La Programación de Horarios

Silvina Smith, Cristina Turner

Introducción

Suponga que está estudiando una carrera cuyas materias se dictan en distintos horarios que puede elegir libremente y que, además, le interesa concurrir a esas clases de inglés que hace tanto tiempo debería haber tomado, todo esto sin dejar de asistir a yoga dos veces por semana, que tanto bien le hace. Seguramente se las ingeniará para acomodar todas sus actividades a lo largo de cada día de la semana, no sin antes haber pasado un buen rato, lápiz y papel en mano, evaluando las distintas posibilidades.

Este es un problema típico –en pequeña escala– de planificación y programación de horarios. En la vida actual, este tipo de problemas se presenta con frecuencia y la cantidad de variables involucradas (muchas personas, máquinas, tareas) hace que su resolución sea una cuestión compleja. En estos casos no se puede pretender considerar *todas* las posibilidades y luego elegir *la mejor*, pues la cantidad de tiempo necesaria para ello puede resultar excesivamente grande. Esto ha motivado el desarrollo de algoritmos (ver apéndice) que permiten obtener soluciones *razonables* en tiempos también razonables.

El problema general se puede plantear de la siguiente manera: se debe realizar un cierto trabajo que consta de una serie tareas, para cuya ejecución se dispone de varias máquinas (usamos esta palabra en sentido amplio, pudiendo también referirse a robots, personas, etcétera).

A partir de aquí surgen distintos subproblemas. Uno de ellos consiste en distribuir las tareas en las máquinas de modo tal que el tiempo requerido para terminar el trabajo sea el menor posible. Otro es distribuir las tareas en las máquinas de manera que el tiempo ocioso de las mismas sea mínimo. Finalmente, también podemos estar interesados en deteminar la cantidad mínima de máquinas que permiten terminar el trabajo en un tiempo dado.

Minimización del tiempo total

Consideremos el primer problema. Para poder resolverlo matemáticamente, es necesario que el mismo esté expresado en términos matemáticos. Esta traducción del lenguaje coloquial al lenguaje formal de la matemática recibe el nombre de *modelización* y constituye un paso fundamental en la resolución del problema [1], [2], [3]. La modelización consiste en la selección de una estructura lógica matemática que refleje el comportamiento de las variables involucradas en el problema. En general, esto se logra a través de ecuaciones o inecuaciones, a las que se agregan condiciones (normalmente restricciones) que aseguran que el problema esté bien planteado. En el caso que estamos considerando las condiciones que definen el problema se pueden expresar como restricciones, no siendo necesaria ecuación alguna.

Una simplificación importante del problema considerado es asumir que cada una de las máquinas puede efectuar cualquiera de las tareas. Aún así, se presentan otras cuestiones, como ser, ¿hay tareas más importantes que otras?, ¿qué se debe hacer si en un momento dado dos tareas tienen exactamente la misma importancia? Por ejemplo, si al mismo tiempo concurren a la guardia de un hospital una persona con hemorragia nasal y otra con politraumatismo de cráneo, es claro a quien se deberá atender primero. Pero habrá que decidir qué hacer en caso de recibir dos personas con politraumatismo de cráneo. Otra cuestión importante es si hay tareas que requieren de la ejecución de otra previa para poder llevarse a cabo; por ejemplo, no puede montarse el motor de un auto si previamente no se armó el chasis.

Aún con la simplificación hecha hasta el momento, el problema sigue siendo muy general. Por ello, antes de seguir analizándolo, fijaremos algunas pautas más. Supondremos que:

1. Cada vez que una máquina comience una tarea, la terminará sin interrupciones.

2. No es posible que en un momento dado haya máquinas disponibles y tareas disponibles, es decir, siempre que exista una máquina desocupada y haya una tarea que puede ser ejecutada, la misma será asignada en forma inmediata a esa máquina.

3. La jerarquía de las tareas se conoce de antemano. La misma se indica frecuentemente mediante un grafo dirigido (ver apéndice). Por ejemplo, si el trabajo a realizar es la construcción de una vivienda, el orden en que deben realizarse las tareas podría estar dado por el grafo dirigido de la figura 1.

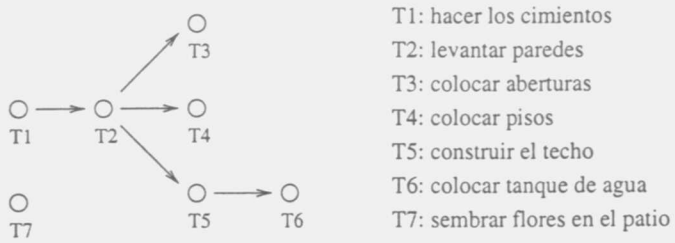

Figura 1: Grafo dirigido con jerarquía de tareas.

4. Las tareas están ordenadas en una lista de prioridades, que es independiente de la jerarquía. Por ejemplo, si al dueño de la casa le interesa por sobre todas las cosas tener listo el techo, la tarea T5 figurará al comienzo de la lista de prioridades. Esto no significa que se ejecutará primero, pues el grafo de jerarquías no lo permite; simplemente indica que la tarea se hará ni bien estén dadas las condiciones para su ejecución. Claramente, distintas listas de prioridades producen resultados diferentes; parte del problema es determinar una lista de prioridades adecuada.

Un algoritmo para procesar listas de prioridades

Un método de distribución de las tareas en las máquinas, teniendo en cuenta

un grafo de jerarquías y una lista de prioridades fijados de antemano, es el que provee el *algoritmo de procesado de listas* [1], que funciona como se describe a continuación.

Supongamos que se han numerado correlativamente todas las máquinas disponibles. El algoritmo asigna a la máquina libre con el número más bajo (esta es sólo una política de "desempate"; podría fijarse otro criterio) la primera tarea de la lista de prioridades que está en condiciones de ser ejecutada –según el grafo dirigido– y aún no ha sido asignada a máquina alguna. Para una mejor comprensión, apliquemos el algoritmo al ejemplo de la figura 2 [1], donde los números dentro de los círculos indican el tiempo necesario para terminar la tarea respectiva. Supongamos que disponemos de dos máquinas y que la lista de

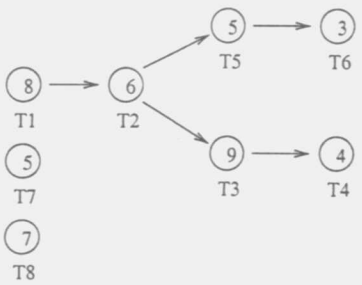

Figura 2: Grafo dirigido con jerarquía de tareas y tiempos de ejecución.

prioridades es $T_8, T_7, T_6, T_5, T_4, T_3, T_2, T_1$. En el tiempo cero, ambas máquinas están libres. La primera tarea de la lista de prioridades, T_8, está en condiciones de ser ejecutada, por lo que se asigna a la máquina 1. La siguiente tarea de la lista también puede ser ejecutada, por lo tanto se asigna a la máquina 2. Al cabo de cinco unidades de tiempo, la tarea T_7 finaliza y la máquina 2 se desocupa. La próxima tarea en la lista de prioridades es T_6, que no puede ser ejecutada en este momento, puesto que el grafo de jerarquías indica que previamente es necesario ejecutar T_1, T_2 y T_5. La única tarea en condiciones de ejecutarse es T_1, por lo que es asignada a la máquina 2. Dos unidades de tiempo

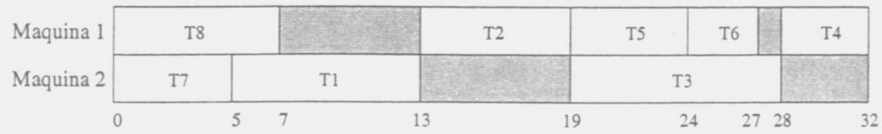

Figura 3: Distribución de tareas en dos máquinas para la lista de prioridades $T_8, T_7, T_6, T_5, T_4, T_3, T_2, T_1$ y el grafo de jerarquías de la figura 2. Los sectores sombreados corresponden al tiempo que la máquina permanece ociosa.

después, la máquina 1 termina la tarea T_8, pero no hay ninguna tarea de la lista en condiciones de comenzarse, por lo que la máquina queda ociosa. Al llegar al tiempo trece, la máquina 2 finaliza la tarea T_1 y ambas máquinas quedan libres. La próxima tarea de la lista que puede ejecutarse es T_2, que es asignada a la máquina 1. La máquina 2 queda ociosa, puesto que el grafo de jerarquías no permite la ejecución de otra tarea de la lista. Al finalizarse la tarea T_2 (tiempo diecinueve), ambas máquinas quedan libres nuevamente. La siguiente tarea de la lista de prioridades es T_5, que puede ser ejecutada y se asigna a la máquina 1. La máquina 2 comienza con la tarea T_3. Y así se continúa hasta terminar la lista de tareas. La figura 3 muestra la distribución de tareas lograda.

¿Es bueno el resultado obtenido? La gran cantidad de tiempo muerto que se observa en la figura 3 nos hace pensar que no. Si se aplica el mismo algoritmo a la lista de prioridades $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8$, se obtiene la distribución de tareas de la figura 4, que evidentemente es mejor que la anterior, ya que el trabajo se termina en menor tiempo. Más aún, esta distribución es óptima, puesto que en el grafo de jerarquías el camino T_1, T_2, T_3, T_4 tiene longitud (ver apéndice) $8 + 6 + 9 + 4 = 27$. Observemos que el tiempo necesario para ejecutar el trabajo completo no puede ser inferior a la longitud del camino más largo en el grafo de jerarquías. Tal camino se denomina crítico (ver apéndice). Otra forma de determinar una cota inferior para el tiempo total requerido es sumar los tiempos de todas las tareas y dividir este número por la cantidad de

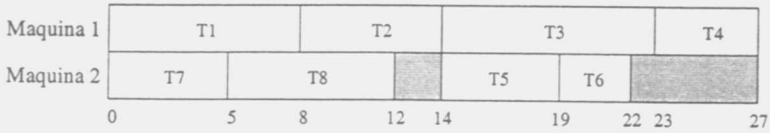

Figura 4: Distribución de tareas en dos máquinas para la lista de prioridades $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8$.

máquinas disponibles. En ambos casos, el resultado obtenido puede no ser una buena estimación del tiempo real necesario para terminar el trabajo.

Programación de horarios con caminos críticos

En los ejemplos anteriores comprobamos que distintas listas de prioridades pueden dar lugar a tiempos de terminación diferentes. Si hay n tareas por realizar, la cantidad de listas de prioridades posibles está dada por la cantidad de permutaciones de dichas tareas, es decir por $n!$. Este número puede ser muy grande, haciendo que no resulte conveniente aplicar el algoritmo anterior a todas las listas de prioridades posibles y luego seleccionar la mejor. ¿Habrá alguna forma de generar una lista de prioridades que produzca un horario óptimo o "casi óptimo"? El hecho que el tiempo de terminación no pueda ser inferior a la longitud de un camino crítico nos induce a pensar que lo mejor es realizar primero las tareas que están al principio de un camino crítico, a fin de no obstaculizar el desarrollo de las tareas posteriores que dependen de éstas. El método es el siguiente [1]:

1. Se identifican los caminos críticos.
2. Si hay sólo un camino crítico, se selecciona la primer tarea de dicho camino. En caso de haber más de un camino crítico, se considera la primera tarea de cada uno de ellos y se elige aquella identificada con el menor número.
3. Se elimina la tarea seleccionada en el grafo de jerarquías y se vuelve al

punto 1.

Consideremos, por ejemplo, el grafo de jerarquías de la figura 5a. Hay dos

Figura 5: (a) Grafo de jerarquías original usado para generar una lista de prioridades con el método de los caminos críticos. (b) Grafo de jerarquías luego de haber borrado las dos primeras tareas seleccionadas.

caminos críticos, de longitud 25, que son T_1, T_2, T_4, T_7 y T_8, T_6, T_4, T_7. Seleccionamos, pues, la tarea T_1 y la borramos del grafo. Después de esto, el único camino crítico es T_8, T_6, T_4, T_7, por lo que la segunda tarea seleccionada es T_8. Al borrar esta tarea se obtiene el grafo de la figura 5b. En él, el único camino crítico es T_5, T_6, T_4, T_7, de longitud 20. Así, la tarea siguiente es T_5. Al culminar el proceso, se obtiene la lista de prioridades $T_1, T_8, T_5, T_6, T_2, T_4, T_9, T_7, T_3$. Aplicando luego el algoritmo de procesado de listas para el caso de dos máquinas se obtiene el horario de la figura 6.

Figura 6: Horario obtenido a partir de la lista de prioridades producida por el método de los caminos críticos.

En algunas ocasiones el método de los caminos críticos da lugar a listas de prioridades que, al aplicar el algoritmo de procesado de listas, producen horarios óptimos. Pero también hay ejemplos en los que el resultado obtenido es malo. En general, no hay ningún método que permita determinar la lista de prioridades que producirá *siempre* un horario óptimo al aplicar el algoritmo de procesado de listas.

Tareas independientes

Un caso muy particular de grafo de jerarquías es el que se presenta cuando no hay flechas, es decir, cuando ninguna tarea requiere de la realización de otra previa para poder ser ejecutada. En este caso se dice que las tareas son independientes entre sí.

Cuando las tareas son independientes, el algoritmo de procesado de listas produce resultados dispares, a veces buenos y otras no tanto. Pero se puede demostrar que cualquiera sea la lista de prioridades LP que se elija, si el horario óptimo requiere un tiempo T_{op}, entonces el tiempo de terminación del horario producido con el algoritmo de procesado de listas aplicado a LP con m máquinas es menor o igual que $(2 - 1/m)T_{op}$. Esta cota se obtiene haciendo un análisis del peor caso (ver apéndice) y tiene más importancia teórica que práctica.

Afortunadamente, si las tareas son independientes existe un método para determinar una lista de prioridades que dará lugar a horarios relativamente buenos. Es el *algoritmo de los tiempos decrecientes* [1] y como su nombre lo indica, la estrategia consiste en ordenar las tareas de mayor a menor, según los tiempos requeridos para terminarlas. Consideremos, por ejemplo, las mismas tareas de la figura 5a, pero independientes. El algoritmo de los tiempos decrecientes produce la lista de prioridades $T_9, T_7, T_1, T_3, T_6, T_8, T_2, T_4, T_5$ y el horario obtenido con el algoritmo de procesado de listas para dos máquinas es el de la figura 7. El horario es óptimo, pues en ningún momento hay máquinas ociosas.

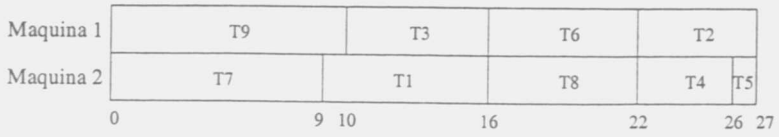

Figura 7: Horario producido por el algoritmo de procesado de listas a partir de la lista de prioridades generada por el algoritmo de los tiempos decrecientes, considerando las tareas de la figura 5a como independientes.

Sin embargo, cabe destacar que el algoritmo de los tiempos decrecientes no siempre produce listas de prioridades que dan lugar a horarios óptimos. Una cota inferior para el tiempo total requerido cuando las tareas son independientes es la suma de los tiempos de cada tarea dividido la cantidad de máquinas. Para el ejemplo que estamos analizando, este número es 54/2=27.

Nótese que si las tareas son independientes, el método de los caminos críticos produce la misma lista de prioridades que el algoritmo de los tiempos decrecientes.

Minimización de la cantidad de máquinas

Como dijimos al principio, otro enfoque del problema de la programación de horarios es distribuir las tareas de modo tal que se requiera la menor cantidad de máquinas posible. En este caso, asumiremos:

1. Cada máquina puede funcionar una cantidad de tiempo fija T, cubierta la cual se debe habilitar otra máquina.
2. Las tareas son independientes.

Como ocurre con el problema analizado anteriormente, no existe ningún algoritmo que, en un tiempo prudencial, calcule *siempre* la cantidad óptima de máquinas. De todos modos, hay algoritmos que permiten resolver el problema

razonablemente en muchos casos.

Los algoritmos SP, PrP y PP

Un método sencillo para resolver el problema es el que provee el *algoritmo de la siguiente posibilidad* (SP) [1]. Consiste en asignar las tareas de la lista (en el orden que aparecen) a la primera máquina, hasta llegar al tiempo T. La primera tarea que no pueda ser ejecutada por esta máquina se asigna a la siguiente; y así sucesivamente.

Consideremos, por ejemplo, la lista de tareas independientes $T_1(9)$, $T_2(7)$, $T_3(6)$, $T_4(5)$, $T_5(5)$, $T_6(2)$, $T_7(4)$, $T_8(1)$, $T_9(2)$, $T_{10}(3)$, $T_{11}(6)$, donde los números entre paréntesis indican el tiempo de ejecución que requiere cada tarea y supongamos que $T = 15$. El algoritmo SP asigna la tarea T_1 a la máquina 1. Como $9 + 7 = 16 > 15$, la tarea T_2 no puede ser ejecutada por la primera máquina, por lo que se asigna a la máquina 2. Y así sucesivamente hasta agotar las tareas de la lista. Finalmente, el algoritmo SP produce el resultado que muestra la figura 8. Este algoritmo cuenta con la ventaja de poder aplicarse sin

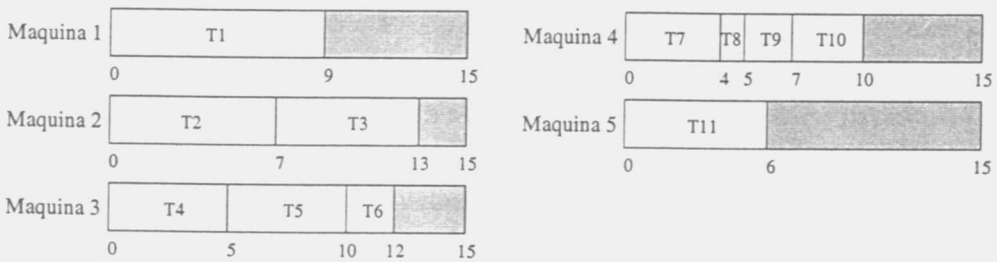

Figura 8: Distribución de tareas del algoritmo SP.

necesidad de conocer a priori toda la lista de tareas. Por otra parte, es claro que puede resultar sumamente ineficiente, sobre todo si las tareas de corta duración figuran consecutivamente en la lista.

El gran tiempo ocioso que eventualmente puede resultar con el algoritmo SP podría evitarse si se permitiera "volver atrás" y completar con tareas el tiempo ocioso de las máquinas ya habilitadas. Esto es lo que hace el *algoritmo de la primera posibilidad* (PrP) [1]: cada tarea de la lista se asigna a la máquina con el número más bajo que aún tiene tiempo suficiente para ejecutarla.

Para el ejemplo anterior, el algoritmo de la primera posibilidad produce el resultado de la figura 9. Se aprecia que este algoritmo aprovecha mejor

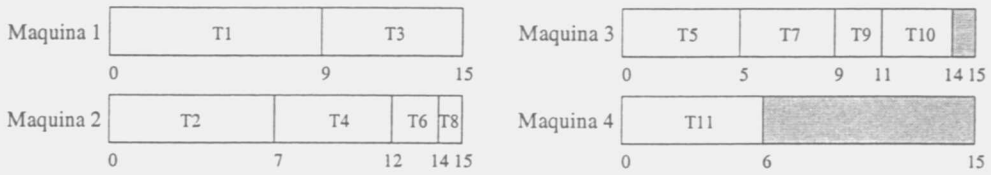

Figura 9: Distribución de tareas del algoritmo PrP.

las máquinas. Como contrapartida, es necesario llevar un registro del espacio disponible en cada una de las máquinas que ya ha sido habilitada.

Otra estrategia para ocupar máquinas con tiempo ocioso es la que propone el *algoritmo de la peor posibilidad* (PP) [1]. Consiste en lo siguiente: cada tarea debe asignarse a la máquina ya habilitada que tenga la mayor cantidad de tiempo disponible. En caso de empate entre dos o más máquinas, se elije aquella con el número más bajo. Si se aplica este algoritmo al mismo ejemplo con el que venimos trabajando, el resultado que se obtiene es el que se muestra en la figura 10. Notemos que los dos últimos algoritmos requieren la misma cantidad de máquinas y que el tiempo ocioso total también es el mismo, aunque las tareas quedan distribuidas de diferente manera.

Hemos dicho que las tareas son independientes. Entonces, podríamos pensar en aplicar el algoritmo de los tiempos decrecientes para generar la lista de prioridades y recién entonces usar alguno de los algoritmos anteriores. Así sur-

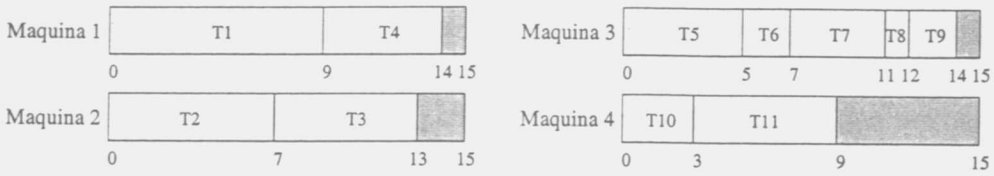

Figura 10: Distribución de tareas del algoritmo PP.

gen los *algoritmos de la siguiente posibilidad decreciente* (SPD), *de la primera posibilidad decreciente* (PrPD) y *de la peor posibilidad decreciente* (PPD) [1].

Nuevamente, no se puede decir que alguno de los algoritmos descriptos sea mejor que los demás. La elección del algoritmo dependerá, en general, de las condiciones relacionadas con el problema específico planteado.

Aplicaciones de algoritmos para planificación de horarios a otros problemas

El problema de la planificación de horarios aparece como tal en muchos ámbitos, por ejemplo, en la distribución de docentes y aulas para los distintos cursos de un establecimiento educativo, en la planificación de guardias médicas y atención de especialistas en un hospital, en la asignación de horarios y plataformas en la terminal de ómnibus de transportes de larga distancia, etcétera. Sin embargo, el espectro de problemas no se agota aquí. Hay problemas que, por su planteo, parecen distintos a éstos, pero que en realidad pueden resolverse aplicando los algoritmos antes descriptos. A modo de ejemplo, mencionaremos el problema del empaquetado en cajas, el cual consiste en guardar objetos de distintos tamaños en cajas con una cierta capacidad, con el objetivo de ocupar la menor cantidad posible de cajas. En términos de la programación de horarios, este problema es equivalente al de distribuir tareas en máquinas, cada una de las cuales puede trabajar a lo sumo T unidades de tiempo, siendo el

objetivo la minimización de la cantidad de máquinas requeridas. Por lo tanto, los algoritmos SP, SPD, PrP, PrPD, PP y PPD pueden usarse para encontrar una solución a este problema. (En la nota editorial de la Revista de Educación Matemática n°3, Vol. 14, 1999, se menciona que ya ha sido resuelto el problema de encontrar el mejor modo de empaquetar objetos).

Otra variante del problema del empaquetado consiste en guardar objetos de distintos tamaños en una caja, tratando de desperdiciar la menor cantidad de espacio posible, es decir, maximizar la cantidad de objetos guardados. Este problema se presenta, por ejemplo, cuando hay que almacenar productos en un depósito. En términos de la programación de horarios, es equivalente a distribuir las tareas en las máquinas (en este caso una), minimizando el tiempo ocioso.

Conclusiones

El problema de la planificación de horarios aparece en muchos ámbitos de la vida cotidiana y la matemática propone diversos métodos para resolverlo. No existe algoritmo que obtenga la solución óptima en todos los casos. Por otra parte, considerar todas las soluciones posibles y luego seleccionar la mejor resulta generalmente impracticable. Es por esto que el estudio del comportamiento de los algoritmos resulta imprescindible. Ahora bien, este es uno de los roles importantes que desempeña la matemática, pero para llegar a este punto es necesario antes haber dado un paso fundamental: la modelización. La correcta modelización matemática es de vital importancia, puesto que es el puente que une a la realidad con la abstracción matemática, es el camino que permite aprovechar todas las herramientas de la ciencia, poniéndolas al servicio de la comunidad.

Apéndice: descripción de algunos términos frecuentes

Para el lector poco familiarizado con el tema, incluimos a continuación una

breve descripción de algunos de los términos usados en el texto, que son de importancia para la buena comprensión del tema desarrollado.

-*Algoritmo.* Es un conjunto de instrucciones a ser ejecutadas en un cierto orden; una descripción concisa y clara de un método. Por ejemplo, un algoritmo para efectuar una llamada desde un teléfono público podría ser: 1) Levantar el tubo; 2) Insertar la moneda; 3) Discar el número deseado; 4) Hablar.

-*Grafo.* Es un esquema que se usa para representar objetos y relaciones entre ellos. Por cada objeto de interés se dibuja un círculo, que recibe el nombre de *vértice.* Si dos objetos están relacionados entre sí, se dibuja una línea que los une, la cual recibe el nombre de *arista* [1], [5]. Por ejemplo, un mapa vial esquemático es un grafo en el que los vértices son las ciudades y las aristas las rutas que las unen. En algunos casos se asignan números a las aristas, los cuales representan el costo, la distancia o el tiempo asociado a ellas. Estos números se llaman *pesos.* En el ejemplo anterior, los pesos serían las distancias entre ciudades unidas por una arista. También es posible asignar los pesos a los vértices, como en los ejemplos discutidos en el texto.

-*Grafo dirigido.* Es un grafo en el cual las aristas son flechas, indicando un sentido de circulación sobre esa arista. Un grafo dirigido no excluye la posibilidad de tener flechas en ambos sentidos entre dos vértices.

-*Camino.* En un grafo, un camino es una secuencia de aristas conectadas entre sí.

-*Longitud de un camino.* Cuando las aristas tienen pesos asociados, la longitud de un camino es la suma de los pesos de las aristas involucradas. Lo mismo ocurre cuando los pesos están en los vértices. En los ejemplos del texto, la longitud de un camino es la suma de los tiempos de las tareas que figuran en ese camino.

-*Camino crítico.* Es el camino más largo en un grafo dirigido. Puede haber más de un camino crítico.

-*Solución óptima.* Cuando un problema tiene varias soluciones, las cuales pueden ser clasificadas según algún criterio, la mejor de ellas recibe el nom-

bre de solución óptima. En los ejemplos discutidos en el texto, los criterios de clasificación son el tiempo total requerido y la cantidad de máquinas necesarias.
-Análisis del peor caso. La utilidad de un algoritmo depende en gran medida del costo (medido en tiempo, esfuerzo, recursos, etcétera) que implique aplicarlo. El análisis del peor caso consiste en evaluar el costo del algoritmo cuando se aplica al peor caso posible. Si bien el resultado de este análisis provee una cota para el costo, frecuentemente suele no ser representativo de lo que ocurre en la mayor parte de los casos. Por ello existe también el *análisis del caso medio*, el cual considera como costo del algoritmo al que se obtiene en general para un problema típico.

Bibliografía

[1] J. Malkevitch, R. Meyer, W. Meyer et al, *Las matemáticas en la vida cotidiana*, director del proyecto: S. Garfunkel, editor coordinador primera edición: L.A. Steen, Addison Wesley Iberoamericana España, 1999.

[2] K. Devlin, *Mathematics: The Science of Patterns*, Scientific American Library, 1994.

[3] P.Davis y R. Hersch, *Experiencia Matemática*, Labor, 1998.

[4] *Ciência Hoje*, Sociedade Brasileira para o Progresso da Ciência, Vol. 8, 1999.

[5] *Revista de Educación Matemática*, Vol. 15, N° 1, 2000.

Silvina Smith, Cristina Turner
Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
e-mail: smith@mate.uncor.edu, turner@mate.uncor.edu