Resolución de un problema de horarios con algoritmos genéticos

Giusseppe Bervis

2 de septiembre de 2021

Capítulo 1

Introducción

La planificación de horarios tiene que ver con la asignación de recursos escasos a actividades, con el objetivo de optimizar uno o más factores de interés. Cabe destacar que los recursos pueden tomar una variedad de formas, dependiendo del problema; al igual que los factores a optimizar. Así, por ejemplo, los recursos pueden ser automóviles, máquinas en una línea de producción, técnicos en un centro de reparación, entre otros. Algunos factores a optimizar pueden ser, la minimización de trabajos que se terminan posterior al tiempo delimitado o la minimización de conflictos, es decir de la superposición de tareas.

El estudio de los problemas de planificación de horarios data desde los años 50 del siglo pasado. Muchos investigadores empezaron a tratar con problemas de asignación de tareas en centros de trabajos. Aunque los problemas que se trataron entonces eran problemas relativamente simples, se desarrollaron algoritmos eficientes para resolver este tipo de problemas, capaces de generar soluciones óptimas. Pero a medida que se fue profundizando en la investigación de esta área se dieron cuenta de que habían problemas para los cuales eran incapaces de desarrollar algoritmos eficientes para resolverlos. Ya en los 70 se demostró que varios de los problemas de horarios eran NP-duros.

Resolver este tipo de problemas puede ser de gran importancia, ya que el uso eficiente de recursos limitados puede llevar al mejor aprovechamiento de los mismos; es por eso que una estrategia para resolverlos fue el desarrollo de estrategias *no deterministas* que tratan de superar las carencias de las técnicas clásicas de optimización. Una de estas técnicas son los *algoritmos genéticos*.

Así pues, en este trabajo se planteará un problema de horarios con características peculiares y desarrollará un paquete, en el lenguaje Python, que ayudará a resolver este tipo de problemas de horarios, a saber un problema de planificación de exámenes sujeto a condiciones específicas.

Capítulo 2

Características del problema

Los problemas de horarios poseen características diversas, ya que los factores que se contemplan cuando se modelan estos problemas son variados y dependen del contexto del problema específico que se quiera resolver. Así, en este capítulo se pretende definir el problema a tratar, así como describir el modelo matemático asociado al mismo.

2.1. Planteamiento «*naive*»

Este parte de una cantidad finita de profesores, exámenes (a su vez, una cantidad de estudiantes por cada examen) y salones (con capacidad finita, «aforo»). Se debe crear un horario que contemple las siguientes condiciones:

- No es posible que un profesor o un alumno tengan examen de dos cosas a la vez,
- ni que un aula esté ocupada simultáneamente, ni que se supere su capacidad.
- Se consideran soluciones mejores las que ocupan menos fechas, menos aulas, con más distancia entre exámenes para alumnos y con más holgura de aforos.

2.2. Clasificación de las restricciones

En nuestro problema se presenta la clasificación de las restricciones del mismo.

2.2.1. «*Hard constraints*»

- Cada profesor no debe cuidar más de 1 examen a la vez.
- Cada estudiante no debe tener más de 1 examen a la vez.
- A cada salón, en determinado período, no se le podrá asignar más 2 exámenes.
- El número de estudiantes en un examen debe ser menor que la capacidad del salón al que es asignado.
- Cada examen debe ser asignado a un período y salón. (Extra).

2.2.2. «*Soft constraint*»

Estas se presentan con el tipo de optimización que se desea sobre las mismas.

- Minimizar la cantidad de días ocupados.
- Minimizar la cantidad de salones ocupados.
- Maximizar holguras de los exámenes y los salones a los que han sido asignados.
- Maximizar la distancia entre exámenes de cada estudiante.

2.3. Notación

El problema de horarios que se va a tratar posee los siguientes conjuntos y parámetros, presentados con la siguiente notación.

- **Exámenes:** $E := \{e_1, \ldots, e_n\}$; $n = card(E)$
- **Estudiantes:** $S := \{s_1, \ldots, s_m\}$; $m = card(S)$.
- **Profesores:** $T := \{t_1, \ldots, t_w\}$; $w = card(T)$.
- **Salones:** $R := \{r_1, \ldots, r_x\}$; $x = card(R)$.
- **Días:** $D := \{u_1, \ldots, u_d\}$ $days := card(D)$
- **Períodos:** $P := \{p_1, \ldots, p_z\}$; $z = 4 * days$.

Usando la notación anterior se definirán algunos datos de relevancia para la resolución del problema, como los datos de entrada y matrices generadas para el modelado del problema.

2.4. Datos de entrada

Estos son los datos que deberán proporcionarse para poder iniciar a resolver el problema.

- **Matriz «profesores-exámenes»**:

$$\texttt{TE} := (t_k e_i)_{w \times n}\text{, donde } t_k e_i = \begin{cases} 1, & \text{si el profesor } k \text{ cuida el examen } i \\ 0, & \text{en caso contrario} \end{cases}$$

- **Matriz «estudiantes-exámenes»**:

$$\texttt{SE} := (s_j e_i)_{m \times n}\text{, donde } s_j e_i = \begin{cases} 1, & \text{si el estudiante } j \text{ toma el examen } i \\ 0, & \text{en caso contrario} \end{cases}$$

- **Matriz de «capacidades» de los salones**:

$$\texttt{C} := \Big(cap(r_l)\Big)_{x \times 1}\text{, donde } cap(r_l) = \text{ no. de asientos en el salon } l$$

- **Número de días:** Se contempla que se determine el número de días en los que se desea hacer la asignación.

$$days := \text{ no. de días en que se realizarán todos los exámenes}$$

Asunciones. En este punto asumimos en los datos de entrada que:

1. todo profesor tiene asignado por lo menos un examen;
2. todo estudiante tomará por lo menos un examen;
3. todo salón tiene capacidad mayor que 0;
4. la cantidad de exámenes debe ser mayor o igual a la cantidad de profesores;
5. la cantidad de estudiantes por examen debe ser mayor que 0;
6. debe haber salones donde quepan los grupos (no es posible que un grupo no pueda ser asignado a ningún salón, por la cantidad de estudiantes que toman el examen).

2.5. Variables de decisión

La variable de decisión, que corresponde con el cromosoma el algoritmo genético, queda planteado en la siguiente matriz; donde se considera los exámenes en contraposición con los salones en determinados períodos. Se usará la fuente *courier* para los nombres de las matrices, estos serán los nombres en el código.

$$\texttt{EPR} := (e_i\ p_q r_l)_{n\times(z*x)} = \begin{bmatrix} e_1\ p_1 r_1 & \cdots & e_1\ p_1 r_x & \cdots\cdots & e_1\ p_z r_1 & \cdots & e_1\ p_z r_x \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ e_n\ p_1 r_1 & \cdots & e_n\ p_1 r_x & \cdots\cdots & e_n\ p_z r_1 & \cdots & e_n\ p_z r_x \end{bmatrix}$$

$$\text{donde } e_i\ p_q r_l = \begin{cases} 1, & \text{si el examen } i \text{ se asigna al periodo } q \text{ en el salón } l \\ 0, & \text{en caso contrario} \end{cases}$$

2.6. Matrices generadas

A partir de los datos de entradas y la matriz de variables de decisión se generan las siguientes matrices, estas serán de relevancia para plantear las restricciones y las funciones objetivo. Estas matrices se producen usando álgebra matricial, así como redimensionamiento de matrices; estas solo se presentarán, sin citar las operaciones realizada para su obtención.

- **Cantidad de exámenes de cada profesor por período y salón:**

$$\texttt{TPR} := (t_k\ p_q r_l)_{w\times(z*x)}\,,$$

  donde $t_k\ p_q r_l :=$ cantidad de exámenes que cuida el profesor k en el periodo q y salón l

- **Cantidad de exámenes de cada estudiante por período y salón:**

$$\texttt{SPR} = (s_j\ p_q r_l)_{m\times(z*x)}\,,$$

  donde $s_j\ p_q r_l =$ cantidad de exámenes del estudiante j en el periodo q y salón l

- **Cantidad de estudiantes por periodo y salón:** Generada a partir de `SPR`.

$$\texttt{NSPR} := \Big(NS(p_q r_l)\Big)_{z\times x},$$

  donde $NS(p_q r_l) =$ no. de estudiantes que hacen examen en el periodo q y salón l

- **Cantidad de exámenes de cada estudiante por período:** Generada a partir de `SPR`.

$$\texttt{NE\_SP} := \Big(NE(s_j p_q)\Big)_{m\times z},$$

  donde $NE(s_j p_q) =$ no. de exámenes del estudiante j en el período q

- **Cantidad de exámenes que debe cuidar cada profesor por período:** Generada a partir de `TPR`.

$$\texttt{NE\_TP} := \Big(NE(t_k p_q)\Big)_{w\times z},$$

  donde $NE(t_k p_q) =$ no. de exámenes que debe cuidar el profesor j en el período q

2.6.1. «*Hard constraints*»

- **Cada examen debe ser asignado a un período y salón:**

$$\forall i \in \{1, \ldots, n\}, \sum_{q=1}^{z} \sum_{l=1}^{x} e_i \; p_q r_l = 1$$

- **A cada salón, en determinado período, no se le podrá asignar más 2 exámenes:**

$$\forall l \in \{1, \ldots, x\}, q \in \{1, \ldots, z\}, \sum_{i=1}^{n} e_i \; p_q r_l \leq 1$$

- **Cada estudiante no debe tener más de 1 examen a la vez:**

$$\forall j \in \{1, \ldots, m\}, q \in \{1, \ldots, z\}, NE(s_j p_q) \leq 1$$

- **Cada profesor no debe cuidar más de 1 examen a la vez:**

$$\forall k \in \{1, \ldots, w\}, q \in \{1, \ldots, z\}, NE(t_k p_q) \leq 1$$

- **El número de estudiantes en un examen debe ser menor que la capacidad del salón al que es asignado:**

$$\forall q \in \{1, \ldots, z\}, l \in \{1, \ldots, x\}, NS(p_q r_l) \leq cap(r_l)$$

2.6.2. «*Soft constraint*»

- **Cantidad de días ocupados.**

$$Min \sum_{g=1}^{days} o(d_g)$$

Donde $o(d_g) = \begin{cases} 1, \text{ si en el día } g \text{ tiene asignado por lo menos a un examen (en algún período)} \\ 0, \text{ en caso contrario} \end{cases}$

- **Cantidad de salones ocupados.**

$$Min \sum_{l=1}^{x} o(r_l)$$

Donde $o(r_l) = \begin{cases} 1, \text{ si el salón } r \text{ se asigna por lo menos a un examen (en algún período)} \\ 0, \text{ en caso contrario} \end{cases}$

- **Holguras de los exámenes y el salón al que ha sido asignado.**

$$Max \, \text{mín} \Big(h(p_q r_l) \Big); i \in \{1, \ldots, n\}, l \in \{1, \ldots, x\}$$

Donde $h(p_q r_l) = \begin{cases} cap(r_l) - NS(p_q r_l), \text{ si } NS(p_q r_l) > 0 \\ \infty, \text{ en caso contrario} \end{cases}$

- **Distancia entre exámenes de cada estudiante**

$$Max \, \text{mín} \Big(d\Big(NE(s_j p_q)_f, NE(s_j p_v)_{f+1} \Big) \Big)$$

Donde $d(NE(s_j p_q)_f, NE(s_j p_v)_{f+1}) = v - q, \; NE(s_j p_q), NE(s_j p_v) > 0$

2.7. Modelo matemático

Para la resolución del problema, ya que se trata de un problema «*ManyObjetivo*», se aplicará un método de solución agregativo, el de las sumas ponderadas (donde $w_i, i \in \{1, \ldots, 4\}$ son parámetros a elegir por el usuario). Así, el modelo es:

$$\begin{aligned}
\textbf{minimizar} \quad & f(x) = w_1 * \sum_{g=1}^{days} o(d_g) + w_2 * \sum_{l=1}^{x} o(r_l) - w_3 * \text{mín}\Big(h(p_q r_l)\Big) \\
& -w_4 * \text{mín}\left(d\Big(NE(s_j p_q)_f, NE(s_j p_v)_{f+1}\Big)\right) \\
\textbf{sujeto a} \quad & \\
& \sum_{q=1}^{z}\sum_{l=1}^{x} e_i \; p_q r_l = 1 \\
& \sum_{i=1}^{n} e_i \; p_q r_l \leq 1 \\
& NE(s_j p_q) \leq 1 \\
& NE(t_k p_q) \leq 1 \\
& NS(p_q r_l) \leq cap(r_l) \\
& i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}, k \in \{1, \ldots, w\}, \\
& l \in \{1, \ldots, x\}, q \in \{1, \ldots, z\}
\end{aligned}$$

2.8. Implementación computacional

Para la resolución de este problema se desarrolló la clase **"Schedule"**, en Python. En el archivo de Jupyter adjunto se muestra las características de la clase y se resuelve un problema específico (también se adjunta un documento en PDF del archivo Jupyter).

Bibliografía

[1] E. Chong y S. Zak. *An introduction to optimization.* Wiley & Sons, inc., 4ta. ed. edition, 2013.

[2] K. Dahal, K. Chen Tan y P. (Eds) Cowling. *Evolutionary scheduling.* Springer-Verlag Berlin Heidelberg, 2007.

[3] D. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley publishing company, Inc., 1989.

[4] H. Rudova y K. Murry. University course timetabling with soft constraints. *Proceedings of the Practice and Theory of Automated Timetabling (PATAT)*, 2740, 2004. doi: https://doi.org/10.1007/978-3-540-45157-0-21.

[5] R. Sarker y C. Newton. *Optimization Modelling : A Practical Approach.* CRC Press, 2008.

[6] J. D. L. Silva, E. K. Burke y S. Petrovic. An introduction to multiobjective metaheuristics for scheduling and timetabling. *Metaheuristic for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems*, 535:91,129, 2004. doi: https://doi.org/10.1007/978-3-642-17144_4.

[7] K.S. Tang, T.M. Chan, R.J Yin y K.F. Man. *Multiobjective optimization methodology. A jumping gene approach.* CRC Press, 2012.