# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

## PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

# UN ALGORITMO HÍBRIDO PARA UN PROBLEMA DE HORARIOS CON RESTRICCIONES ESPECIALES

# T E S I S

QUE PARA OPTAR POR EL GRADO DE:

## MAESTRO EN INGENIERÍA

INGENIERÍA DE SISTEMAS- INVESTIGACIÓN DE OPERACIONES

P R E S E N T A :


## CARLOS PÉREZ DE LA CRUZ


TUTOR:
**RAMÍREZ RODRÍGUEZ JAVIER**

2011

## DEDICATORIA

**A mi esposa, padres y hermanos.**

**A mi tutor Dr. Javier Ramírez Rodríguez.**

**Resumen:**

El presente trabajo expone una metaheurística híbrida que permite encontrar buenas soluciones para un problema de horarios con restricciones especiales, el cual se modela como un Problema de Coloración Robusta Generalizado (PCRG).

El algoritmo híbrido (Algoritmo genético y Búsqueda de vecindad variable reducida) desarrollado para el Problema de Coloración Robusta Generalizado halla mejores soluciones para algunos casos reportados en Lara 2010 [2] y como el algoritmo genético y caso reportados en Ramírez 2001 [1] encuentra la solución óptima; además se presentan casos de mayor tamaño siguiendo la metodología en Lara 2010 [2] y sus soluciones. En la sección 2 se presenta el marco teórico, se recuerdan algunas definiciones Ramírez 2001 [1] y el marco de referencia; en la Sección 3 se exhibe la metodología utilizada para abordar el problema; en la Sección 4 se presenta un problema simplificado y se describe el algoritmo híbrido para el PCRG; en la Sección 5 se presentan resultados computacionales y se demuestra que la solución encontrada para algunos casos es óptima; en la Sección 6 se realizan algunos experimentos computacionales; finalmente en la sección 7 se presentan algunas conclusiones.

**Palabras clave:** Metaheurística, optimización combinatoria, problemas de horarios.

**AMS Subject Classification:** 05C15, 90C59, 68T20

# Contenido

# 1. Introducción

La asignación de horarios es un problema muy común en la planeación de actividades en todas las empresas, poder modelar y resolver estos problemas de una manera precisa permite lograr mejoras significativas en los procesos. Existen modelos en los cuales se ven reflejadas restricciones tales como que dos o más procesos o actividades no puedan compartir un recurso, muchos de estos modelos no contemplan restricciones en cuanto a qué tan lejos en el tiempo deben estar estas actividades.

En Ramírez 2001 [1] se introdujo el problema de coloración robusta generalizado (PCRG), con el cual se pueden modelar problemas de horarios que consideran restricciones del tipo: dos eventos no pueden realizarse a la misma hora y debe haber al menos *d* días entre dos eventos. Se demostró que el problema es NP- Difícil, por lo que es necesario utilizar métodos aproximados para encontrar buenas soluciones, en Lara 2010 [2] se presenta un procedimiento glotón aleatorizado, GRASP por sus siglas en inglés.

Una metaheurística es una buena opción si proporciona con una alta probabilidad, iguales o mejores soluciones a las ya obtenidas con otros métodos, en un tiempo adecuado según el tipo problema. En este trabajo se presenta un algoritmo híbrido para el PCRG que cuenta con las características anteriores.

En la sección 2 se presenta el marco teórico, se recuerdan algunas definiciones Ramírez 2001 [1] y el marco de referencia; en la Sección 3 se exhibe la metodología utilizada para abordar el problema; en la Sección 4 se presenta un problema simplificado y se describe el algoritmo híbrido para el PCRG; en la Sección 5 se presentan resultados computacionales y se demuestra que la solución encontrada para algunos casos es óptima; en la Sección 6 se realizan algunos experimentos computacionales; finalmente en la sección 7 se presentan algunas conclusiones.

## *Objetivos*

El objetivo principal de este trabajo es desarrollar un algoritmo híbrido que combine las ventajas de dos de las metaheurísticas más utilizadas actualmente (los algoritmos genéticos y los algoritmos de búsqueda en vecindades variables), para un problema de horarios que es modelado como un PCRG.

Un objetivo secundario es demostrar qué el algoritmo aquí desarrollado es una buena opción para tratar estos tipos de problemas de horarios y cualquier otro que se pueda modelar como un PCRG.

## 2. Marco Teórico

### *Optimización combinatoria*

En matemáticas existe una amplia variedad de problemas que son considerados difíciles de resolver debido a la dificultad de escoger la mejor solución de entre un gran número de soluciones posibles o debido al hecho de que simplemente encontrar una solución de estos problemas es bastante tardado.

Dentro de este tipo de problemas se encuentran los que tratan la Optimización Combinatoria como una rama de la Programación Matemática.

Una definición de Programación Matemática y de Optimización Combinatoria dada por Salazar 2001 [3] es:

"La Programación Matemática es una parte de la Matemática Aplicada que trata de resolver problemas de decisión en los que se deben determinar acciones que optimicen un determinado objetivo, pero satisfaciendo ciertas limitaciones en los recursos disponibles. "

"La optimización combinatoria es una parte de la programación matemática que estudia los problemas $min\{f(x): x \in S\}$

Siendo $|S| < \alpha$, es decir la Optimización Combinatoria trata de desarrollar algoritmos para afrontar problemas de optimización caracterizados por tener un número finito de soluciones factibles. Aunque por definición puede parecer que la "simple" enumeración de las soluciones de $S$ es suficiente para resolverlos, esta idea no es razonable en la práctica ya que el número $|S|$ pueden ser exageradamente enorme. Por otra parte, particularidades del conjunto $S$ permiten en ocasiones diseñar algoritmos efectivos que permiten determinar (con tiempo de cálculos razonables) alguna solución al problema. "

Los problemas combinatorios anteriores se pueden plantear como un modelo matemático consistente en un sistema de ecuaciones y expresiones matemáticas relacionadas que describen la esencia del problema con los siguientes componentes:

- ✓ Una serie de decisiones cuantificables, relacionadas unas con otras, llamadas variables de decisión.
- ✓ Una función objetivo que mide la efectividad de cada sistema de decisiones.
- ✓ Una serie de restricciones sobre las decisiones a tomar que delimitan el conjunto de soluciones posibles.

## Resolución de Problemas Combinatorios

Existen muchos métodos que se han aplicado he inventado con el paso del tiempo para poder resolver los problemas combinatorios, muchos de ellos, dada su ejecución, aseguran encontrar la solución óptima, estos métodos caen dentro de la categoría de métodos exactos.

Los métodos exactos proporcionan el óptimo de los problemas, pero son costosos en términos de tiempo ya que su ejecución podría llevar varios años para problemas grandes aun en las computadoras más recientes. Un método heurístico o aproximado proporciona una buena solución del problema no necesariamente óptima.

Una definición dada por De Werra [4] de un método heurístico es:

"Un método heurístico es un procedimiento para resolver un problema matemático bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución"

El encontrar una aproximación, y no el óptimo forzosamente, permite reducir considerablemente el tiempo que llevaría ejecutar un algoritmo de resolución, esta es la idea con la que Reeves [5] define método heurístico:

"Un heurístico es una técnica que busca buenas soluciones con un tiempo de computación razonable sin garantizar la optimalidad"

Las dos definiciones anteriores de método heurístico nos permiten ver cuáles son las características deseables para estos:

- ✓ Bueno: que proporcione soluciones mejores a las ya obtenidas con otros métodos o que se pueda probar que se encuentran no muy lejos del óptimo.
- ✓ Robusto: que cada vez que se aplique proporcione con una alta probabilidad buenas soluciones.
- ✓ Eficiente: que su ejecución lleve un tiempo relativamente adecuado.

Para calificar un algoritmo como bueno se pueden hacer diferentes comparaciones de la solución que proporciona con:

- ✓ La solución óptima (en caso de que se conozca para algún problema)
- ✓ Con una cota
- ✓ Con la solución de un método exacto truncado
- ✓ Con la solución de otros heurísticos

- ✓ Con un análisis del peor caso

La robustez del algoritmo se pone a prueba al ser corrido varias veces y notando en qué proporción nos arroja buenas soluciones.

Una manera de medir lo eficiente que es un algoritmo para así poderlo comparar con otros es mediante su tasa u orden de crecimiento, estudiada por una rama de la teoría de la computación llamada complejidad computacional. Varias ideas y párrafos del tema de complejidad computacional son tomados o resumidos del trabajo de Clara María Segura [6].

## *Complejidad computacional*

La complejidad computacional estudia de manera teórica los recursos requeridos durante el cómputo de un algoritmo. Una función de trabajo, $f(n)$, describe el comportamiento del algoritmo ya que es una función que describe cuánto tiempo de procesamiento o espacio en disco o memoria utiliza el procedimiento para llegar a una solución en función de los parámetros de entrada.

El orden de esta función de trabajo es la forma en cómo la función varía en magnitud a medida que sus parámetros tienden a un límite.

Ejemplo 1:

Buscar en un diccionario tiene complejidad logarítmica. Se puede iniciar la búsqueda de una palabra por la mitad del diccionario. Inmediatamente se sabe si se ha encontrado la palabra o, en el caso contrario, en cuál de las dos mitades hay que repetir el proceso (es un proceso recursivo) hasta llegar al resultado. En cada (sub)búsqueda el problema (las páginas en las que la palabra puede estar) se ha reducido a la mitad, lo que se corresponde con la función logarítmica. Este procedimiento de búsqueda (conocido como búsqueda binaria) en una estructura ordenada tiene complejidad logarítmica $O\ (ln\ n)$.

Ejemplo 2:

El proceso más común para ordenar un conjunto de elementos tiene complejidad cuadrática. El procedimiento consiste en crear una colección vacía de elementos. A ella se añade, en orden, el menor elemento del conjunto original que aún no haya sido elegido, lo que implica hacer un recorrido completo del conjunto original, $O(n)$, siendo $n$ el número de elementos del conjunto). Este recorrido sobre el conjunto original se realiza hasta que

todos sus elementos están en la secuencia de resultado. Se puede ver que hay que hacer n selecciones (se ordena todo el conjunto) cada una con un costo $n$ de ejecución: el procedimiento es de orden cuadrático $O(n^2)$. Hay que aclarar que hay diversos algoritmos de ordenación con mejores resultados.

## Problemas P, NP

La clase Polinomial o clase P consiste de todos aquellos problemas de decisión que pueden ser resueltos en una máquina determinística secuencial en un período de tiempo polinomial en proporción a los datos de entrada. En la teoría de complejidad computacional, la clase P es una de las más importantes.

Cualquier problema para el que hayamos encontrado un algoritmo polinomial es un problema de la clase P.

Por ejemplo:

Ordenación: $O(n\ log\ n)$.

Búsqueda en un vector ordenado: $O(log\ n)$.

Multiplicación encadenada de matrices: $O(n^3)$.

Problema de los caminos más cortos: $O(n^3)$.

La clase NP consiste de todos aquellos problemas de decisión cuyas soluciones positivas/afirmativas pueden ser verificadas en tiempo polinomial a partir de ser alimentadas con la información apropiada, o en forma equivalente, cuya solución puede ser hallada en tiempo polinomial en una máquina no-determínistica.

Cualquier problema de optimización tiene un problema de decisión correspondiente

Ejemplo:

Problema del agente viajero: Dado un gráfica dirigida y ponderada, el problema de optimización del agente viajero consiste en encontrar un circuito de costo mínimo que empiece en un vértice, acabe en ese vértice, y visite al resto de vértices exactamente una vez.

El problema de decisión del agente viajero consiste en determinar, dado un valor positivo $d$, si la gráfica tiene un circuito de costo no mayor que $d$.

El problema de decisión correspondiente no es más difícil que el de optimización.

Un algoritmo no determinista está compuesto por dos fases:

- ✓ Fase de adivinación (no determinista): Dada una instancia del problema, produce una salida S que puede entenderse como una supuesta solución.
- ✓ Fase de verificación (determinista): Dada la instancia y la salida S, y procediendo de forma determinista, o devuelve cierto, lo cual significa que se ha verificado que la respuesta para esta instancia es "sí", o devuelve falso.

Un algoritmo no determinista de tiempo polinomial es un algoritmo no determinista cuya fase de verificación es un algoritmo de tiempo polinomial.

Entonces NP es el conjunto de todos los problemas de decisión que pueden ser "resueltos" por algoritmos no deterministas de tiempo polinomial.

Para los problemas de la clase P hay algoritmos no deterministas que los "resuelven". Es por esto que la clase P es un subconjunto de NP, aunque no se sabe si P=NP.

La palabra "resolver" en las 2 oraciones anteriores podría interpretarse como verificar en este contexto para una salida proveniente de la fase de adivinación.

Continuando con el ejemplo del problema del agente viajero, esté se encuentra en NP ya que existe un algoritmo que hace la verificación en tiempo polinomial, aunque esto no quiere decir que haya algoritmos de tiempo polinomial que lo resuelvan en el sentido estricto de la palabra.

## NP-Completos y NP-Difíciles

Un problema B es NP-Completo si

- ✓ Está en NP
- ✓ Para cualquier otro problema A en NP, A se puede reducir polinomialmente a B. Esto significa que existe un algoritmo polinomial que construye una instancia "y" de B para toda instancia "x" de A de tal forma que un algoritmo para B responde "sí" para "y" si y sólo si la respuesta al problema A para "x" es "sí".

Si pudiéramos mostrar que un problema NP-Completo cualquiera está en P, podríamos concluir que P = NP.

Un problema NP-Completo tiene la propiedad de que puede ser resuelto en tiempo polinomial si y sólo si todos los problemas NP-Completos pueden ser resueltos en tiempo polinomial.

Un problema B es NP Difícil si

Para algún problema NP-Completo A, A puede resolverse en tiempo polinomial utilizando un algoritmo polinomial hipotético para el problema B.

Todo problema NP-Completo es NP-Difícil. Los problemas de optimización correspondientes a problemas NP-Completos son NP-Difíciles.

Si un problema NP-Difícil puede ser resuelto en tiempo polinomial, entonces todos los problemas NP-Completos pueden ser resueltos en tiempo polinomial.

## Problemas intratables

Un problema es intratable si no se puede resolver en tiempo polinomial. (Es una propiedad del problema.) (Algunos problemas de NP pueden ser intratables (NP Completos) pero no se ha demostrado)

Problemas para los que se ha demostrado su intratabilidad. Dos tipos:

- ✓ Problemas que requieren una cantidad no polinomial de salidas.
  Determinar todos los ciclos Hamiltonianos de una gráfica. Podría haber $(n-1)!$ ciclos.
- ✓ Problemas que no requieren salida no polinómica pero podemos probar que no se pueden resolver en tiempo polinomial. Por extraño que parezca, se han encontrado pocos problemas de este tipo.
  Problemas indecidibles: no existe algoritmo que los resuelva. Problema de parada.
  Problemas decidibles: generalmente construidos “artificialmente”.

***Definiciones:***

## Problema de Coloración Robusta (PCR)

De Ramírez 2001 [1]. Dada una gráfica $G = (V, A)$, donde $V$ son los vértices del grafo $G$ y $A$ las aristas, con $|V| = n$ y $k > 0$ entero y una matriz de penalizaciones $P = \{p_{ij}, (i,j) \in \bar{A}\}$, el PCR busca construir una coloración válida C: $V \rightarrow \{1,2,\dots,k\}$, donde $k$ no es necesariamente el mínimo, *i.e.*, el número cromático, que cuente con el menor grado de rigidez $R(C)$, entendiéndose por rigidez de la coloración $C$ a la suma de las penalizaciones de las aristas complementarias cuyos extremos están igualmente coloreados, el objetivo es:

$$min R(C) = \sum_{\{i,j\} \in \bar{A}, C(i)=C(j)} p_{ij}$$

s.a $\quad C(i) \neq C(j) \quad \forall \{i,j\} \in A$

Distintas penalizaciones de las aristas complementarias permiten obtener coloraciones válidas con diferentes propiedades. Por ejemplo se podrían hacer que ciertos vértices en la medida de la posible no tuvieran el mismo color (restricción blanda), etc.

## $k$-distancia $d$

De Lara 2010 [2]. Dada una $k$-coloración $C$ sobre una gráfica $G = (V, A)$, se define una $k$-distancia $d$ como una distancia en el conjunto de colores $\{1,2,\dots,k\}$

Un ejemplo de c-distancia es la trivial:

$$d^0(c,c') = \begin{cases} 0 & c = c' \\ 1 & c \neq c' \end{cases} \quad \forall c, c' \in \{1,2,\dots,k\}$$

Otra $k$-distancia $d$ puede ser la basada en el valor absoluto de la diferencia de los colores:

$$d^1(c,c') = |c - c'| \quad \forall c, c' \in \{1,2,\dots,k\}$$

Esta última definición es la que se usará de ahora en adelante en este trabajo.

## Problema de Coloración Robusta Generalizado (PCRG)

De Lara 2010 [2]. El PCRG para el problema tratado en este trabajo, busca construir una coloración válida con un número fijo de colores (no necesariamente el mínimo) que cuente con el menor número violaciones a restricciones del tipo "debe haber al menos $d$ días entre dos eventos de un mismo tipo". Para esto se define el $\bar{d}$-grado de rigidez generalizado de la $k$-coloración $(C)$, siendo $\bar{d} \geq 0$, como la suma de las penalizaciones de las aristas complementarias cuyos extremos están pintados con colores que tienen una distancia menor o igual $\bar{d}$.

$$R^{\bar{d}}(C) = \sum_{\{i,j\} \in \bar{A}, d(C(i),C(j)) \leq \bar{d}} p_{ij}$$

s.a $\quad C(i) \neq C(j) \quad \forall \{i,j\} \in A$

Al igual que el PCR, lo que se busca es encontrar la coloración con menor grado de rigidez Generalizado.

# *Marco de Referencia*

## Problemas de horarios

Un problema de horarios es según Burke 2004[7] un problema con cuatro parámetros: T, una colección finita de periodos de tiempo; R, una colección finita de recursos; M, una colección finita de reuniones y C, una colección finita de restricciones. El problema consiste en asignar, sujeto a limitaciones, los recursos disponibles a objetos colocados en el tiempo de manera que se satisfagan lo más posible el grupo de objetivos deseados.

## Problema de horarios en Universidades

A través de décadas se han desarrollado varios modelos para resolver problemas de horarios para universidades, como los clasifica Burke 1997[8] estos problemas pueden ser divididos en dos clases; de horarios de curso y de horarios de evaluaciones. Lewis 2007[9]

describe las principales diferencias entre ambos: en los problemas de horarios de evaluaciones, múltiples eventos pueden ser programados en el mismo salón en el mismo periodo de tiempo, sujeto a las restricciones de cupo, mientras que en los problemas de horarios de curso se tiene permitido, por lo general, un solo evento por salón por periodo de tiempo. Otra diferencia es que en los problemas de horarios de curso se tiene un número fijo de periodos de tiempo mientras que en problemas de horarios de evaluaciones esto es más flexible por lo general.

Diversos modelos se han propuesto para cada una de las variantes de estos problemas, del mismo modo que se han propuesto métodos ya sean deterministas o heurísticos, presentando ventajas cada uno en determinadas instancias; tamaños y tipos de problemas. Un resumen de estos trabajos puede ser encontrado en Carter 1996 [10], Carter 1998 [11] y en Lewis 2007 [9].

El problema tratado en este trabajo cae dentro de los problemas de programación de curso. El método para abordarlo puede ser considerado como un algoritmo meta heurístico, en particular un algoritmo memético según lo describe Burke 2002 [12] ya que incorpora elementos de búsqueda en un algoritmo genético, y como un algoritmo de dos etapas según Lewis 2007 [9] ya que se busca en cada generación producir hijos que cumplan con las restricciones fuertes para después mejorarlos de acuerdo al número de restricciones suaves que se cumplen.

Como restricciones fuertes en este trabajo tenemos a las de tipo: “dos eventos no pueden realizarse a la misma hora” y como restricciones suaves a las del tipo “debe haber al menos $d$ días entre dos eventos”

En el algoritmo presentado se permitió relajar el número de salones disponibles por periodo de tiempo por lo que según Lewis 2007 [9] este algoritmo podría ser clasificado también como un algoritmo que permite relajaciones.

El problema de horarios específico tratado en esta tesis y algunas de sus instancias también han sido abordados por Ramírez 2001 [1] y Lara 2010 [2]. En ambos trabajos se modela el problema de horarios con el Problema de Coloración Robusta Generalizado, en el primero se utiliza un algoritmo genético voraz para tratar de resolverlo y en el segundo se utiliza un GRASP.

En esta tesis se hace uso también del PCRG para modelar el problema de horarios específico antes mencionado, por lo que desarrollar un algoritmo que encuentre buenas soluciones permite tratar además cualquier problema que pueda ser visto como un Problema de Coloración Robusta Generalizado (más adelante se dan ejemplos de estos).

## Otros problemas que pueden modelarse como un PCRG

Diversos problemas además del problema de horarios tratado en esta tesis pueden ser modelados como un PCRG. Por lo que el algoritmo desarrollado en este trabajo también se puede utilizar para encontrar soluciones a estos problemas.

T- Coloración

Sea $G$ un grafo y $T$ un conjunto de enteros no negativos. Una T-coloración de $G$ es una asignación de un entero positivo $f(i)$ a cada vértice $i$ de $G$ tal que si $i$ y $j$ están unidos por una arista de $G$, entonces $|f(i) - f(j)|$ no está en $T$. Roberts [13] comenta que las T-Coloraciones fueron introducidas por Hale con el problema de asignación de canales en comunicaciones. Este problema puede ser abordado modelándolo también como un PCRG, donde las aristas de la T-coloración pasarían a ser las aristas complementarias para el PCRG y $T$, el conjunto de enteros no negativos en la T-coloración, se transformaría en el vector $\bar{d}$ del PCRG.

Problema de mantenimiento de Flotas

Los vértices son los vehículos que llegan para mantenimiento regular y hay una arista entre dos vehículos si son programados en un puesto de mantenimiento en periodos de tiempo que se intersecan. Se busca asignar en el puesto de mantenimiento un periodo de tiempo $L(i)$ a cada vehículo $i$, de tal manera que los vehículos tengan asignados intervalos que no se traslapen. De nueva cuenta este problema puede ser visto como un PCRG, donde se buscaría minimizar el número de violaciones a restricciones del tipo “debe de haber al menos d periodos de tiempo entre el mantenimiento de vehículos del mismo tipo, con alta demanda o escasos”

Las aristas complementarias del PCRG no sólo pueden modelar restricciones sobre asignaciones temporales próximas, si no cualquier otro tipo, donde la distancia entre colores tenga una interpretación valiosa (como en la T- coloración).

## Complejidad algorítmica del Problema

Como se menciona en Ramírez 2001 [1] y Lara 2010 [2] el PCRG es una generalización del PCR y por tanto es también un problema NP-Difícil, hecho que justifica la realización de algoritmos heurísticos para obtener soluciones satisfactorias en determinadas instancias.

# 3. Metodología

## *Algoritmos genéticos*

### Antecedentes históricos

A mediados del siglo XVIII se desarrollan las primeras ideas evolucionistas en contra de la tesis creacionista (teoría de la creación de todo ser viviente por Dios). Jean Baptiste Lamarck destaca la importancia de la naturaleza en los cambios de las especies. El Lamarckismo sostenía que los cambios en características físicas sufridos por un organismo en su vida podían ser transmitidos a los descendientes de los organismos.

En el siglo XIX August Weismann desarrolla la teoría del germoplasma, contrapuesta al Lamarckismo sostiene la existencia de las células germinales, las cuales trasmiten la información hereditaria, y de las células somáticas que no trasmiten esta información. Según Weismann la selección natural es el único mecanismo que puede cambiar el germoplasma y que el germoplasma y el ambiente pueden influir en el somatoplasma no así el somatoplasma en el germoplasma.

Charles Darwin y Alfred Russell Wallace desarrollan de manera paralela su propia teoría evolutiva. El origen de las especies (1859) tiene un gran éxito, se sostiene que la evolución se origina a través de cambios aleatorios de características hereditarias, combinados con un proceso de selección natural. Las especies sin cambios son incompatibles con su ambiente ya que este tiende a cambiar con el tiempo. De generación en generación, los nuevos individuos se vuelven más aptos para sobrevivir (aprendizaje).

También a finales del siglo XIX Gregor Mendel formuló una nueva teoría de la herencia expresada en lo que luego se llamaría “Leyes de Mendel”. La teoría evolutiva propuesta originalmente por Charles Darwin, combinada con el seleccionismo de Weismann y la genética de Mendel dieron lugar al paradigma Neo-Darwiniano. Según el Neo-Darwinismo la historia de la mayoría de la vida puede ser explicada a través de un puñado de procesos estadísticos que actúan sobre y dentro de las poblaciones y especies

- ✓ la reproducción
- ✓ la mutación
- ✓ la competencia
- ✓ la selección

Desde la década de 1930 la evolución natural fue vista como un proceso de aprendizaje:

- ✓ W.D. Cannon, The Wisdom of the Body: el proceso evolutivo es similar al aprendizaje por ensayo y error
- ✓ A.M. Turing: existe una conexión “obvia” entre la evolución y el aprendizaje de máquina

A finales de la década de 1950 y principios de la década de 1960 biólogos evolutivos hicieron simulaciones de la evolución de sistemas bilógicos en computadoras digitales, se hacía uso de operadores que se han vuelto clásicos hoy en día como son la representación binaria, los mecanismos de selección y de recombinación. En 1962, investigadores como G.E.P. Box, G.J. Friedman, W.W. Bledsoe y H.J. Bremermann habían desarrollado independientemente algoritmos inspirados en la evolución para optimización de funciones y aprendizaje automático, pero sus trabajos generaron poca reacción. En 1965 surgió un desarrollo más exitoso, cuando Ingo Rechenberg, entonces de la Universidad Técnica de Berlín, introdujo una técnica que llamó estrategia evolutiva, aunque se parecía más a los trepa colinas que a los algoritmos genéticos. En esta técnica no había población ni cruzamiento; un padre mutaba para producir un descendiente, y se conservaba el mejor de los dos, convirtiéndose en el padre de la siguiente ronda de mutación. Versiones posteriores introdujeron la idea de población. Las estrategias evolutivas todavía se emplean hoy en día por ingenieros y científicos, sobre todo en Alemania.

El siguiente desarrollo importante en el campo vino en 1966, cuando L.J. Fogel, A.J. Owens y M.J. Walsh introdujeron en América una técnica llamada programación evolutiva. En este método, las soluciones candidatas para los problemas se representaban como máquinas de estado finito sencillas; al igual que en la estrategia evolutiva de Rechenberg, su algoritmo funcionaba mutando aleatoriamente una de estas máquinas simuladas y conservando la mejor de las dos. También al igual que las estrategias evolutivas, hoy en día existe una formulación más amplia de la técnica de programación evolutiva que todavía es un área de investigación en curso. Sin embargo, lo que todavía faltaba en estas dos metodologías era el reconocimiento de la importancia del cruzamiento.

En una fecha tan temprana como 1962, el trabajo de John Holland sobre sistemas adaptativos estableció las bases para desarrollos posteriores; y lo que es más importante, Holland fue también el primero en proponer explícitamente el cruzamiento y otros operadores de recombinación. Sin embargo, el trabajo fundamental en el campo de los algoritmos genéticos apareció en 1975, con la publicación del libro ``Adaptación en Sistemas Naturales y Artificiales''. Basado en investigaciones y publicaciones anteriores del propio Holland y de colegas de la Universidad de Michigan, este libro fue el primero en presentar sistemática y rigurosamente el concepto de sistemas digitales adaptativos utilizando la mutación, la selección y el cruzamiento, simulando el proceso de la evolución

biológica como estrategia para resolver problemas. El libro también intentó colocar los algoritmos genéticos sobre una base teórica firme introduciendo el concepto de esquema. Ese mismo año, la importante tesis de Kenneth De Jong estableció el potencial de los AGs demostrando que podían desenvolverse bien en una gran variedad de funciones de prueba, incluyendo paisajes de búsqueda ruidosos, discontinuos y multimodales.

Estos trabajos fundacionales establecieron un interés más generalizado en la computación evolutiva. Entre principios y mediados de los 80, los algoritmos genéticos se estaban aplicando en una amplia variedad de áreas, desde problemas matemáticos abstractos como el "problema de la mochila" (bin-packing) y la coloración de gráficas hasta asuntos tangibles de ingeniería como el control de flujo en una línea de ensamble, reconocimiento y clasificación de patrones y optimización estructural.

| Algoritmo genético | Programación Evolutiva | Estrategias Evolutivas |
|---|---|---|
| Selección estocástica | Selección estocástica | Selección determinista |
| Nivel genotípico (representación binaria) | Nivel fenotípico (no hay codificación) | Nivel fenotípico (no hay codificación) |
| Cruza = operador principal | No hay recombinación | Recombinación= operador secundario |
| Mutación= operador secundario | Mutación= único operador | Mutación= operador principal |

## Componentes de un algoritmo genético

Ya que los algoritmos genéticos están basados en la evolución se puede hablar de una metáfora evolutiva en cada uno de sus componentes:

- ✓ Población: Contiene (la representación de) los individuos, el tamaño de esta puede ser fijo o variable de acuerdo según el diseño.
- ✓ Representación: Vínculo entre el espacio original del problema (fenotipo) y el espacio de resolución (genotipo). Ejemplo de representaciones: binaria, real (floating-point), por árboles, etc.
- ✓ Función de evaluación/aptitud: Representa la adaptación de un individuo a su entorno, su papel es asignar una medida de calidad calculada en base a la F.O. del problema.

- ✓ Mecanismo de selección de padres: Regularmente, se seleccionan los mejores individuos probabilísticamente aunque debe existir un balance entre intensificación/exploración
- ✓ Operadores de variación / genéticos: Mutación y Cruce. La mutación es un operador unario que causa pequeñas alteraciones a un genotipo dando como resultado un mutante que de acuerdo al diseño puede o no ser incluido obligatoriamente en los individuos de la población en la siguiente generación, su papel es mantener la diversidad en la población y conectar el espacio de búsqueda. El cruce es un operador generalmente binario (pueden haber más de dos padres) que da como resultado 1 o dos hijos, su papel es la recombinación de rasgos deseables presentes en los padres.
- ✓ Mecanismo de selección ambiental y reemplazo: Son el conjunto de reglas (deterministas o estocásticas) relacionadas con la aptitud de los individuos para determinar que individuos estarán en la población en la siguiente generación.
- ✓ Técnica de inicialización: Se refiere a la manera de crear a la población inicial. Típicamente se hace de manera aleatoria.
- ✓ Criterio de paro: Criterio para detener el algoritmo. Puede ser por la precisión alcanzada, por el número de iteraciones, por un tiempo límite o por la no-evolución (estancamiento).

## Etapas de un algoritmo genético

El algoritmo genético enfatiza la importancia de la cruza sexual (operador principal) sobre el de la mutación (operador secundario), y usa selección probabilística.

Un algoritmo genético básico consta de las siguientes etapas:

1. Generación de la población inicial.

De manera aleatoria se crean posibles soluciones (cromosomas) que en conjunto formarán la población inicial.

2. Evaluación de los elementos de la población.

Los elementos de la población son evaluados por medio de una función de aptitud.

3. Selección.

Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.

4. Producir una nueva generación

Se aplican a los elementos seleccionados los operadores genéticos de cruce y mutación y se actualiza la población.

## *Búsqueda de vecindades variables*

### Búsqueda local

Estructura de vecindad: Función $\aleph: \Omega \rightarrow 2^{\Omega}$, que define para cada $s \in \Omega$, un conjunto $\aleph(s) \subseteq \Omega$ de soluciones llamadas "vecinas" de $s$. El conjunto $\aleph(s)$ se llama "vecindad" de $s$ y cada elemento $s' \in \aleph(s)$ es una solución "vecina" de $s$.

La búsqueda local es una heurística que en la que dada una solución se obtienen sus "vecinos" por medio de una regla o función, se sustituye la solución por el mejor vecino si cumple el criterio de aceptación y se repiten los pasos anteriores mientras se satisfaga el criterio de continuación.

### Búsqueda de vecindad variable (Variable Neighbourhood Search (VNS))

Metaheurística para resolución de problemas de optimización combinatoria Propuesto por P.Hansen y N.Mladenovic en 1997.

La idea básica es ir cambiando en forma sistemática la vecindad al momento de realizar la búsqueda. La mayoría de los algoritmos de búsqueda local usan una sola vecindad.

VNS se basa en tres hechos simples:

- ✓ Un mínimo local con respecto a una estructura de vecindad no lo es necesariamente con respecto a otra.
- ✓ Un mínimo global es un mínimo local con respecto a todas las posibles estructuras de vecindades.
- ✓ En muchos problemas el mínimo local con respecto a una o varias estructuras de vecindad están relativamente cerca.

La última observación es empírica e implica que un mínimo local muchas veces nos dé información acerca del óptimo global.

Se pueden hacer tres estrategias: determinística, estocástica, y la combinación de las dos anteriores.

## Determinística: Búsqueda de vecindad variable descendente (Variable neighborhood descent (VND))

La idea es buscar sistemáticamente en diferentes esquemas de vecindad hasta llegar a un mínimo local que es mínimo con respecto a todos los esquemas de vecindad.

Sea $N_k$ un conjunto finito de vecindades predefinidas y $N_k(x)$ el conjunto de soluciones en la $k$-ésima vecindad de $x$.

Sea $N_k$, para $k = 1, \dots, k_{max}$ un conjunto de estructuras de vecindad a seguir en ese orden.

El algoritmo básico es el siguiente:

$k = 1$

*While* $k \leq k_{max}$

*Encontrar al mejor vecino* $x'$ *de* $x$ $(x' \in N_k(x))$.

*if "la solución obtenida* $x'$ *es mejor que la anterior* $(x)$*"*

$x = x'$

$k = 1$

*Else*

$k = k + 1$

*End if*

*Wend*

## Estocástica: Algoritmo de búsqueda de vecindad variable reducida (Reduced VNS (RVNS))

Es un método aleatorio. Es exactamente igual que la anterior, sólo que los vecinos en cada vecindad son seleccionados aleatoriamente.

## Estocástica y determinística: Búsqueda de vecindad variable básico (VNS)

La idea es seleccionar un punto aleatorio en cierta vecindad y luego aplicarle a ese punto búsqueda local. Como en los casos anteriores, si se encuentra una mejor solución, se reemplaza la solución actual y se empieza otra vez con el primer esquema de vecindad.

Sea $N_k$, para $k = 1, \dots, k_{max}$ un conjunto de estructuras de vecindad a usar en la búsqueda.

El algoritmo básico es el siguiente:

*Definir un criterio de paro*
$k = 1$
*While "criterio de paro no se cumpla"*
    $k = 1$
    *While* $k \leq k_{max}$
        *Elegir aleatoriamente un* $(x')$ *de la* $k$*-ésima vecindad de* $x$, $(x' \in N_k(x))$.
        *Aplicar algún método de búsqueda local usando a* $x'$ *como punto inicial (sea* $x''$ *la solución encontrada)*
        *if "la solución obtenida* $x'$ *es mejor que la anterior* $(x)$
            $x = x''$
            $k = 1$
        *Else*
            $k = k + 1$
        *End if*
    *Wend*
*Wend*

Existen diferentes variantes del algoritmo básico.

## 4. Desarrollo de un algoritmo híbrido para los problemas modelados como PCRG

El algoritmo desarrollado en este trabajo considera varias etapas. Las primeras corresponden a un algoritmo genético, y en la parte intermedia, después del cruce, es donde se introduce una búsqueda de vecindad variable reducida en los individuos generados en cada generación. Finalmente se actualiza la población y se repite el proceso a partir del cruce, hasta que se cumpla la condición de paro.

El siguiente ejemplo es presentado en Ramírez 2001 y en Lara 2010. Se toma también en este trabajo como caso de ejemplo con la finalidad de describir cada uno de los pasos de la metodología propuesta.

Se trata de planificar los cursos de un diplomado que consta de 15 materias distribuidas en tres módulos. Los estudiantes se inscribirán al módulo que sea de su interés y de manera obligatoria llevaran todas las clases de las materias de dicho módulo:

| CURSO | ASIGNATURAS | HORAS/SEMANA |
|---|---|---|
| I | A,B,C<br>K | 3<br>1 |
| II | D,E<br>F,G | 3<br>2 |
| III | H,I,J<br>L, M, N, 0 | 2<br>1 |

A la administración le toca planificar las clases en los horarios y salones disponibles de modo que; clases de la misma materia o del mismo módulo no se impartan a la misma hora y además deberá de haber al menos 2 días entre clases de la misma materia. Para este ejemplo, en cada día se cuenta con tres horarios diferentes y se debe procurar no utilizar más de dos salones.

Los días hábiles en la semana son: lunes, martes, miércoles, jueves y viernes.

### *Representación de la solución*

De acuerdo al ejemplo citado, y ya que cada materia tiene un número de clases que deben ser impartidas, a cada una de ellas se les puede etiquetar de la siguiente manera:

| Materia | Clase | Etiqueta |
|---|---|---|
| **A** | 1 | 1 |
| **A** | 2 | 2 |
| **A** | 3 | 3 |
| **B** | 1 | 4 |
| **B** | 2 | 5 |
| **B** | 3 | 6 |
| **.** | . | . |
| **.** | . | . |
| **.** | . | . |
| **O** | 1 | 30 |

A las horas disponibles también se les puede etiquetar de manera análoga:

| Día | Hora | Etiqueta |
|---|---|---|
| **Lunes** | 1 | 1 |
| **Lunes** | 2 | 2 |
| **Lunes** | 3 | 3 |
| **Martes** | 1 | 4 |
| **Martes** | 2 | 5 |
| **Martes** | 3 | 6 |
| **.** | . | . |
| **.** | . | . |
| **.** | . | . |
| **Viernes** | 3 | 15 |

La solución se representa asignándole a cada uno de los vértices (clases) un color que representa la hora en que serán impartidas.

Un ejemplo de una solución es:

| | |
|---|---|
| **1** | **3** |
| **2** | 9 |
| **3** | 15 |
| | |
| **.** | . |
| **.** | . |
| **30** | 1 |
| | |

En esta solución la clase de la materia A etiquetada con 1 se impartirá en la hora etiquetada con 3, esto es en la tercera hora del lunes, la clase de la materia A etiquetada con 2 se impartirá en la hora etiquetada con 9, esto es en la tercera hora del miércoles, la clase de la materia A etiquetada con 3 se impartirá en la hora etiquetada con 15, que corresponde a la tercera hora del viernes etc.

Si a dos o más clases se les asigna impartirse a la misma hora, aun sin saber cual es ésta, diremos que pertenecen al mismo conjunto. Por tamaño de conjunto se entenderá en este caso por el número de clases que lo conforman.

### *Creación de la población inicial*

Se crea una población inicial de tamaño 20, utilizando el siguiente procedimiento:

Se crea una solución que cumpla con la familia de restricciones del tipo “la clase x no puede darse a la misma hora que la clase y”, de manera aleatoria. Esto dará como resultado conjuntos que contienen clases que se impartirán a la misma hora. La familia de restricciones del tipo: “debe haber $d$ días entre clases de la misma materia” se corrige con el algoritmo.

Tomando el ejemplo de las 30 clases:

| | $a1$ | $a2$ | $a3$ | $b1$ | $b2$ | $b3$ | $c1$ | $c2$ | $c3$ | $k1$ | $d1$ | $d2$ | $d3$ | $e1$ | $e2$ | $e3$ | $f1$ | $f2$ | $g1$ | $g2$ | $h1$ | $h2$ | $i1$ | $i2$ | $j1$ | $j2$ | $l1$ | $m1$ | $n1$ | $o1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| $a1$ | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $a2$ | 2 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $a3$ | 3 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $b1$ | 4 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $b2$ | 5 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| $b3$ | 6 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| $c1$ | 7 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| $c2$ | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| $c3$ | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| $k1$ | 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | |
| $d1$ | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |
| $d2$ | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | |
| $d3$ | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | | |
| $e1$ | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| $e2$ | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | |
| $e3$ | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| $f1$ | 17 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | |
| $f2$ | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | |
| $g1$ | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | |
| $g2$ | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | |
| $h1$ | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |
| $h2$ | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | |
| $i1$ | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | |
| $i2$ | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | |
| $j1$ | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | |
| $j2$ | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | |
| $l1$ | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | |
| $m1$ | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| $n1$ | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| $o1$ | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

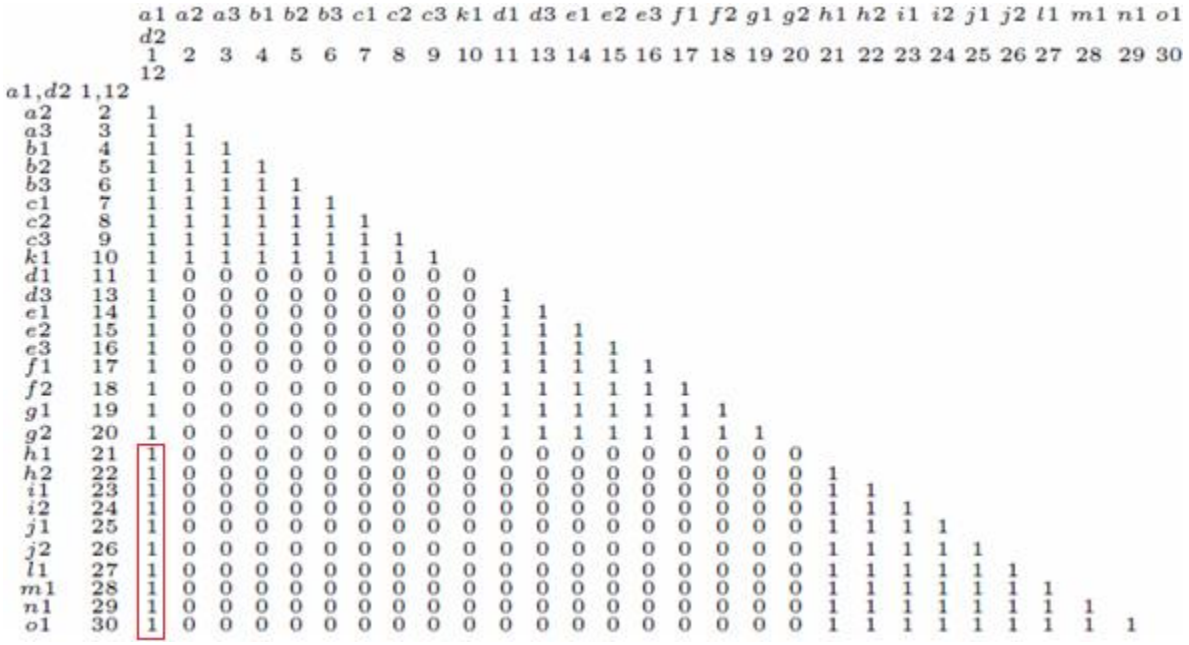
Los unos representan las restricciones de que dos conjuntos no se puedan unir, los ceros que si se puedan unir. Se elige al azar un cero, y se unen los conjuntos asociados en caso de que el conjunto resultante sea de tamaño menor o igual al máximo fijado, por ejemplo:

Supongamos que se elige al cero que relaciona la clase de la materia A etiquetada con 1 y la clase de la materia D etiquetada con 12. Se unirán dichos conjuntos y se agregarán las restricciones derivadas de esa unión.

| | | a1,d2 1,12 | a2 2 | a3 3 | b1 4 | b2 5 | b3 6 | c1 7 | c2 8 | c3 9 | k1 10 | d1 11 | d3 13 | e1 14 | e2 15 | e3 16 | f1 17 | f2 18 | g1 19 | g2 20 | h1 21 | h2 22 | i1 23 | i2 24 | j1 25 | j2 26 | l1 27 | m1 28 | n1 29 | o1 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a2 | 2 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a3 | 3 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b1 | 4 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b2 | 5 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| b3 | 6 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| c1 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| c2 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| c3 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| k1 | 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | |
| d1 | 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | | |
| d3 | 13 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | | | | | | | | |
| e1 | 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | | |
| e2 | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | | | | | | | | | | |
| e3 | 16 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | |
| f1 | 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| f2 | 18 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | |
| g1 | 19 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | |
| g2 | 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | |
| h1 | 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |
| h2 | 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | |
| i1 | 23 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | |
| i2 | 24 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | |
| j1 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | |
| j2 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | |
| l1 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | |
| m1 | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| n1 | 29 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| o1 | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

Si ese conjunto ya tiene el número máximo de elementos que se permiten (para conjuntos que forman las soluciones de la población inicial este número máximo es el doble del $\textit{número de clases por hora permitidas} = 4$), se añaden nuevas restricciones (unos) con el fin de evitar que se unan nuevos elementos a ese conjunto. En caso de que la unión de dos conjuntos de cómo resultado un conjunto de tamaño mayor al máximo fijado, se reparten al azar los elementos de este último conjunto de tal manera que se creara un conjunto que cuente con el número máximo de elementos fijado y otro conjunto que contendrá los elementos sobrantes. Para el conjunto con el número máximo de elementos se añaden unos con el fin de que no se puedan añadir más elementos.

Sólo en caso de que este número máximo se hubiera fijado como 2, los unos que se tendrían que agregar serían los encerrados con rojo en la figura de abajo:



| | a1 d2 | a2 | a3 | b1 | b2 | b3 | c1 | c2 | c3 | k1 | d1 | d3 | e1 | e2 | e3 | f1 | f2 | g1 | g2 | h1 | h2 | i1 | i2 | j1 | j2 | l1 | m1 | n1 | o1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 12 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| a1,d2 | 1,12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a2 | 2 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| a3 | 3 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| b1 | 4 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| b2 | 5 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| b3 | 6 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| c1 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| c2 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| c3 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | |
| k1 | 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | |
| d1 | 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | | |
| d3 | 13 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | | | | | | | |
| e1 | 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | | |
| e2 | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | | | | | | | | | |
| e3 | 16 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| f1 | 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | |
| f2 | 18 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | |
| g1 | 19 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | |
| g2 | 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | |
| h1 | 21 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | |
| h2 | 22 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | |
| i1 | 23 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | |
| i2 | 24 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | |
| j1 | 25 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | |
| j2 | 26 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | |
| l1 | 27 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | |
| m1 | 28 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| n1 | 29 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |
| o1 | 30 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

La elección de ceros, unión de conjuntos y actualización de unos se lleva a cabo hasta que ya no queda cero alguno por elegir.

### *Evaluación de la soluciones de la población inicial*

A los conjuntos, se les asigna al azar un número (que representa una hora) de tal manera que cada hora disponible quede asignada a un conjunto (si hay menos conjuntos que horas disponibles se crean conjuntos vacios). Se evalúa la solución de acuerdo al número de restricciones que son violadas (estas restricciones son del tipo x clase debe darse a m días de distancia o más de y clase).

Por ejemplo a un conjunto formado por alguna de las clases de la materia A (supongamos la etiquetada con 1) y alguna de las clases de la materia D (supongamos la etiquetada con 12), se les asigna al azar que se impartan a la quinta hora, esto es, las dos clases se impartirán a la segunda hora del martes: (1,5), (12,5)

A otro conjunto formado por alguna de las clases de la materia A (supongamos la etiquetada con 2) y la clase de la materia O etiquetada con 30 se les asigna al azar la novena hora, esto es, las dos clases de este conjunto se impartirán la tercera hora del miércoles: (2,9), (30,9).

Ya que clases de la misma materia deben de ser impartidas a dos días de distancia, el que se asigne impartir dos clases de la materia A en días consecutivos genera un violación al

tipo de restricciones “x clase debe de darse a m d días de distancia o más de y clase”. Por lo que una solución que presente esta asignación será penalizada en su evaluación, y así se irá penalizando a las soluciones por el número de restricciones que violen de este tipo. De esta forma la función de aptitud que evaluará qué tan buena es una solución será el grado de rigidez generalizado.

$$R^{\bar{d}}(C) = \sum_{\{i,j\} \in A, d(C(i),C(j)) \leq \bar{d}} p_{ij}$$

### *Búsqueda local en la población inicial*

Se evalúa la mejora de intercambiar la hora asignada de dos conjuntos cualquiera. Si hay mejora, se realiza el intercambio que resulta mejor evaluado y se actualiza la evaluación de la solución.

Se repite el procedimiento anterior hasta que no haya mejora posible con este k2 intercambio.

### *Operadores genéticos*

#### Cruce

Para generar nuevos elementos en la población se ordenan las soluciones de manera aleatoria (para tamaño de población 20 hay $20!$ formas de hacerlo), y ya que están ordenadas las soluciones se toman en forma creciente, de dos en dos, y para cada 1 de las 10 parejas se realiza lo siguiente:

Fase 1

Fase 1.1

Clases que tuvieron el mismo color en cada una de las soluciones (individuos), aunque no sea precisamente el mismo color en las dos soluciones, con una probabilidad grande se creara un conjunto en que el estas clases pertenezcan al mismo.

$$P = MID * PG$$

Si en los conjuntos a los que pertenecen las clases en los padres, en cada conjunto alguno de sus elementos genero violación $MID = .7$, en caso contrario $MID = 1$.

Las clases de la fase 1.1 para las que no se les creó un conjunto son eliminadas para la fase 1.2

Fase 1.2

Clases que tuvieran el mismo color en el padre mejor evaluado, con una probabilidad mediana se creara un conjunto en que el estas clases pertenezcan al mismo.

$$P = MID * (PG - .15)$$

En este caso, si en el conjunto al que pertenecen las clases en el padre mejor evaluado ninguno de sus elementos genero violación $MID = 1$, en caso contrario $MID = 0.8$.

Fase 2

Supongamos que producto de la fase 1 se crearon 3 conjuntos, el primer conjunto formado por la clase de la materia A etiquetada con 1 y la clase de la materia D etiquetada con 12, el segundo conjunto formado por la clase de la materia B etiquetada con 6 y la clase de la materia J etiquetada con 25 y un tercer conjunto formado por la clase de la materia E etiquetada con 14 y por la clase de la materia J etiquetada con 26, podríamos representar esto como:

| | | $a1$ | $a2$ | $a3$ | $b1$ | $b2$ | $b3$ | $c1$ | $c2$ | $c3$ | $k1$ | $d1$ | $d3$ | $e1$ | $e2$ | $e3$ | $f1$ | $f2$ | $g1$ | $g2$ | $h1$ | $h2$ | $i1$ | $i2$ | $l1$ | $m1$ | $n1$ | $o1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $d2$ | | | | | $j1$ | | | | | | | $j2$ | | | | | | | | | | | | | | |
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 27 | 28 | 29 | 30 |
| | | 12 | | | | | 25 | | | | | | | 26 | | | | | | | | | | | | | | |
| $a1,d2$ | 1,12 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $a2$ | 2 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| $a3$ | 3 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | |
| $b1$ | 4 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | |
| $b2$ | 5 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | |
| $b3,j1$ | 6,25 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | |
| $c1$ | 7 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| $c2$ | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | | |
| $c3$ | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | | |
| $k1$ | 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | | | | | |
| $d1$ | 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | | | | | | | | |
| $d3$ | 13 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | | | | | | | | | | |
| $e1,j2$ | 14,26 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | | | | | | | | | | |
| $e2$ | 15 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| $e3$ | 16 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | | | | | | | | | | |
| $f1$ | 17 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | | |
| $f2$ | 18 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | | |
| $g1$ | 19 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | | |
| $g2$ | 20 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | | | | | | | | |
| $h1$ | 21 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | | | | | | | |
| $h2$ | 22 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | | | | | | |
| $i1$ | 23 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | | | | | |
| $i2$ | 24 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | | | | |
| $l1$ | 27 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | | | |
| $m1$ | 28 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | | | |
| $n1$ | 29 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| $o1$ | 30 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

A partir de este punto se elige al azar un cero, se unen conjuntos y se actualizan unos con el mismo procedimiento como se crearon los elementos de la población inicial, con la diferencia que el máximo número de elementos que puede contener un conjunto varia cada vez que se elige un cero de la siguiente manera

$$n.m.e = (Número\ de\ clases\ por\ hora\ permitidas) + INT(RND() * HOLG + .5)$$

$HOLG$ es la holgura en el número clases por hora permitidas que se puede fijar para permitir que algunos conjuntos tengan más elementos de los permitidos. En la mayoría de los problemas sirvió $HOLG = 0$, pero para aquellos que no se encontró una solución con cero violaciones se puso $HOLG > 0$

Después de que la solución está formada, a los conjuntos se les asigna al azar un número (que representa una hora) de tal manera que cada hora disponible quede asignada a un conjunto y se evalúa la solución tal y como se hizo anteriormente con los miembros de la población inicial. Si hay menos conjuntos que $No.De\ horas\ disponibles$ se crean conjuntos vacios, si hay más conjuntos que $No.De\ horas\ disponibles$ para esta solución no se realiza búsqueda local ni tampoco pasara por la etapa de actualización ni de preservación.

### *Búsqueda de vecindad variable reducida*

Para la búsqueda de vecindad variable reducida se definen los siguientes procedimientos:

Procedimiento 1:

Se iguala $T = 0$. Existen $(No.De\ horas\ disponibles) * (No.De\ horas\ disponibles - 1)/2$ intercambios del tipo: dos conjuntos diferentes intercambian el número que les fue asignado (hora), estos intercambios se ordenan de forma aleatoria. Se eligen los primeros $(No.de\ horas\ disponibles)^{\frac{3}{2}}$ intercambios, si hay mejora en alguno de ellos, el intercambio mejor evaluado se realiza, se actualiza la evaluación de la solución y se iguala $T = 1$. Si no hay mejora se repite lo siguiente hasta que, se realice un intercambio o se evalúen todos los intercambios: se evalúa el siguiente intercambio, si se encuentra mejora, se realiza el intercambio, se actualiza la evaluación de la solución y se iguala $T = 1$.

Procedimiento 2:

Se iguala $T = 0$. Existen $(No.total\ de\ clases) * (No.total\ de\ clases - 1)/2$ intercambios del tipo: dos clases diferentes intercambian su conjunto contenedor, estos intercambios se ordenan de forma aleatoria. Se eligen los primeros $(No.total\ de\ clases)^{\frac{3}{2}}$ intercambios, y de estos se vuelven a elegir sólo los que no violen restricciones del tipo "x clase no puede darse a la misma hora que y clase", si hay mejora en alguno de ellos, el intercambio mejor evaluado se realiza, se actualiza la evaluación de la solución y se iguala $T = 1$. Si no hay mejora se repite lo siguiente hasta que, se realice un intercambio o se evalúen todos los intercambios: se evalúa el siguiente intercambio, si se encuentra mejora y no viola restricciones del tipo "x clase no puede darse a la misma hora que y clase", se realiza el intercambio, se actualiza la evaluación de la solución y se iguala $T = 1$.

Procedimiento 3:

Se iguala $T = 0$. Se ordenan de forma aleatoria los cambios del tipo: se cambia una clase cuyo conjunto contenedor tuviese un tamaño mayor al "$número\ de\ clases\ por\ hora\ permitidas$", a un conjunto cuyo tamaño fuese menor al "$número\ de\ clases\ por\ hora\ permitidas$". Se eligen los primeros $(No.total\ de\ clases)^{\frac{3}{2}}$ cambios, y de estos se vuelven a elegir sólo los que no violen restricciones del tipo "x clase no puede darse a la misma hora que y clase", si en alguno de ellos hay mejora o no se modifica la evaluación de individuo, el cambio mejor evaluado se realiza, se actualiza la evaluación de la solución y se iguala $T = 1$. Si no ocurre lo anterior se repite lo siguiente hasta que, se realice un cambio o se evalúen todos los cambios: se evalúa el siguiente cambio, si se encuentra mejora o no se modifica la evaluación y no viola restricciones del tipo "x clase no puede darse a la misma hora que y clase", se realiza el cambio, se actualiza la evaluación de la solución y se iguala $T = 1$.

Procedimiento 4:

Se iguala $T = 0$. Existen $(No.total\ de\ clases) * (No.total\ de\ clases - 1)/2$ intercambios del tipo: dos clases diferentes intercambian su conjunto contenedor, estos intercambios se ordenan de forma aleatoria. Se repite lo siguiente hasta que, se realice un intercambio o se evalúen todos los intercambios: se evalúa el siguiente intercambio, si se encuentra mejora o no se modifica la evaluación, no viola restricciones del tipo "x clase no puede darse a la misma hora que y clase" y las dos clases no son de la misma materia, se realiza el intercambio, se actualiza la evaluación de la solución y se iguala $T = 1$.

Procedimiento 5:

Se iguala $T = 0$. Se ordenan de forma aleatoria los cambios del tipo: Se cambia una clase a un conjunto cuyo tamaño fuese menor al "$número\ de\ clases\ por\ hora\ permitidas\ +\ HOLG$". Se eligen los primeros $(No.total\ de\ clases)^{\frac{5}{2}}$ cambios, y de estos se vuelven a elegir sólo los que no violen restricciones del tipo "x clase no puede darse a la misma hora que y clase", si hay mejora en alguno de ellos, el cambio mejor evaluado se realiza, se actualiza la evaluación de la solución y se iguala $T = 1$. Si no hay mejora se repite lo siguiente hasta que, se realice un cambio o se evalúen todos los cambios: se evalúa el siguiente intercambio, si se encuentra mejora y no viola restricciones del tipo "x clase no puede darse a la misma hora que y clase", se realiza el intercambio, se actualiza la evaluación de la solución y se iguala $T = 1$.

## Estructura de la búsqueda de vecindad variable reducida

*Do*
    *Procedimiento 1*
*While* $T = 1$
*Do*
    $Y = evaluación\ del\ individuo$
    *Do*
        $.X = evaluación\ del\ individuo$
        *Do*
            *Procedimiento 2*
        *While* $T = 1$
        *Do*
            *Procedimiento 3*
        *While* $T = 1$
        $.W = 1$
        *Do*
            *Procedimiento 4*
            $.= W + 1$
        *While* $T = 1$ *And* $W < "No.de\ horas\ disponibles"$
    *While* $X \neq evaluación\ del\ individuo$
    *Do*
        *Procedimiento 5*
    *While* $T = 1$
*While* $Y \neq evaluación\ del\ individuo$

### *Actualización de la siguiente generación*

Si la solución generada es parecida al mejor de los padres, es igualmente evaluada y las horas fueron asignadas a las clases de manera igual o más homogéneamente, se cambia en la población el padre mejor evaluado por el hijo.

Si no pasa lo anterior, la solución generada es igualmente evaluada que el peor de los padres y las horas fueron asignadas a las clases de manera igual o más homogéneamente, se cambia en la población el padre peor evaluado por el hijo.

Si la solución generada es mejor evaluada que cualquiera de los padres, se cambia en la población el padre peor evaluado por el hijo.

Si no se cumple nada de los tres párrafos anteriores permanecen en la población ambos padres.

### *Preservación de la solución más homogénea mejor evaluada*

Dado que en la actualización de las soluciones se dio prioridad a reemplazar en cada generación con aquellas soluciones mejor evaluadas en su función de aptitud, se decidió crear el individuo 21, el cual no se cruza con otros pero sí se actualiza en cada cruce con la información del hijo generado (por otros) de la siguiente manera:

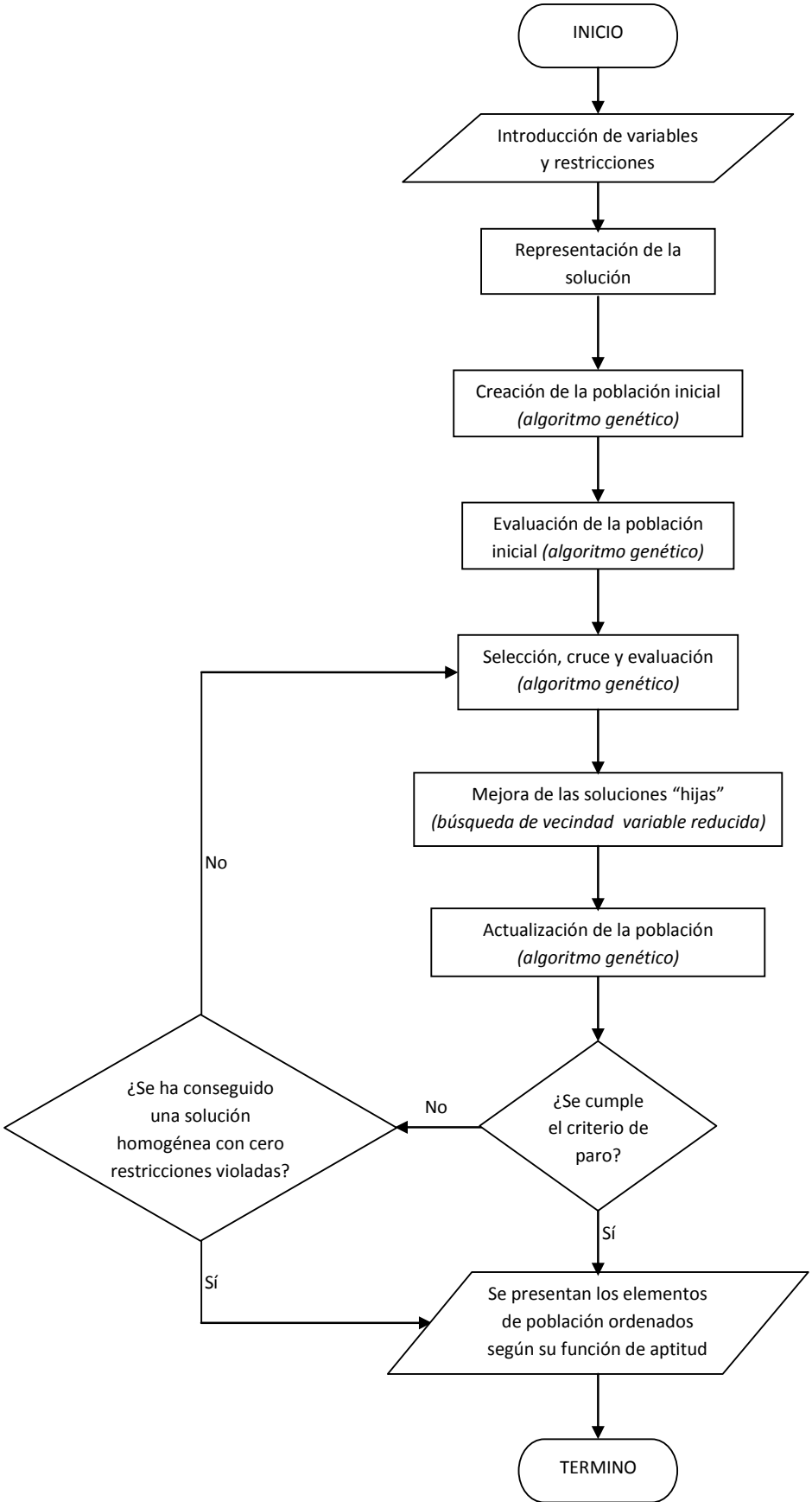
Si las horas en el hijo fueron asignadas a las clases más homogéneamente que en el individuo 21 (en el principio del algoritmo se copia la información del individuo 1 y se pega en el individuo 21), se copia la información del hijo y con esta se actualiza al individuo 21.

Si no pasa lo anterior, las horas en el hijo fueron asignadas a las clases igual de homogéneamente que en el individuo 21 y el hijo es mejor evaluado que el individuo 21, se copia la información del hijo y con esta se actualiza al individuo 21.

Si no se cumple nada de los dos párrafos anteriores la información del individuo 21 no sufre modificaciones.

* La información del individuo 21 nos es útil cuando el número de salones por hora no puede aumentar en ningún caso (es una restricción dura).

## *Diagrama de flujo general del Algoritmo Híbrido*



## 5. Cota del ejemplo citado y resultados computacionales

### *Cota del ejemplo citado*

Considérese el problema con el que se ha venido trabajando.

| CURSO | ASIGNATURAS | HORAS/SEMANA |
|---|---|---|
| I | A,B,C<br>K | 3<br>1 |
| II | D,E<br>F,G | 3<br>2 |
| III | H,I,J<br>L, M, N, 0 | 2<br>1 |

Proposición: No existe una solución en la que se asignen las horas a las clases de manera homogénea (para este ejemplo cada hora sería asignada dos veces) sin violar alguna de las restricciones.

Demostración por contraejemplo:

Supongamos que existe una solución donde se violen cero restricciones y en la que las horas son asignadas máximo 2 veces. Existen 5 materias de tres clases que deben ser repartidas en L, Mi y V para que no se viole ninguna restricción. Ya que cada hora sólo puede ser asignada a cuando más dos materias, sólo quedaría libre por asignar una hora del lunes, una hora del miércoles y una hora del viernes.

En el mejor de los casos podríamos elegir qué hora queda libre* en cada uno de esos tres días, por ejemplo:

| Hora\día | Lunes | | Martes | | Miercoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | * | | | * | | | | | * |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |

Por otro lado, al sólo haber 3 horas libres por asignar en L, Mi y V, forzosamente tres de las 5 materias de dos clases deberán de ser programadas para ser impartidas en Martes y Jueves. Utilizando el principio del palomar podemos inferir que al menos una de esas tres materias corresponde al módulo III.

| Hora\día | Lunes | | Martes | | Miercoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | * | | | * | | | | | * |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | | | |

De esta forma lo podemos ver como que existen sólo tres casos, las tres materias anteriores pertenecen al módulo 3 o dos de las tres materias pertenecen al módulo 3 o sólo una de las tres materias pertenece al módulo 3.

Analizando el caso en que las tres materias pertenecen al módulo 3 quedarían por asignar del módulo 3 las cuatro materias de una sola clase. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas cuatro clases la hora libre del lunes, la hora libre del miércoles y la hora libre del viernes. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

| Hora\día | Lunes | | Martes | | Miercoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | * | H | no (L o M o N o O | * | | H | no (L o M o N o O) | | * |
| 2 | | | I | no (L o M o N o O) | | | I | no (L o M o N o O) | | |
| 3 | | | J | no (L o M o N o O) | | | J | no (L o M o N o O) | | |

Analizando el caso en que dos de las materias pertenecen al módulo 3, por ejemplo:

| Hora\día | Lunes | | Martes | | Miercoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | * | F | * | * | | F | * | | * |
| 2 | | | H | no (L o M o N o O o J) | | | H | no (L o M o N o O o J) | | |
| 3 | | | I | no (L o M o N o O o J) | | | I | no (L o M o N o O o J) | | |

Quedarían por asignar del módulo 3 las cuatro materias de una sola clase y una de las materias de dos clases; en total seis clases. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas 6 clases la hora libre del lunes, la hora libre del miércoles, la hora libre del viernes, una de las horas del martes y una de las horas del jueves, en total 5 horas. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

Analizando el caso en que sólo una materia pertenece al módulo 3, por ejemplo:

| Hora\día | Lunes | | Martes | | Miercoles | | Jueves | | Viernes | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | * | F | * | * | | F | * | | * |
| 2 | | | G | * | | | G | * | | |
| 3 | | | H | no (L o M o N o O o I o J) | | | H | no (L o M o N o O o I o J) | | |

Quedarían por asignar del módulo 3 las cuatro materias de una sola clase y dos de las materias de dos clases; en total ocho clases. Ya que clases del mismo módulo no deben de impartirse a la misma hora sólo quedarían libres por asignar a estas 8 clases la hora libre del lunes, la hora libre del miércoles, la hora libre del viernes, dos de las horas del martes y dos de las horas del jueves, en total 7 horas. Por lo que en este caso no hay forma de lograr una asignación con cero restricciones violadas.

En los tres casos no hay forma de lograr una asignación con cero restricciones violadas, por lo que podemos concluir que no existe tal solución bajo las consideraciones antes planteadas.

Del mismo modo podemos concluir que una solución donde no se violen ninguno de los dos tipos de restricciones mencionadas en ente trabajo, debe permitir al menos que en una hora sean impartidas 3 materias. Y en caso de que se logre esto permitiendo que sólo en una hora se den 3 clases, esta solución será la óptima.

### *Resultados computacionales*

Siguiendo la metodología expuesta en Lara 2010 [2] se generaron las instancias de ese mismo artículo y además, se generaron otras instancias (una de más de doble de tamaño con respecto a la instancia más grade generada en el articulo antes citado); con la finalidad de poner a prueba al algoritmo hibrido aquí desarrollado.

La siguiente tabla muestra las instancias creadas y los resultados obtenidos en Lara 2010 [2], utilizando la metaheurística GRASP, y se comparan con los resultados obtenidos utilizando el algoritmo híbrido de este trabajo.

| No. De vértices | Instancia | No. de colores máx. | No. de clases "fuera de lugar" con GRASP (no se violan restricciones del tipo 1 ni del 2) | No. de clases "fuera de lugar" con Alg. Híbrido (no se violan restricciones del tipo 1 ni del 2) | ¿Es el óptimo? | No. de restricciones del tipo 2 violadas en una solución donde las horas se asignaron de manera totalmente homogénea (no se viola restricciones del tipo 1) | ¿Es el óptimo?. |
|---|---|---|---|---|---|---|---|
| 30 | ED4 | 15 | 1 | 1 | sí | 1 | sí |
| 30 | A42 | 15 | 0 | 0 | sí | 0 | sí |
| 60 | ECA864 | 20 | 0 | 0 | sí | 0 | sí |
| 60 | 976532 | 20 | 0 | 0 | sí | 0 | sí |
| 90 | EDDC96441 | 30 | 1 | 0 | sí | 0 | sí |
| 90 | DCB875322 | 30 | 0 | 0 | sí | 0 | sí |
| 120 | EEDCCBA87644 | 30 | 0 | 0 | sí | 0 | sí |

Para una de las dos instancias (EDDC96441) en las que el GRASP no pudo encontrar una solución con cero violaciones en la que cada hora fuese asignada un número igual de veces, el Algoritmo Genético logró encontrarla; para la otra instancia (ED4) se demostró que aunque hay una clase fuera de lugar (fue asignada una hora a más clases de las permitidas) en las soluciones encontradas con GRASP y el Algoritmo Genético, no hay mejor solución que pueda ser encontrada.

Los resultados de las nuevas instancias se presentan en la tabla inferior.

| No. De vértices | Instancia | No. de colores máx. | No. de clases "fuera de lugar" con Alg. Híbrido (no se violan restricciones del tipo 1 ni del 2) | ¿Es el óptimo? | No. de restricciones del tipo 2 violadas en una solución donde las horas se asignaron de manera totalmente homogénea{3} (no se violan restricciones del tipo 1) | ¿Es el óptimo?. |
|---|---|---|---|---|---|---|
| 60 | EEDD44 | 15 | 2{1} | sin probar | 1 | sin probar |
| 90 | EEEDDD444 | 15 | 3{2} | sin probar | 1 | sin probar |
| 180 | EEDDDDCC9966444411 | 30 | 0 | sí | 0 | sí |
| 180 | EEDDDDCC9966444411 | 60 | 0 | sí | 0 | sí |
| 240 | EEEEDDCCCCBBAA8877664444 | 30 | 0 | sí | 0 | sí |
| 240 | EEEEDDCCCCBBAA8877664444 | 60 | 0 | sí | 0 | sí |
| 270 | EEEEEEEEEDDDDDDDDD444444444 | 45 | 0 | sí | 0 | sí |
| {1} se debe permitir que en dos horas se impartan 5 clases (una clase más de lo deseable) | | | | | | |
| {2} se debe permitir que en tres horas se impartan 7 clases (una clase más de lo deseable) | | | | | | |
| {3} todas las horas fueron asignadas el mismo número de veces | | | | | | |

El algoritmo híbrido también alcanza buenos resultados para instancias de mayor tamaño. Estos resultados también servirán en estudios posteriores para comparar nuevas propuestas.

## 6. Experimentos computacionales

Un experimento es un conjunto de pruebas realizadas en condiciones controladas para un fin específico: para demostrar una verdad conocida, para comprobar la validez de una hipótesis, o para examinar el desempeño de algo nuevo. Investigadores en todos los ámbitos de estudio realizan experimentos para demostrar la teoría, para descubrir conocimiento sobre un proceso en particular, y para medir el efecto de uno o más factores en algunos fenómenos. Un factor es una variable controlable en un experimento que influye en el resultado de un experimento.

Nuestro experimento en esta tesis consiste en tomar varios problemas conocidos (condiciones controladas) y se tratan de resolver con diferentes heurísticos con el objetivo de conseguir información sobre el método heurístico más robusto y eficiente.

La primera prueba se hace con la instancia EEEDDD444 (Ilustración 1) midiendo la calidad de la mejor solución encontrada por cada uno de los heurísticos vs el tiempo (segundos) que se tardaron en encontrarla, durante un intervalo de 30 minutos.

Los métodos heurísticos utilizados tienen relación directa con el algoritmo tratado en esta tesis ya que son casos especiales de este o se basan en partes del mismo. De esta forma se evidencia el desempeño de cada una de las partes por separado (el factor en este caso sería la aplicación o la no aplicación de alguna de las partes, variable 0 o 1), y las ventajas de combinarlos en un algoritmo híbrido.

El primer algoritmo (GENyRVNS) utilizado es el algoritmo híbrido tal y como se describe en esta tesis, sus parámetros son: tamaño de población 20 y es la mezcla de un algoritmo genético y un algoritmo de búsqueda en vecindades variables reducida.

El segundo algoritmo (BUSQITER) utilizado corresponde algoritmo híbrido mencionado anteriormente pero con un tamaño de población 2, lo que lo convierte en un algoritmo de búsqueda local iterativa ya que al ser sólo dos elementos el cruce puede ser considerado como la perturbación en este algoritmo.

El tercer algoritmo (RVNS) toma del algoritmo híbrido sólo la parte de la búsqueda en vecindades variables reducida, por lo que es necesario utilizarlo como multiarranque para reiniciar la búsqueda una vez que se ha “atorado en un óptimo local”.

El cuarto algoritmo (GEN) toma del algoritmo híbrido la parte del algoritmo genético para después aplicarle una búsqueda local bastante más sencilla que el método completo de la búsqueda en vecindades variables reducida.

Los resultados de la prueba son los siguientes:

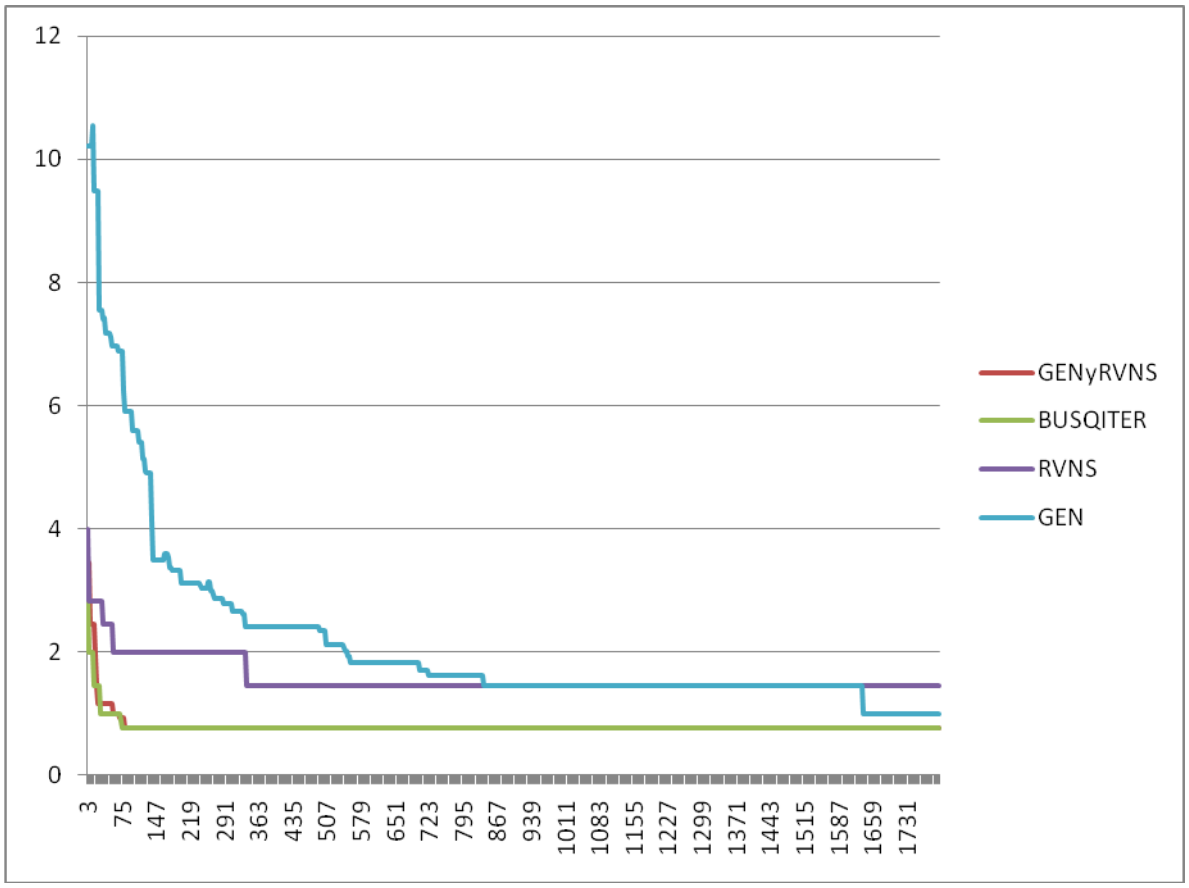



**Ilustración 1. Instancia EEEDDD444 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

Por sí solos el algoritmo genético (GEN) y el algoritmo multiarranque de búsqueda en vecindades variables reducida (RVNS) no logran llegar a la mejor solución conocida para este problema. Combinándolos tanto en la búsqueda local iterativa (BUSQITER) como en el algoritmo híbrido (GENyRVNS) logran llegar a la mejor solución conocida. En términos de tiempo el algoritmo de búsqueda local iterativa es sólo un poco más eficiente que el algoritmo híbrido por lo que se podría concluir que aumentar la población a 20 no vuelve lento el proceso si no que esto permite aportar las ventajas de la combinación de patrones de un algoritmo genético clásico.

Para las demás instancias se graficó también la calidad de la mejor solución encontrada por cada uno de los heurísticos vs. el tiempo (segundos) que se tardaron en encontrarla, durante un intervalo de 30 minutos (ver anexo), encontrándose resultados similares a los ya comentados para la instancia EEEDDD444.

## 7. Conclusiones

Con base en los resultados (págs. 38 y 39) se puede afirmar que se desarrolló un algoritmo híbrido que logra combinar las ventajas de dos metaheuristicas comúnmente utilizadas (los algoritmos genéticos y el los algoritmos de búsqueda en vecindades variables) para el problema de horarios que se aborda en esta tesis, logrando con esto el objetivo principal.

Ya que el problema de horarios en esta tesis se modeló como un PCRG, y que el algoritmo híbrido se desarrolló para resolver este tipo de problemas, cualquier problema que pueda ser modelado como un PCRG podrá ser tratado haciendo uso del algoritmo diseñado en el presente trabajo.

El algoritmo híbrido es una buena opción ya que cuenta con las características deseables de una metaheuristica:

- Bueno. Proporciona mejores soluciones a las ya obtenidas con otros métodos (pág. 38).
- Robusto. Tiene las ventajas de las dos meteheurísticas en las que se basa, incorporando características que hacen que pueda escapar de óptimos locales en un gran número de casos conservando parte de la información (pág. 41).
- Eficiente. Para el problema tratado los tiempos en los que arroja buenas soluciones son adecuados (págs. 41 y Anexo).

En el desarrollo del algoritmo se manejó holgura en el número de colores (horas) y holgura en el número de clases permitidas por hora; con tal de obtener soluciones en las que no se violaran ninguno de los siguientes dos tipos de restricciones: dos eventos no pueden realizarse el mismo día y debe haber al menos *d* días entre dos eventos. Dejar que algunos miembros de la población tuviesen un número de colores mayor al establecido no aportó nada de información pero dejar en algunas soluciones holgura en el número de clases permitidas por hora aportó valiosa información que permitió llegar al óptimo más rápidamente para las instancias citadas.

Queda como trabajo futuro el desarrollo de otras metaheurísticas para este tipo de problemas (por ejemplo Redes Neuronales) y su comparación con las ya existentes.

## Anexo

## *Gráficas generadas en los experimentos computacionales*

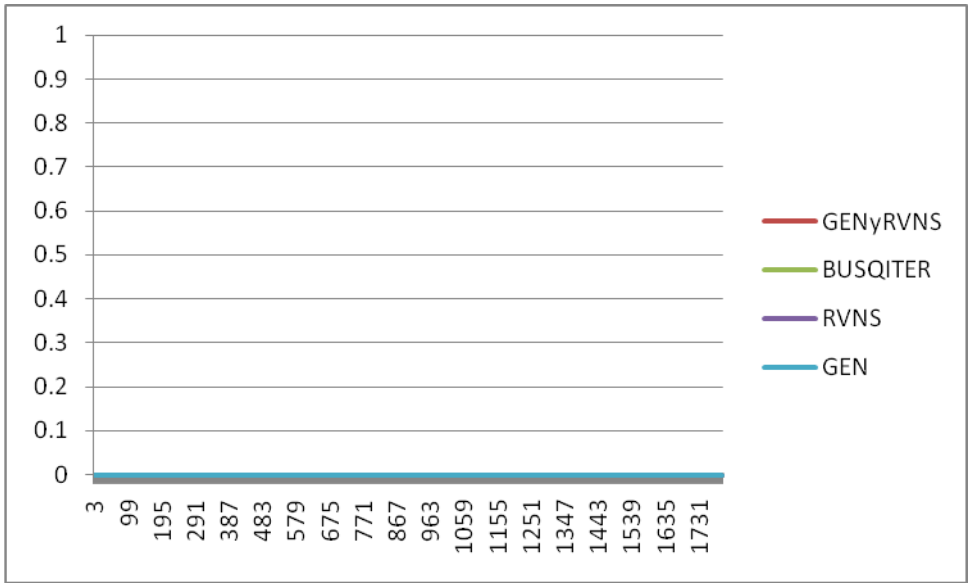



**Ilustración 2. Instancia A42 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

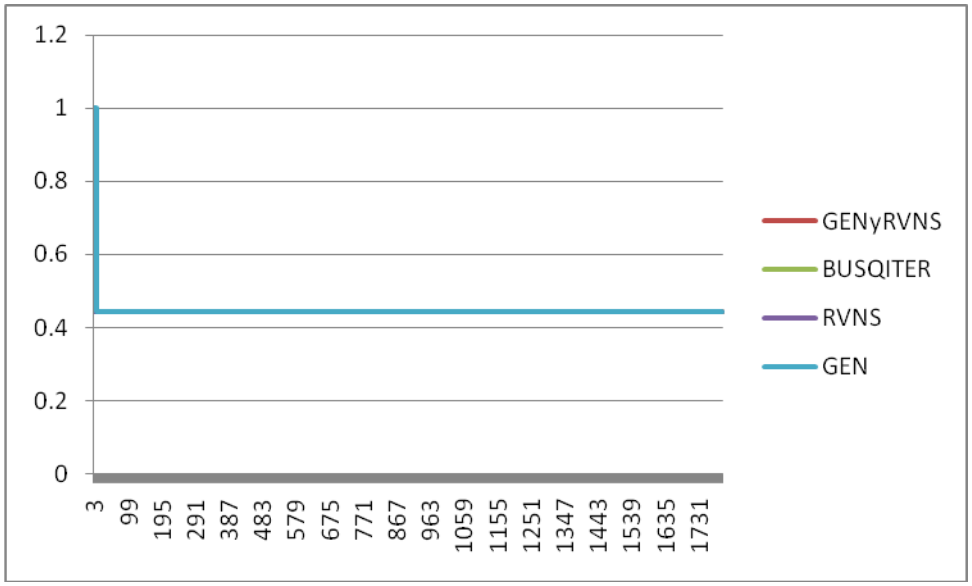



**Ilustración 3. Instancia AD4 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

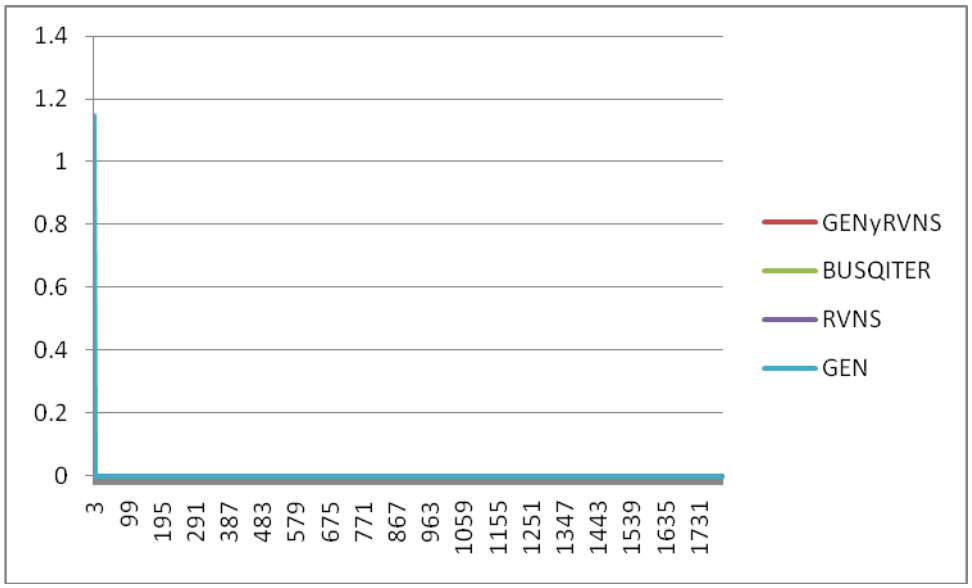



**Ilustración 4. Instancia 976532 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

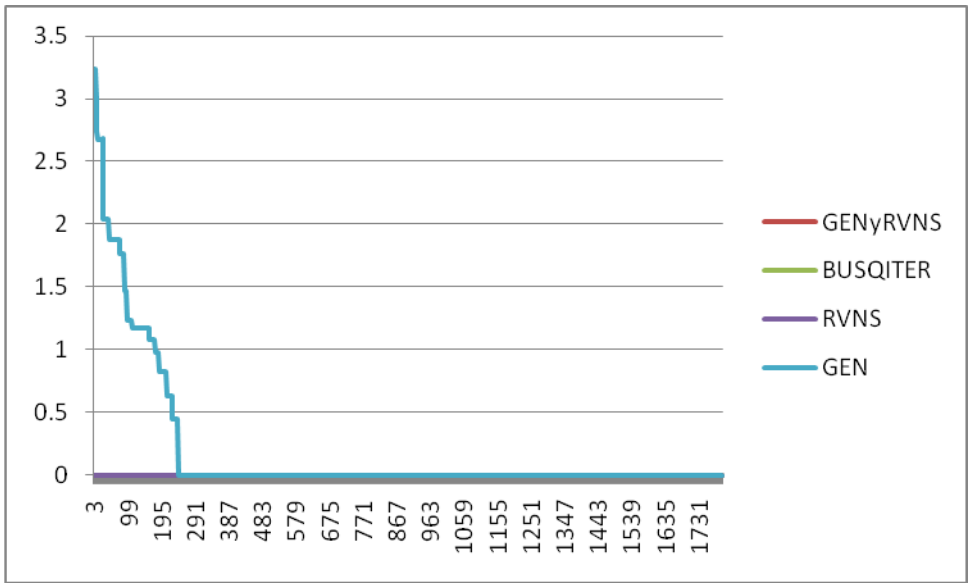



**Ilustración 5. Instancia ECA864 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

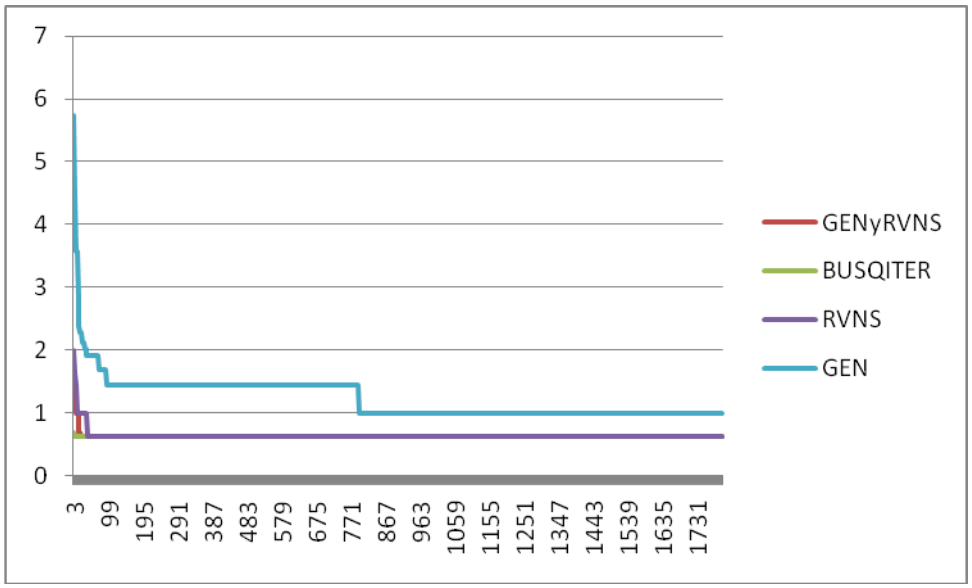



**Ilustración 6. Instancia EEDD44 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

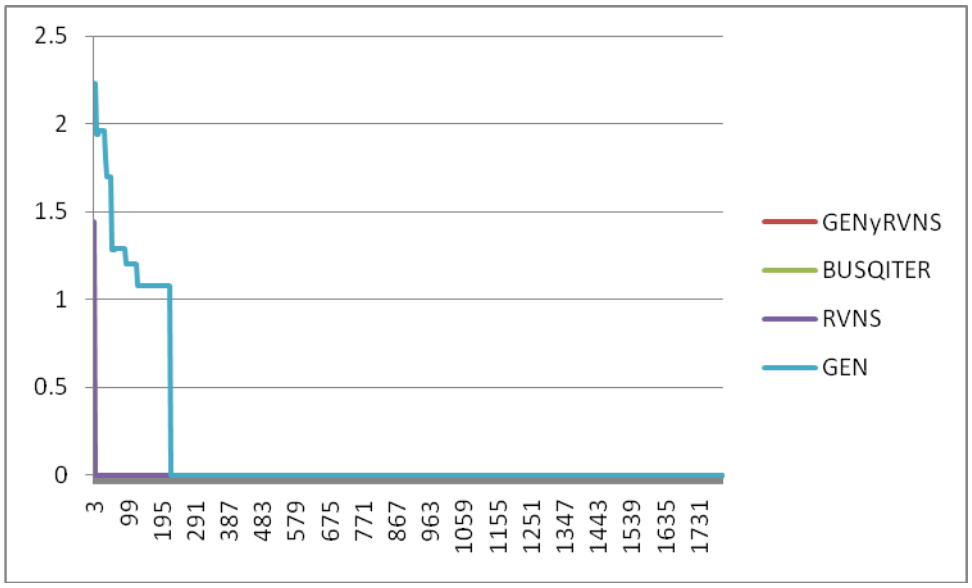



**Ilustración 7. Instancia DCB875322 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

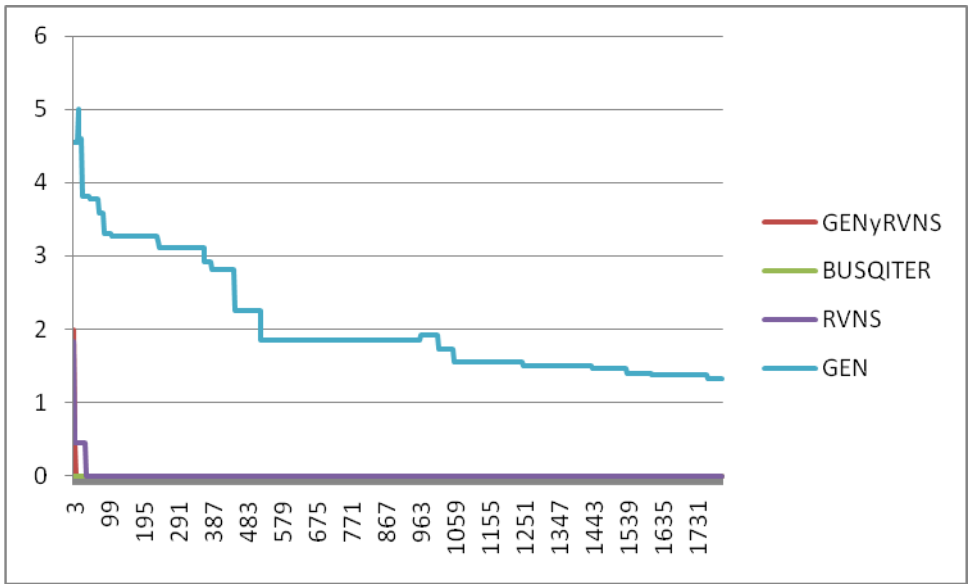



**Ilustración 8. Instancia EDDC96441 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

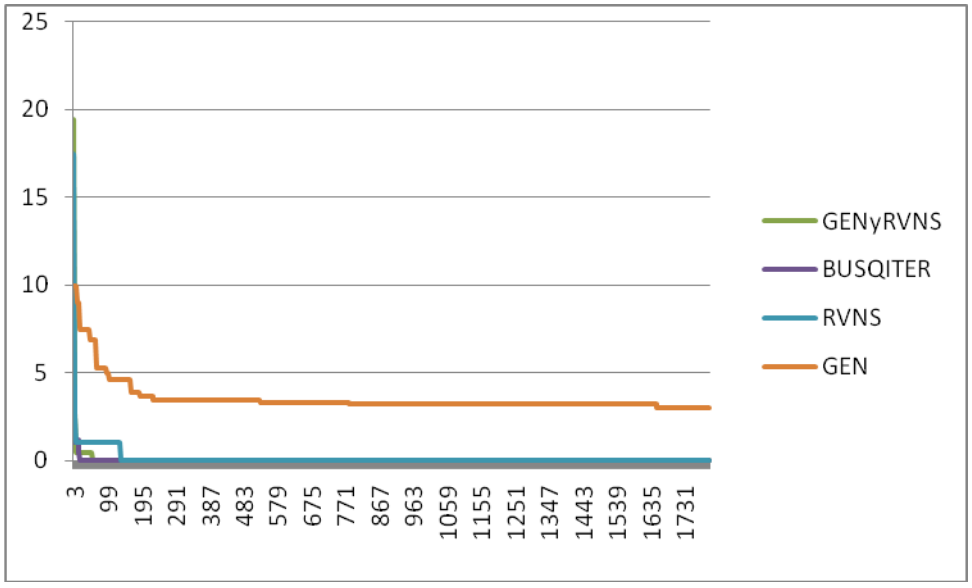



**Ilustración 9. Instancia EEDCCBA87644 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

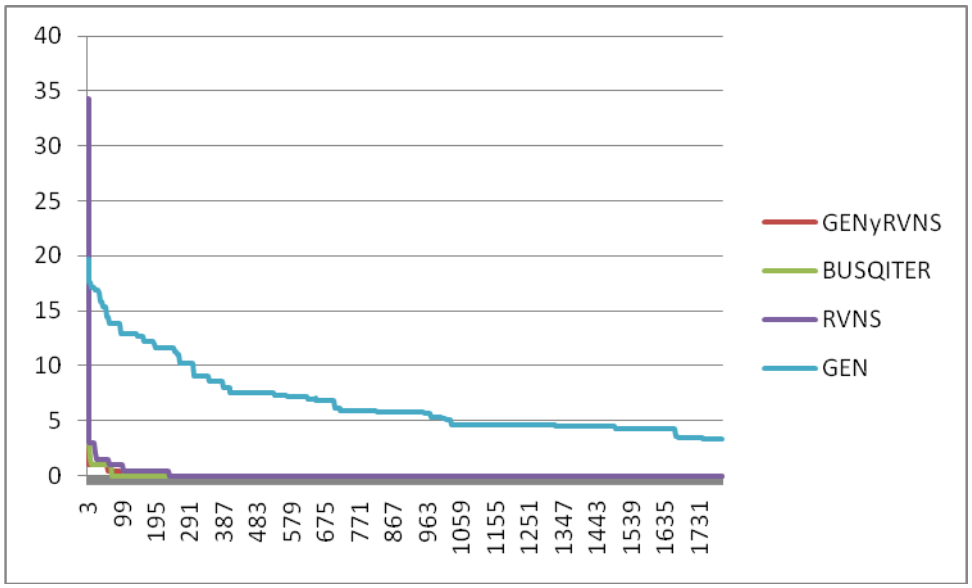



**Ilustración 10. Instancia EEDDDDCC9966444411(30) "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

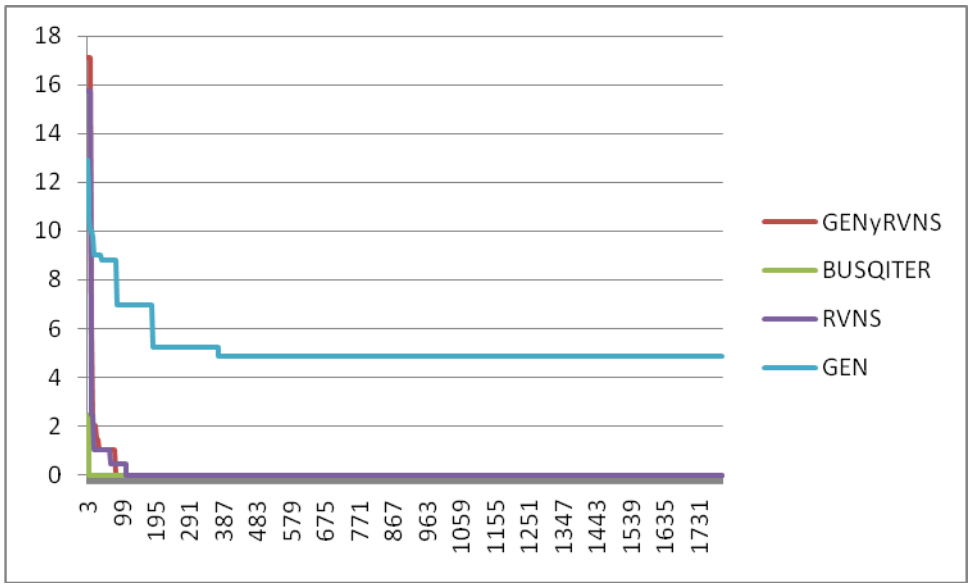



**Ilustración 11. Instancia EEDDDDCC9966444411(60) "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

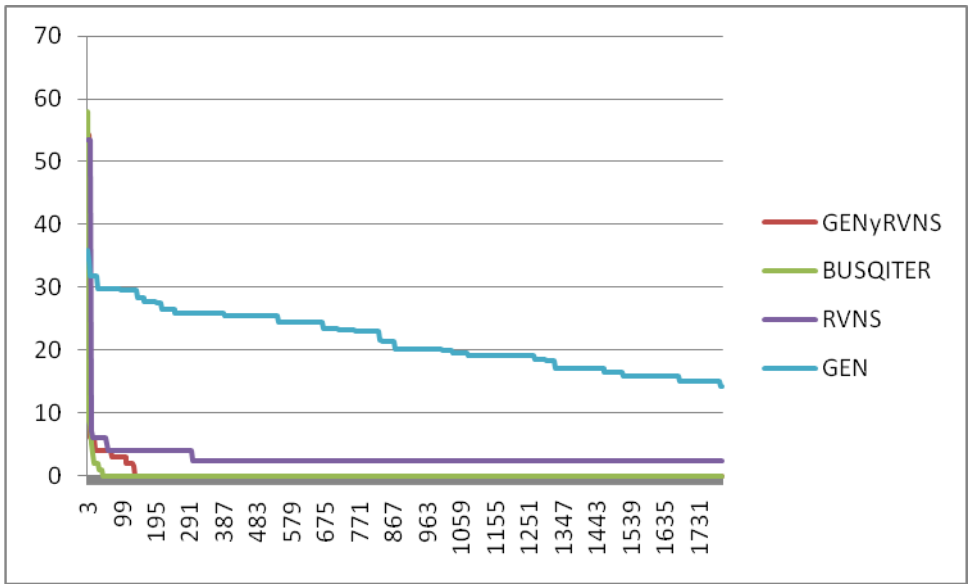



**Ilustración 12. Instancia EEEEDDCCCCBBAA8877664444(30) "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

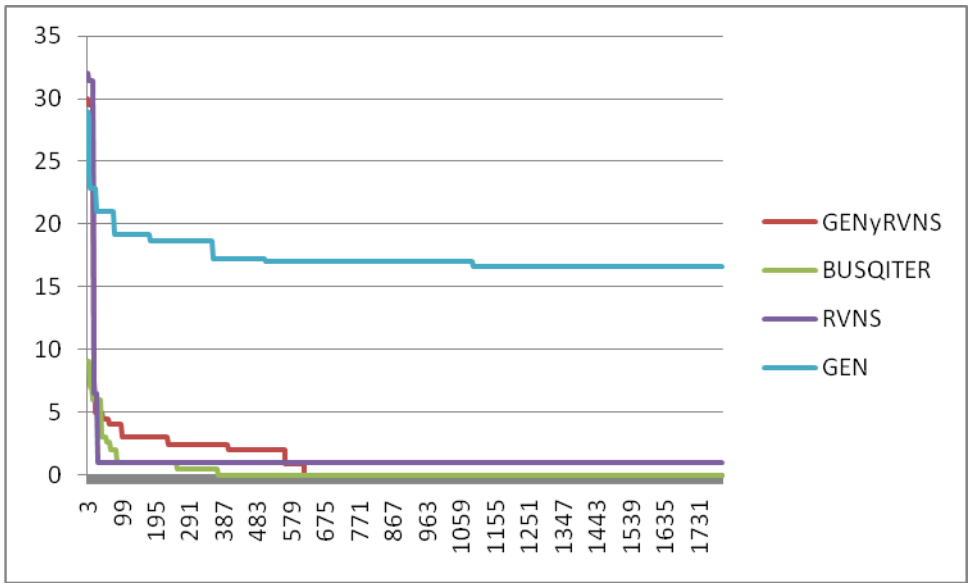



**Ilustración 13. Instancia EEEEDDCCCCBBAA8877664444(60) "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

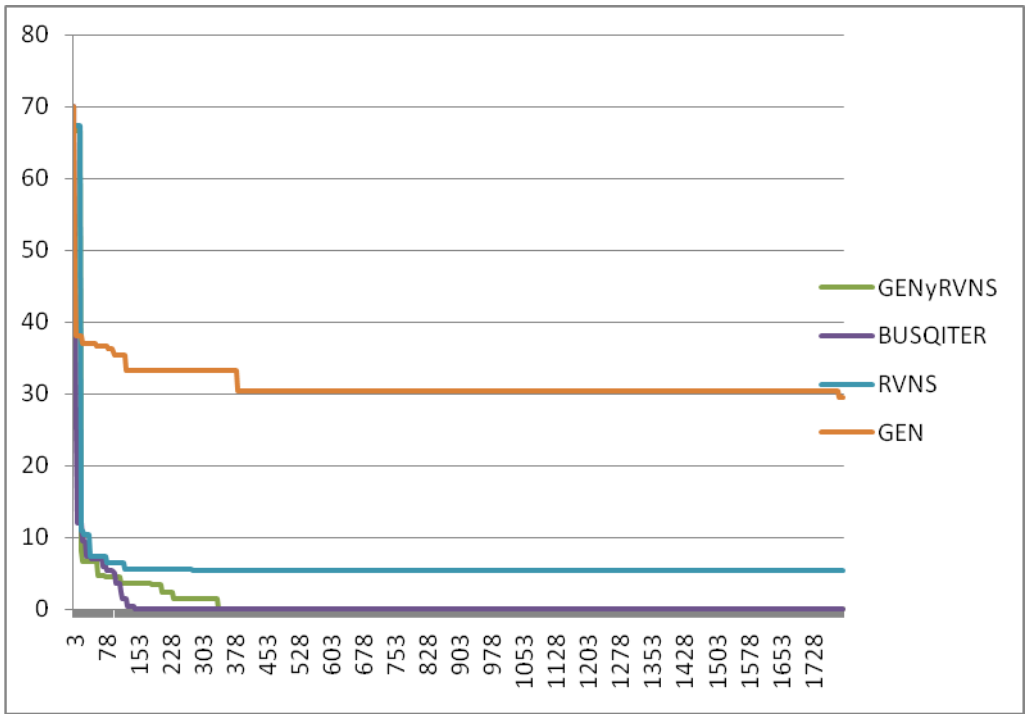



**Ilustración 14. Instancia EEEEEEEEEDDDDDDDDD444444444 "calidad de la mejor solución encontrada por el método heurístico vs el tiempo que se tardo en encontrarla (segundos)"**

## Referencias


[1] Ramírez J (2001). Extensiones del problema de coloración de grafos. Universidad Complutense de Madrid; España, ISBN 84-669-1855-8. http://www.ucm.es/BUCM/tesis/mat/ucm-t24770.pdf.

[2] Lara Velázquez P, López Bracho R, Ramírez Rodríguez J, Yáñez J (2010). A model for timetabling problems with period spread constraints. Journal of Operational Research Society 62, pp 217-222.

[3] Salazar Gonzales Juan José (2001).Programación matemática. Editorial Díaz de Santos.

[4] Edward A. Silver, R. Victor, V. Vidal, Dominique de Werra (1980). A tutorial on heuristic methods. European Journal of Operational Research 5, 3, pp 153-162

[5] Reeves, C.R. (1995). Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, UK.

[6] Segura Clara María (2004). Introducción a la teoría de la NP-completitud. http://www.fdi.ucm.es/profesor/csegura/mtp0304/apuntes/NPcompletitud.pdf

[7] Burke EK., Werra D. and Kingston J. (2004).“Applications to Timetabling,”, In: J.Gross and J.Yellan(eds). The Handbook of Graph Theory, Chapman Hall/CRC Press, pp 445-474.

[8] Burke EK, Kingston J, Jackson K, Weare R (1997). Automated university timetabling: The state of the art. The Computer Journal 40,9, pp 565-571.

[9] Rhydian Marc Rhys Lewis (2007). A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectrum (2008) 30, pp 167–190

[10] Carter M, Laporte G (1998). Recent developments in practical course timetabling. In: Burke E, Carter M (eds) Practice and theory of automated timetabling (PATAT) II, vol 1408. Springer, Berlin, pp 3–19

[11] Carter M, Laporte G, Lee SY (1996). Examination timetabling: algorithmic strategies and applications. J Oper Res Soc 47, pp 373–383

[12] Burke EK, Petrovic Sanja (2002). Recent research directions in automated timetabling. European Journal of Operational Research 140, pp 266–280

[13] Roberts, F.S.(1979). T-Colorings of graphs: recent results and open problems. Discrete Mathematics 93, pp 229-245.