de autómata celular dirigidos por John Holland de la Universidad de Michigan. En los que se definió a los AG como una técnica evolutiva, que puede mejorar su comportamiento de acuerdo con la experiencia que adquiere a medida que va haciendo la misma tarea en diversas oportunidades, simulando el comportamiento evolutivo de la naturaleza, que hace que los individuos evolucionen de acuerdo con las condiciones que se les plantean en su medio ambiente, utilizando técnicas como herencia, mutación, selección natural y recombinación.

Su aplicación en las ciencias de la computación ha servido para encontrar buenas soluciones a problemas cotidianos de búsqueda y optimización en ingeniería, partiendo de una representación abstracta de posibles soluciones, llamada cromosoma, que va evolucionando desde una población inicial aleatoria a la que se le cuantifica su desempeño y se va modificando para crear una nueva población, repitiendo este procedimiento generación tras generación hasta llegar a una adecuada combinación de genes que permiten obtener un buen desempeño.

- Funcionamiento de un AG

Un cromosoma es un individuo que representa una solución al problema que está siendo resuelto, que no tiene una estructura específica, sino que se adapta a las condiciones de cada problema y al método de análisis que se está utilizando, por lo tanto hay una gran variedad de representaciones que pueden ser implementadas para la solución de un solo problema.

Inicialmente los individuos de la primera generación pueden ser concebidos aleatoriamente, para dar la posibilidad de evaluar diferentes espacios de

soluciones o se pueden generar a través de una semilla, algún tipo de orientación que le permita a los cromosomas colocarse en un espacio de posibles soluciones.

Durante cada generación cada individuo de la población es evaluado de acuerdo con su desempeño, esto es, de acuerdo con el resultado de la función objetivo que se obtiene de la solución planteada por cada cromosoma de la población. Luego se hace un ordenamiento de los cromosomas de acuerdo con sus resultados, de tal manera que se obtiene en la parte superior a los cromosomas con mejor respuesta, las mejores soluciones al problema respecto a las otras respuestas del resto de la población.

Lo que se busca luego es originar la siguiente generación con la misma cantidad de individuos que la población anterior, por medio de los procesos de selección y reproducción de determinados individuos a través de los operadores genéticos: cruce y mutación. Para la selección de cada individuo padre, los mecanismos más conocidos están enfocados en escoger a los individuos con el mejor desempeño sin eliminar del todo al resto de la población, para evitar que se generen poblaciones que converjan tempranamente a un óptimo local.

La operación cruce entre los cromosomas seleccionados está dirigida por una probabilidad de cruce, que determina si dos individuos seleccionados interactúan. Del cruce de dos cromosomas padres resultan dos cromosomas hijos que hacen parte de la siguiente generación. Este proceso se repite con diferentes padres hasta que esté la población completa de individuos que van a formar la siguiente generación.

El siguiente paso es la mutación de los cromosomas hijos dirigida por una probabilidad de mutación, que se lleva a cabo intercambiando los genes del cromosoma que se va a mutar.

Todo este proceso da como resultado una nueva generación de cromosomas que tiene información genética de la población preliminar; normalmente a medida que se conciben nuevas generaciones, el desempeño de la población va mejorando ya que se van seleccionando los mejores ejemplares de cada una de ellas para dar origen a la nueva raza de individuos.

Todo el proceso de evaluación, selección y reproducción se repite hasta generar que la condición de terminación es alcanzada:

- Se obtiene un número fijo de generaciones
- Se agota el tiempo presupuestado
- Se encuentra un individuo que satisface las expectativas
- Se llega al punto en el que no se producen mejores resultados, se han encontrado los individuos con el mejor desempeño.

▪ Observaciones

De acuerdo con la experiencia de los investigadores que han trabajado con los algoritmos genéticos se han generado opiniones sobre el desempeño de estos en las aplicaciones en las que han sido implementados.

- Los AG tienen la tendencia de converger hacia un óptimo local en lugar de alcanzar el óptimo global.
- La selección de los cromosomas padre requiere del ordenamiento de estos de acuerdo con su desempeño, lo que aumenta el tiempo computacional.
- Debido a que los cromosomas empiezan a converger tempranamente, el manejo dinámico de los datos es complicado, a medida que se llega a soluciones que no son válidas para ser tenidas en cuenta en las siguientes generaciones. Para incrementar la diversidad genética y prevenir de esta manera la convergencia temprana, ya sea

incrementando la probabilidad de mutación cuando la calidad de las soluciones se estanca o introduciendo ocasionalmente nuevos elementos generados aleatoriamente en los cromosomas.

- Los AG pueden encontrar rápidamente buenas soluciones en cualquier tipo de ambientes y en espacios de búsqueda muy amplios, debido a que pueden evaluar soluciones en diversas direcciones.
- Los AG tienen la habilidad de manipular muchos parámetros simultáneamente, es decir, pueden manejar problemas con múltiples objetivos.
- Los AG tienen la capacidad de hacer cambios aleatorios en sus soluciones, como consecuencia de esto todos los espacios de búsqueda son posibles para ellos, lo que les permite considerar múltiples soluciones, evaluar su desempeño y determinar si los cambios realizados conducen a mejoras o si por el contrario deben ser descartados.
- Los AG requieren de alguna medida de desempeño que le permita hacer la comparación entre las soluciones que va generando, es decir, no pueden ser empleados en problemas en los que la respuesta sea bien/mal, no tiene manera de converger a una solución. Este tipo de problemas son los llamados *aguja en un pajar*.
- Debido a que es una técnica que evoluciona de acuerdo con la fijación de sus parámetros, hay que ajustar las probabilidades de mutación y cruce, así como también el tamaño de la población, hasta encontrar una buena combinación para el problema que se está trabajando. Hay límites superiores e inferiors teóricos, que pueden servir de referencia, pero necesariamente hay que hacer un trabajo de ajuste hasta encontrar la combinación práctica más adecuada.
- Tiene mucha influencia en la eficiencia del algoritmo la manera en que se plantea la estructura de los cromosomas, esta debe ser robusta, capaz de tolerar los cambios aleatorios que se hacen en el cromosoma,

ya que una mala definición en su organización puede conllevar a comportamientos erráticos del algoritmo .

- Variantes

Debido a la amplia capacidad de adaptación de los AG a cualquier aplicación, se han desarrollado diferentes variantes del algoritmo original, que incluyen:

- En el algoritmo simple original cada gen de cada cromosoma es un número binario, haciendo de los cromosomas una cadena de bits. Se plantean entonces cromosomas de números enteros, de números decimales, alfanuméricos, listas de números, en fin, el diseño del cromosoma se adapta a cualquier estructura de datos y está limitado solamente por las condiciones del problema que se quiere resolver.
- El algoritmo original plantea el cruce y la mutación a nivel de genes, sin embargo, para los diferentes tipos de datos con los que se construyen los cromosomas, variaciones específicas de los operadores puedenser diseñadas.
- Una nueva variante, del proceso original, en la construcción de una nueva población es el elitismo, que permite el paso a la siguiente generación de los mejores individuos sin ser alterados. Pero como es sujeto a mejoras en los AG se han implementado variantes también de esta técnica.
- Algunos AG introducen dependencia del tiempo o ruido en la función de desempeño para sacar al algoritmo de óptimos locales.

- Aplicaciones

Los problemas en los que generalmente los AG han mostrado un buen desempeño, es decir, en los que los AG se constituyen en una herramienta apta para encontrar buenas soluciones, incluyen problemas de programación, problemas de logística, problemas de ingeniería en general

que no puedan ser resueltos de manera analítica, ya que los métodos analíticos tradicionales consumen mucho menos tiempo y potencia computacional que los AG y a diferencia de los AG ofrecen una única solución exacta comprobable matemáticamente.

Como regla general los AG pueden ser una herramienta útil en problemas cuyos dominios son amplios, en los cuales hay que optimizar múltiples objetivos, y que tienen funciones objetivo complejas, es decir, en ambientes discontinuos, ruidosos, dinámicos, lineales o no lineales.

# 4. Formulación del problema

Dadas las ecuaciones para el modelamiento de los FMS con las RdP y con la aplicación de los AG, es posible plantear un problema de programación de trabajos cuyo objetivo sea la minimización del Makespan ($C_{max}$), es decir, del tiempo de terminación del último trabajo que queda en el sistema, lo que implica encontrar una secuencia de disparos que maximice la utilización de los recursos disponibles.

El problema quedaría así:

Función Objetivo:

$$MIN \qquad C_{\max} = \sum_{i=0}^{m} \tau(k) \quad \text{A través de las generaciones}$$

Sujeto a:

Dinámica del sistema

$$\overline{X}(k+1) = \begin{bmatrix} \overline{M}^{p}(k+1) \\ \overline{M}^{r}(k+1) \end{bmatrix} = \begin{bmatrix} \overline{I} & \overline{0} \\ -\tau(k) \bullet \overline{P} & \overline{I} \end{bmatrix} \bullet \begin{bmatrix} \overline{M}^{p}(k) \\ \overline{M}^{r}(k) \end{bmatrix} + \begin{bmatrix} \overline{A} \\ \overline{W} \bullet \overline{A^{+}} \end{bmatrix} \bullet \overline{u}(k)$$

Disparo de transiciones disponibles

$$\overline{u}(k) \geq 0$$

Tokens disponibles

$$\overline{M}^{p}(k) \geq \overline{A}^{-} \bullet \overline{u}(k)$$

Trabajos incompletos

$$\overline{M}^{r}(k) \geq \tau(k) \bullet \overline{P} \bullet \overline{A}^{-} \bullet \overline{u}(k)$$

Duración del algoritmo

p < población

g < Generaciones

# 5. Implementación

## 5.1. Modelando con Rdp

Para la aplicación de las RdP en la solución de problemas de programación de sistemas de manufactura flexible, es necesario tener claramente especificada la estructura de la red que va a ser utilizada para modelar las diferentes operaciones de los trabajos, que van a ser programadas en cada una de las máquinas del sistema. Para este trabajo, se tuvieron en cuenta las relaciones de precedencia y los tiempos de setups, o de preparación, que requiere cada máquina cuando hay cambios en la naturaleza de las actividades que tiene programadas.

### 5.1.1. Estructura de la RdP

Para modelar los *n* diferentes trabajos de SMF se emplean *n* subredes acíclicas que modelan las relaciones de precedencia de cada uno de las actividades que constituyen un trabajo y están interconectadas por las plazas de recurso[17].

- Las plazas que están contenidas en cada una de las subredes, pueden desempeñar tres tipos de funciones en la red:

    - Cada una de las operaciones o tareas que componen los diferentes trabajos, siguiendo una secuencia predeterminada de actividades hasta llegar a la culminación del trabajo. Estas plazas operación, pueden tener el factor tiempo asociado, indicando el tiempo de duración de dicha operación o el tiempo que debe permanecer el token en esa plaza, tiempo en el que no están disponibles los recursos asociados.

- Son buffers, almacenamientos temporales, que indican la presencia de algún tipo acumulación de stock en el sistema.

- Son los recursos asociados a una operación, el número de plazas que son recursos asociados a una plaza operación depende de la cantidad de máquinas que tengan que ser utilizadas para llevar a cabo dicha tarea. La presencia de tokens en las plazas recurso indican la disponibilidad de dicho recurso para desarrollar una actividad determinada que lo tenga asociado. Estas plazas por lo tanto, son plazas de entrada a la transición que da inicio a una operación determinada y son plazas de salida de la transición que da por terminada dicha operación; esto asegura que, mientras que una operación esté en proceso los recursos asociados a ella no estén disponibles para ejecutar otra actividad.

- Las transiciones en cada una de las subredes, representan eventos que ocurren en el proceso de ejecutar las actividades de cada uno de los trabajos. Estos eventos incluyen el inicio de un trabajo, la finalización de un trabajo, el inicio de una operación y la finalización de una operación. Las transiciones en cada una de las subredes solamente se disparan cuando los recursos de una operación están disponibles, cuando se han cumplido las condiciones de precedencia de los trabajos, es decir, cuando los tokens han pasado por las operaciones anteriores requeridas para llevar a cabo la operación actual y se ha preparado la máquina para ejecutar dicha operación. Entonces la ocurrencia de dichos eventos en una red tiene implícita la integración de tres diferentes factores que son esenciales para llevar a cabo cada una de los trabajos, la disponibilidad y preparación de los recursos asociados y las precedencias de una operación de un trabajo.

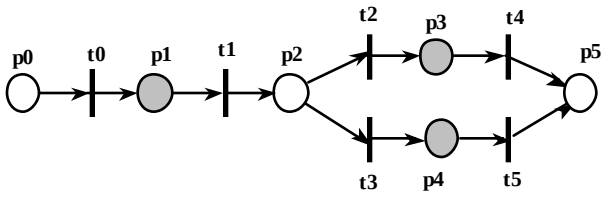
Ejemplo 1:

Se tiene un sistema de manufactura con 2 trabajos y 3 máquinas (Mach0, Mach1 y Mach 2). La ruta de cada trabajo está dada en la tabla. Los tiempos de proceso de cada operación se muestran en paréntesis.

Modelar con una red de Petri:

| | Trabajo 1 | Trabajo 2 |
|---|---|---|
| Operación 1 | Mach 0 (2) | Mach 0 (6) |
| Operación 2 | Mach 1 (5) ó Mach 2 (5) | Mach 1 (7) |

La ruta del trabajo 1 puede modelarse como en la siguiente figura

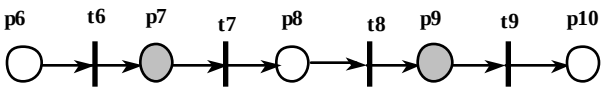



La descripción de cada plaza es como sigue:

| | |
|---|---|
| p0 | Trabajo 1 disponible |
| p1 | Trabajo 1 en proceso por máquina 0 |
| p2 | Trabajo 1 finaliza operación 1 |
| p3 | Trabajo 1 en proceso por máquina 1 |
| p4 | Trabajo 1 en proceso por máquina 2 |
| p5 | Trabajo 1 finaliza operación 2 |

Nótese que la operación 2 puede realizarse en 2 máquinas (1 ó 2) y esto genera que dos arcos emanen de la plaza p2.

La ruta del trabajo 2 puede modelarse así:

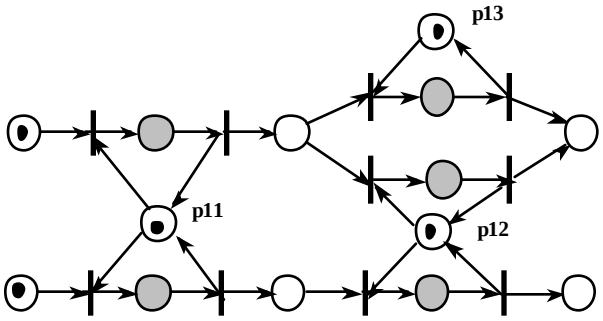


| | |
|---|---|
| p6 | Trabajo 2 disponible |
| p7 | Trabajo 2 en proceso por máquina 0 |
| p8 | Trabajo 2 finaliza operación 1 |
| p9 | Trabajo 2 en proceso por máquina 2 |
| p10 | Trabajo 2 finaliza operación 2 |

Notar que las plazas p0 a p5 pertenecen al trabajo 1 y las plazas p6 a p10 al trabajo 2. Las plazas de recurso no han sido añadidas. Se tienen 3 recursos que requieren del mismo número de plazas de recurso. Estas plazas son p11, y p12 y p13 que corresponden a Mach0, Mach1 y Mach 2 respectivamente.

Ahora falta la adición de recursos que se realiza como en la siguiente figura (plazas duplicadas corresponden a la misma plaza):

| | |
|---|---|
| p11 | Mach 0 disponible |
| p12 | Mach 1 disponible |
| p13 | Mach 2 disponible |




### 5.1.2. La RdP en el algoritmo

Para que los modelos de los SMF con las RdP puedan ser recorridas por medio de secuencias de disparos de transiciones, dirigidas por los Algoritmos genéticos, es necesario buscar la mejor manera de ingresar la información que viene implícita en los parámetros de las RdP. Es así como se utilizaron gran cantidad de matrices, vectores y variables en la

implementación del algoritmo, que ayudan a tener de una manera clara y ordenada toda la información inmersa en una RdP.

Una RdP en el algoritmo está definida, en el algoritmo, de la siguiente manera:

- La matriz de incidencia es descompuesta en dos sub matrices. Ambas son vectores de vectores, ya que las filas no tienen todas las misma dimensión $A_{min}$ (n x $\nu_i$) y $A_{plus}$ (n x $\xi_i$) con $i$ = 0, 1, 2, ...., n – 1, donde $n$ = cantidad de transiciones en la red, siendo $\nu_i$ la cantidad de plazas de entrada a la transición i y $\xi_i$ la cantidad de plazas de salida de la transición i.

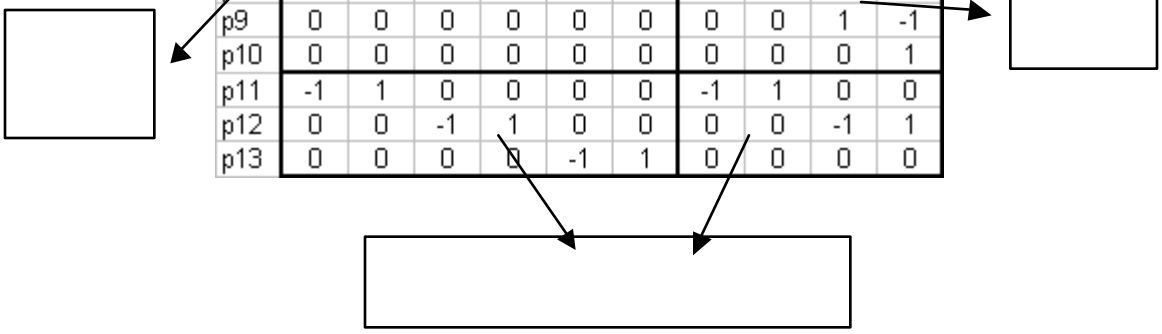
    - $A_{min}$ matriz de incidencia negativa, contiene la información de los places de entrada a cada una de las transiciones de la red. La primera posición es una plaza operación y las demás son plazas recurso.
    - $A_{plus}$ matriz positiva, contiene la información de los places de salida de cada una de las transiciones de la red. La primera posición es una plaza operación y las demás son plazas recurso que se liberan al momento de disparar la transición correspondiente.

    En esta estructura la matriz de incidencia para esta red es como sigue:

| | |
|---|---|
| **Sub-matriz trabajo 1** | **0** |
| **0** | **Sub-matriz trabajo 2** |
| **Sub-matriz recurso - trabajo 1** | **Sub-matriz recurso - trabajo 2** |

La matriz para esta red sería:

| | t0 | t1 | t2 | t3 | t4 | t5 | t6 | t7 | t8 | t9 |
|---|---|---|---|---|---|---|---|---|---|---|
| p0 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p1 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| p2 | 0 | 1 | -1 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| p3 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 |
| p4 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 |
| p5 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| p6 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| p7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 |
| p8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 0 |
| p9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 |
| p10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| p11 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 |
| p12 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | -1 | 1 |
| p13 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 0 |



Los vectores implementados Cmin y Cmax son:

| Cmin | | |
|---|---|---|
| t0 | 0 | 11 |
| t1 | 1 | |
| t2 | 2 | 12 |
| t3 | 3 | |
| t4 | 2 | 13 |
| t5 | 4 | |
| t6 | 6 | 11 |
| t7 | 7 | |
| t8 | 8 | 12 |
| t9 | 9 | |

| Cplus | |
|---|---|
| 1 | |
| 2 | 11 |
| 3 | |
| 5 | 12 |
| 4 | |
| 5 | 13 |
| 7 | |
| 8 | 11 |
| 9 | |
| 10 | 12 |

- El marcaje inicial $M_0$, es un vector fila de (1 x m - 1) con m = número de plazas en la red.

  El marcaje inicial se establece marcando las plazas “source” (sin transiciones de entrada) de cada trabajo y colocando un token en las plazas de recurso.

- El Marcaje actual $M_p$, es un vector fila de (1 x m - 1) con m = número de plazas en la red, que se va actualizando a medida que se van disparando las transiciones.

- El tiempo de proceso remanente $M_r$, es un vector fila de (1 x m - 1) con m = número de plazas en la red, tiene la información del tiempo que falta para terminar cada operación de cada trabajo.

- El tiempo de proceso PT, es un vector fila de (1 x m - 1) con m = número de plazas en la red, que contiene los tiempos de proceso de cada una de las plazas operación de la red.

- El tiempo de trabajo remanente rwt, es un vector fila de (1 x m - 1) con m = número de plazas en la red, muestra el tiempo que falta para terminar cada trabajo.

- El vector p_id, es un vector fila de (1 x m - 1) con m = número de plazas en la red, que muestra a qué trabajo corresponde cada plaza operación de la red, numerando cada plaza desde 0 hasta njobs (cantidad de trabajos) - 1. Los recursos se identifican con números negativos desde –1 hasta nres (cantidad de recursos).

Los otros componentes de la red son:

| | p0 | p1 | p2 | p3 | p4 | p5 | p6 | p7 | p8 | p9 | p10 | p11 | p12 | p13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| M0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Mr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PT | 0 | 4 | 0 | 5 | 5 | 0 | 0 | 6 | 0 | 7 | 0 | 0 | 0 | 0 |
| rwt | 9 | 5 | 5 | 0 | 0 | 0 | 13 | 7 | 7 | 0 | 0 | 0 | 0 | 0 |
| p_id | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | -1 | -2 | -3 |
| dd | 9 | 13 | | | | | | | | | | | | |

Nota: dd es la fecha de entrega de cada trabajo que es la suma de los tiempos de las operaciones de cada trabajo. Generalmente para

estudios computacionales la fecha de entrega es un múltiplo de ese número.

Como cada trabajo se modela con una sub-red de Petri. Cada sub-red contiene su propia matriz de incidencia, marcaje, vector de tiempos, etc.

- El vector *um*, es el vector de transiciones activas, de longitud variable, ya que su dimensión depende de la cantidad de transiciones disponibles en cada momento en el que se evalúa la red.

- Los setups se manejan con la variable setup_aplicado que es actualizada con la información del tiempo requerido para la preparación del recurso a programar, cuando ya se a determinado cuál es la transición a disparar. Esta información es aportada por los vectores memRes (1 x opRes) con opRes = cantidad de operaciones que utilizan recursos, memStatus (1 x opRes) y memWkc (1 x opRes), que almacenan la información de la secuencia de utilización de los recursos, la secuencia de tipos de trabajo o status que han tenido los recursos y la secuencia de utilización de los centros de trabajo, respectivamente.

  setup_aplicado se utiliza en dos procesos del algoritmo, 1. en la generación del cromosoma inicial, puesto que para la aplicación de las reglas de despacho es de vital importancia no solamente tener en cuenta los tiempos de proceso, sino también los tiempos de setup al momento de disparar las transiciones, y 2. una vez seleccionada la transición a disparar con la utilización del cromosoma, hay que sumar los tiempos de setup a los tiempos de proceso en la respuesta final, el setup hace parte de la evaluación del Cmax.

- El vector wkcMachines, es un vector fila de (1 x nres), que identifica cada recurso con su centro de trabajo, esto es debido a que en la industria es muy frecuente encontrar grupos de máquinas semejantes trabajando en paralelo, lo que se conoce como centros de trabajo. Además, la interfaz utilizada, Lekin[1], proporciona esta información al programa , información que es necesaria para implementar este tipo de sistemas con centros de trabajo.

- El vector machSetup, es un vector fila de (1 x nres), que muestra el tipo de configuración que tienen todos los recursos al momento de iniciar la programación, es decir, la preparación que tiene cada máquina antes de empezar a trabajar, lo que tiene implicaciones en los tiempos de setup iniciales, debido a que hay que cambiar la disposición de la máquina para que quede lista para empezar con el primer trabajo a ejecutar.

## 5.2. Algoritmos Genéticos

### 5.2.1. Estructura del cromosoma

- Longitud: Cada cromosoma tiene tantos genes como el doble de operaciones en la red.

- Gen: Cada gen del cromosoma es un número aleatorio o no aleatorio al que se le aplica el operador Módulo con respecto a la longitud del vector de transiciones disponibles, que va siendo actualizado con cada disparo en la red. El operador módulo dará entonces un número i que selecciona la i-ésima posición del vector de transiciones disponibles, de esta manera se selecciona una de las transiciones entre las que son aptas para ser disparadas.

[1] LEKIN. Flexible job-shop scheduling system. Leonard N. Stern School of Business. New York University. Desarrollado por Michael Pinedo y Andrew Feldman. Versión 2.4. 2001

Se definió esta estructura para el cromosoma, dado que no es posible tener un cromosoma en el que cada gen sea una transición directamente, debido a que se deben tener en cuenta las relaciones de precedencia al momento de llevar a cabo cada una de las operaciones de los diferentes trabajos y que con un cromosoma de transiciones sería no factible de ser cumplida.

Al comienzo de esta sección, se dijo que cada gen puede ser un número aleatorio o no aleatorio, esto se debe a que, para mejorar el desempeño del algoritmo, se utilizaron reglas de despacho al momento de generar el cromosoma inicial, es decir, dos cromosomas de la población inicial son generados de manera no aleatoria[2], mientras que el resto de los cromosomas se generaron aleatoriamente. Estos dos cromosomas no aleatorios, corresponden a números que disparan las transiciones de acuerdo a la regla SPT[3], el primero y a números que disparan las transiciones de acuerdo a la regla LPT[4]. Esto fue esencial en la evolución del algoritmo, ya que, desde un comienzo se tienen buenos cromosomas que dan una dirección segura y pueden ser mejorados a medida que las generaciones van transcurriendo, un mejoramiento de la raza.

### 5.2.2. Operadores Genéticos

Para generar una nueva población a partir de la población inicial deben ser utilizados operadores genéticos que permiten una combinación de la información genética de los padres. Los operadores genéticos que se emplearon en este trabajo son los siguientes:

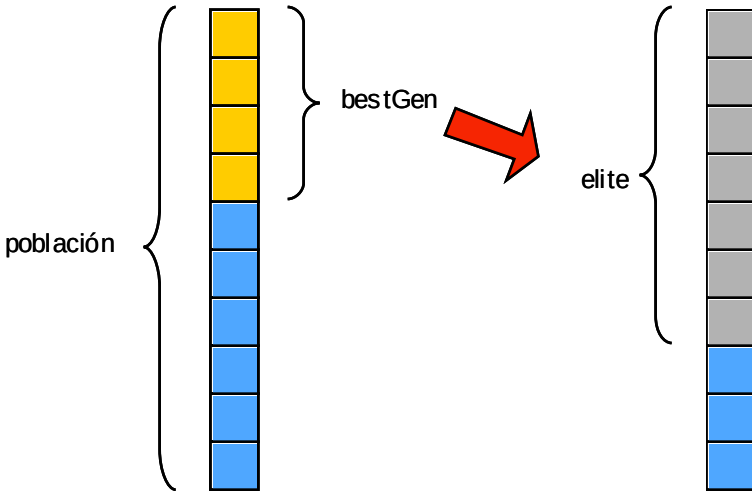
- Elitismo

[2] En la siguiente sección se explica, en detalle, el proceso de generación de estos cromosomas no aleatorios, es decir, generados a través de reglas de despacho.
[3] SPT = Shortest Procesing Time.
[4] LPT = Longest Procesing Time.

Para asegurar que la información genética de los mejores ejemplares pase de generación en generación, de tal manera que con su interacción se mejore la familia de soluciones, se utiliza el operador genético del elitismo, que consiste en seleccionar los mejores cromosomas de una población y aplicar otros operadores genéticos entre ellos, como la mutación de tal manera que conciban un porcentaje de nueva generación de cromosomas combinando la información genética de los mejores ejemplares, para abrir paso a una generación más fuerte, es decir, que de mejores resultados. El resto de la nueva generación se concibe aplicando dicho operador entre padres seleccionados aleatoriamente, sin importar su desempeño.

El elitismo se implementó de la siguiente manera:

- Se ordenan los cromosomas en orden descendente de acuerdo con su desempeño.
- Se determina un porcentaje de población elitista, es decir, el porcentaje de la población entre la cual se va a aplicar el operador,
- Se determina un porcentaje de la población generada por la élite,
- El resto de la población que no es generada por la élite, se genera aplicando el operador entre la población independientemente de su desempeño, es decir, se seleccionan los padres aleatoriamente entre toda la población, cuidando que siempre se seleccionen padres diferentes.



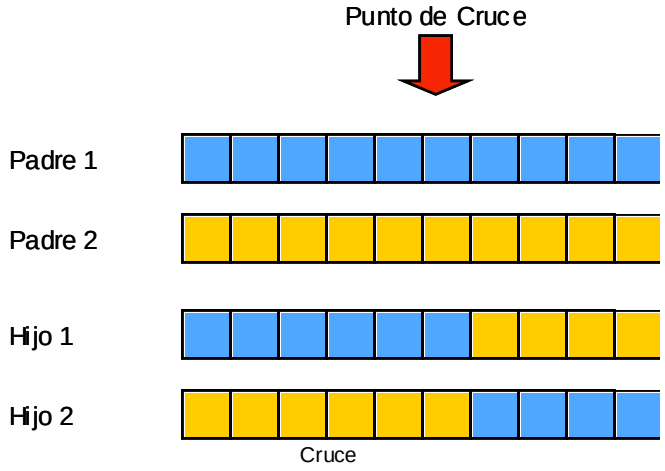
Elitismo

- Cruce por un punto.

El operador cruce utilizado se implementó de la forma clásica:

- Se determina una tasa de cruce de manera experimental,
- Se seleccionan dos padres de la población, de acuerdo con la tasa de cruce,
- Se genera un número aleatorio *alea*, si se cumple que alea ≤ la tasa de cruce determinada, se lleva a cabo el cruce entre los dos padres seleccionados, no se cruzan de lo contrario,
- Se selecciona aleatoriamente una posición de los cromosomas *pos*,
- Para el hijo 1. La información genética del padre 1, los genes del padre 1, desde el inicio del cromosoma hasta pos, se mantienen intactos en el hijo 1, desde pos hasta el final del cromosoma, se copia la información genética, los genes, del padre 2 desde pos hasta el final.
- Para el hijo 2. Se mantienen intactos los genes del padre 2 desde el inicio hasta pos y se copia la información genética del padre 1 desde pos hasta el final del cromosoma padre 1.

- Y se continua con este procedimiento hasta que sea generada una población con la misma cantidad de individuos que la anterior, verificando que siempre se seleccionen dos padres diferentes para llevar a cabo el cruce.



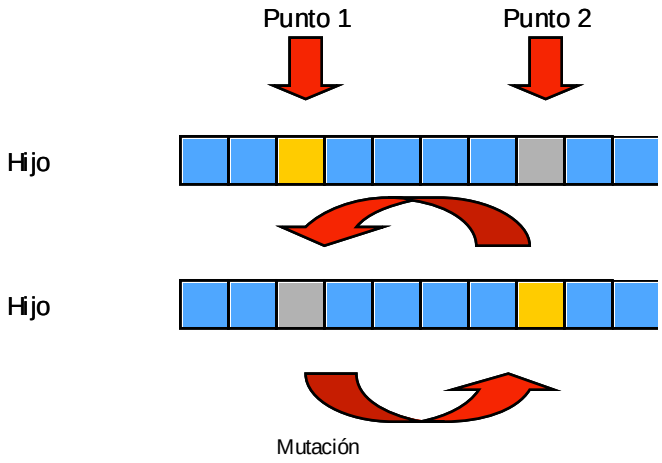

- Mutación

La mutación se implementó también de la manera clásica, y se aplicó en el algoritmo para generar una perturbación en los cromosomas, de tal manera que se pueda acceder a otra secuencia de disparos para indagar otras territorios a los que no se accedieron con la población anterior. Es una herramienta para evitar que el algoritmo se quede estancado y evolucione mejorando la raza en cada generación.

La mutación se implementó de la siguiente manera:

- Se determina una taza de mutación de manera experimental,
- Se seleccionan dos genes, o dos posiciones, del cromosoma,
- Se genera un número aleatorio *alea*, si se cumple que alea ≤ tasa de mutación determinada, se efectúa la mutación, no se efectúa de lo contrario.
- Se intercambia la información que hay en esos dos genes.

- Se efectúa este mismo procedimiento para todos los cromosomas, verificando siempre que se seleccionen dos posiciones diferentes en el cromosoma.




## 5.3. Algoritmo General

El algoritmo se estructuró de la siguiente manera:

- Generación de la población inicial.

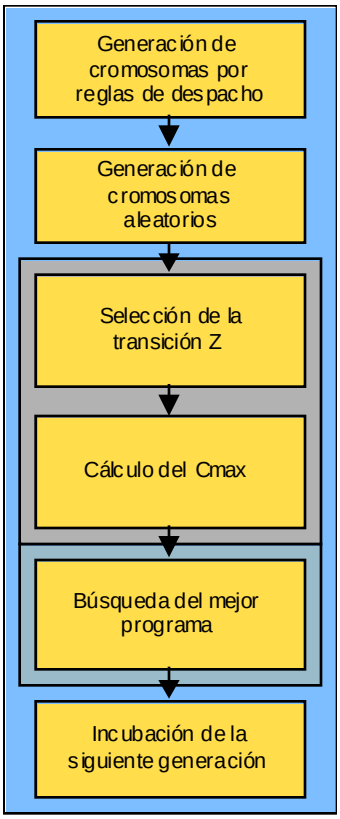
  Para generar la población inicial hay que tener en cuenta toda la información de la red, es decir, hay que recorrer toda la red para disparar todas las transiciones con las reglas de despacho, teniendo en cuenta las siguientes variables:

  - El vector de transiciones activas
  - Setup_aplicado

  En la población inicial, entonces hay dos tipos de cromosomas:

  - Los que se generaron a través de reglas de despacho.

Las reglas de despacho que se utilizaron para la generación de los dos primeros cromosomas son la SPT y la LPT[5]. Los procedimientos para la generación de estos dos cromosomas son idénticos excepto

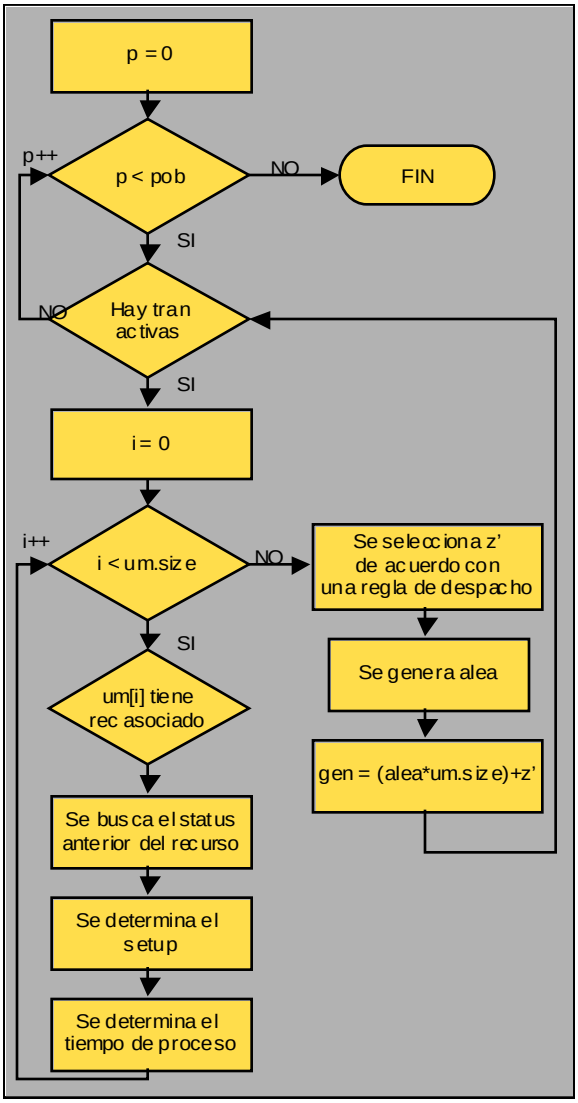


Algoritmo General

por la regla que se empleó. El procedimiento es el siguiente:

- Una a una se recorren las posiciones del vector de transiciones activas *um*.
- Si la transición activa *z'* tiene algún recurso asociado:

[5] Se contempló la utilización de otras reglas de despacho, pero debido a que debe ser tenido en cuenta el tiempo de setup y que este tiempo depende del trabajo en el que se empleó una máquina y el trabajo que esa misma máquina va a empezar, y este no es un tiempo de preparación igual para todos los cambios de actividad de un recurso, entonces es imposible saber los tiempos remanentes para la finalización de cada trabajo, lo que descarta la utilización de reglas que tengan en cuenta los tiempos remanentes.




Generación de cromosomas a partir de las reglas de despacho LPT y SPT[6]

- Se busca el estatus que tiene el recurso actualmente.
- Se determina el setup a aplicar con el cambio de actividad del recurso.

o Se almacena el tiempo de proceso asociado a la plaza operación de salida de *z'*.

o Se le suma el tiempo de setup.

[6] p lleva la cuenta de la cantidad de cromosomas han sido generados. i contador que recorre el vector de transiciones activas.

- Una vez recorrido um, con los resultados de los tiempos de proceso más los tiempos de setup de cada transición, se selecciona a *z*, una de las transiciones activas de acuerdo con una regla de despacho.
- Se genera un número aleatorio.
- Se genera cada gen de la siguiente manera:

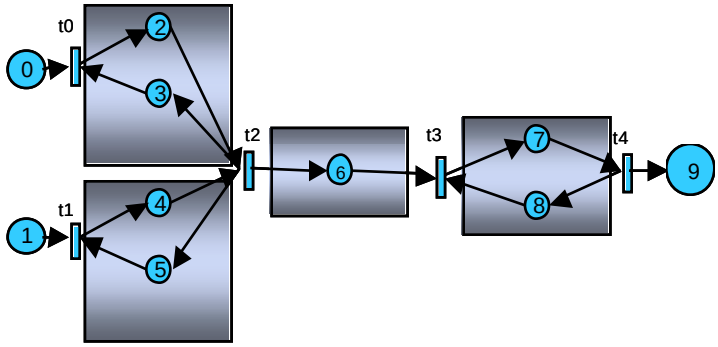
$$gen = ( aleatorio * longitud\ de\ um ) + z$$

esto es para asegurar que al momento de decodificar el gen, es decir, de seleccionar una transición para la programación de trabajos, con el módulo, se van a obtener las secuencias de transiciones determinadas por las reglas de despacho aquí empleadas.

- Se actualizan las memorias del algoritmo:
  - memWk
  - memRes
  - memStatus
  - memWkc
- Se actualiza *um.*
- Se continua con este proceso hasta que no haya más transiciones para disparar.

Ejemplo 2: La siguiente figura muestra una RdP cuyo primer vector *um* está formado por las transiciones $t_0$ y $t_1$, $um = \{t_0, t_1\}$. Debido a que ambas plazas operación tienen un recurso asociado, hay que buscar el status anterior de cada uno de los recursos para determinar el setup, si el recurso 3 estuvo

efectuando tareas tipo A anteriormente y esta es una tarea tipo B, el setup a aplicar es 3, si el recurso 5 estuvo efectuando tareas tipo C anteriormente y esta es una tarea tipo B, el setup a aplicar es 2. Los tiempos de proceso son 8 y 10 respectivamente para cada plaza operación que sale de las transiciones 0 y 1.




De acuerdo con la regla SPT, se seleccionaría la $t_0$ para ser disparada debido a que su tiempo total es de 11 mientras que el de $t_1$ 12. Por la regla LPT, se seleccionaría primero $t_1$.

Los genes para cada regla serían: *aleatorio* = 25, longitud de *um* = 2

$$Gen_{SPT} = (25 * 2) + 0 = 50$$
$$Gen_{LPT} = (25 * 2) + 1 = 51$$

- Los que se generaron aleatoriamente. El resto de la población se genera con números aleatorios sin ningún tipo de limitación.

■ Selección de las transición *z* a disparar.

Para disparar una transición de la red se requiere:

- o Vector de transiciones disponibles *um*
- o Cromosoma

- La posición actual en la secuencia de disparos

El proceso para seleccionar una transición es el siguiente:

- Se ubica el gen correspondiente a la posición actual en la secuencia de disparos.
- Se decodifica el gen de la siguiente manera[7]:

$$z = gen\ Mod\ (longitud\ um)$$

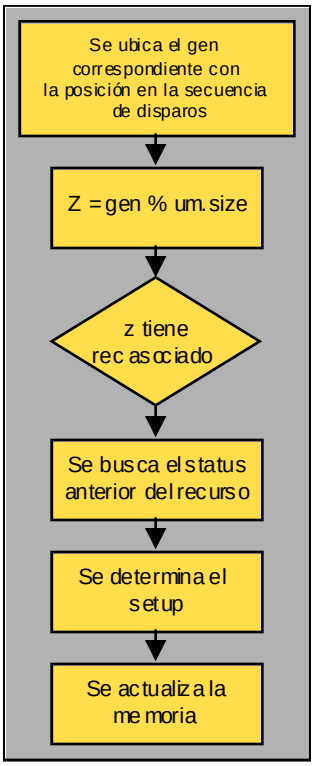
- Si la plaza correspondiente a la transición activa *z* tiene algún recurso asociado:
  - Se busca el estatus que tiene el recurso actualmente.
  - Se determina el setup a aplicar con el cambio de actividad del recurso.
- Se actualizan las memorias del algoritmo:
  - memWk
  - memRes
  - memStatus
  - memWkc
  - memTransition

Ejemplo 3: Continuando con la red del ejemplo 2 tenemos que $um = \{t_0, t_1\}$. Si el cromosoma en la posición 0 es 37, por ejemplo.

$$z = 37\ Mod\ (2) = 1$$

memWk = 0, memRes = 5, memStatus = B, memWkc = 1 y memTransition = 1

[7] Mod = Residuo de la división




Selección de la transición a disparar

- Cálculo del Cmax

Para encontrar el tiempo de finalización de la operación correspondiente a la transición disparada, se busca el máximo tiempo remanente de las plazas de entrada a la transición seleccionada *delta* y este será el tiempo tenido en cuenta en la respuesta final. delta se suma al acumulador iterTime junto con el setup asociado, para determinar el tiempo de finalización total de cada operación, cuando se terminen de disparar todas las transiciones de la red el acumulador va a tener la respuesta del tiempo de proceso de todo el programa de producción.

Después de encontrar el tiempo de finalización de cada operación se actualiza um y se repite el procedimiento anterior hasta que no haya más transiciones para disparar.

- Almacenamiento del mejor programa

  - **En la población**

    Cuando se disparan todas las transiciones de la red con la información del cromosoma y se obtiene el tiempo empleado para completar todos los trabajos, el Cmax, se almacena cada tiempo de cada individuo, en iterTime (cuya dimensión es la cantidad de individuos en la población), que hace parte de la población, para poder hacer un ordenamiento de menor, mejor desempeño, a mayor, peor desempeño, el orden se encuentra en cromoSec (1xpoblación) . Este ordenamiento es indispensable para el elitismo y para saber cuál es mejor individuo, el cromosoma que genera el mejor desempeño, el mejor tiempo de proceso en cada población y la mejor secuencia de transiciones.

  - **A través de las generaciones**

    A medida que se van obteniendo los mejores tiempos de cada población se va buscando el mejor global, es decir, el mejor Cmax de todas las generaciones, la mejor secuencia global de transiciones y el mejor cromosoma

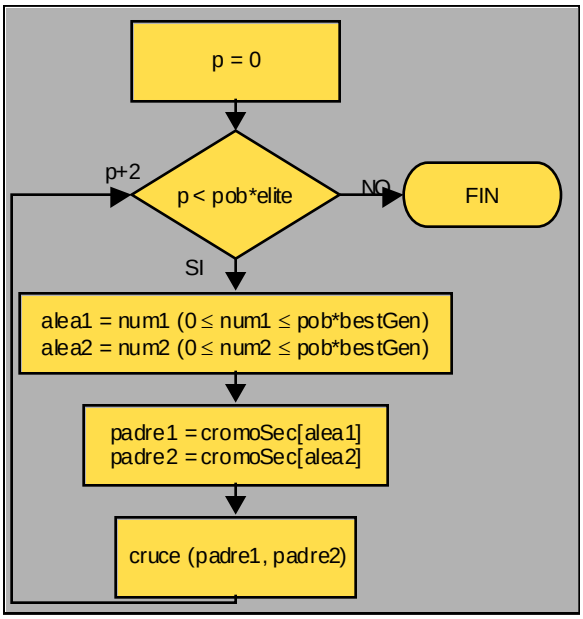
- Incubación de la siguiente generación.

Para generar los cromosomas hijos se implementaron los operadores genéticos de la siguiente manera:

- **Cruce por un punto, con elitismo**

Una vez hecho el ordenamiento de los individuos de cada población de acuerdo con su desempeño, se procede a hacer el cruce entre los cromosomas en dos tiempos:

- Se genera la población élite, determinada por la tasa de población elite que se desea (pob*elite), por medio del cruce entre la población de los mejores (bestGen), esta población está determinada por la tasa de padres mejores generados que se quieran cruzar para la próxima generación élite.



Cruce con elitismo

Para la selección de los padres que están entre los mejores, se hizo lo siguiente:

- Se generaron dos números aleatorios enteros entre 0 y pob*bestGen. Por ejemplo:

- Los dos números aleatorios van a dar dos posiciones en el vector cromoSec, que contiene el orden de los cromosomas, entonces se seleccionan los cromosomas padres, verificando que sean diferentes, con los números que se encuentran en las posiciones indicadas por los números aleatorios.

- Se genera el resto de la población por el cruce entre todos los cromosomas de la población, generando dos números aleatorios entre 0 y pob, verificando que sean números diferentes.

- **Mutación**

Se trabaja la mutación sobre la nueva población, de la manera como se explicó en la sección 4.2.2.

# 6. Resultados

Para saber cuáles son los niveles de los parámetros que brindan un mejor desempeño al algoritmo, se efectuó un experimento de efectos fijos, en el que se tuvieron en cuenta los siguientes tratamientos con los siguientes niveles:

- Mutación, 2 niveles (0.1 y 0.3)
- Cruce, 2 niveles (0.7 y 0.9)
- bestGen, 2 niveles (0.2 y 0.4)
- elite, 3 niveles (0.3, 0.5 y 0.7)

Debido a la diferente naturaleza de las instancias con las que se va a probar el algoritmo, se pensó en esto como una fuente de variación no planificada. Con el fin de controlar esta variación se incluyeron en el modelo dos tipos diferentes de problemas que fueron circunscritos como bloques, es decir, debido a la diferente naturaleza entre los problemas con los que se está probando el algoritmo, por ejemplo los LA[8] y los ORB[9], es posible que esto implique un nivel de perturbación cuyo efecto puede ser controlado a través de un factor de bloque.

El modelo del experimento planteado es el siguiente:

$$x_{ijklm} = c + cruce_i + mutación_j + bestGen_k + elite_l + tipos_m + \varepsilon_t$$

Los parámetros fueron sintonizados utilizando los problemas clásicos LA16 y ORB 1, ya que son problemas con las mismas dimensiones, que son lo suficientemente grandes como para ser una medida el comportamiento del algoritmo. Se fijó la población en 100 individuos, ya que experimentalmente una población mayor no

[8] LA16 Problema de 10 Trabajos en 10 Máquinas. Presentado por Lawrence. Referenciado [32].
[9] ORB1 Problema de 10 Trabajos en 10 Máquinas. Presentado por Bill Cook (BIC2). Referenciado [32].

generaba mejoras significativas y una población menor limitaba considerablemente el espacio de búsqueda. Se determinó que para las pruebas se iban a reproducir 20 generaciones, ya que experimentalmente con esta población es posible apreciar los efectos de los niveles de los diferentes factores y así comprobar cuáles estos factores son significativos en el modelo y, de esta manera, hacer el análisis de variabilidad entre los niveles de los tratamientos.

Los resultados obtenidos con los problemas LA16 y ORB1 son los siguientes:

| | | | | MUTACION | | | | | | MUTACION | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0,1 | | | 0,3 | | | 0,1 | | | 0,3 | | |
| | | | | elite | | | elite | | | elite | | | elite | | |
| | | | | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 |
| CRUCE | 0,7 | bestGen | 0,2 | 1014 | 996 | 1017 | 1009 | 1014 | 1020 | 1156 | 1156 | 1168 | 1186 | 1159 | 1192 |
| | | | | 1034 | 1002 | 1029 | 1023 | 1011 | 1022 | 1191 | 1160 | 1186 | 1180 | 1201 | 1160 |
| | | | 0,4 | 1040 | 1007 | 1018 | 1022 | 1024 | 1026 | 1183 | 1180 | 1196 | 1133 | 1167 | 1145 |
| | | | | 1014 | 1008 | 1029 | 1012 | 1015 | 1029 | 1178 | 1202 | 1189 | 1188 | 1172 | 1169 |
| | 0,9 | bestGen | 0,2 | 1015 | 994 | 1021 | 1022 | 1013 | 1016 | 1144 | 1131 | 1158 | 1176 | 1173 | 1176 |
| | | | | 1017 | 996 | 1026 | 1028 | 1008 | 1026 | 1159 | 1135 | 1152 | 1148 | 1169 | 1172 |
| | | | 0,4 | 1017 | 1011 | 1012 | 1028 | 1020 | 1049 | 1153 | 1179 | 1169 | 1199 | 1181 | 1192 |
| | | | | 1016 | 1013 | 1019 | 1038 | 1021 | 1021 | 1148 | 1163 | 1170 | 1196 | 1161 | 1164 |

Hay cinco hipótesis a tener en cuenta en este experimento que establecen que los diferentes niveles de cada uno de los factores no tienen efecto en el modelo y sus respectivas hipótesis alternativas, que algún nivel de cada factor, sí tenga efecto en el modelo:

$H_{0CRUCE}$ : $cruce_1 = cruce_2 = 0$

$H_{ACRUCE}$ : al menos un $cruce_i \neq 0$

$H_{0MUTACION}$ : $mutación_1 = mutación_2 = 0$

$H_{AMUTACION}$ : al menos un $mutación_j \neq 0$

$H_{0bestGen}$ : $bestGen_1 = bestGen_2 = 0$

$H_{AbestGen}$ : al menos un $bestGen_k \neq 0$

$H_{0elite}$ : $elite_1 = elite_2 = elite_3 = 0$

$H_{Aelite}$ : al menos un $elite_l \neq 0$

$H_{0tipo}$ : $tipo_1 = tipo_2 = 0$

$H_{Atipo}$ : al menos un $tipo_m \neq 0$

El Análisis de Varianza (ANOVA), de los factores y sus interacciones en el modelo es el siguiente:

**General Linear Model: X versus bloques. cruce. mutación. bestGen. elite**

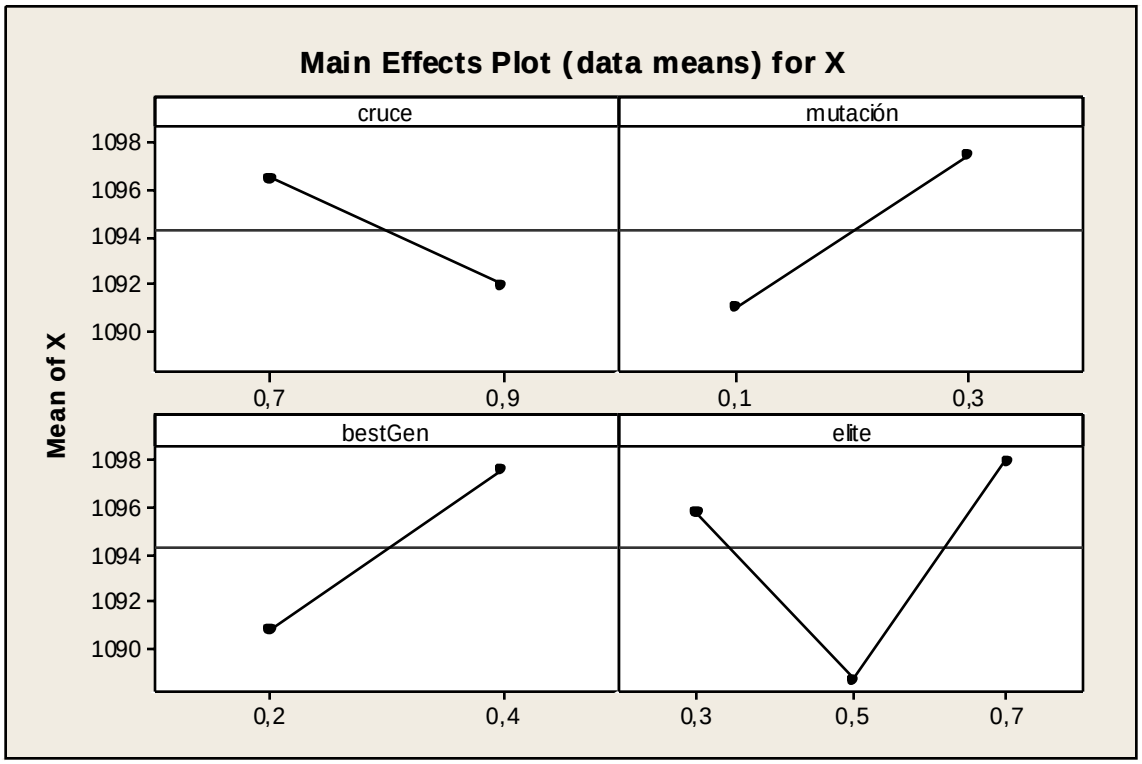
```
Factor    Type   Levels  Values
bloques   fixed       2  1. 2
cruce     fixed       2  0,7. 0,9
mutación  fixed       2  0,1. 0,3
bestGen   fixed       2  0,2. 0,4
elite     fixed       3  0,3. 0,5. 0,7
```

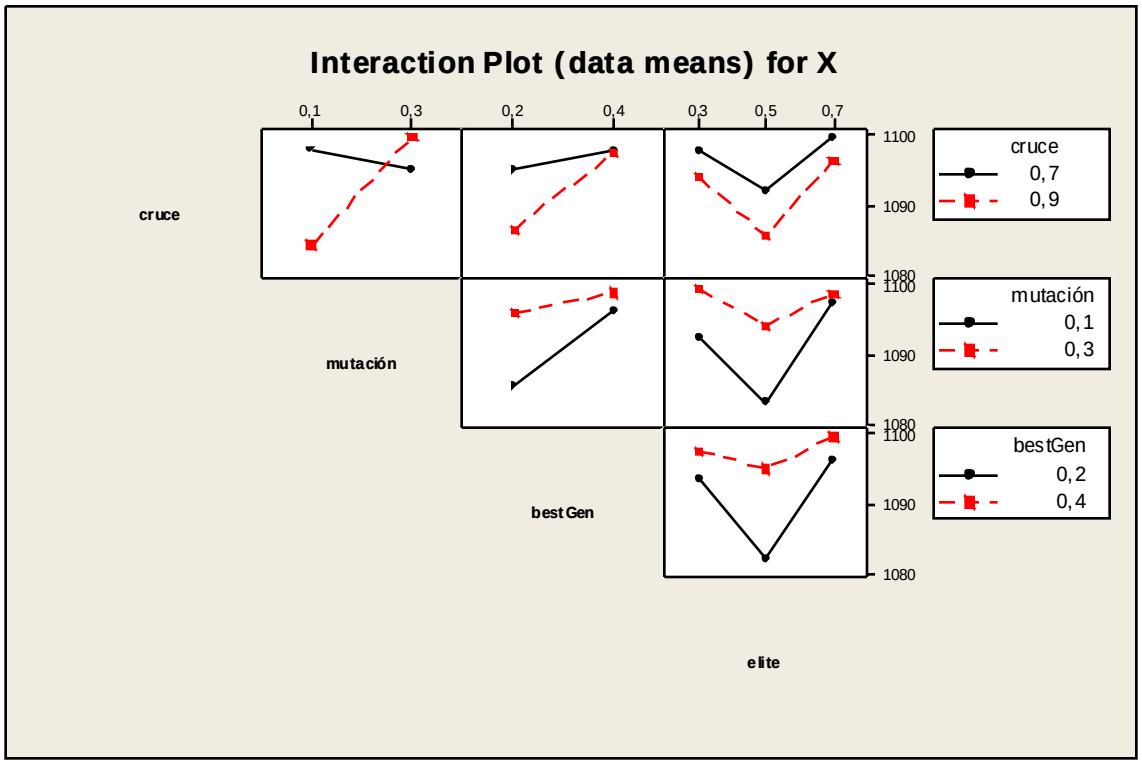
```
Analysis of Variance for X, using Adjusted SS for Tests
```

| Source | DF | Seq SS | Adj SS | Adj MS | F | P |
|---|---|---|---|---|---|---|
| bloques | 1 | 552522 | 552522 | 552522 | 3494,33 | 0,000 |
| cruce | 1 | 491 | 491 | 491 | 3,10 | 0,082 |
| mutación | 1 | 969 | 969 | 969 | 6,13 | 0,016 |
| bestGen | 1 | 1100 | 1100 | 1100 | 6,96 | 0,010 |
| elite | 2 | 1493 | 1493 | 746 | 4,72 | 0,012 |
| cruce*mutación | 1 | 2138 | 2138 | 2138 | 13,52 | 0,000 |
| cruce*bestGen | 1 | 438 | 438 | 438 | 2,77 | 0,101 |
| cruce*elite | 2 | 54 | 54 | 27 | 0,17 | 0,844 |
| mutación*bestGen | 1 | 372 | 372 | 372 | 2,35 | 0,129 |
| mutación*elite | 2 | 383 | 383 | 191 | 1,21 | 0,304 |
| bestGen*elite | 2 | 448 | 448 | 224 | 1,42 | 0,249 |
| cruce*mutación*bestGen | 1 | 556 | 556 | 556 | 3,52 | 0,065 |
| cruce*mutación*elite | 2 | 359 | 359 | 179 | 1,13 | 0,328 |
| cruce*bestGen*elite | 2 | 80 | 80 | 40 | 0,25 | 0,778 |
| mutación*bestGen*elite | 2 | 696 | 696 | 348 | 2,20 | 0,118 |
| cruce*mutación*bestGen*elite | 2 | 357 | 357 | 178 | 1,13 | 0,329 |
| Error | 71 | 11226 | 11226 | 158 | | |
| Total | 95 | 573679 | | | | |

```
S = 12,5746   R-Sq = 98,04%   R-Sq(adj) = 97,38%
```

El ANOVA anterior muestra que los factores principales son significativos en el modelo, es decir, se rechaza la $H_0$ de que no hay efecto de ninguno de los factores, la respuesta del algoritmo depende de los niveles que tomen cada uno de sus factores y del tipo de instancia que se esté corriendo (bloques).




Debido a que los factores son significativos en el modelo, es decir se rechazaron las hipótesis nulas, se utilizó el procedimiento de Tukey de comparación de

medias, para identificar diferencias importantes entre los distintos niveles de los factores bajo observación. Los resultados son los siguientes[10]:

### One-way ANOVA: X versus cruce

```
Source  DF      SS    MS     F      P
cruce    1     491   491  0,08  0,777
Error   94  573189  6098
Total   95  573679

S = 78,09   R-Sq = 0,09%   R-Sq(adj) = 0,00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level   N    Mean  StDev  -------+---------+---------+---------+--
0,7    48  1096,5   80,5     (--------------*--------------)
0,9    48  1092,0   75,6  (--------------*--------------)
                          -------+---------+---------+---------+--
                              1080      1095      1110      1125

Pooled StDev = 78,1


Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of cruce

Individual confidence level = 95,00%


cruce = 0,7 subtracted from:

cruce   Lower  Center  Upper  --------+---------+---------+---------+-
0,9    -36,17   -4,52  27,13  (---------------*---------------)
                              --------+---------+---------+---------+-
                                    -20         0        20        40
```

### One-way ANOVA: X versus mutación

```
Source    DF      SS    MS     F      P
mutación   1     969   969  0,16  0,691
Error     94  572710  6093
Total     95  573679

S = 78,06   R-Sq = 0,17%   R-Sq(adj) = 0,00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level   N    Mean  StDev  --------+---------+---------+---------+-
0,1    48  1091,1   78,2  (--------------*--------------)
0,3    48  1097,4   77,9       (--------------*--------------)
                          --------+---------+---------+---------+-
                               1080      1095      1110      1125

Pooled StDev = 78,1
```

[10] Se utilizó MINITAB para realizar este análisis

```
Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of mutación

Individual confidence level = 95,00%


mutación = 0,1 subtracted from:

mutación   Lower  Center  Upper  ------+---------+---------+---------+---
0,3       -25,28    6,35  37,99     (---------------*---------------)
                                 ------+---------+---------+---------+---
                                     -20         0        20        40
```

## One-way ANOVA: X versus bestGen

```
Source   DF      SS    MS     F      P
bestGen   1    1100  1100  0,18  0,672
Error    94  572579  6091
Total    95  573679

S = 78,05   R-Sq = 0,19%   R-Sq(adj) = 0,00%


                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level   N    Mean  StDev  --------+---------+---------+---------+-
0,2    48  1090,9   77,5  (--------------*--------------)
0,4    48  1097,6   78,6       (--------------*--------------)
                          --------+---------+---------+---------+-
                               1080      1095      1110      1125

Pooled StDev = 78,0


Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of bestGen

Individual confidence level = 95,00%


bestGen = 0,2 subtracted from:

bestGen   Lower  Center  Upper  ------+---------+---------+---------+---
0,4      -24,86    6,77  38,40      (--------------*---------------)
                                 ------+---------+---------+---------+---
                                     -20         0        20        40
```

## One-way ANOVA: X versus elite

```
Source  DF      SS    MS     F      P
elite    2    1493   746  0,12  0,886
Error   93  572187  6153
Total   95  573679

S = 78,44   R-Sq = 0,26%   R-Sq(adj) = 0,00%
```

```
                          Individual 95% CIs For Mean Based on
                          Pooled StDev
Level   N    Mean  StDev  ---------+---------+---------+---------+
0,3    32  1095,8   76,9     (------------*------------)
0,5    32  1088,8   81,9  (------------*------------)
0,7    32  1098,1   76,5      (------------*------------)
                          ---------+---------+---------+---------+
                                1080      1100      1120      1140

Pooled StDev = 78,4


Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of elite

Individual confidence level = 98,08%


elite = 0,3 subtracted from:

elite   Lower  Center  Upper  ---------+---------+---------+---------+
0,5    -53,76   -7,03  39,70   (---------------*--------------)
0,7    -44,51    2,22  48,95      (---------------*--------------)
                           ---------+---------+---------+---------+
                                  -30         0        30        60


elite = 0,5 subtracted from:

elite   Lower  Center  Upper  ---------+---------+---------+---------+
0,7    -37,48    9,25  55,98         (--------------*---------------)
                           ---------+---------+---------+---------+
                                  -30         0        30        60
```

De acuerdo con los resultados anteriores, no hay diferencias significativas entre los diferentes niveles de los diferentes factores, ya que todos los intervalos de confianza de las diferencias de medias contienen al 0.

Con este resultado, se fijan los niveles de cada uno de los factores de acuerdo con el comportamiento de los efectos principales y sus interacciones, que muestran que el algoritmo tiene un mejor desempeño cuando:

- La tasa de cruce es de 0.9
- La tasa de mutación es de 0.1
- bestGen es 0.2
- elite es 0.5

Para ver los efectos de los factores cuándo aumenta el número de generaciones a 100, es decir, cuando el algoritmo se está estabilizando, se hizo el siguiente análisis de varianza:

| | | | | MUTACION | | | | | | MUTACION | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0,1 | | | 0,3 | | | 0,1 | | | 0,3 | | |
| | | | | elite | | | elite | | | elite | | | elite | | |
| | | | | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 | 0,3 | 0,5 | 0,7 |
| CRUCE | 0,7 | bestGen | 0,2 | 1015 | 1021 | 1002 | 1008 | 1023 | 1007 | 1133 | 1169 | 1147 | 1153 | 1114 | 1177 |
| | | | | 1035 | 1008 | 1013 | 1009 | 1007 | 1014 | 1143 | 1179 | 1145 | 1151 | 1171 | 1177 |
| | | | 0,4 | 1007 | 994 | 1002 | 1004 | 1015 | 989 | 1180 | 1153 | 1165 | 1145 | 1165 | 1179 |
| | | | | 1021 | 1014 | 1008 | 1001 | 1004 | 1002 | 1150 | 1154 | 1149 | 1119 | 1148 | 1162 |
| | 0,9 | bestGen | 0,2 | 1002 | 998 | 1022 | 1006 | 1008 | 1028 | 1154 | 1145 | 1148 | 1167 | 1102 | 1167 |
| | | | | 1019 | 1012 | 1023 | 1010 | 1008 | 1028 | 1183 | 1158 | 1160 | 1160 | 1131 | 1172 |
| | | | 0,4 | 1016 | 995 | 1006 | 1004 | 1006 | 994 | 1160 | 1161 | 1163 | 1114 | 1157 | 1129 |
| | | | | 1000 | 1012 | 1024 | 1014 | 1000 | 1009 | 1127 | 1156 | 1155 | 1144 | 1134 | 1156 |

**General Linear Model: X versus bloque. Cruce. Mutacion. bestGen. elite**

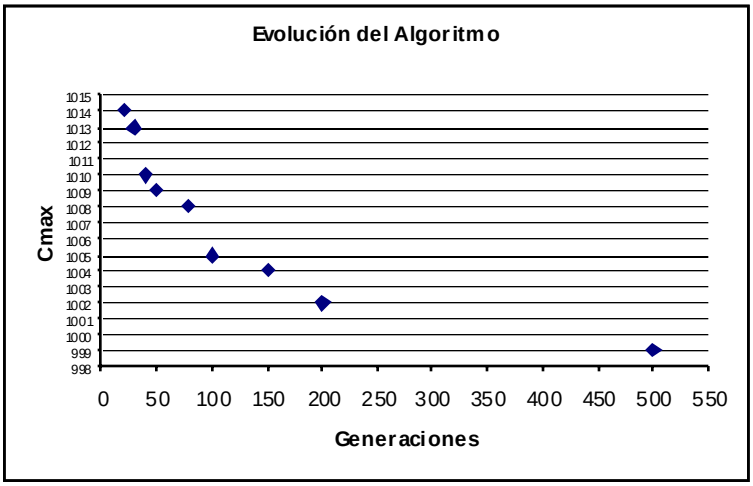
```
Factor    Type   Levels  Values
bloque    fixed       2  1. 2
Cruce     fixed       2  0,7. 0,9
Mutacion  fixed       2  0,1. 0,3
bestGen   fixed       2  0,2. 0,4
elite     fixed       3  0,3. 0,5. 0,7
```

```
Analysis of Variance for X, using Adjusted SS for Tests

Source                        DF  Seq SS  Adj SS  Adj MS        F      P
bloque                         1  490776  490776  490776  2617,28  0,000
Cruce                          1     113     113     113     0,60  0,441
Mutacion                       1     477     477     477     2,54  0,115
bestGen                        1     737     737     737     3,93  0,051
elite                          2     721     721     361     1,92  0,154
Cruce*Mutacion                 1      81      81      81     0,43  0,514
Cruce*bestGen                  1      74      74      74     0,39  0,533
Cruce*elite                    2     715     715     358     1,91  0,156
Mutacion*bestGen               1     210     210     210     1,12  0,293
Mutacion*elite                 2     784     784     392     2,09  0,131
bestGen*elite                  2     494     494     247     1,32  0,274
Cruce*Mutacion*bestGen         1       1       1       1     0,01  0,929
Cruce*Mutacion*elite           2     346     346     173     0,92  0,402
Cruce*bestGen*elite            2     821     821     410     2,19  0,120
Mutacion*bestGen*elite         2    1319    1319     659     3,52  0,035
Cruce*Mutacion*bestGen*elite   2     271     271     135     0,72  0,489
Error                         71   13314   13314     188
Total                         95  511253
```

```
S = 13,6936   R-Sq = 97,40%   R-Sq(adj) = 96,52%
```

Como resultado de este análisis se aprecia que a este nivel ninguno de los factores es significativo, ya que, los cambios radicales se produjeron por influencia de los factores en las generaciones iniciales.




Lo anterior sirve para determinar la cantidad de generaciones que se van reproducir en el algoritmo. Se determinó que para las pruebas 100 generaciones son suficientes para obtener una respuesta aceptable, puesto que con un menor número de generaciones se producen resultados que pueden ser mejorados con un mayor número de generaciones y con un mayor número de generaciones no hay mejoras sustanciales y si un consumo significativo de tiempo.

Para ver el comportamiento del algoritmo en otras instancias se corrieron otro tipo de problemas, que ya han sido estudiados con anterioridad y de los que se tiene registro de la mejor respuesta encontrada hasta el momento, que es considerada el óptimo y el punto de referencia.

Debido a que los problemas iniciales son de pequeñas dimensiones, el algoritmo trabaja muy bien, encontrando la solución óptima, básicamente porque la selección de cada transición es hace de entre un conjunto pequeño de posibilidades, encontrando la mejor solución. A medida que aumenta la dimensión de los problemas, el conjunto del cual se selecciona una transición se hace más

grande, lo que se presta a que se generen muchas posibles soluciones sin que necesariamente se llegue a la mejor secuencia de disparos.

| Problemas | Resultado ($C_{max}$) | Solución ($C_{max}$) | Desviación | Tiempo (seg) |
|---|---|---|---|---|
| LA01 | 666 | 666 | 0,00% | 48 |
| LA02 | 700 | 655 | 6,87% | 49 |
| LA03 | 634 | 597 | 6,20% | 48 |
| LA04 | 611 | 590 | 3,56% | 49 |
| LA05 | 593 | 593 | 0,00% | 49 |
| LA06 | 926 | 926 | 0,00% | 98 |
| LA07 | 906 | 890 | 1,80% | 98 |
| LA08 | 863 | 863 | 0,00% | 98 |
| LA09 | 951 | 951 | 0,00% | 98 |
| LA10 | 958 | 958 | 0,00% | 98 |
| LA11 | 1222 | 1222 | 0,00% | 166 |
| LA12 | 1039 | 1039 | 0,00% | 164 |
| LA13 | 1150 | 1150 | 0,00% | 168 |
| LA14 | 1292 | 1292 | 0,00% | 169 |
| LA15 | 1238 | 1207 | 2,57% | 165 |
| LA16 | 1007 | 945 | 6,56% | 160 |
| LA17 | 823 | 784 | 4,97% | 162 |
| LA18 | 863 | 848 | 1,77% | 165 |
| LA19 | 875 | 842 | 3,92% | 163 |
| LA20 | 941 | 902 | 4,32% | 162 |
| LA21 | 1148 | (1040 , 1053) | (10.39 , 9.02)% | 350 |
| LA22 | 1003 | 927 | 8,20% | 351 |
| LA23 | 1045 | 1032 | 1,26% | 344 |
| LA24 | 1019 | 935 | 8,98% | 343 |
| LA25 | 1096 | 977 | 12,18% | 342 |
| LA27 | 1363 | (1235 , 1269) | (10.36 , 7.4)% | 588 |
| LA28 | 1350 | 1216 | 11,02% | 588 |
| LA29 | 1306 | (1120 , 1195) | (16,6% , 10,1)% | 587 |
| LA30 | 1433 | 1355 | 5,76% | 580 |
| LA33 | 1782 | 1719 | 3,66% | 1281 |
| LA36 | 1404 | 1268 | 10,73% | 730 |
| ORB1 | 1143 | 1059 | 7,93% | 162 |
| ORB2 | 990 | 888 | 11,49% | 162 |
| ORB3 | 1132 | 1005 | 12,64% | 164 |
| ORB4 | 1132 | 1005 | 12,64% | 165 |
| ORB5 | 944 | 887 | 6,43% | 166 |
| ORB10 | 990 | 944 | 4,87% | 169 |

# 7. Aplicación[11]

Una vez desarrollado el algoritmo en sus diferentes etapas, es necesario hacer una aplicación práctica con un elemento real de producción, se ha seleccionado para este fin el sector metalmecánico, específicamente la producción de frenos delanteros de un automóvil Corsa Chevrolet GL y para hacer un poco más atractiva la aplicación se modelaron los frenos delanteros de una Luv 4x2 HEC CS.

- Descripción general

El freno de este tipo de carros está compuesto por un cilindro maestro, que envía una multiplicación de fuerza al caliper, que ajusta o cierra las pastillas sobre el disco, impidiendo que este gire y ejerciendo la acción de frenado.

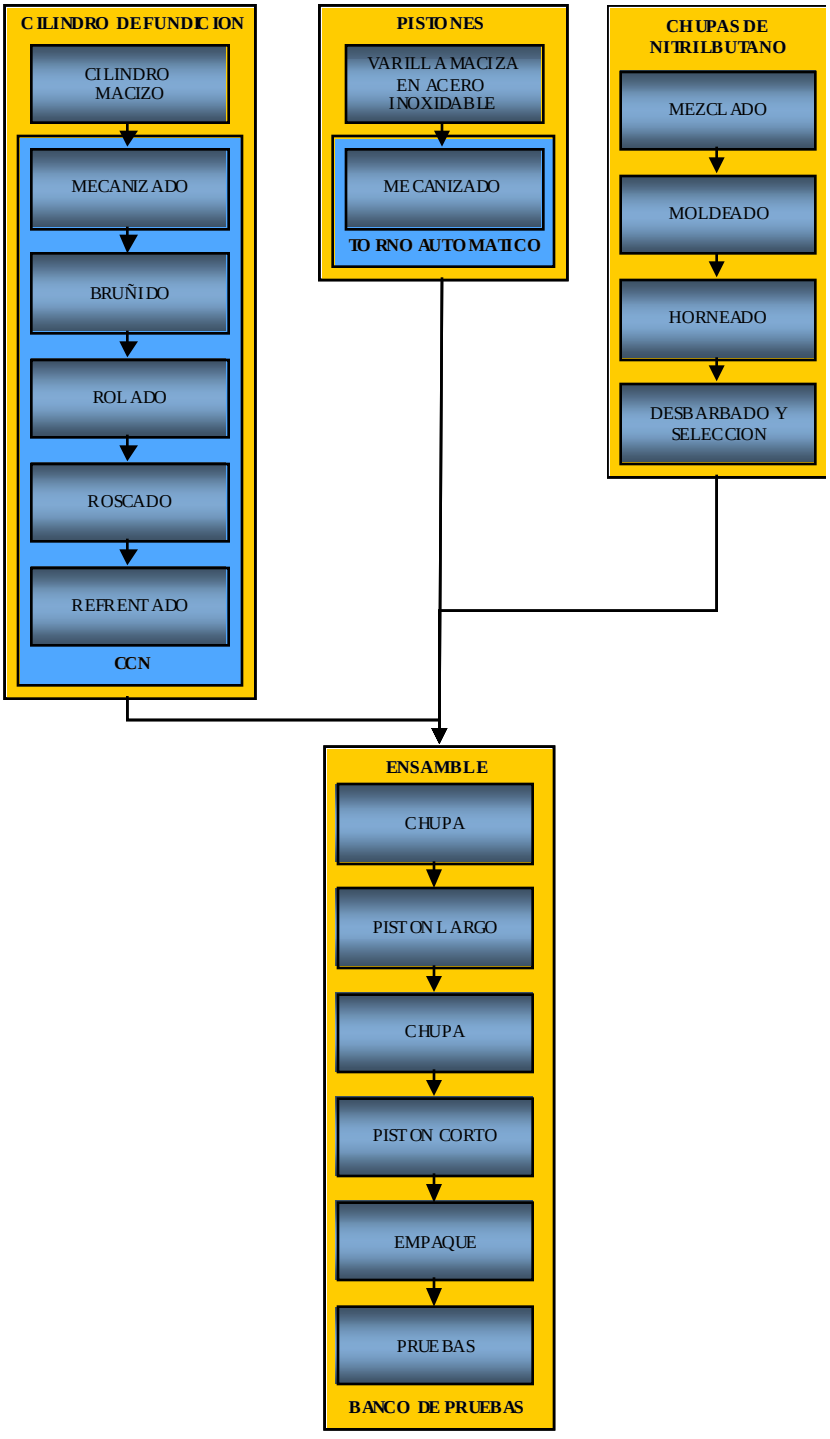
- Cilindro maestro

El cilindro maestro está constituido por tres tipos de piezas:

  - El cilindro, como tal que se fabrica en un centro de control numérico donde las herramientas realizan operaciones de taladro, fresado, bruñido, rolado, roscado y refrentado, que partiendo de una pieza maciza de fundición en acero al carbon, deja la pieza con una terminación con tolerancias admisibles de 0.0001 de milímetro, ya que es considerada una pieza de alta seguridad en el vehículo.
  - Los pistones, son fabricados a partir de una varilla de acero SAE 1040 que es mecanizada en un torno automático.
  - Las chupas, se hacen por lotes de 1000 unidades, el proceso inicia mezclado de nitrilbutano con estabilizantes y colorantes, se lleva a

[11] La información del proceso fue obtenida gracias a la colaboración del Ing. Luis Eduardo Galves Rey

una temperatura de fusión controlada con un margen de tolerancia de +/- 2ºC, hasta que la mezcla adquiera una consistencia líquida; se vierte en moldes y pasa a un proceso de horneado a temperatura descendente controlada. Luego se hace el desbarbado y el

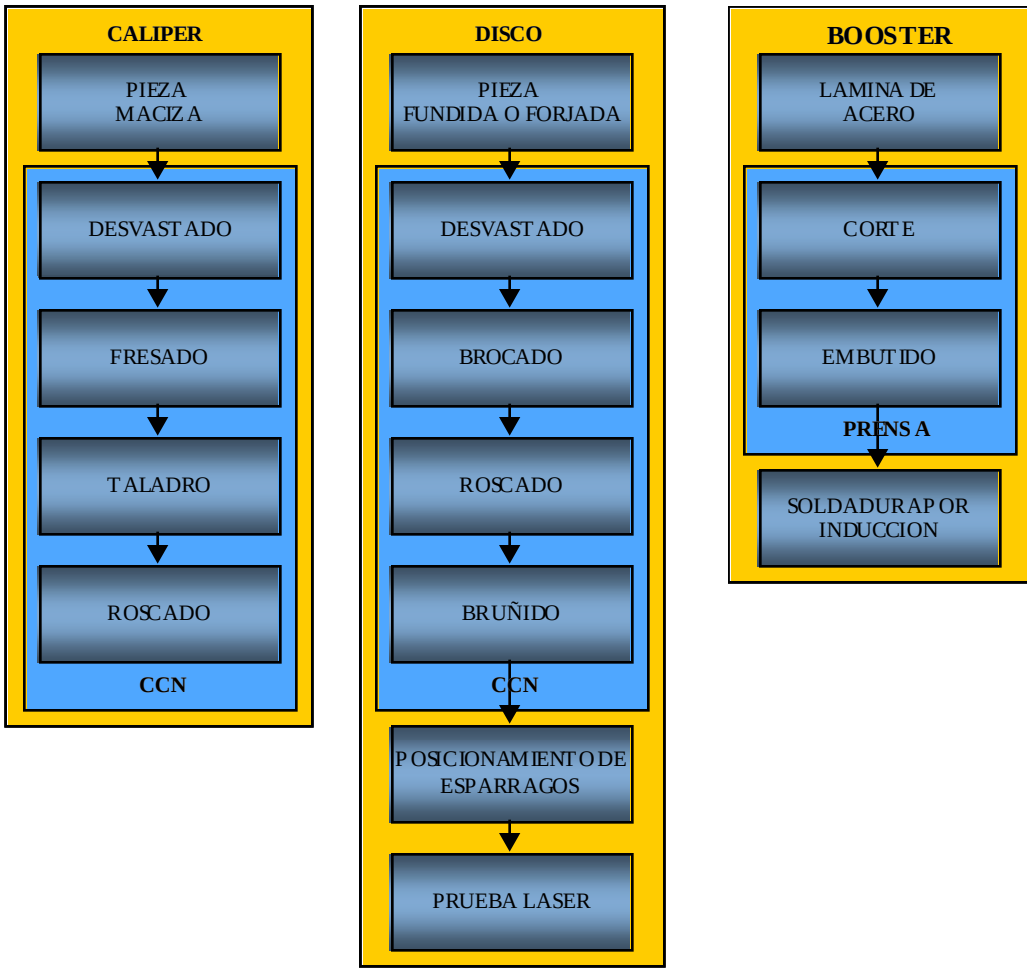



patronado uno a uno con un calibrador pasa no pasa.

Luego estas tres piezas van a un ensamble para obtener el producto final del cilindro maestro de freno

- Caliper

Es una pieza maciza en fundición de acero al carbono, que se monta en un centro de control numérico que tiene unas herramientas especializadas, debido a que es una pieza de geometría compleja a la cual se le hacen fresados de desbaste frontales y longitudinales, taladrados roscados, obteniéndose una pieza de alta seguridad, se consideran tolerancias de 0.0001 de milímetro.



- Disco

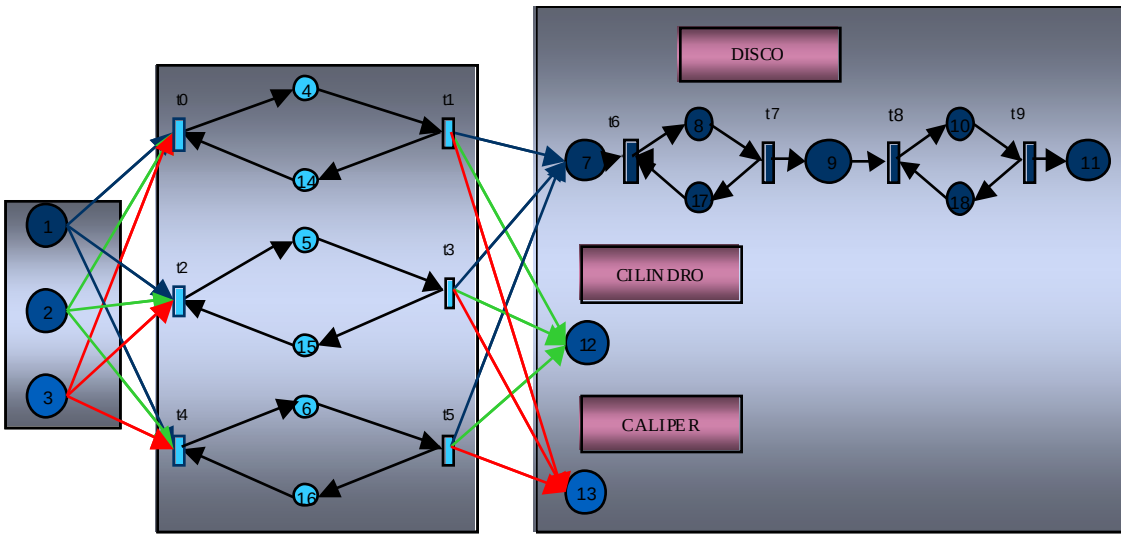
Se elabora en un centro de control numérico partiendo de una pieza fundida o forjada en el que se realizan operaciones de desbaste, fresado, perforación, roscado, bruñido. La pieza obtenida debe ser probada en un dispositivo laser para medir la concentridad de sus agujeros por tratarse de la pieza más delicada del conjunto freno, ya que es la que soporta la rueda del vehículo.

- Booster

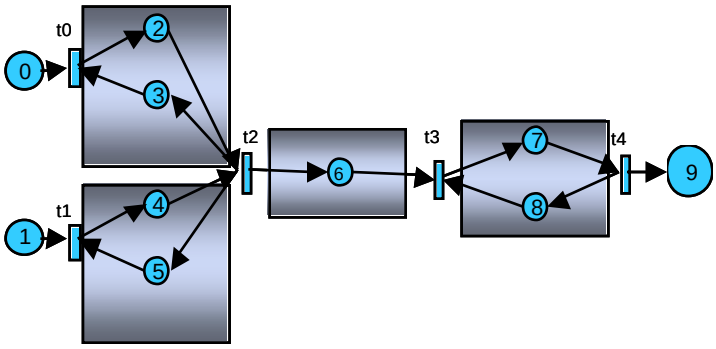
Se fabrica en dos procesos, ya que tiene 2 tipos de tapas, partiendo de una lámina de acero en una prensa de 1 ton con un troquel especial, se hace un corte y embutido tanto para la tapa A como para la tapa B y luego las dos son unidas mediante un proceso de soldadura por inducción .

La representación de la aplicación con RdP, sería la siguiente:

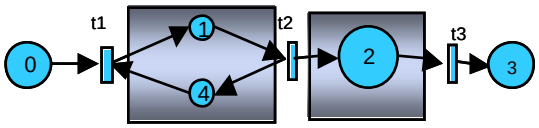
Centro de Control Numérico (CCN):

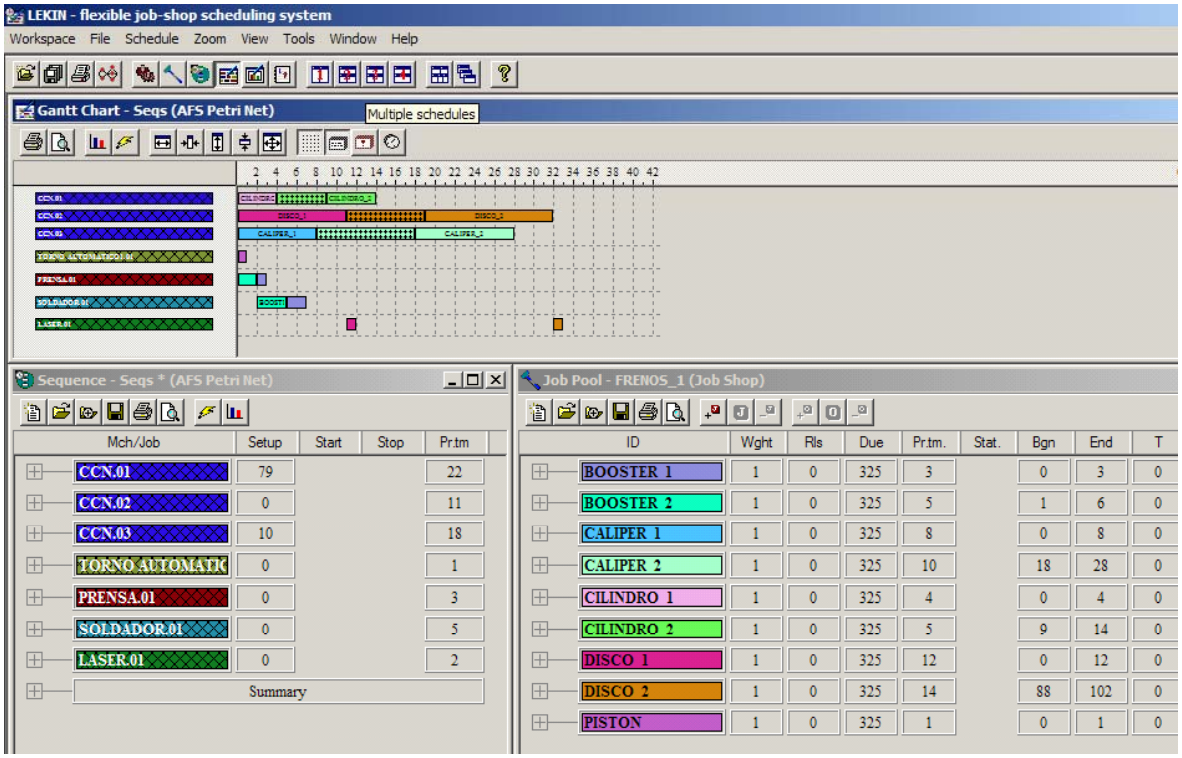


Prensa:




Torno Automático:



Los resultados de LEKIN nos dan el resultado esperado para este problema aplicado. Los setups para el CCN, fueron determinados para los frenos tipo 1, los frenos delanteros del CORSA, inicialmente, luego se quiere ver el desempeño de la red cuando se quiere cambiar el tipo de freno a fabricar a los tipo 2, los frenos delanteros de la LUV.

El status inicial del CCN, con 3 máquinas en paralelo, cada una fabricando un componente diferente, el CCN 1 está programado inicialmente para fabricar CILINDROS tipo 1, el CCN 2 está programado inicialmente para fabricar DISCOS tipo 1 y el CCN 3 está programado inicialmente para fabricar CALIPER tipo 1. El cambio en la preparación de una máquina que viene trabajando CALIPER y se quiere hacer con ella otro tipo de CALIPER no es muy significativo, 10 minutos. Lo mismo sucede con los setups de los cilindros en una misma máquina, con un setup de 5 minutos y con los discos de 8 minutos. En cambio, si se quiere cambiar el tipo de pieza a fabricar, por ejemplo de DISCO a CALIPER, tiene un tiempo de preparación de 99 minutos.

Por lo anterior la respuesta encontrada por el algoritmo es la adecuada, la de seguir la programación en una misma máquina con piezas del mismo tipo para evitar que los tiempos de setup sean grandes y esto aumente el $C_{max}$

# 8. Conclusiones y Recomendaciones

- Los resultados obtenidos de las diferentes instancias que se corrieron aplicando la combinación de las RdP y los AG, demuestran que esta técnica es una buena alternativa para encontrar soluciones a problemas NP-Hard, como los que se encuentran en la industria. Aunque, a medida que la complejidad de los trabajos aumenta se aleja de la solución óptima conocida, ya que aumenta la cantidad de transiciones disponibles, es decir, aumenta el universo de posibilidades que pueden ser seleccionadas cada vez, generando entonces gran cantidad de soluciones diferentes. A pesar de esto, las soluciones se encuentran en un rango en el las soluciones se pueden considerar buenas, obtenidas en un buen tiempo.

- El algoritmo planteado en este trabajo puede estar sujeto a mejoras:
    - La implementación del elitismo común, que traslada sin alteración el o los mejores individuos de la población anterior a la nueva generación, lo que podría ser una buena alternativa, ya que se inicia la nueva población con un buen individuo que podría encargarse de mantener y mejorar sus genes a través de las generaciones.
    - La tasa de mutación podría ser dinámica, esto podría ampliar el rango de soluciones o podría asegurar que los buenos genes no se pierdan en el camino por una tasa de mutación alta.

- La sintonización de los factores del AG es muy importante, y consume tiempo, ya que no se puede crear una aplicación que solo de buenos resultados para cierto tipo de problemas, sino que debe ser pensado, de tal manera que sea lo suficientemente flexible para que sea utilizado en diferentes tipos de actividades. Es por esto que hay que tener en cuenta niveles que abarquen todos los posibles comportamientos del algoritmo y que podrían llegar a determinar su comportamiento, pero como se vio con

los problemas que se utilizaron para este fin, el algoritmo demuestra un comportamiento robusto ante la variación de los niveles de los factores, ya que las soluciones en los diferentes niveles no muestra una gran variación.

- La posibilidad de aplicación práctica del algoritmo, demuestra su capacidad de adaptación a diferentes procesos. Gracias a la estructura de las RdP, que pueden ser adecuadas para modelar diferentes tipos de situaciones ya la estructura flexible y simple, de los AG que pueden ser planteados de acuerdo con el tipo de problema que se quiera solucionar, lo que puede potencializar a esta combinación como una herramienta de uso continuo en las diferentes empresas que requieran programación de sus tareas

- El tiempo empleado por el algoritmo para generar buenas soluciones es relativamente bueno, de acuerdo con las condiciones en las que se hizo la implementación del algoritmo, Este tiempo aumenta en la medida en que la dimensión de los problemas que se están trabajando aumenta, ya que aunque sea la misma cantidad de individuos en cada generación y la misma cantidad de descendencia la que se está procreando en todos los problemas, el aumento en la cantidad de trabajos y máquinas a programar aumenta la cantidad de transiciones a disparar y por lo tanto la cantidad de operaciones y tiempo computacional que emplea el algoritmo

## 9. Bibliografía

*[1] Mu Der Jeng, Reyu Wen Jaw, Pen Li Hung. Sceduling FMS with Due Dates Based on Petri State Equations. Proceedings of the 1997IEEE. p 2724, 5p*

*[2] Huanxin Henry Xiong, MengChu Zhou, Reggie J. Caudill. A Hybrid Heuristic Search Algorith For Scheduling Flexible Manufacturing Systems. Proceedings of the 1996 IEEE. April 1996. p2793, 4p*

*[3]* D *J Stockton*, L Quinn and R A Khalil. Use of genetic algorithms in operations management Part 1: applications. Proceedings of the Institution of Mechanical Engineers -- Part B -- Engineering Manufacture; 2004, Vol. 218 Issue 3, p315, 13p*

*[4]. Stockton, D.J.; Quinn, L.; Khalil, R.A. Use of* genetic algorithms *in operations management Part 2: results. Proceedings of the Institution of Mechanical Engineers -- Part B -- Engineering Manufacture, 2004, Vol. 218 Issue 3, p329, 15p;*

*[5] Ahmad, R.; Jamaluddin, H.; Hussain, M. A. Model structure selection for a discrete-time non-linear system using a genetic algorithm. Proceedings of the Institution of Mechanical Engineers -- Part I -- Journal of Systems & Control Engineering, 2004, Vol. 218 Issue 2, p85, 14p*

*[6] S. Lloyd, H. Yu y N. Konstans. FMS Scheduling Using Petri Nets Modeling and Branch & Bound Search. Proceedings of th IEEE. International Symposium on Assembly. 1995. p141, 5p*

*[7]* W. J. ZHANG, Q. LI, Z. M. BI and X. F. ZHA. *A generic Petri net model for flexible manufacturing systems and its use for FMS control software testing.* INT. J. PROD. RES., 2000, VOL. 38, NO. 5, 1109 - 1131

*[8]* H. Yu, A. Reyes, S. Cang and S. Lloyd. *Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems—part 1. Petri net modeling and heuristic search. Computers & Industrial Engieering, 44(2003) 527-543*

*[9] ]* H. Yu, A. Reyes, S. Cang and S. Lloyd. *Combined Petri net modelling and AI-based heuristic hybrid searchfor flexible manufacturing systems—part II. Heuristic hybrid search. Computers & Industrial Engieering, 44(2003) 527-543*

*[10] Jean-Marie Proth and Nathalie Sauer. Schedulling of picewise constant product flows: A Petri net approach. European Journal of Operational Research 106(1998) 45-56.*

*[11]Dongsheng Liu , Jianmin Wang, Stephen C.F. Chan, Jiaguang Sun and Li Zhang. Modeling workflow processes with colored Petri nets. Computers in Industry 49(2002) 267-281.*

*[12] A. Reyes, H. Yu, G. Kelleher and S. Lloyd. Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS. Computers in Industry 47(2002) 123-138.*

*[13]* Jyh-Horng Chen, Li-Chen Fu, Ming-Hung Lin, and An-Chih Huang. *Petri-Net and GA-Based Approach to Modeling, Scheduling, and Performance Evaluation for Wafer Fabrication.* IEEE Transactions on Robotics and Automation, Vol. 17, No. 5, October 2001.

*[14] James T. Lin and Chia-Chu Lee. A Petri net-based integrated control and scheduling scheme for flexible manufacturing cells. Computer Integrated Manufacturing Systems. Vol 10. No. 2, pp. 109-122, 1997*

*[15]* Christian Artigues and François Roubellat. *A Petri net model and a general method for on and of-line multi-resource shop floor scheduling with setup times.* Int. J. Production Economics 74 (2001) 63-75

[16] Miguel A. Gutierrez, Sergio G. De los Cobos and Blanca R. Perez. Revista enLinea Número 1 , Junio 1997. Consultado en Abril 23 de 2005 http://www.azc.uam.mx/publicaciones/enlinea2/1.htm

[17] Juan Pablo Caballero and Gonzlo Mejía. A hybrid approach that combines Petri Net models and Genetic algorithms for scheduling of flexible manufacturing systems. III Congreso Colombiano y I Conferencia Andina Internacional de Investigación de Operaciones

*[18]* Mu Der Jeng and Shih Chang Chen. *Heuristic Search Based on Petri Net Structures for FMS Scheduling.* IEEE Transactions on Industry Applications. Vol. 35, No. 1, January/February 1999

*[19*] S.I. Han, I. Muta, T. Hoshino, T. Nakamura and N. Maki. Optimal design of superconducting generator using genetic algorithm and simulated annealing. IEE Proc.-Electr. Power Appl., Vol. 151, No. 5, September 2004

*[20]* P. K. Hota, R. Chakrabarti and P. K. Chattopadhyay. *A Simulated Annealing-Based Goal-Attainment Method for Economic Emission Load Dispatch with Nonsmooth Fuel Cost and Emission Level Functions.* Electric Machines and Power Systems, 28:1037–1051, 2000

*[21]* B. J. Reynolds and S. Azarm. *A multi-objective heuristic-based hybrid genetic algorithm.* Mechanics of Structures and Machines Vol. 30, No. 4, pp. 463–491, 2002

*[22]* Balram Suman. *Simultaed Annealing-based multiobjective algorithms and their application for system reliability. Eng. Opt.,* Vol. 35, No. 4, August 2003, pp. 391–416

[23] R. Bryan Kethley and Bahram Alidaee. Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. Computers & Industrial Engineering 43 (2002) 509–528

[24] In Lee. Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs. Computers & Operations Research 28 (2001) 835-852

[25] Roberto Spina, Luigi M. Galantucci and Michele Dassisti. A hybrid approach to the single line scheduling problem with multiple products and sequence-dependent time. Computers & Industrial Engineering 45 (2003) 573–583

[26] Ling Wang, Liang Zhang and Da-Zhong Zheng. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. Computers & Operations Research.

[27] Chichang Jou. A genetic algorithm with sub-indexed partitioning genes and its application to production scheduling of parallel machines. Computers & Industrial Engineering 48 (2005) 39–54.

[28] Paulo M. França, Jatinder N.D. Gupta, Alexandre S. Mendes, Pablo Moscato and Klaas J. Veltink. Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. Computers & Industrial Engineering 48 (2005) 491–506

[29] R. Tavakkoli-Moghaddam and M. Daneshmand-Mehr. A computer simulation model for job shop scheduling problems minimizing makespan. Computers & Industrial Engineering 48 (2005) 811–823

[30] Masato Watanabe, Kenichi Ida and Mitsuo Gen. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Computers & Industrial Engineering 48 (2005) 743–752

*[31] Hervé P. Hillion and Jean-Marie Proth. Using Timed Petri Nets for the scheduling of job-shop systems. Engineering costs and Production economics, 17(1989) 149-154*

[32] Gonzalo Mejía and Nicholas g. Odrey. Petri Net Models and Heuristic Search for Scheduling of Maufacturing Systems: A Comparative Study.

[33] Tadao Murata. Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, No. 4, April 1989.

[34] Liu, C.S., Ma, Y.H., y Odrey, N.G. Hierarchical Petri Net Modeling for System Dynamics and Control of Manufacturing Systems. International Conference on Flexible Automation Intelligent Manufacturing (FAIM'97), Sponsored by University of Teesside, Middlesbrough, UK, June 25-27, 1997.

[35] Montgomery, Douglas C. Diseño y análisis de experimentos. Limusa Wiley. Mexico 2002.

[36] Mitchel, Melanie. An Introduction to genetic algorithms. Cambbridge, Mass.; London: MIT Press 1999