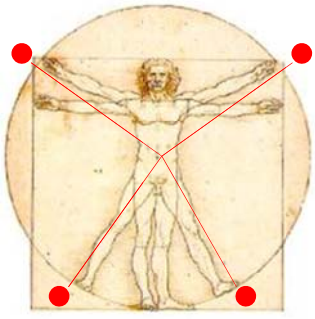
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN XV. AÑO 2017

SEPARATA



REVISIÓN DE ALGORITMOS DE BÚSQUEDA APLICADAS AL PROBLEMA DE CREACIÓN DE CALENDARIOS DE EXÁMENES

Pilar Moreno Díaz, Jesús Sánchez Allende

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior
Villanueva de la Cañada (Madrid)

Febrero, 2017.
http://www.uax.es/publicacion/revision-de-algoritmos-de-busqueda-aplicadas-al-problema-de-creacion-de.pdf

Escuela Politécnica Superior.
Universidad Alfonso X el Sabio.
28691, Villanueva de la Cañada (Madrid).
ISSN: 1696-8085
Editor: Javier Morales Pérez – tecnologia@uax.es

Revisión de algoritmos de búsqueda aplicadas al problema de creación de calendarios de exámenes

Pilar Moreno Díaz[(1)], Jesús Sánchez Allende[(2)]

(1) Licenciada en Ciencias (Matemáticas)
Area de tecnologías de información y comunicaciones, Escuela Politécnica Superior, Universidad Alfonso X el Sabio. Avda. De la Universidad nº1, Villanueva de la Cañada, 28691 Madrid. España.
Tlf.:918109237, email: pilar@myuax.com

(2) Dr Ing. de Telecomunicación
Area de tecnologías de información y comunicaciones, Escuela Politécnica Superior, Universidad Alfonso X el Sabio. Avda. De la Universidad nº1, Villanueva de la Cañada, 28691 Madrid. España.
Tlf.:918109135, email: jallende@myuax.com

RESUMEN: La creación de los calendarios de exámenes es una de las actividades que tienen que realizar todas las Universidades e instituciones educativas una o varias veces al año. En este artículo se hace una revisión de las técnicas más utilizadas en la resolución de este problema así como se presentan los resultados publicados. Del análisis realizado se ha podido comprobar que se han utilizado muchas técnicas diferentes pero no todas ellas han resultado igual de eficaces. Sin embargo, no hay ninguna que de manera especial destaque sobre las demás lo que deja abierto el camino a seguir investigando y proponer nuevas técnicas y estrategias para este problema.

PALABRAS CLAVE: Algoritmos, calendario de exámenes, metaheurísticas, optimización combinatoria, basadas en trayectorias, basadas poblaciones, algoritmos evolutivos, busqueda tabú, Hill-Climbing, Simulated Annealing.

ABSTRACT: Examination timetiming is one of the activities that universities and educational institutions have to do at least once every year. In this article a review is made of the most common techniques of solving the problem as it is revealed by the results of the investigations. From the present survey, it has been proven that there have been used many different techniques but not all of them have resulted equally efficient. Nevertheless, none of them specially stands out above the others, which leaves an open path to continue investigating and coming up with new techniques for this problem.

KEY-WORDS: Algorithm, timetabling problem, metaheuristics, combinatorial optimization, based on trajectories, based on populations, Evolutionary Algorithms, Taboo Search, Hil-Climbing, Simulated Annealing.

1. Introducción

La creación de horarios es una de las tareas cotidianas a las que nos enfrentamos todos, desde decidir cuándo programar una reunión, cuando quedar con los amigos, cuando planificar las vacaciones, etc. En el ámbito educativo es una de las tareas habituales que hay que realizar periódicamente, bien anualmente, semestralmente, trimestralmente, etc, dependiendo de los calendarios académicos para planificar las actividades docentes de profesores y estudiantes.

En el ámbito educativo se suelen distinguir tres tipos de horarios:

- Horarios clase-profesor, habitual en los institutos, donde los grupos son compactos por cursos académicos.
- Horarios universitarios, donde prima la flexibilidad de elección de materias y alumnos con muy distinta elección de asignaturas.
- Horarios de exámenes, donde se asignan las fechas de realización de exámenes dentro de un periodo más o menos reducido.

De manera más formal un problema de horarios es un problema con cuatro parámetros: Un conjunto finito de franjas horarias, un conjunto finito de recursos, un conjunto finito de asignaciones, un conjunto finito de restricciones.

El problema consiste en asignar tiempos y recursos a las aulas de manera que se satisfagan el mayor número posible de restricciones.

En este artículo se trata de forma exclusiva el problema de calendarios de exámenes. El objetivo del mismo es realizar una revisión de las distintas técnicas que se han utilizado hasta el momento para resolver el problema. Aunque es imposible recoger de forma exhaustiva todos los trabajos publicados, se ha realizado una selección de las técnicas más relevantes y, para cada una de ellas, los trabajos que se consideran los más importantes.

1.1 Revisiones anteriores

En los artículos de investigación se pueden encontrar varias revisiones anteriores centradas en la creación de calendarios de exámenes, que en orden cronológico son las siguientes:

- La primera que se publicó fue la de (Carter, 1986). En su revisión ninguno de los trabajos descritos se había probado en más de una Universidad, lo que a pesar del estudio no permitía disponer de una base comparativa de los algoritmos.
- Posteriormente actualizó la revisión en (Carter & Laporte, 1995). En esta revisión se centra en que los algoritmos propuestos hubiesen sido

implementados en Universidades reales o probados con datos reales desde 1986 a 1995.

- En (Schaerf, 1999) se realiza una revisión tanto de los horarios universitarios como los calendario de exámenes. Se indica en el trabajo que no existe una diferencia que impida utilizar los mismos modelos de resolución.
- La revisión más reciente publicada es la de (Qu, Burke, McCollum, Merlot & Lee, 2009), tomando como base la de (Carter & Laporte, 1995) y se centra, por tanto, en los artículos publicados desde 1996 hasta 2009.

2. Definición del problema

El problema de creación de calendarios de exámenes trata de la asignación de un conjunto finito de asignaturas a franjas horarias (día y hora de realización). La dificultad del problema estriba en que en el calendario de exámenes se debe conseguir, si es posible, que a ningún alumno le coincidan dos exámenes en la misma franja horaria. Adicionalmente puede requerirse algunos restricciones adicionales que mejoren la calidad del calendario de exámenes, atendiendo a restricciones de uso de equipos o aulas, disponibilidad de profesores o vigilantes para los exámenes, disponibilidad de aulas con capacidad suficiente, etc.

Se distingues entre dos tipos de restricciones:

- Restricciones fuertes: son las restricciones que deben cumplirse obligatoriamente. No es factible crear un calendario que incumpla este tipo de restricciones. Las restricciones más habituales de este tipo son:
  - Un alumno no puede tener dos exámenes en la misma franja horaria.
  - Dependiendo de la institución, los exámenes pueden compartir, o no el aula de examen (más de un examen se puede realizar en el mismo aula)
  - Se debe disponer de suficiente capacidad de aula para realizar los exámenes (dependiendo de las instituciones los exámenes se puede partir entre varias aulas o no).
- Restricciones débiles: son las restricciones que mejoran la calidad del calendario de exámenes, atendiendo a condiciones de percepción de los estudiantes o de la utilización de recursos. Entre las restricciones de este tipo más habituales se encuentran:
  - Dejar tiempo suficiente entre exámenes para que los estudiantes puedan prepararlos, extendiéndolos lo máximo posible en el calendario disponible.
  - Restricción en el uso de determinadas franjas horarias (ej: cierto laboratorio no está disponible los viernes por la mañana)

- Minimizar el número de exámenes consecutivos o en el mismo día que puede tener un estudiante.
- Conseguir cierto orden en la realización de los exámenes.
- Limitar el número máximo de estudiantes que realizan exámenes de forma simultánea en una determinada franja horaria.
- Fijar que varios exámenes se realicen en el mismo día o misma franja horaria.
- Poner determinados exámenes en ciertas franjas horarias, por ejemplo, la primera del día o la última del día.
- Asignar en la misma aula solo exámenes de la misma duración.
- Otras muchas consideraciones que pueden depender de las características y cultura de la Universidad.

2.1 Conjuntos de datos estándar

Para poder comparar unos algoritmos con respecto a otros es necesario que estos resuelvan el mismo problema, por esto han surgido conjuntos de datos que utilizar como referencia. Entre los más utilizados en la literatura se encuentran:

- Conjunto de datos de Toronto: Consta de trece problemas de horarios de exámenes reales. Para estos conjuntos de datos se requiere que no haya coincidencia de exámenes para ningún estudiante y que se distribuyan unos exámenes respecto a otros, penalizando si un estudiante tiene exámenes en franjas horarias muy seguidas.
- Conjunto de datos de la Universidad de Nottingham: Consta de tres franjas al día, cinco días a la semana de lunes a viernes con un total de 23 franjas, para 800 exámenes con 7.896 alumnos matriculados en 34.265 asignaturas.
- Conjuntos de datos de la Universidad de Melbourne: Se utilizaban dos conjuntos de datos diferentes con dos franjas por día con capacidades diferentes. Además se incluyen restricciones donde determinados exámenes están preasignados a determinadas franjas horarias o sólo se pueden planificar en un determinado conjunto de ellas. El primer problema consta de 23 franjas, para 521 exámenes con 20.656 alumnos matriculados en 62.248 asignaturas. El segundo problema consta de 31 franjas, para 526 exámenes con 19.816 alumnos matriculados en 60.637 asignaturas.

El más utilizado en la literatura es el conjunto de datos de Toronto, también porque tiene una cierta diversidad de conjuntos de datos de diversas universidades reales.

3. Algoritmos de resolución del problema de calendarios de exámenes

La investigación sobre el problema de los horarios de exámenes se lleva desarrollando desde antes de los años 80. Desde entonces se han propuesto muchos algoritmos diferentes para abordar el problema. En esta sección se realiza una extensa revisión de los métodos más importantes, donde se describen los métodos empleados para su resolución, sus fundamentos y los resultados obtenidos de acuerdo con las investigaciones publicadas.

A modo de clasificación (Carter & Laporte, 1995) dividieron las técnicas en cuatro categorías: métodos de cluster, métodos secuenciales, métodos basados en restricciones y metaheurísticas. Posteriormente (Petrovic & Burke, 2004) añadieron las categorías: multicriterio, razonamiento basado en casos e hiperheurísticas.

En una clasificación más general se pueden dividir los métodos de resolución en métodos basados en trayectorias y métodos basados en poblaciones. Los métodos basados en trayectorias son aquellos que utilizan una única solución y van explorando de forma aleatorizada el espacio de estados para encontrar soluciones cada vez mejores hasta que se llega al criterio de parada determinado. Entre los algoritmos de este tipo se encuentran:

- **Hill-Climbing** (Merlot, Boland, Hughes & Stuckey, 2003), (Burke & Bykov, 2008).
- **Tabu Search** (White & Xie, 2000), (Kendall & Hussin, 2004).
- **Simulated Annealing** (Wright, 2001), (Burke, Eckersley, McCollum, Sanja & Qu, 2003), (Battistutta, Schaerf & Urli, 2015).
- **Great Deluge Algorithm** (Abdullah, Shaker, McCollum & McMullan, 2009), (Turabieh & Abdullah, 2011).
- **Variable Neighbourhood Search** (Abdullah, Burke & Mccollum, 2005)**,** (Burke, Eckersley, McCollum, Petrovic & Qu, 2010).

Los métodos basados en poblaciones tratan con varias soluciones y van mejorándolas hasta obtener la mejor solución posible. Entre los ejemplos de este tipo de métodos están los siguientes:

- **Genetic Algorithm** (Chu & Fang, 1999), (Erben, 2000).
- **Memetic Algorithm** (Burke, Newall & Weare, 1996)**,** (Burke & Silva, 2005).
- **Ant Colony Optimisation** (Dowsland & Thompson, 2005)**,** (Eley, 2006a)**,** (Eley, 2006b).
- **Particle Swarm Optimization** (Fealko, 2005)**,** (Ahmed, Sajid, Ali & Bukhari, 2011)**.**

En el resto de apartados de esta sección se van a describir los principales algoritmos indicados, mantenido su nombre original, lo que permite referirse a ellos de forma consistente con la literatura que se puede encontrar, al igual que se ha hecho anteriormente al referirse a los mismos.

3.1. Graph heuristic (GH)

Se trata de una de las técnicas más antiguas utilizadas para resolver el problema de creación de horarios y en particular de los horarios de exámenes. Para ello se realiza un proceso de coloreado de grafos donde se asignan colores a los vértices, de forma que no puede haber dos vértices adyacentes con el mismo color. Para el problema de horarios de exámenes, los vértices representan las asignaturas y los arcos entre vértices representan las restricciones fuertes. Las restricciones débiles se suelen evaluar de forma separada después del proceso de coloreado del grafo. Como ejemplos se pueden citar los artículos de (de Werra, 1985), (Burke, Elliman & Weare, 1994) o (Burke, Kendall, Mısır, Özcan, Burke, Kendall, Özcan & Mısır, 2004).

Este método de coloreado de grafos se utilizaba inicialmente por sí mismo como método de resolución del problema (Carter, 1986). Sin embargo, en la actualidad suele verse como método para crear una solución inicial a partir de la cual iniciar procesos de optimización adicionales creando métodos híbridos junto con otros métodos (Qu, Burke & McCollum, 2009) (Mandal & Kahar, 2015). GH es un método relativamente fácil de implementar y genera soluciones razonables en poco tiempo. El proceso ordena las asignaturas de acuerdo a un determinado criterio (por ejemplo, las restricciones que tiene) y, después, se van planificando una a una en las franjas que resultan más apropiadas. Las estrategias más habituales para ordenar las asignaturas son (Burke, McCollum, Meisels, Petrovic & Qu, 2007):

a) **Saturation degree (SD)**: Ordena las asignaturas empezando por aquellas que tienen menor número de franjas disponibles para planificarlas. La prioridad de las asignaturas va cambiando dinámicamente según se construye la solución.
b) **Largest degree (LD)**: Se ordenan las asignaturas según el número de conflictos que genera con otras asignaturas. Se planifican primero las asignaturas que tienen un mayor número de conflictos.
c) **Largest weighted degree (LWD)**: Es similar a la anterior excepto que se ponderan las asignaturas teniendo en cuenta el número de estudiantes matriculados. Si existen asignaturas con el mismo orden se da prioridad a aquellas con un mayor número de alumnos matriculados.
d) **Largest enrollment (LE)**: Ordena las asignaturas teniendo en cuenta el número de estudiantes matriculados, planificando primero las que tiene el mayor número de alumnos.

e) **Color Degree (CD)**: Las asignaturas se ordenan teniendo en cuenta el número de conflictos con aquellas que se han planificado. Este orden va cambiando según se van planificando asignaturas.

f) **Random ordering (RO)**: Ordena las asignaturas aleatoriamente.

De acuerdo con (Qu, Burke, McCollum, Merlot & Lee, 2009) en su revisión indica que normalmente parece que Largest degree (LD) y Saturation degree (SD) generan mejores resultados que el resto de estrategias. Sin embargo, (Carter, 1986) en su estudio con las distintas estrategias sobre problemas tanto reales como generados aleatoriamente indican que ninguna de las estrategias mostraba diferencias significativas en su rendimiento sobre el resto teniendo en cuenta todos los problemas sobre los que se probaron.

En (Qu & Burke, 2009) se investiga el uso de las distintas estrategias de GH junto con las metodologías Hiperheurísticas (HH), donde las HH se utilizan para elegir la estrategia GH más apropiada para generar los horarios. La idea se basa en que por sí mismas no son siempre un método apropiado para tratar la complejidad de los problemas de horarios, generando malas soluciones para algunos problemas. Por otra parte, en investigaciones más recientes se están utilizando para generar la solución inicial para metaheurísticas (Mandal & Kahar, 2015; Qu & Burke, 2009).

En los últimos años, GH ha evolucionado de forma que se está utilizando de distintas formas como en la hibridación con otros métodos de búsqueda.

3.2. Hill-Climbing (HC)

Hill-Climbing (HC), también llamado ascenso de colinas es una técnica de las clásicas. En cada paso de la iteración, se selecciona una solución candidata s' de forma aleatoria dentro de la vecindad $N(s)$ de la solución actual. Esta solución candidata s' se acepta y sustituye a la solución actual si $f(s')$ supone una mejora con respecto a $f(s)$, como se puede observar en la **¡Error! No se encuentra el origen de la referencia.**. Este método es muy sencillo y fácil de implementar. Como desventaja hay que indicar que se puede ver atrapado fácilmente en un óptimo local. Es por ello que se suele hibridar HC con otros métodos como Simulated annealing (SA) o algoritmos evolutivos. Por ejemplo (Merlot, Boland, Hughes & Stuckey, 2003) utiliza para resolver el problema de horarios de exámenes un método en varias fases que incluye programación con restricciones, Simulated annealing y Hill-Climbing.

En (Burke, Newall & Weare, 1996) presentan una hibridación de algoritmos genéticos con Hill-Climbing para mejorar las soluciones individuales. Este método de hibridar algoritmos se llama también algoritmos meméticos que se verán más adelante. En (Kendall & Hussin, 2005) se aplican hiperhurísticas y Hill-Climbing al problema de horarios de exámenes. En (Müller, 2009) se utiliza la combinación de Hill-Climbing con

Great Deluge y con Simulated annealing para el problema de horarios de exámenes del concurso ITC 2007.

En (Burke & Bykov, 2008) proponen una modificación de Hill-Climbing que denominan Late Acceptance Hill-Climbing. Este método retrasa un número de pasos la comparación entre soluciones candidatas y la solución actual. En el artículo se muestra como esta variación es capaz de generar buenas soluciones comparadas con otros métodos. Este método intenta buscar un mecanismo para escapar de óptimos locales.

```
sinceLastMove := 0
While sinceLastMove < MaxMoves do
Choose exam e and period t at random with t != period(e)
If penalty(e, t) < penalty(e, period(e)) then
      Move exam e to period t
      sinceLastMove := 0
Else
      sinceLastMove += 1
Endif
Done
```

Figura 1: Algoritmo de Hill-Climbing (Burke & Newall, 2002).

3.3. Tabu search (TS)

Tabu search (TS) también llamado búsqueda tabú, es un método propuesto por (Glover, 1986) que funciona de forma muy parecida a Hill-Climbing pero incorpora una memoria para diversificar en la exploración del espacio de estados. Según (Glover & Laguna, 1997) TS es "*una metaheurística que guía a un proceso de búsqueda heurístico locas para explorar el espacio de soluciones más allá del óptimo local*".

En la Figura 2 puede verse el proceso de funcionamiento de TS. El proceso empieza con una solución inicial s_0. La búsqueda explora un subconjunto $N'(s)$ de la vecindad $N(s)$ de la solución actual s. El proceso de búsqueda favorece el proceso de exploración del espacio de estados aceptando soluciones de la vecindad con el menor valor, incluso aunque dicho valor sea peor que la solución actual. Esta aceptación de valores que no mejoran el estado actual permite que en el proceso de exploración se salga del óptimo local. Sin embargo, al elegir una solución del subconjunto $N'(s)$ se puede caer en un bucle. Para evitarlo, se usa una memoria, llamada lista tabú, para guardar las últimas soluciones elegidas, que no se utilizan para realizar más un cierto número de iteraciones (que depende del tamaño de la lista tabú). Sin embargo, se utiliza también un mecanismo denominado criterio de aspiración para que la solución no esté en la lista tabú si su evaluación es una solución mejor que la actual.

```
Step 1. Choose an initial solution i in S. Set i*= i and k = 0.
Step 2. Set k=k+1 and generate a subset V* of solution in N(i, k)
such that either one of the tabu conditions t_r(i, m) ∈ T_r is
violated (r = 1,...,t) or at least one of the aspiration
conditions ar(i, m) ∈ A_r(i, m) holds (r = 1,...,a).
Step 3. Choose a best j=i of m in V* (with respect to f or to the
function f') and set i=j.
Step 4. If f{i)<f{i*) then set i*=i.
Step 5. Update tabu and aspiration conditions.
Step 6. If a stopping conditions is met then stop. Else go to
Step 2.
```

Figura 1: Algoritmo de Tabu Search (Hertz, Taillard & De Werra, 1995).

En (Di Gaspero & Schaerf, 2000) se añade un mecanismo de penalización e incumplimientos de restricciones en la creación de los horarios de exámenes. El mecanismo de penalización usa un peso variable a las restricciones (tanto duras como blandas) para mejorar la exploración del espacio de estados. Consideran dos posibilidades: el incumplimiento de las restricciones duras o el incumplimiento de las restricciones duras y de las blandas. Estas dos posibilidades, combinadas con una lista tabú de tamaño variable y una buena solución inicial, son capaces de generar buenas soluciones, según indica. En (Di Gaspero, 2002) de título "*Recolour, shake and kick: A recipe for the examination timetabling problem*" aplica una combinación Búsqueda Tabú con múltiples vecindades. La estrategia aplica una optimización de la función objetivo (*recolor*), perturbando la solución actual (*shake*) u obteniendo más mejoras (*kick*). El algoritmo de *recolor* y *shake* se aplican en secuencia hasta que no existen más mejoras y se continúa después con el algoritmo *kick*. Indica que esta estrategia mejora la Búsqueda Tabú con una vecindad.

En (White & Xie, 2000) se implementa un algoritmo TS sobre dos de los conjuntos de datos de Toronto (Carter, Laporte & Lee, 1996). Usan dos listas tabú: una de corto plazo para soluciones recientes y otra de largo plazo por frecuencia. También se incluyó un proceso de relajación. Los resultados mostraban que con mayores listas tabú de largo plazo generaban mejores soluciones que otros algoritmos. Más tarde en (White, Xie & Zonjic, 2004) aplicaron esta estrategia al resto de conjuntos de datos de Toronto. Los resultados mostraban que con mayores listas tabú de largo plazo mejoran la calidad de las soluciones e indican que la relajación tabú es una buena estrategia por que ayuda a que se explore una mayor porción del espacio de estados.

En (Paquete & Stützle, 2002) desarrollan TS tras un proceso de búsqueda iterativa en grafos para ordenar las prioridades de las restricciones. Las restricciones se consideran de dos formas: una restricción cada vez desde la prioridad más alta y otra

con todas las restricciones a la vez empezando con la de mayor prioridad. La segunda estrategia consiguió mejores resultados aunque la primera era más consistente en ellos. En el algoritmo se modificaba dinámicamente el tamaño de la lista tabú teniendo en cuenta el número de incumplimiento de las restricciones en las soluciones. También indican que el tamaño de la lista tabú debe aumentar con el tamaño del problema.

En (Wilke & Ostler, 2010) aplicaron la búsqueda tabú al problema de horarios académicos y lo compararon con varios métodos (Simulated annealing, algoritmos genéticos y Branch & bound) para crear un programa que sea capaz de resolver distintos problemas de horarios. SA generaba los mejores resultados, pero TS era capaz de generar un buen resultado en muy poco tiempo. (Mushi, 2006) implementó un algoritmo TS para generar horarios de clases para minimizar la penalización de la solución con incumplimiento de restricciones. En la investigación compara los resultados con los que se generaban manualmente. El sistema propuesto genera mejores resultados que los manuales. Utilizan dos estrategias de movimientos, mover a otra franja e intercambiar asignaturas, junto con un criterio de aspiración que permite sacar una solución de la lista tabú si supone una mejora sustancial de la solución.

También TS se ha hibridado con otros métodos. (Abdullah, Burke & Mccollum, 2005) implementa una lista tabú junto con el método Variable neighbourhood search (VNS). (Abdullah, Shaker, McCollum & McMullan, 2009) hibrida TS con algoritmos meméticos (MA), donde la lista tabú se usa para contener las estructuras de la vecindad que no pueden generar mejores soluciones tras las operaciones de cruze y mutación. Se usan las vecindades que permiten mejoras hasta que no se consigue mejorar más la solución. Esta estrategia generar buenas soluciones para cuatro de los conjuntos de datos de Toronto. (Frausto-Solís & Alonso-Pecina, 2007) hibridaron SA con TS. Posteriormente (Abdullah & Turabieh, 2012) extendió el trabajo anterior para investigar estructuras multi-vecindad. Por otra parte (Kendall & Hussin, 2004) investigaron hyper-heurísticas de TS con los datos de la institución de uno de los autores.

3.4. Simulated annealing (SA)

El método de Simulated Annealing (SA) fue propuesto por Kirkpatrick en 1983 (Kirkpatrick, Gelatt, Vecchi & others, 1983). Viene motivado por el proceso de enfriamiento de los metales calentados a alta temperatura y dejándolos enfriar suavemente hasta que cristalizan y no se producen más cambios. Este proceso de enfriamiento para cada material puede variar y resulta muy importante.

En SA se empieza desde una solución inicial, generada por ejemplo con una heurística constructiva, y aceptará nuevas soluciones de la vecindad siempre que sean mejores que la actual. Si son peores se aceptarán con una probabilidad $p = e^{-\frac{d}{t}}$ donde d es la diferencia entre el valor objetivo y el actual para la solución actual, t es un

parámetro temperatura que va descendiendo en el proceso de acuerdo a una estrategia de enfriamiento. Este proceso se muestra en la **Figura 2**.

```
T = To
While T>Tmin do
      Choose exam e and period t at random with t != period(e)
      If penalty (e, t) <= penalty (e, period (e)) then
            Move exam e to period t
      Else
            Move exam e with probability
            exp((penalty(e, period(e)) - (penalty (e, t)/T))
      Endif
      T = reduce(T)
Done
```

Figura 2: Algoritmo de Simulated Annealing (Burke & Newall, 2002).

Según (Thompson & Dowsland, 1998) el proceso de enfriamiento influye de manera muy importante en la calidad de solución final. Los procesos de enfriamiento más rápidos tienden a hacer que el proceso converja en un óptimo local, mientras que los procesos de enfriamiento más lentos generalmente generan mejores soluciones pero aumenta el tiempo de búsqueda. Normalmente se utiliza un proceso de enfriamiento geométrico que termina tras un número dado de pasos, cuando se alcanza cierta temperatura o se obtiene una solución.

SA se ha empleado con éxito en muchas áreas, entre ellas la generación de horarios de exámenes. (Thompson & Dowsland, 1995) resolvieron el problema en dos fases, una fase constructiva, generando una solución inicial factible, y una de fase de mejora, mejorando la calidad de la solución. Usaron un proceso de enfriamiento adaptativo. Sus resultados mostraban que mejoraban la calidad que usando un enfriamiento geométrico. (Thompson & Dowsland, 1996, 1998) investigaron con distintos enfriamientos y generación de la vecindad. Los resultados mostraban que la vecindad de cadenas de Kempe (*kempe chains*) generaba las mejores soluciones. El resultado se debe a que este proceso permite generar un mayor número de exámenes que mover, lo que redunda en una mejora de la calidad de la solución.

En (Bullnheimer, 1997) trata cómo se adapta un modelo para los problemas de asignación cuadrática ("Quadratic Assignment Problems") para el problema de horarios de exámenes de tamaño pequeño de la Universidad de Maddeburg. El modelo permite controlar cuando debes espaciarse los exámenes en conflicto. Se utiliza SA con dos estructuras de vecindad.

En (Merlot, Boland, Hughes & Stuckey, 2003) se emplea SA a partir de las soluciones iniciales que se crean utilizando programación con restricciones y seguida después de HC para intentar mejorar la solución final obtenida. Así mismo se emplea una vecindad con una cadena Kempe. Obtuvieron muy buenos resultados y sigue teniendo algunos de los mejores resultados conocidos. Los autores sugieren que el uso de generación de soluciones con búsqueda local será el dominante en el futuro.

En (Duong & Lam, 2004) se emplea SA sobre las soluciones iniciales que se genera utilizando programación con restricciones para el problema de la Universidad de tecnología de Ho Chi Minh City. Utilizan una cadena Kempe en el SA, con un proceso de enfriamiento que se establece afinando los parámetros del proceso. Los autores indican que cuando el tiempo disponible es limitado es crucial ajustar los parámetros del SA al problema concreto.

En (Wright, 2001) utilizan un SA con búsqueda guiada por sub-costes para el problema de horarios académicos. Los sub-costes que incorpora a SA se usan para modificar la función de probabilidad estándar de aceptación de soluciones peores utilizando un incremento de coste en la fórmula de probabilidad. Sus resultados muestran que esta característica adicional mejora los resultados del SA.

(Frausto-Solís & Alonso-Pecina, 2007) hibridaron SA con TS para resolver el concurso de ITC 2007. Su método se divide en dos fases: una primera fase es la construcción de una solución factible con SA y otra una fase de mejora también con SA añadiendo las franjas adicionales. Sin embargo, si SA no muestra ninguna mejora durante esta segunda fase el algoritmo continúa con TS. Este algoritmo genera buenas soluciones aunque no las mejores.

(Zhang, Liu, M'Hallah & Leung, 2010) aplicó SA al problema de creación de horarios de exámenes proponiendo una nueva estructura de vecindad que intercambia exámenes entre pares de franjas. En su artículo indican que esta estructura de vecindad aumenta la eficiencia y rendimiento de SA. Sus resultados muestran que esta heurística probándola en dos conjuntos de datos de prueba se comporta mejor que otros métodos publicados.

En (Wilke & Ostler, 2010) comparan el rendimiento de los algoritmos TS, SA, GA y Ramificación y poda con problemas reales utilizando una plataforma homogénea. En sus resultados recomiendas utilizar SA pues es el algoritmo con el mejor equilibrio entre tiempo de cómputo y calidad de las soluciones.

3.5. Great Deluge Algorithm (GDA)

En 1993 Dueck presenta el algoritmo Great Deluge (GOA) que funciona de forma muy similar a SA. La diferencia principal con SA es que utiliza un límite superior, denominado normalmente nivel de agua (WATER-LEVEL), como límite de aceptación

de soluciones en lugar de una probabilidad de aceptación asociada a una temperatura. El algoritmo empieza con un límite igual a la calidad de solución inicial. Sólo acepta soluciones peores si el nuevo coste es menor que el límite, que se va reduciendo con cada iteración siguiendo una estrategia de reducción. En la **Figura 3** se puede ver la estructura de este algoritmo. La ventaja más importante de este algoritmo con respecto a SA es que sólo existe un único parámetro (la tasa de reducción del límite), ya que además las técnicas metaheurísticas suelen ser muy dependientes del ajuste de sus parámetros (Sanja Petrovic & Edmund Burke, 2004).

```
Choose an initial configuration
Choose the "rain speed" UP > 0
Choose the initial WATER-LEVEL> 0
      Opt: choose a new configuration which is a stochastic small
pertubation of the old configuration
      Compute E := quality (new configuration)
      If E>WATER_LEVEL then
            old configuration := new configuration
            Water_level := water_level+ up
      If a long time no increase in quality or too many
iterations
      Then stop
Goto Opt
```

Figura 3: Algoritmo Great Deluge para maximizar (Dueck, 1993).

(Dueck, 1993) aplicó GDA al problema del viajante. El factor de descenso era la diferencia entre los extremos y el tamaño del camino actual divido por 500 o un factor fijo de 0,01. Con este algoritmo generaba buenas soluciones. (Burke & Newall, 2002) aplicaron GDA a los problemas de horarios. El factor de descenso que utilizaban lo calculaban como la solución inicial multiplicado por un factor que indicaba el usuario dividido por el número de iteraciones. El algoritmo lo ejecutaban 200.000.000 de iteraciones o terminaba cuando no conseguía ninguna mejora durante 1.000.000 de iteraciones. Indican que el resultado final que era mejor que otras soluciones propuestas.

(Burke, Bykov, Newall & Petrovic, 2004) implementaron un GDA con tiempo predefinido para el problema de horarios de exámenes. El algoritmo incluye dos parámetros de ajuste: el tiempo de cómputo y una estimación del coste de la solución. La velocidad de reducción se calcula como la diferencia entre la solución inicial y la solución deseada dividida por el tiempo de cómputo. Según sus autores este algoritmo genera buenas soluciones.

(Mcmullan, 2007) realizó una extensión a GDA, donde utiliza una velocidad de reducción en pasos (con una reducción proporcional al 50% de la diferencia total en la primera fase y el 25% en el resto de pasos. Este proceso fuerza al algoritmo a alcanzar mejores soluciones lo antes posible. Además permite que el algoritmo se recaliente, de forma similar a como se hace en SA, lo que amplía el límite permitiendo, por tanto aceptar movimiento peores. Posteriormente (McCollum, Schaerf, Paechter, McMullan, Lewis, Parkes, Gaspero, Qu & Burke, 2010) aplicaron esta extensión de GDA a los conjuntos de datos de horarios de exámenes del ITC 2007 usando una estrategia en dos fases, construcción y mejora. La solución inicial se construye usando una heurística de ordenación adaptativa (Burke & Newall, 2004). La mejora se consigue utilizando GDA extendido que incluye un mecanismo de recalentamiento. Este método generó buenas soluciones comparadas con otros resultados publicados.

(Abdullah, Shaker, McCollum & McMullan, 2009) hibrida GDA con TS. El algoritmo aplica cuatro movimientos de vecindad en cada iteración y selecciona de ellos la mejor solución generada. Si no hay mejora en un tiempo dado, se aumenta el límite aleatoriamente entre 0 y 3. (Turabieh & Abdullah, 2011) hibridan GDA con mecanismos electromagnéticos (EM). El método EM utiliza un mecanismo de atracción/repulsión que mueve la vecindad hacia mejores soluciones. Indican que este método generó buenos resultados en algunos de los conjuntos de datos.

3.6. Variable neighbourhood search (VNS)

(Mladenović & Hansen, 1997) presentaron Variable Neighbourhood Search (VNS) con la idea de disponer de más de una estructura de vecindad e ir cambiándola durante el proceso de búsqueda local. Esta idea se basa en que la mayoría de las metaheurísticas son muy dependientes de los ajustes en sus parámetros (Petrovic & Burke, 2004). Así mismo, el comportamiento de las metaheurísticas depende tanto de su mecanismo de búsqueda como de la estructura de vecindad utilizada (Thompson & Dowsland, 1998), (Ahuja, Orlin & Sharma, 2000).

En la **Figura 4** se muestra el algoritmo VNS. En primer lugar determina el conjunto de estructuras de vecindad predefinidas *k*, donde $k = 1, \dots, K$, siendo K el número total de estructuras de vecindad que se usa en la búsqueda. El proceso genera una solución s' usando la k-ésima vecindad. A partir de esta solución s' se utiliza un método de búsqueda local hasta que se obtiene un óptimo local s''. Esta solución s'' se acepta si $f(s'')$ es mejor que $f(s)$. Siempre que una estructura de vecindad genera una mejor solución, la búsqueda comienza de nuevo desde la primera estructura de vecindad $k = 1$. En caso contrario, se usa la siguiente vecindad $k = k + 1$.

```
Initialization: Select the set of neighbourhood structures N_k,
for k=1,... ,k_max, that will be used in the search; find an
```

initial solution x; choose a stopping condition; Repeat the following sequences until the stopping condition is met:
(1) Set k = 1;
(2) *Repeat* the following steps until $k=k_{max}$:
(3) *Shaking*. Generate a point x' at random from the kth neighbourhood of ($x' \in N_k(x)$);
(4) *Local search*. Apply some local search method with x' as initial solution; denote with x'' the so obtained local minimum;
(5) *Move or not*. If the local minimum x'' is better than the incumbent x, move there (x=x''), and continue the search with $N_1(k=1)$; otherwise, set k=k+1;

Figura 4: Algoritmo Variable neighbourhood search (Mladenović & Hansen, 1997).

(Abdullah, Burke & Mccollum, 2005) proponen una estrategia combinada de VNS con TS. La lista tabú se fija en 2 y se usa para mantener las estructuras de vecindad que se comportan peor, evitando que se elijan en las siguientes iteraciones, por lo que se fomenta explorar áreas diferentes del espacio de estados. También investigan las distintas estrategias de ordenación de las vecindades, forzando a que la búsqueda vuelva a la primera de ellas si se encuentra una mejora.

(Burke, Eckersley, McCollum, Petrovic & Qu, 2010) hibridan VNS con algoritmos genéticos. Investigan el uso de diferentes estructuras de vecindad entre las que se incluyen: ascenso/descenso que acepta movimientos a peor con una cierta probabilidad, VNS con límites que implica mover un examen que genera una gran penalización, vecindades específicas del problema que implica reducir el número de vecindades y diferentes estrategias de inicialización, como por ejemplo inicialización ávida o aleatorizada. Del análisis estadístico que realizan se demuestra que los problemas son dependientes de sus vecindades, donde ciertas vecindades pueden generar mejoras en un problema pero no en otros.

3.7. Genetic algorithm (GA)

Los Algoritmos genéticos (Holland, 1975) son un mecanismo muy popular de búsqueda basado en poblaciones que utiliza la teoría de la evolución biológica para generar mejores soluciones de generación en generación. Este método utiliza los conocidos operadores biológicos de selección, cruce y mutación para manipular las soluciones, que se denominan cromosomas en el algoritmo. Estas operaciones se utilizan durante un número de generaciones para ir mejorando el coste de las soluciones. Los cromosomas se representan como una cadena que contiene la información de la solución. Durante su aplicación hay que considerar varios parámetros como el tamaño de la población, la tasa de cruce, la tasa de mutación, el número de generaciones, etc. (Golberg, 1989; Pham & Karaboga, 2012).

Los algoritmos genéticos empiezan con una población inicial, que suele estar compuesta de individuos aleatorios, donde cada uno de ellos es una solución. Cada uno de estos individuos tiene un coste asociado (*fitness*) que se evalúa con su función objetivo. A continuación se produce una fase de selección donde se eligen algunos de los individuos según un operador de selección, antes de pasar al proceso de combinación. En el proceso de combinación se utilizan los operadores de cruce y mutación para explorar el espacio de estados, creando nuevos individuos o soluciones. Estos nuevos individuos sustituyen a individuos ya existentes, normalmente eliminando los peores según su *fitness*[1]. Este proceso se repite hasta que se cumple la condición de terminación, que puede ser un cierto número de iteraciones o un cierto tiempo. En la **Figura 5** se puede ver la estructura de un algoritmo genético.

```
InitialisePopulation P
For each sol_i from P
      CalculateFitness (sol)
Repeat
      select two parents sol_1 and sol_2 from P
      child = crossover (sol_1, sol_2)
      mutate (child)
      calculateFitness (child)
      replaceSome (p, child)
until stop condition not satisfied
```

Figura 5: Estructura de Algoritmo genético (Čupić, Golub & Jakobović, 2009).

(Corne, Fang, Mellish & Corne, 1993) utilizan un GA para el problema de horarios de exámenes. Utilizan un cromosoma cuyo tamaño es el número de exámenes. Para evitar soluciones imposibles proponen en (Ross & Corne, 1995) utilizar sólo el operador de mutación para generar soluciones viables. Esta modificación genera mejores resultados que un operador de cruce uniforme. Añaden también un mecanismo de reparación que resuelve posibles soluciones imposibles debidas a la representación del cromosoma.

(Chu & Fang, 1999) investigaron el uso de los algoritmos genéticos y TS para el problema de horarios de exámenes y compararon sus resultados, concentrándose en la calidad de las soluciones y el tiempo necesario para generarlas. El resultado es que TS genera mejores soluciones con menor tiempo de cómputo que GA. Sin embargo, GA puede generar varias distintas soluciones cercanas al óptimo simultáneamente.

[1] Se mantiene el término original de *fitness* ya que es el que se encuentra en la literatura, de forma que no induzca a confusión, por ser un término propio de los algoritmos genéticos.

Normalmente la calidad de las soluciones que generan los algoritmos basados en poblaciones son peores que los métodos basados en trayectorias. La razón estriba en una prematura convergencia, ya que los algoritmos basados en poblaciones se basan más en la exploración que en la explotación de soluciones.

3.8. Ant Colony Optimisation (ACO)

(Dorigo, 1992) es la tesis en la que se propuso un método de optimización inspirado en el comportamiento de las hormigas y la forma en la buscan la comida, un método cooperativo con el que las hormigas consiguen buscar un camino entre su hormiguero y la comida basado en el depósito de feromonas en su camino. La estructura de este algoritmo se puede ver en la **Figura 6**.

1. Assign parameters
2. Calculate s_0 and initialize pheromone matrix
3. Assign a random starting member to all ants
4. Add the rest of the member groups sequentially
5. Evaluate objective function and rank ants
6. Check for convergence
7. Update pheromone matrix using the top 15% of ants
8. Repeat steps 3–7 until convergence

Figura 6: Estructura de un algoritmo de ACO (Dowsland & Thompson, 2005).

(Dowsland & Thompson, 2005) investigan el uso de ACO al problema de horarios de exámenes. El objetivo de la investigación era por una parte comprobar cómo se comportaba la herramienta ANTCOL en grafos de horarios con los conjuntos de grafos que habían creado (Costa & Hertz, 1997) y, por otra parte, identificar combinaciones de heurísticas constructivas que funcionasen con el problema. Los resultados experimentales mostraban que las propuestas para la herramienta ANTCOL aplicada al problema de horarios de exámenes resultaban competitivas con otras estrategias publicadas minimizando el número de franjas horarias necesarias para generar horarios factibles.

En (Eley, 2006a, 2006b) utilizan un Max-Min y ANTCOL para el problema de horarios de exámenes. Se prueban dos algoritmos con los conjuntos de datos de Toronto. Sin embargo, incluyen un valor de penalización de 10.000 ya que el algoritmo propuesto no garantizaba soluciones sin conflictos. Se utilizaban 50 hormigas con valores fijos para la tasa de evaporación y del valor de intervalo de feromonas. Además prueban distintos factores de ponderación (a y p). Los resultados muestran que estas estrategias no generan muy buenos resultados aunque su rendimiento es similar a otros métodos.

3.9 Memetic Algorithm (MA)

Los algoritmos genéticos realizan la búsqueda en todo el espacio de estados sin centrarse en una parte concreta del mismo, lo que puede hacer que se pierda información muy valiosa de uno de los individuos (Acan & Tekol, 2003). Sin embargo, la ventaja de los algoritmos genéticos es que realizan la búsqueda en muchas direcciones y desde muchos puntos distintos utilizando un conjunto de soluciones candidatas lo que puede resultar una ventaja si se incluye un proceso de búsqueda local para mejorar el mejor cromosoma solución. A este proceso general sobre búsqueda de poblaciones se le conoce como algoritmo memético (Moscato, 1989). Los MA son estrategias evolutivas combinadas con búsqueda local.

El origen del concepto de memético se basa en las ideas de Dawkins (Dawkin, 1976), que describe los memes actuando como unidades de información que se van expandiendo por la sociedad. La desventaja es que el paso de una generación a la siguiente tarda más tiempo, pero se puede justificar si se consigue mejorar más en cada generación que si no se utiliza la búsqueda local. En la **Figura 7** se puede observar la estructura de un MA.

En (Burke, Newall & Weare, 1996) utilizan un MA para el problema de horarios de exámenes. Se incluye una tasa de mutación baja y otra alta así como HC determinista. El objetivo era generar una solución factible con una penalización lo más baja posible. Se probó con los conjuntos de datos de Nottingham y de Toronto. Los resultados demostraron que el método puede generar soluciones factibles y de buena calidad. Posteriormente, en (Burke & Newall, 1999) extendieron el trabajo anterior y propusieron un MA multiestado. El algoritmo utiliza un subconjunto de los exámenes mientras el siguiente subconjunto se planifica sobre los eventos ya planificados. Se utiliza un horario de tamaño fijo para generar el horario. Para evitar que se produzcan incompatibilidades, los exámenes se ordenan por dificultad (Largest Degree, Color Degree y Saturation Degree), junto un una estrategia de prueba previa. Los resultados indicaban que la calidad de las soluciones obtenidas es superior a las que se obtienen con los MA por sí mismos.

```
Create initial population
Repeat
      1Take each individual in tum:
            Choose a mutation method (light or heavy mutation)
            Apply mutation operator to chosen individual
            Apply hill-climbing to individual just created.
            Insert it into the population.
      2 Select a half of them to reduce the population to its
original size
Until termination condition is true
```

Figura 7: Esquema de algoritmo Memetic (Hung, Binh & Anh, 2005).

En (Hung, Binh & Anh, 2005) se presenta una modificación al algoritmo de (Burke, Newall & Weare, 1996) en el que generar horarios de exámenes para la Ho Chi Minh City University of Technology. Aplican los mismos operadores evolutivos que Burke et al, pero añaden HC con penalización y con restricciones. Su modificación genera buenas soluciones pero a costa de tardar un mayor tiempo.

Otros trabajos de MA incluyen los de (Abdullah & Turabieh, 2012) con TS o los de (Krasnogor & Smith, 2005).

3.10. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO), (Optimización por enjambres de partículas), es un método de optimización que se presentó inicialmente en 1995 (Eberhart, Kennedy & others, 1995). Se trata de un algoritmo de tipo evolutivo que se inspira en las bandadas de pájaros o en los bancos de peces.

En este algoritmo los elementos que se manejan son partículas que se mueven en un espacio multidimensional. Cada una de las partículas tiene una velocidad y una mejor posición que ha alcanzado en un momento dado. En este algoritmo se utiliza una población, un enjambre, en donde cada partícula representa una posible solución. El enjambre se inicializa de forma aleatoria. El proceso de búsqueda se realiza realizando cambios en la velocidad y posición de cada una de las partículas según se generan nuevas generaciones. De hecho, las partículas se mueven en un espacio de búsqueda N-dimensional siguiendo a la mejor partícula del enjambre. En cada iteración se evalúa la posición de cada una de las partículas. Cada una de las partículas guarda la mejor posición que haya encontrado en algún momento P_{best} y la mejor posición de la mejor partícula del enjambre, llamada G_{best}. Este proceso se puede ver en la **Figura 8**.

```
1. Initialize an array of particles with random positions and
velocities on D dimensions
2. Evaluate the desired minimization function in D
variables
3. Compare evaluation with particle's previous best value
(PBEST[]): If current value < PBEST[] then PBEST[] =
current value and PBESTx[][d] = current position in Ddimensional
hyperspace
4. Compare evaluation with group's previous best
(PBEST[GBEST]): If current value < PBESTCGBEST]
then GBEST=particle's array index
5. Change velocity by the following formula:
W[dI = W[dI +ACC-CONST*rand()*(PBESTx[] [d] - PresentX[] [d])
+ ACC-CONST*rand()*(PBESTx[GBEST[]d ] - PresentX[l[d]), and
6. Move to PresentX[][d] + v[][d]: Loop to step 2 and
repeat until a criterion is met
```

Figura 8: Algoritmo de Optimización por enjambre de partículas (Eberhart, Kennedy & others, 1995).

En (Fealko, 2005) investiga el uso de PSO al problema de creación de calendarios de exámenes. Se investiga de forma metódica el impacto de los distintos parámetros en el resultado generando un perfil de rendimiento del algoritmo al conjunto de prueba utilizado. Compara sus resultados con los del conjunto de datos de Toronto, generando unos resultados en el entorno de los publicados.

En (Chu, Chen & Ho, 2006) confirman que PSO se puede aplicar a la resolución de calendarios de exámenes. Para ello proponen un conjunto de datos de ejemplo y lo utilizan para ver el comportamiento del algoritmo, concluyendo su utilidad para este tipo de problemas.

En (Ahmed, Sajid, Ali & Bukhari, 2011) se utiliza PSO como hiperheurística para resolver el problema para escoger entre heurísticas más simples. Aplican su algoritmo al conjunto de datos de su Universidad y concluyen que el método es muy eficiente y los resultados son muy prometedores.

En (Montero, Riff & Altamirano, 2011) utilizan el algoritmo para aplicarlo a la generación de horarios de la Universidad Técnica Federico Santa María de Chile (UTFSM). Comparan la ejecución del algoritmo con la creación a mano o mediante un algoritmo de examen hacia adelante (forward checking). Los resultados obtenidos son mejores que los dos casos con los que compara.

(Ahandani, Baghmisheh, Zadeh & Ghaemi, 2012) resuelve un subconjunto de los conjuntos de datos de Toronto utilizando un algoritmo en dos fases empezando con PSO y usando Hill-Climbing para terminar el proceso de búsqueda. En sus resultados indican que son muy competitivos con otros algoritmos propuestos.

3.11. Hyper-heuristics (HH)

El desarrollo de las hiperheurísticas (HH) se desarrolló por la necesidad de aumentar el nivel de generalidad en la resolución automática de problemas (Burke, Kendall & Soubeiga, 2003). La mayoría de las meteheurísticas ya comentadas tratan directamente con el espacio de estados del problema. Las HH tratan el espacio de estados de las heurísticas (Burke, Petrovic & Qu, 2006).

Al comienzo se trataban las HH como una estrategia de elegir inteligentemente en cuál era la mejor metaheurística para resolver un problema (Kendall & Hussin, 2004; Pillay & Banzhaf, 2009; Qu & Burke, 2009). En este sentido el sistema se dotaba de un conjunto de heurísticas y el problema era encontrar la mejor secuencia de las mismas para resolver el problema de fondo de manera indirecta.

En (Burke, Kendall & Soubeiga, 2003) se utiliza TS como heurística de alto nivel para buscar en el espacio de estados de estrategias de movimiento para el problema de horarios de clases y de turnos de enfermeras. Indican que el resultado obtenido es bueno teniendo en cuenta la generalidad del método. Más tarde en (Burke, Silva & Soubeiga, 2005) extienden el trabajo anterior investigando qué heurísticas de bajo nivel son más apropiadas y efectivas para distintos objetivos individuales en asignación de espacios mutiobjetivo y en problemas de horarios, utilizando también TS como selector. También indican resultados prometedores comparados con otras técnicas.

En (Burke, McCollum, Meisels, Petrovic & Qu, 2007) utilizan una HH basada en TS usando heurísticas de grafo para el problema de los horarios. El espacio de estados representa un conjunto de heurísticas de bajo nivel, en lugar de ser soluciones el problema. TS se utiliza para buscar en la lista de heurísticas de bajo nivel de forma aleatoria sin considerar los detalles de las soluciones reales. Esta elección de heurísticas se utiliza para ordenar los eventos que todavía no están planificados. Los resultados obtenidos se encuentran dentro del rango de los buenos resultados en otros trabajos. En (Qu & Burke, 2009) se extiende el trabajo anterior proponiendo una estrategia adaptativa, en lugar de TS, de forma que se hibridan dinámicamente las heurísticas durante la generación de la solución. El resto de las heurísticas (LD, LWD y LE) se hibridan aleatoriamente en la lista de SO. Indican que esta estrategia es un método más eficiente y simple que requiere menos tiempo de cómputo y los resultados son comparables con los mejores publicados.

También se pueden ver otros trabajos sobre HH en (Burke, Kendall, Newall, Hart, Ross & Schulenburg, 2003) donde utilizan HH con un GA, como extensión de (Ross, Hart & Corne, 1997) donde ya en 1997 indicaban «*Sin embargo, todo esto sugiere que vale la pena el uso de GA en la investigación sobre generación de horarios. Sugerimos que es mejor utilizar un GA en la búsqueda de un buen algoritmo en lugar de utilizarlo para buscar una solución específica a un problema concreto*». En (Kendall & Hussin, 2005) utilizan HH con TS.

4. Conclusiones

De todos los algoritmos descritos anteriormente se puede extraer que se han utilizado técnicas muy numerosas para la resolución del problema de creación de horarios de exámenes. Entre las más habituales se pueden citar los algoritmos evolutivos y la búsqueda tabú (Tabu Search). También se pueden encontrar numerosas referencias donde se utilizan las técnicas de Hill-Climbing y Simulated Annealing.

En esta revisión se han dejado aparte las técnicas de resolución paralela, pues aportan mecanismos de paralelismos sobre los algoritmos anteriores, no siendo algoritmos básicamente diferentes, sino solo distintas formas de implementación que faciliten su ejecución paralela.

Cada vez se utilizan más técnicas híbridas que utilizan un sistema en dos a tres fases. Inicialmente se construye una solución que cumpla con las restricciones fuertes y, posteriormente se optimiza esa solución utilizando alguno de los algoritmos descritos.

En general los resultados de los distintos algoritmos descritos y los distintos trabajos de investigación revisados indican que se obtienen buenos resultados, aunque no se pueden comparar los trabajos entre sí ni unos con otros debido a los diferentes problemas en que se aplican. Sí hay que indicar que unos de los conjuntos de datos más utilizados es el conjunto de datos de Toronto.

Sería necesario aplicar los algoritmos a conjuntos de datos reales, pero cada universidad tiene distinta concepción de cómo se crean estos calendarios de exámenes. En este sentido no existe en la actualidad ningún algoritmo o conjunto de ellos que pueda declararse como el más apropiado, teniendo que utilizar técnicas que resulten apropiadas a las necesidades de cada universidad.

BIBLIOGRAFÍA CITADA

Abdullah, Salwani, Edmund K. Burke y Barry Mccollum (2005): «An investigation of variable neighbourhood search for university course timetabling», En *The 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA)*, pp. 413–427.

Abdullah, Salwani, Khalid Shaker, Barry McCollum y Paul McMullan (2009): «Construction of course timetables based on great deluge and tabu search», En *Proceedings of MIC 2009: VIII Metaheuristic International Conference*, pp. 13–16.

Abdullah, Salwani y Hamza Turabieh (2012): «On the use of multi neighbourhood structures within a Tabu-based memetic approach to university timetabling problems», *Information Sciences*, 191, pp. 146–168.

Acan, Adnan y Yüce Tekol (2003): «Chromosome reuse in genetic algorithms», En *Genetic and Evolutionary Computation Conference*, Springer, pp. 695–705.

Ahandani, Morteza Alinia, Mohammad Taghi Vakil Baghmisheh, Mohammad Ali Badamchi Zadeh y Sehraneh Ghaemi (2012): «Hybrid particle swarm optimization transplanted into a hyper-heuristic structure for solving examination timetabling problem», *Swarm and Evolutionary Computation*, 7, pp. 21–34.

Ahmed, Aftab, Ahthasham Sajid, Mazhar Ali y AH Shah Bukhari (2011): «Particle Swarm Optimizatin Based Hyper-Heuristic For Tackling Real World Examinations Scheduling Problem», *Australian Journal of Basic and Applied Sciences*, 5, pp. 1406–1413.

Ahuja, Ravindra K., James B. Orlin y Dushyant Sharma (2000): «Very large-scale neighborhood search», *International Transactions in Operational Research*, 7/4-5, pp. 301–317.

Battistutta, Michele, Andrea Schaerf y Tommaso Urli (2015): «Feature-based tuning of single-stage simulated annealing for examination timetabling», *Annals of Operations Research*, pp. 1-16.

Bullnheimer, Bernd (1997): «An examination scheduling model to maximize students' study time», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 78–91.

Burke, E. K., D. G. Elliman y R. Weare (1994): «A university timetabling system based on graph colouring and constraint manipulation», *Journal of research on computing in education*, 27/1, pp. 1–18.

Burke, E. K. y JD Landa Silva (2005): «The design of memetic algorithms for scheduling and timetabling problems», En *Recent Advances in Memetic Algorithms*, Springer, pp. 289–311.

Burke, Edmund, Yuri Bykov, James Newall y Sanja Petrovic (2004): «A time-predefined local search approach to exam timetabling problems», *Iie Transactions*, 36/6, pp. 509–528.

Burke, Edmund, Adam Eckersley, Barry McCollum, Petrovic Sanja y Rong Qu (2003): «Using simulated annealing to study behaviour of various exam timetabling data sets».

Burke, Edmund K. y Yuri Bykov (2008): «A late acceptance strategy in hill-climbing for exam timetabling problems», En *PATAT 2008 Conference, Montreal, Canada*.

Burke, Edmund K., Adam J. Eckersley, Barry McCollum, Sanja Petrovic y Rong Qu (2010): «Hybrid variable neighbourhood approaches to university exam timetabling», *European Journal of Operational Research*, 206/1, pp. 46–53.

Burke, Edmund K., Graham Kendall, Mustafa Mısır, Ender Özcan, E. K. Burke, G. Kendall, E. Özcan y M. Mısır (2004): «Applications to timetabling», En *Handbook of Graph Theory, chapter 5.6*, Citeseer.

Burke, Edmund K., Graham Kendall y Eric Soubeiga (2003): «A tabu-search hyperheuristic for timetabling and rostering», *Journal of Heuristics*, 9/6, pp. 451–470.

Burke, Edmund K., Barry McCollum, Amnon Meisels, Sanja Petrovic y Rong Qu (2007): «A graph-based hyper-heuristic for educational timetabling problems», *European Journal of Operational Research*, 176/1, pp. 177–192.

Burke, Edmund K. y James P. Newall (1999): «A multistage evolutionary algorithm for the timetable problem», *IEEE transactions on evolutionary computation*, 3/1, pp. 63–74.

Burke, Edmund K. y James P. Newall (2002): «Enhancing timetable solutions with local search methods», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 195–206.

Burke, Edmund K. y James P. Newall (2004): «Solving examination timetabling problems through adaption of heuristic orderings», *Annals of operations Research*, 129/1-4, pp. 107–134.

Burke, Edmund K., James P. Newall y Rupert F. Weare (1996): «A memetic algorithm for university exam timetabling», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 241–250.

Burke, Edmund K., Sanja Petrovic y Rong Qu (2006): «Case-based heuristic selection for timetabling problems», *Journal of Scheduling*, 9/2, pp. 115–132.

Burke, Edmund K., J.Dario Landa Silva y Eric Soubeiga (2005): «Multi-objective hyper-heuristic approaches for space allocation and timetabling», En *Metaheuristics: Progress as Real Problem Solvers*, Springer, pp. 129–158.

Burke, Edmund, Graham Kendall, Jim Newall, Emma Hart, Peter Ross y Sonia Schulenburg (2003): «Hyper-heuristics: An emerging direction in modern search technology», En *Handbook of metaheuristics*, Springer, pp. 457–474.

Carter, Michael W. (1986): «A Survey of Practical Applications of Examination Timetabling Algorithms», *Operations Research*, 34/2, pp. 193-202.

Carter, Michael W. y Gilbert Laporte (1995): «Recent developments in practical examination timetabling», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 1–21.

Carter, Michael W., Gilbert Laporte y Sau Yan Lee (1996): «Examination timetabling: Algorithmic strategies and applications», *Journal of the Operational Research Society*, 47/3, pp. 373–383.

Chu, S. C. y H. L. Fang (1999): «Genetic algorithms vs. tabu search in timetable scheduling», En *Knowledge-Based Intelligent Information Engineering Systems, 1999. Third International Conference*, IEEE, pp. 492–495.

Chu, Shu-Chuan, Yi-Tin Chen y Jiun-Huei Ho (2006): «Timetable scheduling using particle swarm optimization», En *First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06)*, IEEE, pp. 324–327.

Corne, David, Hsiao-Lan Fang, Chris Mellish y Dave Corne (1993): *Solving the modular exam scheduling problem with genetic algorithms*, Department of Artificial Intelligence, University of Edinburgh.

Costa, Daniel y Alain Hertz (1997): «Ants can colour graphs», *Journal of the operational research society*, 48/3, pp. 295–305.

Čupić, Marko, Marin Golub y Domagoj Jakobović (2009): «Exam Timetabling Using Genetic Algorithm», En *31st International Conference on Information Technology Interfaces, ITI2009*.

Dawkin, Richard (1976): «The selfish gene», *Oxford University Press*, 1, p. 976.

Di Gaspero, Luca (2002): «Recolour, shake and kick: A recipe for the examination timetabling problem», En *Proceedings of the fourth international conference on the practice and theory of automated timetabling, Gent, Belgium*, pp. 404–407.

Di Gaspero, Luca y Andrea Schaerf (2000): «Tabu search techniques for examination timetabling», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 104–117.

Dorigo, Marco (1992): «Optimization, learning and natural algorithms», *Ph. D. Thesis, Politecnico di Milano, Italy*.

Dowsland, K. A. y J. M. Thompson (2005): «Ant colony optimization for the examination scheduling problem», *Journal of the Operational Research Society*, 56/4, pp. 426–438.

Dueck, Gunter (1993): «New optimization heuristics: The great deluge algorithm and the record-to-record travel», *Journal of Computational physics*, 104/1, pp. 86–92.

Duong, Tuan Anh y Kim-Hoa Lam (2004): «Combining Constraint Programming and Simulated Annealing on University Exam Timetabling.», En *RIVF*, pp. 205–210.

Eberhart, Russ C., James Kennedy y others (1995): «A new optimizer using particle swarm theory», En *Proceedings of the sixth international symposium on micro machine and human science*, New York, NY, pp. 39–43.

Eley, Michael (2006a): «Ant algorithms for the exam timetabling problem», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 364–382.

Eley, Michael (2006b): «Some experiments with ant colony algorithms for the exam timetabling problem», En *International Workshop on Ant Colony Optimization and Swarm Intelligence*, Springer, pp. 492–499.

Erben, Wilhelm (2000): «A grouping genetic algorithm for graph colouring and exam timetabling», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 132–156.

Fealko, Daniel R. (2005): «Evaluating particle swarm intelligence techniques for solving university examination timetabling problems», Nova Southeastern University.

Frausto-Solís, Juan y Federico Alonso-Pecina (2007): «A hybrid simulated annealing-tabu search algorithm for post enrolment course timetabling», En PATAT, en línea: <http://www.patatconference.org/patat2008/proceedings/Alonso-Pecina-WA1a.pdf> [accedido: 21/08/2015].

Glover, Fred (1986): «Future paths for integer programming and links to artificial intelligence», *Computers & operations research*, 13/5, pp. 533–549.

Glover, Fred y Manuel Laguna (1997): «Tabu search, 1997», *Kluwer Academic Publishers*.

Golberg, David E. (1989): «Genetic algorithms in search, optimization, and machine learning», *Addion wesley*, 1989, p. 102.

Hertz, Alain, Eric Taillard y Dominique De Werra (1995): «A tutorial on tabu search», En *Proc. of Giornate di Lavoro AIRO*, pp. 13–24.

Holland, John H. (1975): *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press, en línea: <http://doi.apa.org/psycinfo/1975-26618-000> [accedido: 22/04/2016].

Hung, Nguyen Quoc Viet, Ta Quang Binh y Duong Tuan Anh (2005): «A Memetic Algorithm for Timetabling», En *Proceedings of 3nd Int, Conf. RIVF*, pp. 289–294.

Kendall, Graham y Naimah Mohd Hussin (2004): «A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 270–293.

Kendall, Graham y Naimah Mohd Hussin (2005): «An investigation of a tabu-search-based hyper-heuristic for examination timetabling», En *Multidisciplinary Scheduling: Theory and Applications*, Springer, pp. 309–328.

Kirkpatrick, Scott, C.Daniel Gelatt, Mario P. Vecchi y others (1983): «Optimization by simmulated annealing», *science*, 220/4598, pp. 671–680.

Krasnogor, Natalio y James Smith (2005): «A tutorial for competent memetic algorithms: model, taxonomy, and design issues», *IEEE Transactions on Evolutionary Computation*, 9/5, pp. 474–488.

Mandal, Ashis Kumar y M. N. M. Kahar (2015): «Combination of graph heuristic with hill climbing search for solving capacitated examination timetabling problem», En IEEE, pp. 118-123.

McCollum, Barry, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu y Edmund K. Burke (2010): «Setting the research agenda in automated timetabling: The second international timetabling competition», *INFORMS Journal on Computing*, 22/1, pp. 120–130.

Mcmullan, Paul (2007): «An extended implementation of the great deluge algorithm for course timetabling», En *International Conference on Computational Science*, Springer, pp. 538–545.

Merlot, Liam TG, Natashia Boland, Barry D. Hughes y Peter J. Stuckey (2003): «A hybrid algorithm for the examination timetabling problem», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 207–231.

Mladenović, Nenad y Pierre Hansen (1997): «Variable neighborhood search», *Computers & Operations Research*, 24/11, pp. 1097–1100.

Montero, Elizabeth, Marıa-Cristina Riff y Leopoldo Altamirano (2011): «A PSO algorithm to solve a real course+ exam timetabling problem», En *International conference on swarm intelligence*, pp. 24–1.

Moscato, P. (1989): *On evolution, search, optimization, GAs and martial arts: toward memetic algorithms. California Inst. Technol., Pasadena*, CA, Tech. Rep. Caltech Concurrent Comput. Prog. Rep. 826.

Müller, Tomáš (2009): «ITC2007 solver description: A hybrid approach», *Annals of Operations Research*, 172/1, pp. 429–446.

Mushi, A. R. (2006): «Tabu search heuristic for university course timetabling problem», *African Journal of Science and Technology*, 7/1, en línea: <http://www.ajol.info/index.php/ajst/article/view/55191> [accedido: 21/08/2016].

Paquete, Luis y Thomas Stützle (2002): «An experimental investigation of iterated local search for coloring graphs», En *Workshops on Applications of Evolutionary Computation*, Springer, pp. 122–131.

Petrovic, Sanja y E. K. Burke (2004): «Handbook of Scheduling: Algorithms, Models, and Performance Analysis», *Bölüm: «University Timetabling», Automated Scheduling, Optimisation and Planning (ASAP) Research Group School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK*, en línea: <http://www.crcnetbase.com/doi/abs/10.1201/9780203489802.ch45> [accedido: 17/04/2016].

Pham, Duc y Dervis Karaboga (2012): *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*, Springer Science & Business Media, en línea: <https://books.google.es/books?hl=es&lr=&id=FlndBwAAQBAJ&oi=fnd&pg=PA1&dq=Intelligent+Optimization+Techniques:+Genetic+Algorithms,+Tabu+Search,+Simulated+Annealing+and+Neural+Networks.&ots=KUIIGoQYI5&sig=yxgFX2R2cIjLy-x6P3gw8e0HFoc> [accedido: 22/05/2016].

Pillay, Nelishia y Wolfgang Banzhaf (2009): «A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem», *European Journal of Operational Research*, 197/2, pp. 482–491.

Qu, R., E. K. Burke, B. McCollum, L. T. G. Merlot y S. Y. Lee (2009): «A survey of search methodologies and automated system development for examination timetabling», *Journal of Scheduling*, 12/1, pp. 55-89.

Qu, Rong y Edmund K. Burke (2009): «Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems», *Journal of the Operational Research Society*, 60/9, pp. 1273–1285.

Qu, Rong, Edmund K. Burke y Barry McCollum (2009): «Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems», *European Journal of Operational Research*, 198/2, pp. 392–404.

Ross, Peter y Dave Corne (1995): «Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems», En *AISB Workshop on Evolutionary Computing*, Springer, pp. 94–102.

Ross, Peter, Emma Hart y Dave Corne (1997): «Some observations about GA-based exam timetabling», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 115–129.

Sanja Petrovic y Edmund Burke (2004): «University Timetabling», En *Handbook of Scheduling*, Chapman & Hall/CRC Computer & Information Science Series, Chapman and Hall/CRC.

Schaerf, Andrea (1999): «A survey of automated timetabling», *Artificial intelligence review*, 13/2, pp. 87–127.

Thompson, Jonathan y Kathryn A. Dowsland (1995): «General cooling schedules for a simulated annealing based timetabling system», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 345–363.

Thompson, Jonathan M. y Kathryn A. Dowsland (1998): «A robust simulated annealing based examination timetabling system», *Computers & Operations Research*, 25/7, pp. 637–648.

Thompson, Jonathan M. y Kathryn A. Dowsland (1996): «Variants of simulated annealing for the examination timetabling problem», *Annals of Operations Research*, 63/1, pp. 105-128.

Turabieh, Hamza y Salwani Abdullah (2011): «An integrated hybrid approach to the examination timetabling problem», *Omega*, 39/6, pp. 598–607, en línea: <http://www.sciencedirect.com/science/article/pii/S030504831100003X> [accedido: 17/05/2016].

de Werra, Dominique (1985): «An introduction to timetabling», *European journal of operational research*, 19/2, pp. 151–162.

White, George M. y Bill S. Xie (2000): «Examination timetables and tabu search with longer-term memory», En *International Conference on the Practice and Theory of Automated Timetabling*, Springer, pp. 85–103.

White, George M., Bill S. Xie y Stevan Zonjic (2004): «Using tabu search with longer-term memory and relaxation to create examination timetables», *European Journal of Operational Research*, 153/1, pp. 80–91.

Wilke, P. y J. Ostler (2010): *Solving the School Timetabling Problem Using Tabu Search, Simulated Annealing, Genetic and Branch & Bound Algorithms. In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal*.

Wright, Mike (2001): «Subcost-guided search—experiments with timetabling problems», *Journal of Heuristics*, 7/3, pp. 251–260.

Zhang, Defu, Yongkai Liu, Rym M'Hallah y Stephen C. H. Leung (2010): «A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems», *European Journal of Operational Research*, 203/3, pp. 550-558.