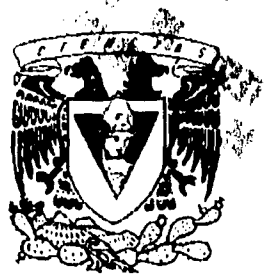
UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS



# Algorítmos tipo subasta para algunos problemas de redes

T E S I S

Que para obtener el título de

A C T U A R I O

presenta

Dulce M[illegible] Rosas Díaz



1996



TESIS CON
FALLA DE ORIGEN

89
25j

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

M. en C. Virginia Abrín Batule
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
P r e s e n t e

**Comunicamos a usted que hemos revisado el trabajo de Tesis:** "ALGORITMOS TIPO SUBASTA PARA ALGUNOS PROBLEMAS DE REDES"

**realizado por** ROSAS DIAZ DULCE MARIA

**con número de cuenta** 8955487-4 **, pasante de la carrera de** ACTUARIA

**Dicho trabajo cuenta con nuestro voto aprobatorio.**

**Atentamente**

| | | |
|---|---|---|
| **Director de Tesis Propietario** | M.en C. BEATRIZ RODRIGUEZ FERNANDEZ | Beatriz Rodríguez Fdz. |
| **Propietario** | MAT. AGUSTIN CANO GARCES | |
| **Propietario** | M.en I.O. MA. DEL CARMEN HERNANDEZ AYUSO | Ma del Carmen Hdez Ayuso |
| **Suplente** | MAT. FELIPE CONTRERAS ALCALA | |
| **Suplente** | M.en C. VIRGINIA ABRIN BATULE | Virginia Abrín Batule |

**Consejo Departamental de Matemáticas**
M.en C. ALEJANDRO BRAVO MOJICA

# Algoritmos tipo subastas para algunos problemas de redes

# Índice General

# Introducción

El algoritmo de subasta fue diseñado en 1979 para resolver el problema clásico de asignación. El motivo era solucionar el problema usando paralelismo en forma natural. Sin embargo, el método resultante fue muy rápido también en forma serial.

En este trabajo se estudiará el algoritmo de subasta para el problema de asignación y después se discutirán varias extensiones a otros problemas. El propósito es que en otros problemas lineales de flujo en redes, como el problema de transporte y el problema de rutas más cortas, puedan aplicar el algoritmo de subasta para el problema de asignación.

Los otros problemas mencionados se pueden resolver por medio del algoritmo original de subasta para el problema de asignación. Ya que por medio de algunas modificaciones se puede llegar a problemas equivalentes de asignación. Una vez que el algoritmo de subasta es aplicado al problema equivalente de asignación se obtiene un algoritmo de subasta para el problema original.

El algoritmo de subasta es un método intuitivo que opera como una subasta verdadera, donde las personas que "compiten"por los objetos, aumentan los precios a éstos. El algoritmo de subasta se ejecuta muy bien en la teorí a y en la práctica gracias a que es muy adecuado para la computación en paralelo.

Después se desarrollará el algoritmo de subasta para el problema de asignación simétrico, es decir, cuando se tiene igual número de personas que de objetos. También se discutirá una alternativa del algoritmo de subasta, llamado subasta *hacia atrás*; esto es, en la subasta regular (hacia adelante) las personas ofrecen cantidades cada vez mayores por los objetos; en cambio en la subasta hacia atrás los objetos compiten por las personas disminuyendo sus precios. Se

describirá un algoritmo de subasta para el problema de rutas más cortas. Se mostrará la extensión al problema de transporte. Finalmente, se discutirán los aspectos computacionales del algoritmo de subasta, incluyendo resultados en paralelo.

# Capítulo 1

# Problema de asignación

El problema de asignación es de gran importancia en muchos contextos prácticos, pero también es de gran importancia teórica a pesar de su sencillez, incorpora una estructura fundamental de programación lineal. El problema de asignación ha servido como un punto de partida conveniente para otros algoritmos importantes de programación lineal. Por ejemplo, el método primal-dual fue motivado y desarrollado por el método Húngaro, el primer método especializado para resolver el problema de asignación.

Algunos ejemplos prácticos del problema de asignación son asignar trabajadores a empleos, tareas a máquinas, etcétera. También hay situaciones donde el problema de asignación aparece como un subproblema para resolver otros problemas de optimización, como son el problema de ruta más corta y el problema de transporte entre otros.

Supóngase que se tienen $n$ personas y $m$ objetos. Si la persona $i$ es asignada al objeto $j$, se tiene un beneficio $a_{ij}$. Se desea encontrar el beneficio máximo de asignar las personas a los objetos. En cada solución factible las variables $x_{ij}$ sólo pueden tomar el valor 0 o 1; si $x_{ij} = 1$ significa que la persona $i$ ha sido asignada al objeto $j$, y $x_{ij} = 0$ indica que la persona $i$ no está asignada al objeto $j$, la restricción que existe en este tipo de problemas es que a cada persona se le asignará un sólo objeto, y a cada objeto se le asignará una sola persona. Matemáticamente, se quiere encontrar un conjunto de parejas, persona-objeto $(1, j_1), ..., (n, j_n)$, tal que los objetos $j_1, ..., j_n$ sean todos distintos, y el beneficio total $\sum_{i=1}^{n} a_{ij_i}$ sea máximo.

Una condición necesaria y suficiente para que el problema de asignación tenga solución, es que sea simétrico; es decir, cuando $m = n$. Si $m > n$ se pueden agregar $m - n$ objetos, con costo cero. Si $m < n$, se agregan $n - m$ personas.

Se puede formular el problema de asignación como un problema lineal:

$$\begin{aligned}
&\text{Maximizar} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} \\
&\quad \text{sujeto a} \\
&\quad \sum_{j=1}^{n} x_{ij} = 1, \qquad \forall \quad i = 1,...,n \qquad (1.1) \\
&\quad \sum_{i=1}^{n} x_{ij} = 1, \qquad \forall \quad j = 1,...,n \\
&\quad 0 \leq x_{ij} \leq 1, \qquad \forall \quad i,j = 1,...,n
\end{aligned}$$

En forma matricial, el problema de asignación resulta:

$$\begin{aligned}
&\text{Maximizar} \quad ax \\
&\text{Sujeta a} \quad Ax = \mathbf{1} \\
&\qquad\qquad x_{ij} = 0 \text{ o } 1 \quad i,j = 1,...,n
\end{aligned}$$

en donde $x = (x_{11},...,x_{1n},...,x_{n1},...,x_{nn})^t$, y $A$ es una matriz $2n \times n^2$ cuya columna $(i,j)$ es $A_{ij} = e_i + e_{n+j}$ donde $e_i$ y $e_{n+j}$ son los vectores unitarios con unos en la $i$-ésima y la $(n+j)$-ésima posición, respectivamente, para $i,j = 1,2,...,n$.

La matriz $A$, con dimensión $2n \times n^2$, tiene la siguiente forma especial:

$$A = \overset{\text{n}^2 \text{ columnas}}{\begin{pmatrix} \mathbf{1} & \mathbf{0} & ... & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & ... & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & ... & \mathbf{1} \\ \mathbf{I} & \mathbf{I} & ... & \mathbf{I} \end{pmatrix}} \quad \text{2n renglones}$$

en donde $\mathbf{1}$ es un $n$-vector renglón de componentes todos iguales a 1; e $\mathbf{I}$ es una matriz identidad de $n \times n$. La matriz $A$ es la que le da al problema de

asignación una estructura especial, en efecto, la matriz $A$ tiene la propiedad de unimodularidad total.

La matriz $A$ es *totalmente unimodular* si el determinante de cualquier submatriz cuadrada de ella tiene valor $-1$, 0 o $+1$. En el caso de la matriz de asignación, puesto que todos sus elementos son 0 o 1, cada submatriz de $1 \times 1$ tiene valor 0 o 1. Además cualquier submatriz de $2n^2$ tiene determinante de valor 0 pues el rango de $A$ es $2n - 1$. Sólo falta demostrar que cualquier submatriz de $k \times k$ con $(1 < k < 2n)$ tiene la propiedad requerida.

Sea $A_k$ cualquier submatriz de dimensión $k \times k$ de $A$. Se debe probar que el $\det A_k = \pm 1$ o 0. Por inducción sobre $k$ supóngase que la propiedad es cierta para $A_{k-1}$ (se sabe que es cierto para $A_1$). Tomando en cuenta que cada columna de $A_k$ tiene, ya sea ningún 1, sólo un 1, o dos 1s, se tiene que, si ninguna columna de $A_k$ tiene $1's$, entonces $\det A_k = 0$. Si cada columna de $A_k$ tiene dos $1's$, entonces uno de los $1's$ está en un renglón origen y el otro 1 está en un renglón destino; en este caso, la suma de los renglones origen de $A_k$ es igual a la suma de los renglones destino de $A_k$; por lo tanto, los renglones de $A_k$ son linealmente dependientes y $\det A_k = 0$. Finalmente, si alguna columna de $A_k$ tiene sólo un 1, entonces expandiendo $\det A_k$ en los menores de esa columna se obtiene:

$$\det A_k = \pm \det A_{k-1}$$

en donde $A_{k-1}$ es una submatriz $(k-1) \times (k-1)$. Pero, por la hipótesis de inducción, $\det A_{k-1} = \pm 1$ o 0. Por lo tanto, la propiedad se cumple para $A_k$, lo cual demuestra el resultado.

Aplicando la propiedad de unimodularidad total de $A$, se tiene que una solución básica factible del problema de asignación, con la restricción $x_{ij} = 0$ o 1 reemplazada por $x_{ij} \geq 0$ será toda entera. Además, por las restricciones ninguna $x_{ij}$ puede ser mayor que 1. Por lo tanto, todas las $x_{ij}$ serán 0 o 1 en una solución óptima del problema lineal. Esto permite reemplazar las restricciones $x_{ij} = 0$ o 1 por la restricción $x_{ij} \geq 0$. Así se obtiene:

$$\begin{array}{lll} \text{Maximizar} & ax & \\ \text{Sujeta a} & Ax & = \mathbf{1} \\ & x_{ij} & \geq 0 \quad i,j = 1,\ldots,n \end{array}$$

Puesto que el problema de asignación es un caso especial del problema de transporte, se puede resolver como si fuera uno de estos, pero sería muy ineficiente

resolver este tipo de problemas, por medio del método simplex o por medio del método de transporte. Se explotará la estructura especial del problema de asignación para obtener un algoritmo más eficiente, que es el algoritmo de subasta.

Otra propiedad importante del problema de asignación es que se puede representar por una gráfica bipartita, como se muestra en la figura 1. Hay $2n$ nodos divididos en dos grupos: uno que corresponde a $n$ personas y el otro a los $n$ objetos. También, para cada $(i,j)$, $i,j = 1,...,n$, hay un arco que conecta a la persona $i$ con el objeto $j$. En términos de problemas de redes, la variable $x_{ij}$ se refiere al flujo del arco $(i,j)$. La restricción $\sum_{j=1}^{n} x_{ij} = 1$, indica que la salida total de flujo del nodo $i$ debe ser igual a 1, lo cual se puede ver como el requerimiento del nodo. Similarmente la restricción $\sum_{i=1}^{n} x_{ij} = 1$, indica que la entrada total de flujo del nodo $j$ debe ser igual a 1, lo cual se puede ver como la demanda del nodo.

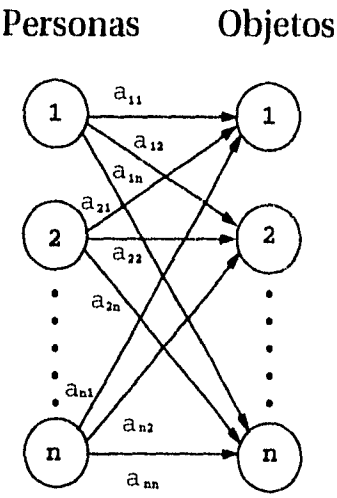



**Fig. 1.** *Problema de asignación*

## 1.1 Dualidad

Si el problema primal de asignación es:

$$\text{Maximizar} \quad Z = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij}$$

sujeto a

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &\leq 1, \qquad \forall \quad j = 1,...,n \\ \sum_{j=1}^{n} x_{ij} &= 1, \qquad \forall \quad i = 1,...n \\ 0 \leq x_{ij} &\leq 1, \qquad \forall \quad i,j = 1,...,n, \end{aligned} \tag{1.2}$$

Sea $X$ un conjunto dado por:

$$X = \left\{ x_{ij} \mid \sum_{j=1}^{n} x_{ij} = 1,\ i = 1,...,n;\ x_{ij} \geq 0 \ y \ x_{ij} \text{ entero para } i,j = 1,...,n \right\}$$

entonces se puede reescribir el problema de asignación de la siguiente manera:

$$\text{Maximizar} \quad Z = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij}$$

sujeto a

$$\begin{aligned} \sum_{i=1}^{n} x_{ij} &\leq 1, \qquad \forall \quad j = 1,...,n \\ x &\in X \end{aligned} \tag{1.3}$$

Utilizando la definición de dualidad no lineal se tiene que el problema dual está dado por:

$$\begin{aligned} &\text{Minimizar} \quad q(p) \\ &\text{Sujeto a} \quad p \geq 0 \end{aligned} \tag{1.4}$$

donde $p$ es el vector de precios de los objetos, y

$$q(p) = \sup_{x \in X} \left\{ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} + p_1(1 - x_{11} - ... - x_{1n}) + ... + p_n(1 - x_{21} - ... - x_{2n}) \right\}.$$

Factorizando a $x_{ij}$ se tiene:

$$q(p) = \sup_{x \in X} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij} + \sum_{j=1}^{n} p_j - \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} p_j \right\}$$

$$q(p) = \sup_{x \in X} \left\{ \sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - p_j) x_{ij} \right\}$$

como $\sum_{j=1}^{n} p_j$ es una constante, puede salir del supremo y la función $q(p)$ quedará:

$$q(p) = \sum_{j=1}^{n} p_j + \sup_{x \in X} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - p_j) x_{ij} \right\}$$

como $X$ es un conjunto finito, el supremo es igual al máximo, entonces

$$q(p) = \sum_{j=1}^{n} p_j + \max_{x \in X} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} - p_j) x_{ij} \right\}$$

como el máximo de la suma es igual a la suma de los máximos, se puede escribir a $q(p)$ como sigue:

$$q(p) = \sum_{j=1}^{n} p_j + \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} \max_{x \in X} \{a_{ij} - p_j\} x_{ij} \right\}$$

como $x_{ij}$ sólo puede ser 1 o 0, si se quitan todas las $x_{ij} = 0$, es decir, se encuentra el máximo de los objetos $j$ que pueden ser asignados con la persona $i$, entonces se tiene que:

$$q(p) = \sum_{i=1}^{n} \max_{j} \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \tag{1.5}$$

entonces la condición de holgura complementaria para una asignación $S$ (no necesariamente completa) y un vector de precios $p$, se puede escribir como sigue:

$$a_{ij_i} - p_{j_i} = \max_{j} \{a_{ij} - p_j\}, \qquad \forall \quad (i,j) \in S \tag{1.6}$$

Esto quiere decir que el objeto $j$ tiene un precio $p_j$, y la persona que sea asignada al objeto $j$ debe pagar el precio $p_j$. El valor neto del objeto $j$ para la persona $i$ es $a_{ij} - p_j$, pero la persona $i$ lógicamente quiere ser asignada a un

objeto $j_i$ con valor máximo.

Se demuestra más adelante, en la proposición 1, que una condición necesaria y suficiente para que $S$ y $p$ sean la solución primal y dual óptima es que $S$ sea completa y que $S$ y $p$ satisfagan la condición de holgura complemetaria (HC) 1.6; el nombre viene de la terminología estándar de programación lineal. Así para una asignación óptima, cada persona es asignada al objeto que le dé el máximo beneficio.

**Proposición 1.** Si una asignación factible y un conjunto de precios satisfacen la condición de holgura complementaria 1.6 para todas las personas $i$, entonces la asignación es óptima y los precios son una solución óptima del siguiente problema:

$$\min_{p_j,\ j=1,\dots,n} \left\{ \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \right\}$$

llamado problema dual. Además, el beneficio de la asignación es óptimo y es igual al óptimo del problema dual.

**Demostración:** El costo total de cualquier asignación factible $\{(i, k_i) \mid i = 1, \dots, n\}$ satisface:

$$\sum_{i=1}^{n} a_{ik_i} \leq \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j$$

ya que por dualidad se tiene que:

$$Z(x) \leq q(p)$$

para cualquier par de soluciones factibles. Por otra parte, dada una asignación y un conjunto de precios denotados por $\{(i, j_i) \mid i = 1, \dots, n\}$ y $\{\bar{p}_j \mid j = 1, \dots, n\}$, respectivamente, que satisfagan la condición HC, se tiene:

$$a_{ij_i} - \bar{p}_{j_i} = \max_j \{a_{ij} - \bar{p}_j\}, \qquad i = 1, \dots, n$$

sumando sobre todas las $i$ se tiene que:

$$\sum_{i=1}^{n} a_{ij_i} = \sum_{i=1}^{n} \left( \max_j \{a_{ij} - \bar{p}_j\} + \bar{p}_{j_i} \right).$$

Por lo tanto, la asignación $\{(i, j_i) \mid i = 1, ..., n\}$ obtiene el máximo del lado izquierdo y es óptima para el problema primal; mientras que, $\{\bar{p}_j \mid j = 1, ..., n\}$ obtiene el mínimo del lado derecho y es óptima para el problema dual, además los dos valores óptimos son iguales. Q.E.D.

# Capítulo 2

# Asignación utilizando el algoritmo ingenuo de subasta

En el problema clásico de asignación simétrico hay $n$ personas y $n$ objetos relacionados uno a uno. Hay un beneficio $a_{ij}$ que une cada persona $i$ con cada objeto $j$, y se quiere asignar personas a objetos de tal manera que se maximice el beneficio.

Una asignación significa un conjunto $S$ de parejas persona-objeto $(i, j)$, tal que, cada persona $i$ y cada objeto $j$ están implicados a lo más en una pareja de $S$. Si el número de parejas en $S$ es $n$, lo cual significa que todas las personas han sido involucradas en una asignación, se dice que $S$ es factible; de otra manera $S$ no es factible. Se busca una asignación factible (un conjunto de parejas $(1, j_1), ..., (n, j_n)$ persona-objeto, tal que los objetos $j_1, ...j_n$ sean todos distintos), la óptima será la que maximice el beneficio total $\sum_{i=1}^{n} a_{ij_i}$.

## 2.1 Algoritmo ingenuo de subasta

Se considera un proceso natural (como una subasta de obras de arte) para hallar una asignación y un vector de precios que cumplan ciertás condiciones de equilibrio que serán llamadas de holgura complemetaria. Se llamará a este proceso el *algoritmo ingenuo de subasta* porque tiene el defecto de que en algunas ocaciones no converge. En el capítulo tres se dará una condición de convergencia que soluciona el problema.

El algoritmo de subasta ingenuo trabaja en iteraciones y genera una secuencia de precios y asignaciones parciales, los precios se van incrementando con las ofertas que hacen las personas por sus objetos preferidos. Por una asignación parcial, se quiere decir que se tiene una asignación donde no todas las personas han sido asignadas a los objetos. Una asignación parcial puede ser diferente de una asignación completa, pues no necesariamente se tienen las mismas las parejas persona-objeto que se tenían en la asignación parcial. Al principio de cada iteración, la condición de HC, (ec. 1.6):

$$a_{ij_i} - p_{j_i} = \max_j \{a_{ij} - p_j\}$$

se satisface para todas las parejas $(i, j_i)$ de la asignación. Si todas las personas son asignadas, el algoritmo termina. De otra manera el subconjunto no vacío $I$ de personas no asignadas es seleccionado para la siguiente iteración.

### Algoritmo de subasta ingenuo

En cada una de las fases que se mencionan a continuación, se denota con $I$ el conjunto no vacío de personas no asignadas:

**Fase de oferta:** Cada persona $i \in I$ halla un objeto $j_i$ que hace que el valor de la oferta sea máximo esto es,

$$j_i = \arg\max_j \{a_{ij} - p_j\}$$

y entonces:

(a) la persona $i$ se asigna al mejor objeto $j_i$; la persona que fue asignada al objeto $j_i$ al principio de la iteración (si existe), ahora no está asignada;

(b) el precio de $j_i$ es $p_{j_i} + \gamma_i$, donde $\gamma_i$ es el incremento de la oferta;

calcúlese

$$\gamma_i = \nu_i - \omega_i$$

donde $\nu_i$, es el valor del mejor objeto para el cual la persona $i$ quiere ser asignada, y está dado por:

$$\nu_i = \max_j \{a_{ij} - p_j\} \qquad (2.1)$$

y $\omega_i$ es el valor del segundo mejor objeto y es:

$$\omega_i = \max_{j,\ j \neq j_i} \{a_{ij} - p_j\} \tag{2.2}$$

(Si $j_i$ es el único objeto en $A(i)$, se define $\omega_i$ como -$\infty$ o computacionalmente, como un número que sea mucho más pequeño que $\nu_i$).

**Fase de Asignación:** Cada objeto $j$ que se selecciona como el mejor objeto de un subconjunto no vacío $P(j)$ de personas en $I$, determina la mejor oferta:

$$i_j = \arg \max_{i \in P(j)} i$$

El algoritmo continúa con una secuencia de iteraciones hasta que todas las personas tengan un objeto asignado.

(Nótese que $p_{j_i}$ se incrementa, entonces el valor $a_{ij_i} - p_{j_i}$ que ofrece el objeto $j_i$ por la persona $i$ se decrementa, $\gamma_i$ es el incremento más grande por el cual $p_{j_i}$ se puede incrementar, de esta manera se mantiene la propiedad de que $j_i$ ofrece el valor máximo por $i$).

$\gamma_i$ no puede ser negativo, ya que $\nu_i \geq \omega_i$ (compare 2.1 y 2.2), así los precios de los objetos tienden a incrementarse. En realidad, cuando $i$ es la única oferta, $\gamma_i$ es el mayor incremento, para el cual la condición de HC se mantiene, teniendo la asignación de $i$ con su objeto preferido. Tal como en una subasta verdadera se incrementa la oferta y el precio crece, lo cual estimula la competencia.

Nótese también que hay una libertad en elegir el subconjunto $I$ de personas no asignadas durante la iteración de la oferta. Una posibilidad es considerar el conjunto $I$ con una sola persona no asignada. A esto se le conoce como la versión de *Gauss-Seidel* porque es similar a los métodos de *Gauss-Seidel* para resolver sistemas de ecuaciones no lineales, usualmente los mejores trabajos en serie, en un ambiente computacional. Si $I$ consiste de todas las personas no asignadas, es más eficiente resolverlo en forma paralela. Esto es conocido como la versión de *Jacobi;* a causa de su semejanza con los métodos de *Jacobi* para solucionar sistemas de ecuaciones no lineales.

**Ejemplo 1**

Encontrar una asignación por medio del algoritmo de subasta ingenuo, para el siguiente problema de asignación:

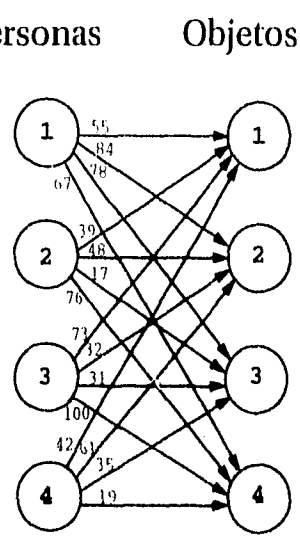



**Fig. 2.** *Problema de asignación*

El problema de asignación se puede representar en forma matricial, donde cada renglón representa a cada una de las personas y las columnas a cada uno de los objetos; el vector de precios inicial es el vector nulo.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 55 & 84 & 78 & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100 \\ 42 & 61 & 35 & 19 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (0,0,0,0)$$

Versión *Gauss-Seidel.* Para cada una de las personas se cálcula el $\max\{a_{ij} - p_j\}$, la representación matricial de la primera asignación parcial se muestra a

continuación:

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84^* & 78 & 67 \\ 39 & 48 & 17 & 76^* \\ 73 & 32 & 31 & 100 \\ 42 & 61 & 35 & 19 \end{array} \right) \end{array} \qquad \begin{array}{rcl} v_3 & = & 100 \\ \omega_3 & = & 73 \\ \gamma_3 & = & 27 \end{array}$$

Vector de precios $(0, 0, 0, 27)$

La persona 3 no es asignada porque su objeto preferido ya fue asignado; por lo tanto, hace una oferta $\gamma_3 = 27$, por el objeto 4 y al hacer esta oferta la persona 2 que tenía el objeto 4 ahora no está asignada:

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100^* \\ 42 & 61 & 35 & 19 \end{array} \right) \end{array} \qquad \begin{array}{rcl} v_4 & = & 61 \\ \omega_4 & = & 42 \\ \gamma_4 & = & 19 \end{array}$$

Vector de precios $(0, 19, 0, 27)$

La persona 4 hace la oferta $\gamma_4 = 19$ por el objeto 2, y al hacer esta oferta la persona 1 que tenía el objeto 2 ahora no está asignada:

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78 & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100^* \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array} \qquad \begin{array}{rcl} v_1 & = & 78 \\ \omega_1 & = & 65 \\ \gamma_1 & = & 13 \end{array}$$

Vector de precios $(0, 19, 13, 27)$

La persona 1 hace la oferta $\gamma_1 = 13$ por el objeto 3.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100^* \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array} \qquad \begin{array}{rcl} v_2 & = & 49 \\ \omega_2 & = & 39 \\ \gamma_2 & = & 10 \end{array}$$

Vector de precios $(0, 19, 13, 37)$

La persona 2 hace la oferta $\gamma_2 = 10$ por el objeto 4.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73 & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} v_3 & = & 73 \\ \omega_3 & = & 63 \\ \gamma_3 & = & 10 \end{array}$$

Vector de precios $(10, 19, 13, 37)$

La persona 3 hace la oferta $\gamma_3 = 10$ por el objeto 1.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73^* & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{pmatrix} \end{array}$$

Fin, todas las personas tienen asignado un objeto.

La asignación total

$$S = \{(1,3),(2,4),(3,1),(4,2)\}$$

y el vector de precios

$$p = (10, 19, 13, 37)$$

satisfacen la condición de holgura complementaria,

$$\sum_{(i,j)\in S} a_{ij} = \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j$$

$$\begin{array}{rcl} 78 + 76 + 73 + 61 & = & (65 + 39 + 63 + 42) + (10 + 19 + 13 + 37) \\ 288 & = & 288 \end{array}$$

por lo tanto, $S$ es una asignacion óptima.

Versión *Jacobi*. Para cada una de las personas se cálcula el $\max\{a_{ij} - p_j\}$, la representación matricial de la primera asignación parcial se muestra a continuación:

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{cccc} \text{objetos} & & & \\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 55 & 84^* & 78 & 67 \\ 39 & 48 & 17 & 76^* \\ 73 & 32 & 31 & 100 \\ 42 & 61 & 35 & 19 \end{pmatrix} \end{array} \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array}$$

$$\begin{array}{llll} \nu_3 = 100 & \nu_4 = 61 \\ \omega_3 = 73 & \omega_4 = 42 \\ \gamma_3 = 27 & \gamma_4 = 19 \end{array}$$

Vector de precios $(0, 19, 0, 27)$

La persona 3 no es asignada porque su objeto preferido ya fue asignado, por lo tanto, hace una oferta $\gamma_3 = 27$, por el objeto 4 y al hacer esta oferta la persona 2 que tenía el objeto 4 ahora no esta asignada. La persona 4 que tampoco ha sido asignada porque su objeto preferido ya fué asignado, hace una oferta $\gamma_4 = 19$, por el objeto 2 y al hacer esta oferta la persona 1 que tenía el objeto 2 ahora no esta asignada.

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{cccc} \text{objetos} & & & \\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 55 & 84 & 78 & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100^* \\ 42 & 61^* & 35 & 19 \end{pmatrix} \end{array} \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array}$$

$$\begin{array}{llll} \nu_1 = 78 & \nu_2 = 49 \\ \omega_1 = 65 & \omega_2 = 39 \\ \gamma_1 = 13 & \gamma_2 = 10 \end{array}$$

Vector de precios $(0, 19, 13, 37)$

La persona 1 hace la oferta $\gamma_1 = 13$ por el objeto 3, y la persona 2 hace una oferta $\gamma_2 = 10$ por el objeto 4

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{cccc} \text{objetos} & & & \\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73 & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{pmatrix} \end{array} \begin{array}{l} 1 \\ 2 \\ 3 \\ 4 \end{array}$$

$$\begin{array}{llll} v3 = 73 \\ \omega_3 = 63 \\ \gamma_3 = 10 \end{array}$$

Vector de precios $(10, 19, 13, 37)$

La persona 3 hace la oferta $\gamma_3 = 10$ por el objeto 1.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73^* & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array}$$

Fin, todas las personas tienen asignado un objeto.

La asignación total

$$S = \{(1,3),(2,4),(3,1),(4,2)\}$$

y el vector de precios

$$p = (10, 19, 13, 37)$$

satisfacen la condición de holgura complementaria,

$$\sum_{(i,j)\in S} a_{ij} = \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j$$

$$\begin{aligned} 78 + 76 + 73 + 61 &= (65 + 39 + 63 + 42) + (10 + 19 + 13 + 37) \\ 288 &= 288 \end{aligned}$$

por lo tanto, $S$ es una asignacion óptima.

Desafortunadamente, el algoritmo ingenuo de subasta no siempre trabaja bien. La dificultad se presenta cuando el incremento $\gamma_i$ es cero; es decir, más de un postor $i$ ofrece el valor máximo por el mismo objeto. Esta situación se crea cuando varias personas compiten por un número muy pequeño de objetos igualmente deseables sin aumentar sus precios, de esta manera se crea un ciclo que nunca termina. Ver el ejemplo 2.

## Ejemplo 2

Sea $a_{ij} = c > 0$ para toda $(i,j)$ con $i = 1,2,3$; y $j = 1,2$; y $a_{ij} = 0$ para toda $(i,j)$ con $i = 1,2,3$ y $j = 3$. Encontrar una asignación por medio del algoritmo ingenuo de subasta.

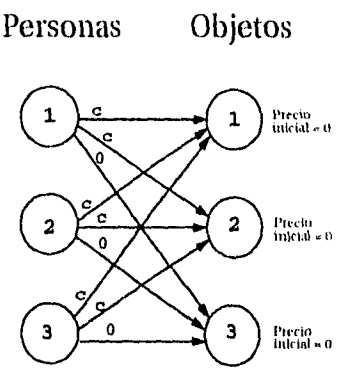



**Fig. 3.** *Problema de asignación*

El problema de asignación se puede representar en forma matricial, donde cada renglón representa a cada una de las personas y las columnas a cada uno de los objetos, el vector de precios inicial es el vector nulo.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c & c & 0 \\ c & c & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (0,0,0)$$

Para cada una de las personas se cálcula el $\max\{a_{ij} - p_j\}$, que se denota por $c^*$, la representación matricial se muestra a continuación:

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c^* & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_3 & = & c \\ \omega_3 & = & c \\ \gamma_3 & = & 0 \end{array}$$

$$\text{Vector de precios} \quad (0,0,0)$$

La persona 3 no es asignada porque su objeto preferido ya fue asignado, por lo tanto, hace una oferta $\gamma_3 = 0$, por el objeto 2 y al hacer esta oferta la persona

2 que tenía el objeto 2 ahora no esta asignada.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} \\ 1 \\ \text{personas} \quad 2 \\ 3 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c & 0 \\ c & c^* & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_2 & = & c \\ \omega_2 & = & c \\ \gamma_2 & = & 0 \end{array}$$

Vector de precios $(0,0,0)$

La persona 2 hace la oferta $\gamma_2 = 0$ por el objeto 1. Con lo cual se regresa a la asignación que se tenia en la primera iteración, y el vector de precios no ha sido modificado

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} \\ 1 \\ \text{personas} \quad 2 \\ 3 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c^* & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_3 & = & c \\ \omega_3 & = & c \\ \gamma_3 & = & 0 \end{array}$$

Vector de precios $(0,0,0)$

Aquí se ve, como el algoritmo de subasta ingenuo puede caer en un ciclo infinito para un problema con tres personas y tres objetos. El algoritmo tiene un ciclo entre la persona 2 y 3 que alternativamente ofrecen por el objeto 2 sin cambiar el precio, además estas personas prefieren de igual manera el objeto 1 y 2, ($\gamma = 0$). La situación es semajente si en la primera iteración la persona 3 ofrece una oferta por el objeto 1.

Para romper tal ciclo se introduce una modificación, motivada por subastas verdaderas, donde cada oferta debe aumentar el precio del objeto por un incremento mínimo positivo. En particular, se fija un escalar positivo, y se dice que una asignación y un vector de precios $p$ satisface $\epsilon$-holgura complementaria ($\epsilon$-HC) si

$$a_{ij_i} - p_{j_i} \geq \max_j \{a_{ij} - p_j\} - \epsilon$$

para todas las parejas $(i, j_i)$ asignadas. En otras palabras, se satisface $\epsilon$-HC, si todas las personas fueran asignadas a objetos que están a $\epsilon$ de ser los mejores.

Se reformulará el proceso de subasta hacia adelante, para que el incremento de la oferta sea al menos igual a $\epsilon$. El método resultante, es el mismo que el

algoritmo de subasta ingenuo, excepto que el incremento de la oferta $\gamma_i$ ahora es:

$$\gamma_i = v_i - \omega_i + \epsilon \tag{2.3}$$

Con esto, la condición $\epsilon$-HC se satisface. El incremento particular $\gamma_i = v_i - \omega_i + \epsilon$ usado en el algoritmo de subasta es la cantidad máxima con esta propiedad. Incrementos más pequeños $\gamma_i$ también trabajarían, ya que $\gamma_i \geq \epsilon$, pero se usará el incremento más grande posible para acelerar el algoritmo. Siendo consistentes con la experiencia de subastas verdaderas se sabe que se termina más rápido cuando la oferta es grande.

Para el caso de un problema completamente denso, hay que tener en cuenta que si un objeto recibe una oferta en $k$ iteraciones, su precio debe exceder el precio inicial por lo menos en $k\epsilon$. Así, para $k$ suficientemente grande, el objeto será bastante caro para ser juzgado como "inferior" y existe un objeto que no ha recibido una oferta hasta este momento y sé vuelve más atractivo. Esto es, un objeto puede recibir una oferta un número límite de iteraciones mientras que otros objetos fijos no tiene todavía alguna oferta. En cambio, una vez que todos los objetos reciban al menos una oferta, termina la subasta. El ejemplo 3 muestra como el algoritmo de subasta basado en el incremento de ofertas $\gamma_i = v_i - \omega_i + \epsilon$, supera el problema de ciclo del ejemplo 2.

Cuando el algoritmo de subasta termina, se tiene una asignación que satisface $\epsilon$-HC, pero, ¿es óptima?. La respuesta aquí depende del tamaño de $\epsilon$. En una subasta verdadera un postor prudente no colocará una oferta excesiva para poder ganar el objeto por un precio inecesariamente alto. Se puede mostrar que si $\epsilon$ es pequeño, entonces la asignación final será también "óptima". En particular, se muestra que el beneficio total de la asignación está a $n\epsilon$ de ser óptimo.

La idea es, que una asignación factible y un conjunto de precios que satisfagan $\epsilon$-HC se pueden ver como un problema que satisface HC (y son, por lo tanto primal y dual respectivamente), con una diferencia pequeña, todos los costos $a_{ij}$ son los mismos, excepto por el costo de las $n$ parejas asignadas, las cuales son modificadas por no más que $\epsilon$.

**Ejemplo 3**

Sea $a_{ij} = c > 0$ para toda $(i,j)$ con $i = 1,2,3$; y $j = 1,2$; y $a_{ij} = 0$ para toda $(i,j)$ con $i = 1,2,3$; y $j = 3$. Encontrar una asignación por medio del algoritmo de subasta

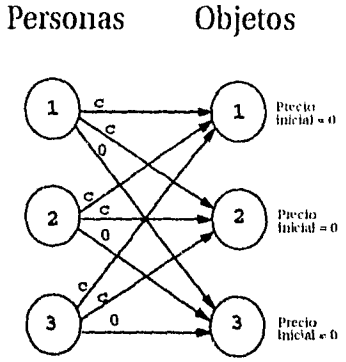



**Fig. 4.** *Problema de asignación*

Representando el problema de asignación en forma matricial, donde cada renglón representa a cada una de las personas y las columnas a cada uno de los objetos y el vector de precios inicial es el vector nulo, se tiene:

$$\begin{array}{rc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c & c & 0 \\ c & c & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (0,0,0)$$

Para cada una de las personas se cálcula el $\max\{a_{ij} - p_j\}$, que se denota por $c^*$, la representación matricial se muestra a continuación:

$$\begin{array}{rc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c^* & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_3 & = & c \\ \omega_3 & = & c \\ \gamma_3 & = & \epsilon \end{array}$$

$$\text{Vector de precios} \quad (0,\epsilon,0)$$

La persona 3 no es asignada porque su objeto preferido ya fue asignado, por lo tanto, hace una oferta $\gamma_3 = \epsilon$, por el objeto 2 y al hacer esta oferta la persona

2 que tenía el objeto 2 ahora no esta asignada.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c & 0 \\ c & c^* & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_2 & = & c \\ \omega_2 & = & c-\epsilon \\ \gamma_2 & = & 2\epsilon \end{array}$$

Vector de precios $(2\epsilon, \epsilon, 0)$

La persona 2 hace la oferta $\gamma_2 = 2\epsilon$ por el objeto 1, la persona 1 que tenía el objeto 1 ahora no esta asignada.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} c & c & 0 \\ c^* & c & 0 \\ c & c^* & 0 \end{pmatrix} \end{array} \qquad \begin{array}{rcl} \nu_1 & = & c-\epsilon \\ \omega_1 & = & c-2\epsilon \\ \gamma_1 & = & 2\epsilon \end{array}$$

Vector de precios $(2\epsilon, 3\epsilon, 0)$

La persona 1 hace una oferta $\gamma_1 = 2\epsilon$ por el objeto 2. Y así susesivamente, hasta llegar a la asignación.

Aquí se muestra como el algoritmo de subasta supera el problema del ciclado que hay en el ejemplo 2, haciendo el incremento de la oferta por lo menos igual a $\epsilon$. Se muestra una serie de posibles ofertas y asignaciones generadas por el algoritmo de subasta, empezando con todos los precios iguales a 0 y con la asignación parcial $\{(1,1),(2,2)\}$. En la última iteración, después que los precios de 1 y 2 exceden a $c$, el objeto 3 recibe una oferta y el algoritmo de subasta termina.

**Proposición 2.** Una asignación factible que satisfaga $\epsilon$-HC junto con un vector de precios está a $n\epsilon$ de una existencia óptima. Además, el vector de precios está a $n\epsilon$ de ser una solución óptima del problema dual.

**Demostración:** Sea $A^*$ una asignación óptima total de beneficios:

$$A^* = \max_{k_i, i=1,\ldots,n \;\; k_i \neq k_m \;\; si \;\; i \neq m} \sum_{i=1}^{n} a_{ik_i}$$

y sea $D^*$ el costo dual óptimo:

$$D^* = \min_{p_j} \min_{j=1,\ldots,n} \left\{ \sum_{i=1}^{n} \max_j \{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j \right\}$$

Si $\{(i, j_i) \mid i = 1, \ldots, n\}$ es la asignación dada que satisface la condición de $\epsilon$-HC junto con un vector de precios $\bar{p}$, se tiene:

$$\max_j \{a_{ij} - \bar{p}_j\} - \epsilon \leq a_{ij_i} - \bar{p}_{j_i}$$

$$\max_j \{a_{ij} - p_j\} + \bar{p}_{j_i} \leq a_{ij_i} + \epsilon$$

Sumando esta relación sobre todas las $i$, se ve que:

$$D^* \leq \sum_{i=1}^{n} \left( \max_j \{a_{ij} - \bar{p}_j\} + \bar{p}_{j_i} \right) \leq \sum_{i=1}^{n} a_{ij_i} + n\epsilon \leq A^* + n\epsilon$$

Como se sabe que $A^* = D^*$, esto muestra que la asignación total de beneficios $\sum_{i=1}^{n} a_{ij_i}$ está a $n\epsilon$ del valor óptimo $A^*$, mientras que el costo dual de $\bar{p}$ está a $n\epsilon$ del costo dual óptimo. Q.E.D.

# Capítulo 3

# Algoritmo de subasta

El algoritmo de subasta descrito antes, procede iterativamente y termina cuando se obtiene una asignación factible. Al empezar a generar las iteraciones se tiene una asignación parcial $S$ y un vector de precios $p$ que satisfacen $\epsilon$-holgura complemetaria ($\epsilon$-HC).

$$a_{ij} - p_j \geq \max_k\{a_{ik} - p_k\} - \epsilon. \qquad \forall \quad (i,j) \in S \tag{3.1}$$

Al iniciar el algoritmo se puede usar el vector de precios $p = (0, 0, ..., 0)$ junto con una asignación vacía, los cuales satisfacen $\epsilon$-HC, o bien, se puede empezar con una asignación parcial $S$ que junto con el vector de precios $p = (0, 0, ..., 0)$ también satisfagan $\epsilon$-HC, por ejemplo se puede elegir a $S$ de la siguiente manera, a cada persona $i$ se le asigna el objeto $j_i = \arg\max_j\{a_{ij}\}$, el cual toma el valor máximo, solo si el objeto $j_i$ no ha sido asignado a otra persona.

### Algoritmo de Subasta hacia adelante

**Fase de oferta:** Sea $I$ un subconjunto no vacío de personas $i$ que no están en la asignación parcial $S$. Para cada persona $i \in I$ :

1. Encontrar al "mejor" objeto $j_i$, es decir el que tenga el valor máximo:

$$j_i = \arg\max_j\{a_{ij} - p_j\},$$

y el correspondiente valor

$$v_i = \max_j\{a_{ij} - p_j\}, \tag{3.2}$$

encontrar el valor del segundo mejor objeto, es decir,

$$\omega_i = \max_{j,\, j \neq j_i} \{a_{ij} - p_j\} \tag{3.3}$$

2. Calcular la oferta de la persona $i$ dada por:

$$b_{ij_i} = p_{j_i} + \nu_i - \omega_i + \epsilon = a_{ij_i} - \omega_i + \epsilon \tag{3.4}$$

Se tiene esta situación cuando la persona $i$ ofrece por el objeto $j_i$, y el objeto $j_i$ recibe una oferta de la persona $i$. El algoritmo trabaja si la oferta tiene cualquier valor entre $p_{j_i} + \epsilon$ y $p_{j_i} + \nu_i - \omega_i + \epsilon$, pero tiende a trabajar más rápido si se escoge el máximo sobre todas las personas que hacen una oferta en la ec. 3.4.

**Fase de asignación:** Para cada objeto $j$, sea $P(j)$ el conjunto de personas de las cuales $j$ recibe una oferta en la iteración durante la fase de oferta. Si $P(j)$ es no vacío, incrementar $p_j$ por la oferta más alta,

$$p_j := \max_{i \in P(j)} b_{ij}, \tag{3.5}$$

remover de la asignación parcial $S$ cualquier pareja $(i, j)$ (si $j$ fué asignada a alguna $i$), y agregar a $S$ la pareja $(i_j, j)$, donde $i_j$ es una persona en $P(j)$ que obtiene el máximo ya mencionado.

Durante una iteración, los objetos cuyos precios fueron cambiados, son aquellos que recibieron una oferta durante la iteración. Cada precio involucrado se incrementa por lo menos en $\epsilon$, note que de la ecuación 3.2 a la 3.4 se tiene que:

$$b_{ij_i} = a_{ij_i} - \omega_i + \epsilon \geq a_{ij_i} - \nu_i + \epsilon = p_{j_i} + \epsilon,$$

así cada oferta por un objeto, incluye la ganancia de la oferta que excede al precio del objeto actual por lo menos en $\epsilon$. Al final de la iteración se tiene que una nueva asignación es diferente a la anterior, porque cada objeto que recibe una oferta, ahora es asignado a una persona que no estaba asignada al principio de la iteración. Sin embargo, al final de la iteración, la asignación no necesita tener más parejas que al principio de la iteración, porque es posible

que todos los objetos que recibieron una oferta estuvieron asignados al principio de la iteración.

El escoger el incremento de la oferta $\nu_i - \omega_i + \epsilon$ para cada persona $i$, ec. 3.4, permite que el algoritmo mantenga la condición $\epsilon$-HC, como se muestra en la siguiente proposición (de hecho se puede ver que es el incremento más grande para el cual se mantiene).

**Proposición 3.** El algoritmo de subasta mantiene la condición $\epsilon$-HC durante toda la ejecución; esto es, si la asignación y el vector de precios disponibles al principio de una iteración satisfacen $\epsilon$-HC, también se satisface para la asignación y para el vector de precios obtenidos al final de la iteración.

**Demostración:** Supóngase que el objeto $j^*$ recibe una oferta de la persona $i$ y que fué asignado a $i$ durante la iteración. Sean $p_j$ y $p'_j$ los precios de los objetos antes y después de la fase de asignación respectivamente. Entonces se tiene, (ver 3.4 y 3.5):

$$p'_{j^*} = a_{ij^*} - \omega_i + \epsilon$$

usando esta ecuación, se tiene:

$$a_{ij^*} - p'_{j^*} = \omega_i - \epsilon = \max_{j,\ j \neq j^*} \{a_{ij} - p_j\} - \epsilon$$

si $p'_j \geq p_j$ para toda $j$, esta ecuación implica que:

$$a_{ij^*} - p'_{j^*} \geq \max_j \{a_{ij} - p'_j\} - \epsilon \tag{3.6}$$

lo cual muestra que la condición $\epsilon$-HC 3.1 se mantiene después de la fase de asignación durante la iteración, para todas las parejas $(i, j^*)$ que entran a la asignación durante la iteración.

Se considera también cualquier pareja $(i, j^*)$ que pertenece a la asignación antes y después de la iteración. Entonces $j^*$ no debe haber recibido una oferta durante la iteración, así $p'_{j^*} = p_{j^*}$. Además, la ec. 3.6 tiene en cuenta la condición $\epsilon$-HC 3.1 que mantuvo antes de la iteración y de hecho, $p'_j = p_j$ para toda $j$. Así, la condición $\epsilon$-HC se matiene para todas las parejas $(i, j^*)$ que pertenecen a la asignación después de la iteración. Q.E.D.

Se presentará ahora algunas consideraciones que llevarán a demostrar que el algoritmo converge, se supone que los costos $a_{ij}$ son todos enteros; (si $a_{ij}$ es un número racional, se puede transformar a un entero por una multiplicación con un número apropiado). Entonces el beneficio total de cualquier asignación es entero. De este modo si $n\epsilon < 1$, cualquier asignación completa que está a $n\epsilon$ de ser óptima debe ser óptima si $\epsilon < 1/n$ y los beneficios $a_{ij}$ son todos enteros, entonces la asignación obtenida en términos del algoritmo de subasta es óptima. Se afirma este resultado con la siguiente proposición. La prueba se basa en los siguientes puntos:

(a) Una vez que un objeto es asignado permanece asignado durante el resto del algoritmo; además siempre existe un objeto que nunca ha sido asignado con el mismo precio inicial excepto al final del algoritmo. La razón es que la fase de oferta y de asignación pueden provocar una reasignación de un objeto que ya está asignado a otra persona diferente, pero no puede pasar que el objeto ya no quede asignado.

(b) Cada iteración en la que un objeto recibe una oferta, su precio se incrementa por lo menos en $\epsilon$, ver 3.4 y 3.5. Además, si el objeto recibe una oferta un número infinito de veces su precio se incrementa a $\infty$.

(c) En cada una de las n ofertas hechas por la persona $i$, el escalar $v_i$ definido por la ecuación:

$$v_i = \max_j \{a_{ij} - p_j\} \tag{3.7}$$

se disminuye por lo menos en $\epsilon$. La razón es que una oferta por una persona $i$ disminuye a $v_i$ por lo menos en $\epsilon$, o bien, $v_i$ no cambia porque hay más de un objeto $j$ que consigue el máximo en la ec. 3.7. Sin embargo, en el último caso, el precio del objeto $j_i$ que recibe la oferta se incrementará por lo menos en $\epsilon$, y el objeto $j_i$ no recibirá ninguna otra oferta por la persona $i$ hasta que $v_i$ disminuya por lo menos en $\epsilon$. La conclusión es que si la persona $i$ ofrece un número infinito de veces, $v_i$ debe tender a $-\infty$.

**Proposición 4.** Si existe al menos una asignación factible, el algoritmo de subasta termina con una asignación factible que está a $n\epsilon$ de ser óptima (y es óptima si los datos del problema son enteros y $\epsilon < 1/n$).

**Demostración:** Como se tiene una gráfica completa, sí existe una asignación factible. La demostración será por contradicción. Si nunca termina, el subconjunto $J^{\infty}$ de objetos que recibieron un número infinito de ofertas es no vacío. También, el subconjunto de personas $I^{\infty}$ que hace un número infinito de ofertas es no vacío. Como se argumentó en $(b)$ los precios de los objetos en $J^{\infty}$ deben tender a $\infty$, mientras que en $(c)$ se argumentó que los escalares $v_i = \max_j\{a_{ij} - p_j\}$ tienden a -$\infty$, para todas las personas $i \in I^{\infty}$, entonces las personas que hacen un número infinito de ofertas las hacen sobre todos los objetos, ya que si no ofrecieran por algún objeto en particular, en algún momento este sería atractivo para las personas que ofrecen un número infinito de veces, pues el precio de este objeto quedaría fijo y los precios de los demás objetos tienden a infinito, por lo tanto se tiene que:

$$\{j \mid j = 1, ..., n\} \subset J^{\infty}, \qquad \forall \quad i \in I^{\infty} \tag{3.8}$$

Esto quiere decir que todos los objetos objetos han pertenecido por lo menos una vez a alguna asignación parcial, considerando que una vez que un objeto es asignado, este permanece asignado durante el resto del algoritmo, se tiene que el número de objetos asignados es mayor al número de personas asignadas, lo cual contradice la existencia de una asignación factible, por lo tanto el algoritmo debe terminar. La asignación factible obtenida al terminar, satisface $\epsilon$-HC por la proposición 3 y por la proposición 1. La asignación está a $n\epsilon$ de ser óptima. Q.E.D.

## 3.1 Subasta hacia atrás

En el algoritmo de subasta, las personas compiten por los objetos ofreciendo por ellos e incrementando el precio del mejor objeto. Como el problema de asignación es simétrico, es posible cambiar los papeles de las personas y objetos. A esto se le llama *subasta hacia atrás* donde los objetos compiten por personas esencialmente ofreciendo descuentos (bajar sus precios).

Para describir este algoritmo se introduce una variable de beneficios $\pi_i$ para cada persona $i$. El papel que juega el beneficio con las personas es análogo al papel que el precio juega con los objetos. Cuando los objetos bajan sus precios tienden a crecer los beneficios de las personas, así las personas que

aprovechan el juego suelen aumentar los precios de los objetos. Se puede describir el algoritmo de subasta hacia atrás de dos maneras diferentes, una donde los objetos no asignados bajan sus precios tanto como sea posible para atraer a una persona sin violar $\epsilon$-HC, y otra donde los objetos no asignados escojen la mejor persona y aumentan su beneficio tanto como sea posible sin violar $\epsilon$-HC. Por conveniencia analítica, se tomará la segunda descripción, ya que con esta descripción, la subasta hacia adelante y hacia atrás serán matemáticamente equivalentes.

Considerando la siguiente condición $\epsilon$-HC para una asignación parcial $S$ y un vector de beneficios $\pi$

$$a_{ij} - \pi_i \geq \max_k \{a_{kj} - \pi_k\} - \epsilon, \qquad \forall \quad (i,j) \in S \tag{3.9}$$

Nótese la simetría de la condición $\epsilon$-HC 3.9 usando los beneficios y la correspondiente condición de holgura complementaria usando los precios como en la ec. 2.3. El algoritmo de subasta hacia atrás mantiene al principio de cada iteración una asignación parcial $S$ y un vector de beneficios $\pi$ que satisface la condición $\epsilon$-HC 3.9, y se termina cuando la asignación es factible.

Al iniciar el algoritmo se puede usar el vector de beneficios $\pi = (0, 0, ..., 0)$ junto con una asignación vacía, los cuales satisfacen $\epsilon$-HC o bien, se puede empezar con una asignación parcial $S$ que junto con el vector de beneficios $\pi = (0, 0, ..., 0)$ también satisfagan $\epsilon$-HC, por ejemplo se puede elegir a $S$ de la siguiente manera, a cada objeto $j$ se le asigna la persona $i_j = \arg\max_i \{a_{ij}\}$, la cual toma el valor máximo solo si la persona $i_j$ no ha sido asignada a otro objeto.

### Algoritmo de Subasta hacia atrás

**Fase inicial:** Sea $J$ un subconjunto no vacío de objetos $j$ que no están asignados y pertenecen a la asignación parcial $S$. Para cada objeto $j \in J$:

1. Hallar la "mejor" persona $i_j$, es decir la que tenga el valor máximo:

$$i_j = \arg\max_i \{a_{ij} - \pi_i\}$$

y el valor correspondiente:

$$\beta_j = \max_i \{a_{ij} - \pi_i\} \tag{3.10}$$

y hallar:

$$\omega_j = \max_{i,\, i \neq i_j} \{a_{ij} - \pi_i\} \tag{3.11}$$

2. Cada objeto $j \in J$ ofrece por la persona $i_j$ un incremento

$$b_{i_j j} = \pi_{i_j} + \beta_j - \omega_j + \epsilon = a_{i_j j} - \omega_j + \epsilon \tag{3.12}$$

**Fase de asignación:** Para cada persona $i$ que recibe por lo menos una oferta, incrementar $\pi_i$ por la oferta más grande

$$\pi_i := \max_{j \in P(i)} b_{ij} \tag{3.13}$$

donde $P(i)$ es el conjunto de objetos para el cual $i$ recibe una oferta; remover de la asignación parcial $S$ cualquier pareja $(i, j)$ (si $i$ fue asignada a algún $j$ bajo $S$), y agregar a $S$ la pareja $(i, j_i)$, donde $j_i$ es un objeto en $P(i)$ que consigue el valor máximo.

Nótese que la subasta hacia atrás es idéntica a la subasta hacia adelante con los papeles de personas y objetos intercambiados así como los beneficios y precios. Para usar la correspondiente subasta hacia atrás se tiene la siguiente proposición:

**Propocición 5.** Si existe por lo menos una asignación factible, el algoritmo de subasta hacia atrás termina con una asignación factible que está a $n\epsilon$ de ser óptima (y es óptima si los datos del problema son enteros y $\epsilon < 1/n$). La demostración es análoga a la del algoritmo de subasta hacia adelante.

## 3.2 Combinación de la subasta hacia adelante y hacia atrás

Una de las razónes por las que se está interesado en el algoritmo de subasta hacia atrás es construir un algoritmo que intercambie la subasta hacia adelante

y hacia atrás. De tal manera que los algoritmos simultáneamente mantengan un vector de precios $p$ que satisface la condición $\epsilon$-HC 2.3 y un vector de beneficios $\pi$ que satisface la condición $\epsilon$-HC 3.9. Por esta razón introduce una condición $\epsilon$-HC para la pareja $(\pi, p)$ en la cual se verán implicadas las otras dos condiciones. Mantener esta condición es esencial para cambiar agraciadamente entre la subasta hacia adelante y hacia atrás.

**Definición:** Una asignación parcial $S$ y una pareja $(\pi, p)$ satisface $\epsilon$-HC si

$$\pi_i + p_j \geq a_{ij} - \epsilon, \qquad \forall \quad i, j = 1, ..., n \tag{3.14}$$

$$\pi_i + p_j = a_{ij}, \qquad \forall \quad (i,j) \in S \tag{3.15}$$

se tiene la siguiente proposición.

**Proposición 6.** Supóngase que una asignación parcial $S$ junto con la pareja $(\pi, p)$ beneficio-precio satisfacen $\epsilon$-HC. Entonces:

(a) $S$ y $\pi$ satisfacen la condición $\epsilon$-HC

$$a_{ij} - \pi_i \geq \max_k \{a_{kj} - \pi_k\} - \epsilon, \qquad \forall \quad (i,j) \in S \tag{3.16}$$

(b) $S$ y $p$ satisfacen la condición $\epsilon$-HC

$$a_{ij} - p_j \geq \max_k \{a_{ik} - p_k\} - \epsilon, \qquad \forall \quad (i,j) \in S \tag{3.17}$$

(c) Si $S$ es factible, entonces $S$ está a $n\epsilon$, de ser una asignación óptima.

**Demostración:**

(a) Para toda $(i,j) \in S$, se tiene que $p_j = a_{ij} - \pi_i$, por la ec. 3.15 y para toda $(i,j)$ se tiene que $p_j \geq a_{kj} - \pi - \epsilon$ por la desigualdad 3.14 lo cual implica que $a_{ij} - \pi_i \geq a_{kj} - \pi_k - \epsilon$ para todo $k$. Esto muestra la desigualdad 3.16.

(b) La prueba es la misma que en la parte (a) con los papeles de $\pi$ y $p$ intercambiados.

(c) En la parte (b), la condición $\epsilon$-HC 3.17 se satisface, por la proposición 2, $S$ está a $n\epsilon$ de la asignación óptima. Q.E.D.

Ahora se introducirá una combinación del algoritmo hacia adelante y hacia atrás. El algoritmo mantiene al principio de cada iteración una asignación parcial $S$ y una pareja de beneficios y precios $(\pi, p)$ que satisfacen la condición $\epsilon$-HC en 3.14 y en 3.15. Termina cuando la asignación es factible. Un manera común para inicializar el algoritmo tal que satisfaga la condición $\epsilon$-HC es tomar $S$ vacío, elegir a $p = (0, 0, ..., 0)$ arbitrariamente y elegir $\pi$ como una función de $p$ vía la relación $\pi_i = \max_k \{a_{ik} - p_k\}$ para cada persona $i$.

### Combinación de la subasta hacia adelante y hacia atrás

**Paso 1. (Ejecutar la subasta hacia adelante).** Ejecutar varias iteraciones del algoritmo de subasta hacia adelante (sujeto a la condición de terminación) y al final de cada iteración, después de incrementar los precios de los objetos que reciben una oferta, calcular

$$\pi_i = a_{ij_i} - p_{j_i} \tag{3.18}$$

para cada pareja persona-objeto $(i, j_i)$ que entra a la asignación durante la iteración. Ir al paso 2.

**Paso 2. (Ejecutar la subasta hacia atrás).** Ejecutar varias iteraciones del algoritmo de subastas hacia atrás (sujeto a la condición de terminación) y al final de cada iteración (después de incrementar los beneficios de las personas que reciben una oferta) calcular

$$p_j = a_{i_j,j} - \pi_{i_j} \tag{3.19}$$

para cada pareja persona-objeto $(i_j, j)$ que entra a la asignación durante la iteración. Ir al paso 1.

Nótese que el gasto (overhead) adicional de combinar el algoritmo hacia adelante con el algoritmo hacia atrás es mínimo; sólo se requiere una actualización de la forma 3.18 ó 3.19 por iteración para cada objeto o persona que reciba una oferta durante la iteración. Una propiedad importante es que las actualizaciones de 3.18 y 3.19 mantienen la condición $\epsilon$-HC 3.14 para la pareja $(\pi, p)$, por lo tanto, por la proposición 6 se mantiene el requerimiento de la condición $\epsilon$-HC 3.16 y 3.17 para $\pi$ y $p$ respectivamente. Esto se afirma en la siguiente

proposición.

**Proposición 7.** Si la asignación y la pareja beneficio-precio disponible al empezar una iteración, ya sea del algoritmo de subasta hacia adelante o hacia atrás, satisfacen la condición $\epsilon$-HC 3.14, entonces también se satisface para la asignación y la pareja de beneficio-precio obtenida al final de la iteración dado que la ec. 3.18 se usa para actualizar $\pi$ (en el caso de la subasta hacia adelante) y la ec. 3.19 se usa para la actualización de $p$ (en el caso de subasta hacia atrás).

**Demostración:** Supóngase por definición que se inicia con la subasta hacia adelante sea $(\pi, p)$ y $(\bar{\pi}, \bar{p})$ la pareja beneficio-precio, antes y después de la iteración, repectivamente. Entonces $\bar{p}_j \geq p_j$ para toda $j$, la desigualdad es estricta si y sólo si $j$ recibe una oferta durante la iteración. Por lo tanto, se tiene $\bar{\pi}_i + \bar{p}_j \geq a_{ij} - \epsilon$ para toda $(i, j)$ tal que $\pi_i = \bar{\pi}_i$. Además, se tiene que $\bar{\pi}_i + \bar{p}_j = \pi_i + p_j = a_{ij}$ para toda $(i, j)$ ya que pertenece a la asignación antes y después de la iteración. También al actualizar la ec. 3.18, se tiene $\bar{\pi}_i + \bar{p}_j = a_{ij_i}$ para todas las parejas $(i, j_i)$ que entran a la asignación durante la iteración. La condición se mantiene para todas las personas $i$ que someten una oferta y fueron asignadas al objeto $j_i$ durante la iteración.

$$\bar{\pi}_i + \bar{p}_j \geq a_{ij} - \epsilon, \qquad \forall \quad j = 1, ..., n \tag{3.20}$$

Desde luego, para cada persona $i$ se tiene por la ec. 3.4 que:

$$\bar{p}_{j_i} = a_{ij_i} - \max_{j,\ j \neq j_i} \{a_{ij} - p_j\} + \epsilon,$$

lo cual implica que:

$$\bar{\pi}_i = a_{ij_i} - \bar{p}_{j_i} \geq a_{ij} - p_j - \epsilon \geq a_{ij} - \bar{p}_j - \epsilon, \qquad \forall \quad j = 1, ..., n.$$

Esto muestra la relación 3.20 deseada. Q.E.D.

Nótese que durante la subasta hacia adelante los precios $p_j$ de los objetos crecen, mientras que los beneficios $\pi_i$ disminuyen, pero pasa exactamente lo contrario en la subasta hacia atrás. Por ésta razón, la prueba de terminación usada para la subasta hacia adelante y hacia atrás no se aplica al método combinado. Desde luego, es posible construir ejemplos de problemas factibles donde al combinar el método nunca se termina si los cambios entre el algoritmo hacia adelante y hacia atrás son hechos arbitrariamente, sin embargo,

es fácil garantizar que al combinar el algoritmo se termina finitamente para un problema factible; es suficiente asegurarse de que se realicen "progresos irreversibles" antes de cambiar entre el algoritmo hacia adelante y hacia atrás. Una implementación fácil se obtiene al abstenerse de cambiar hasta que por lo menos una o más parejas de persona-objeto hayan sido agregadas a la asignación.

De esta manera se puede cambiar a lo más $(n-1)$ tiempos entre los pasos hacia adelante y hacia atrás del algoritmo. Para un problema factible, la subasta hacia adelante y hacia atrás por sí ha garantizado una terminación finita, así en la iteración final se termina con una asignación factible satisfaciendo $\epsilon$-HC. El siguiente ejemplo muestra el uso del algoritmo de subasta combinado

**Ejemplo 4**

Encontrar una asignación por medio del algoritmo de subasta combinado, para el problema de asignación del ejemplo 1.

Subasta hacia adelante

Para cada una de las personas se cálcula el $\max\{a_{ij} - p_j\}$, la representación matricial de la primera asignación parcial se muestra a continuación:

| | | objetos | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | | | | | |
| personas | 1 | 55 | 84* | 78 | 67 | $\nu_3$ | = | 100 | $\nu_4$ | = 61 |
| | 2 | 39 | 48 | 17 | 76* | $\omega_3$ | = | 73 | $\omega_4$ | = 42 |
| | 3 | 73 | 32 | 31 | 100 | $\gamma_3$ | = | $27+\epsilon$ | $\gamma_4$ | = $19+\epsilon$ |
| | 4 | 42 | 61 | 35 | 19 | | | | | |

Vector de precios $(0,0,0,0)$

Vector de beneficios $(84,76,0,0)$

La persona 3 no es asignada porque su objeto preferido ya fue asignado, por lo tanto, hace una oferta $\gamma_3 = 27+\epsilon$, por el objeto 4 y al hacer esta oferta la persona 2 que tenía el objeto 4 ahora no esta asignada. La persona 4 que tampoco ha sido asignada porque su objeto preferido ya fué asignado, hace

una oferta $\gamma_4 = 19 + \epsilon$, por el objeto 2 y al hacer esta oferta la persona 1 que tenía el objeto 2 ahora no esta asignada.

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78 & 67 \\ 39 & 48 & 17 & 76 \\ 73 & 32 & 31 & 100^* \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array} \quad \begin{array}{lcllcl} \nu_1 & = & 78 & \nu_2 & = & 49 \\ \omega_1 & = & 65 & \omega_2 & = & 39 \\ \gamma_1 & = & 13 + \epsilon & \gamma_2 & = & 10 + \epsilon \end{array}$$

$$\text{Vector de precios} \quad (0, 19 + \epsilon, 0, 27 + \epsilon)$$

$$\text{Vector de beneficios} \quad (84, 76, 73 - \epsilon, 42 - \epsilon)$$

La persona 1 hace la oferta $\gamma_1 = 13 + \epsilon$ por el objeto 3, y la persona 2 hace una oferta $\gamma_2 = 10 + \epsilon$ por el objeto 4

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73 & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array} \quad \begin{array}{lcl} v1 & = & \epsilon \\ \omega_1 & = & \epsilon \\ \gamma_1 & = & \epsilon \end{array}$$

$$\text{Vector de precios} \quad (0, 19 + \epsilon, 13 + \epsilon, 37 + \epsilon)$$

$$\text{Vector de beneficios} \quad (65 - \epsilon, 39 - \epsilon, 73 - \epsilon, 42 - \epsilon)$$

Subasta hacia atrás

El objeto 1 ofrece $\gamma_1 = \epsilon$ por la persona 3

$$\begin{array}{cc} & \text{objetos} \\ & \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \text{personas} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \left( \begin{array}{cccc} 55 & 84 & 78^* & 67 \\ 39 & 48 & 17 & 76^* \\ 73^* & 32 & 31 & 100 \\ 42 & 61^* & 35 & 19 \end{array} \right) \end{array}$$

$$\text{Vector de precios} \quad (0, 19 + \epsilon, 13 + \epsilon, 37 + \epsilon)$$

Vector de beneficios $(65-\epsilon, 39-\epsilon, 73-\epsilon, 42-\epsilon)$

Fin, todas las personas tienen asignado un objeto.

La asignación total

$$S = \{(1,3),(2,4),(3,1),(4,2)\}$$

el vector de precios y el de beneficios, son:

$$p = (0, 19+\epsilon, 13+\epsilon, 37+\epsilon)$$

$$\pi_i = (65-\epsilon, 39-\epsilon, 73, 42-\epsilon)$$

satisfacen la condición de holgura complementaria,

$$\sum_{j=1}^{n} p_j + \sum_{i=1}^{n} \pi_i = \sum_{(i,j)\in S} a_{ij}$$

$$\begin{aligned} 78+76+73+61 &= 65-\epsilon+39-\epsilon+73+42-\epsilon+ \\ &\quad 13+\epsilon+37+\epsilon+0+19+\epsilon \\ 288 &= 288 \end{aligned}$$

por lo tanto, $S$ es una asignacion óptima.

La combinación del algoritmo de subasta hacia adelante y hacia atrás típicamente trabaja substancialmente más rápido que la versión hacia adelante. Parece ser menos afectado por el fenómeno de "la guerra de precios" que es provocada por la secuencia de pequeños incrementos en los precios por un número de personas que ofrecen por un número pequeño de objetos. Las guerras de precios que aún pueden ocurrir en el algoritmo combinado, surgen por problemas más complejos. Por ésta razón al combinar el algoritmo de subasta hacia adelante y hacia atrás depende menos de un factor de $\epsilon$ para una buena ejecución.

# Capítulo 4

# Problema de ruta más corta

El problema de ruta más corta es un problema clásico e importante de combinatoria que está implementado en muchos contextos. Dada una gráfica conexa dirigida $(N, A)$ con nodos $1, ..., N$, cada arco $(i, j) \in A$ tiene un costo o "longitud" $a_{ij}$ asociado. La longitud de una ruta $(i_1, i_2, ..., i_k)$ es igual a la longitud de sus arcos

$$\sum_{n=1}^{k-1} a_{i_n i_{n+1}}$$

Una ruta es la más corta si tiene la longitud mínima sobre todas las otras rutas con los mismos nodos origen y destino. La longitud de la ruta más corta también es llamada la *distancia más corta*. La distancia más corta de un nodo a si mismo es por convención cero. El problema de ruta más corta encuentra la distancia más corta entre una pareja de nodos seleccionados. (Nótese que se optimiza sobre todas las rutas dirigidas, esto es, la ruta consiste de arcos dirigidos, de esta manera para cada ruta (o ciclo) que se analice se supondrá que es dirigida).

Todos los mejores algoritmos de rutas más cortas están basados en la siguiente proposición.

**Proposición 8.** Sea $d = (d_1, d_2, ..., d_N)$ un vector que satisface

$$d_j \leq d_i + a_{ij}, \qquad \forall \quad (i, j) \in A \tag{4.1}$$

y sea $P$ una ruta que empieza en el nodo $i_1$ y termina en el nodo $i_k$. Si

$$d_j = d_i + a_{ij}, \qquad \text{para todos los arcos} \quad (i, j) \in P \tag{4.2}$$

entonces $P$ es la ruta más corta de $i_1$ a $i_k$.

**Demostración:** Sumando la ecuación 4.2 sobre todos los arcos de $P$, se ve que la longitud de $P$ es $d_{i_k} - d_{i_1}$. Sumando la ecuación 4.1 sobre todos los arcos de cualquier otra ruta $P'$ que empieza en $i_1$ y termina en $i_k$, se ve que la longitud de $P'$ debe ser por lo menos igual a $d_{i_k} - d_{i_1}$. Por lo tanto $P$ es una ruta más corta. Q.E.D.

Las condiciones 4.1 y 4.2 se les llamara holgura complementaria (HC) para el problema de ruta más corta, las cuales toman la forma de:

$$p_i \leq a_{ij} + p_j, \qquad \forall \quad (i,j) \in A \tag{4.3}$$

$$p_i = a_{ij} + p_j, \qquad \text{para todos los arcos} \quad (i,j) \in P \tag{4.4}$$

## 4.1 Algoritmo hacia adelante para el problema de ruta más corta

En esta sección se considerará un algoritmo para encontrar la ruta más corta de un origen a un solo destino en una gráfica conexa dirigida $(N, A)$. Se verá después que este algoritmo es una aplicación del algoritmo ingenuo de subasta (el algoritmo de subasta con $\epsilon = 0$).

Se supone que se tiene el conjunto de nodos $N$ y el conjunto de arcos $A$ de una gráfica conexa y la longitud $a_{ij}$ para cada arco $(i,j)$. Por una ruta, se entenderá una serie de nodos $(i_1, i_2, ..., i_k)$, tal que, $(i_m, i_{m+1})$ es un arco para toda $m = 1, ..., k-1$. Si, en forma creciente, los nodos $i_1, i_2, ..., i_k$ son distintos la serie $(i_1, i_2, ..., i_k)$ se llamará *ruta simple*, es decir no contiene arcos repetidos ni nodos repetidos. La longitud de una ruta está definida por la suma de sus longitudes de arco. Se quiere hallar una ruta de longitud mínima sobre todas las rutas que empiezan en un origen dado (nodo 1) y terminan en un destino dado (nodo $t$).

Para simplificar la presentación se supone que todos los ciclos tienen longitud positiva, también se supondrá que cada nodo, excepto el destino, tiene al menos un arco que sale de él; cualquier nodo que no satisfaga esta condición se puede conectar al nodo destino con un arco de longitud muy grande sin alterar

el problema y sin alterar al algoritmo siguiente.

En el caso donde sólo se tiene un origen, el algoritmo es muy simple. En cada iteración la ruta es *extendida* agregando un nuevo nodo o *contraida* borrando el nodo final. Cuando el destino es el nodo final de la ruta, el algoritmo termina.

En particular, dada una ruta simple $P$ y un vector de precios $p$, se dice que la pareja $(P,p)$ satisface la condición de holgura complemetaria (HC) si

$$p_i \leq a_{ij} + p_j, \qquad \forall \quad (i,j) \in A \tag{4.5}$$

$$p_i = a_{ij} + p_j, \qquad \text{para todas las parejas de nodos } i \text{ y } j \text{ de } P \tag{4.6}$$

Esto quiere decir que si una pareja $(P,p)$ satisface la condición HC, entonces la porción de $P$ entre cualquier nodo $i \in P$ y cualquier nodo $k \in P$ es la ruta más corta de $i$ a $k$, y $p_i - p_k$ es la correspondiente distacia más corta. Se puede ver esto directamente observando que por la ec. 4.6, $p_i - p_k$ es la longitud entre $i$ y $k$ de la porción de $P$, y cada ruta que conecta a $i$ con $k$ debe ser al menos igual a $p_i - p_k$ (esto es por la ec. 4.5).

### 4.1.1 Descripción y análisis del algoritmo hacia adelante para el problema de ruta más corta

Se describirá el algoritmo de subasta ingenuo directamente en términos del problema original de ruta más corta para el caso de un solo origen y un solo destino. Sea el nodo 1 el nodo origen y $t$ el nodo destino. El algoritmo mantiene todo el tiempo una ruta simple $P = (1, i_1, ..., i_k)$. (Se supone que la ruta es dirigida, esto es, todos los arcos de la ruta son arcos dirigidos). Al nodo $i_k$ se le llama nodo final de $P$. La ruta degenerada $P = (1)$ también se puede obtener en el transcurso del algoritmo. Si $i_{k+1}$ es un nodo que no pertenece a la ruta $P = (1, i_1, ..., i_k)$ y $(i_k, i_{k+1})$ es un arco, *extender* a $P$ por $i_{k+1}$, es decir, remplazar a $P$ por la ruta $(1, i_1, ..., i_k, i_{k+1})$, que es llamada *la extensión* de $P$ por $i_k$. Si $P$ no consiste sólo del nodo origen 1 entonces, *contraer* a $P$, es decir decir, remplazar a $P$ por la ruta $(1, i_1, ..., i_{k-1})$.

El algoritmo mantiene también un vector de precios $p$ que satisface HC junto con $P$. Se supone que se dispone de una pareja inicial $(P,p)$ que satisfaga HC, uno puede usar por default la pareja

$$P = (1), \qquad p_i = 0, \qquad \forall \quad i \in N$$

Ahora se describirá el algoritmo. Inicialmente, $(P, p)$ es cualquier pareja que satisface HC. El algoritmo procede en iteraciones, transformando la pareja $(P, p)$ que satisface HC en otra pareja que satisface HC. En cada iteración la ruta $P$ es *extendida* con un nuevo nodo o *contraida* borrando su nodo final. En el segundo caso, el precio del nodo final se incrementa estrictamente. Un caso degenerado ocurre cuando la ruta consiste únicamente del nodo origen 1; en este caso la ruta o se *extiende* o se deja igual incrementando el precio de $p_1$.

**Algoritmo hacia adelante para resolver el problema de ruta más corta**

**Iteración común:**

Sea $i$ el nodo final de $P$. Si

$$p_i < \min_{(i,j)\in A}\{a_{ij} + p_j\}$$

ir al paso 1; si no ir al paso 2.

**Paso 1 (Contraer la ruta):** Sea

$$p_i := \min_{(i,j)\in A}\{a_{ij} + p_j\} \tag{4.7}$$

y si $i \neq 1$, contraer a $P$. Si no, ir a la sigiente iteración.

**Paso 2 (Extender la ruta):** Extender la ruta $P$ por el nodo $j_i$, donde

$$j_i = \arg\min_{(i,j)\in A}\{a_{ij} + p_j\} \tag{4.8}$$

Si $j_i$ es el destino $t$, parar; $P$ es la ruta más corta deseada. De otra manera ir a la siguiente iteración.

Nótese que una extensión de $P$ (paso 2) es una ruta simple de 1 a $j_i$; si esto no fuera así, entonces al agregar $j_i$ a $P$ se crearía un ciclo, y para cada arco $(i, j)$ de este ciclo se tendría $p_i = a_{ij} + p_j$. Al agregar esta condición a lo largo del ciclo se tendrá una longitud cero, lo cual no es posible por hipótesis.

**Ejemplo 5.**

Encontrar la ruta más corta del siguiente problema:

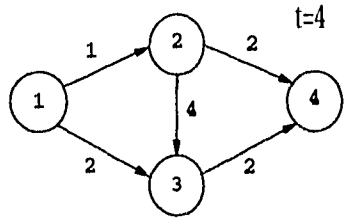



**Fig. 5.** *Problema de ruta más corta*

se comienza con:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (0, 0, 0, 0)$ como aún no se llega al destino y

$$p_1 = 0 < \min\{1 + 0, 2 + 0\} = 1$$

se contrae a 1 y se actualiza $p_1$

ahora se tiene:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (1, 0, 0, 0)$ como aún no se llega al destino y

$$p_1 = 1 \not< \min\{1 + 0, 2 + 0\} = 1$$

se extiende la ruta al nodo 2, ya que,

$$2 = \arg\min\{1 + 0, 2 + 0\}$$

ahora se tiene:
$P = (1, 2)$, el nodo final de la ruta es $i = 2$
$p = (1, 0, 0, 0)$ como aún no se llega al destino, y

$$p_2 = 0 < \min\{2 + 0\} = 2$$

se contrae 2 y se actualiza $p_2$

$$p_2 = \min\{2 + 0\} = 2$$

ahora se tiene:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (1, 2, 0, 0)$ como aún no se llega al destino, y

$$p_1 = 1 < \min\{1 + 2, 2 + 0\} = 2$$

se contrae 1 y se actualiza $p_1$

$$p_1 = \min\{1 + 2, 2 + 0\} = 2$$

ahora se tiene:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (2, 2, 0, 0)$ como aún no se llega al destino y

$$p_1 = 2 \not< \min\{1 + 2, 2 + 0\} = 2$$

se extiende la ruta al nodo 3, ya que,

$$3 = \arg\min\{1 + 2, 2 + 0\}$$

ahora se tiene:
$P = (1, 3)$, el nodo final de la ruta es $i = 3$
$p = (2, 2, 0, 0)$ como aún no se llega al destino y

$$p_3 = 0 < \min\{2 + 0\} = 2$$

se contrae a 3 y se actualiza $p_3$

ahora se tiene:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (2, 2, 2, 0)$ como aún no se llega al destino y

$$p_1 = 2 < \min\{1 + 2, 2 + 2\} = 3$$

se contrae a 1 y se actualiza $p_1$

ahora se tiene:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (3, 2, 2, 0)$ como aún no se llega al destino y

$$p_1 = 3 \not< \min\{1 + 2, 2 + 2\} = 3$$

se extiende la ruta al nodo 2, ya que,

$$2 = \arg\min\{1+2, 2+2\}$$

ahora se tiene:
$P = (1,2)$, el nodo final de la ruta es $i = 2$
$p = (3,2,2,0)$ como aún no se llega al destino y

$$p_1 = 2 \neq \min\{2+0\} = 2$$

se extiende la ruta al nodo 4, ya que,

$$4 = \arg\min\{2+0\}$$

ahora se tiene:
$P = (1,2,4)$, el nodo final de la ruta es $i = 4$
$p = (3,2,2,0)$

FIN, se ha encontrado la ruta más corta $P = (1,2,4)$ de 1 a 4,
con longitud igual a $p_1 - p_4 = 3$

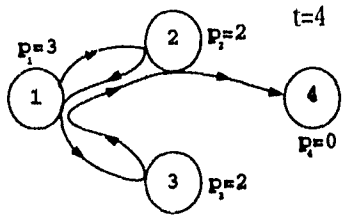



**Fig. 6.** *Trayectoria para encontrar la ruta más corta*

Ahora se deriva una propiedad del algoritmo y se establece su validez

**Proposición 9.** La pareja $(P,p)$ generada por el algoritmo satisface HC. Además, para cada pareja de nodos $i$ y $j$, se tiene que en todas las iteraciones, $p_i - p_j$ es una subestimación de la distancia más corta de $i$ a $j$.

**Demostración**: Primero se mostrará por inducción que $(P,p)$ satisface HC. La pareja inicial satisface HC por hipótesis. Ahora se considerará una iteración que empieza con una pareja $(P,p)$ que satisface HC y produce una pareja $(\bar{P},\bar{p})$. Sea $i$ el nodo final de $P$. Si

$$p_i = \min_{(i,j)\in A}\{a_{ij} + p_j\} \tag{4.9}$$

Entonces $\bar{P}$ es la extensión de $P$ por un nodo $j_i$ y $\bar{p} = p$, esto implica que la condición HC 4.6 se tiene para todos los arcos de $P$ y también para $(i, j_i)$.

Ahora supóngase que:

$$p_i < \min_{(i,j)\in A}\{a_{ij} + p_j\}$$

si $P$ es la ruta degenerada que consistente solo del vértice 1 ($P = (1)$), la condición de HC se mantiene, si no, $\bar{P}$ se obtiene por contracción de $P$, lo cual implica que $\bar{p}_i > p_i$, y que para todos los nodo $j \in \bar{P}$, se tiene que $\bar{p}_j = p_j$, lo cual implica que la condición 4.5 y 4.6 se satisface para los arcos que salen de los nodos de $\bar{P}$. También para el nodo final $i$, se tiene que:

$$\bar{p}_i = \min_{(i,j)\in A}\{a_{ij} + p_j\}$$

esto implica que la condición 4.5 se satisface para los arcos que salen de ese nodo. Además, si $\bar{p}_i > p_i$ y $\bar{p}_k = p_k$, para toda $k \neq i$, se tiene que $\bar{p}_k \leq a_{kj} + \bar{p}_j$ para todos los arcos $(k, j)$ que salen de nodos $k \notin P$. Esto completa la inducción que prueba que $(P, p)$ satisface HC.

Finalmente se considera cualquier ruta de un nodo $i$ a un nodo $j$ . Considerando la condición de HC 4.5 a lo largo de la ruta $(i_i, i_{i+1}, ..., i_{j-1}, i_j)$ que va de $i$ a $j$ se tiene que:

$$\begin{aligned} p_i &\leq a_{i,i+1} + p_{i+1}, \qquad \forall \quad (i, i+1) \in A \\ p_{i+1} &\leq a_{i+1,i+2} + p_{i+2}, \qquad \forall \quad (i+1, i+2) \in A \\ &\vdots \\ p_{j-1} &\leq a_{j-1,j} + p_j, \qquad \forall \quad (j-1, j) \in A \end{aligned}$$

sumando las desigualdades anteriores, se tiene que:

$$p_i + p_{i+1} + ... + p_{j-1} \leq a_{i,i+1} + p_{i+1} + a_{i+1,i+2} + p_{i+2} + ... + a_{j-1,j} + p_j$$

reduciendo términos se llega a que:

$$p_i - p_j \leq a_{i,i+1} + a_{i+1,i+2} + ... + a_{j-1,j}$$

por lo que se vé que la longitud de la ruta es al menos $p_i - p_j$, esto prueba la última afirmación de la proposición. Q.E.D.

**Proposición 10.** Si $P$ es una ruta generada por el algoritmo, entonces $P$ es la ruta más corta del origen al nodo final de $P$.

**Demostración:** Como la pareja $(P, p)$ mantiene la condición de HC (esto se mostró en la proposición 9). En particular si se considera la condición 4.6 a lo largo de la ruta $P = (i_1, i_2, ..., i_k)$ que va de 1 a $k$ se tiene que:

$$p_1 = a_{12} + p_2$$

$$p_2 = a_{23} + p_3$$

$$\vdots$$

$$p_{k-1} = a_{k-1,k} + p_k$$

sumando las igualdades anteriores, se tiene que:

$$p_1 + p_2 + ... + p_{k-1} = a_{12} + p_2 + a_{23} + p_3 + ... + a_{k-1,k} + p_k$$

reduciendo términos se llega a que:

$$p_1 - p_k = a_{12} + a_{23} + ... + a_{k-1,k}$$

lo cual implica que $P$ tiene una longitud de $p_1 - p_k$, y por la proposición 9 cada ruta que va de 1 a $k$ debe ser al menos igual a $p_1 - p_k$, por lo tanto la ruta generada por el algoritmo es la más corta. Q.E.D.

La siguiente proposición muestra la validez del algoritmo.

**Proposición 11.** Si existe al menos una ruta del origen al destino, el algoritmo termina con una ruta más corta. De otra manera el algoritmo nunca termina y $p_1 \to \infty$.

**Demostración:** Primero se supondrá que existe una ruta del nodo origen 1 al nodo destino $t$. Por la proposición 9 se tiene que $p_1 - p_t$ es una subestimación (finita) de la distancia más corta de 1 a $t$, $p_1$ es monótona y no decreciente, y $p_t$ es fijo a lo largo de todo el algoritmo, esto muestra que $p_1$ debe estar acotado, después se pedirá que $p_i$ esté acotado para toda $i$. Para tener que $p_i \to \infty$ el nodo $i$ debe ser el nodo final de $P$ un número infinito de veces, lo cual implica (por la proposición 9) que $p_1 - p_i$ debe ser igual a la distancia más corta de 1

a $i$ infinitamente, lo cual es una contradicción pues $p_1$ es acotado.

Se verá a continuación que el algoritmo termina. Se puede ver por inducción que para cada nodo $i$, $p_i$ es igual a su valor inicial o igual a la longitud de alguna ruta que empieza en $i$ más el precio inicial del nodo final de la ruta; al cual se le llamará la longitud modificada de la ruta. Cada ruta que empieza en $i$ se puede descomponer en una ruta simple y en un número finito de ciclos, cada uno tiene longitud positiva por hipótesis, así el número de longitudes modificadas distintas dentro de cualquier intervalo acotado es acotado. Se mostró que $p_i$ es acotado. Además, cada vez que $i$ se convierte en el nodo final por extensión de la ruta $P$, $p_i$ es estrictamente mayor sobre todas las veces anteriores que $i$ se hace el nodo final de $P$ lo que corresponde a una modificación de la longitud de la ruta estrictamente mayor. Se sigue que el número de veces que $i$ puede llegar a ser el nodo final de $P$ por extensión de la ruta es acotado. Ya que el número de contracciones de ruta entre dos extensiones de ruta consecutivas esta acotado por el número de nodos en la gráfica, el número de iteraciones en el algoritmo es acotado, lo cual implica que el algoritmo termina.

Si ahora se supone que no hay una ruta de 1 al destino. Entonces el algoritmo nunca terminará, así por el argumento anterior algunos nodos $i$ serán el nodo final de la ruta $P$ por extensión infinita, y $p_i \to \infty$. Al final de la iteración cuando esto pasa, $p_1 - p_i$ debe ser igual a la distancia más corta de 1 a $i$, lo cual implica que $p_1 \to \infty$. Q.E.D.

## 4.2 Algoritmo hacia atrás para el problema de ruta más corta

En el problema de ruta más corta se puede cambiar los papeles de los orígenes y destinos intercambiando las direcciones de todos los arcos. Por lo tanto es posible usar el algoritmo que mantiene la ruta $R$, la cual termina en el nodo destino y cada iteración poder cambiarla por una contracción o una extensión. A este algoritmo se le llama algoritmo hacia atrás y es equivalente al algoritmo hacia adelante.

De la condición 4.5 de holgura complementaria del algoritmo hacia adelante

se tiene que:

$$p_i \leq a_{ij} + p_j$$

$$-p_j \leq a_{ij} - p_i$$

remplazando $-p_i$ por $\bar{p}_i$ y $-p_j$ por $\bar{p}_j$ se tiene que la condicion HC para el problema con los arcos en dirección inversa es:

$$\bar{p}_j \leq a_{ij} + \bar{p}_i, \qquad \forall \quad (i,j) \in A,$$

$$\bar{p}_j = a_{ij} + \bar{p}_i, \qquad \text{para todas las parejas de nodos sucesivos } i \text{ y } j \text{ de } R$$

asi se mantiene una condición común para los algoritmos hacia adelante y hacia atrás.

Para ser consistentes con el algoritmo hacia adelante, se supone que cada nodo, excepto el origen, tiene al menos un arco que llega a él. Al inicio del algoritmo hacia atrás, $R$ es cualquier ruta que termina en el destino, y $p$ cualquier vector de precios que satisface la condición HC junto con $R$, por ejemplo,

$$R = (t), \qquad p_i = 0, \qquad \forall \quad i$$

**Algoritmo hacia atrás para resolver el problema de ruta más corta**

**Iteración común:**

Sea $j$ el nodo inicial de $R$. Si

$$p_j > \max_{(i,j)\in A} \{p_i - a_{ij}\}$$

ir al paso 1; si no ir al paso 2.

**Paso 1. (Contraer la ruta).** Calcular

$$p_j := \max_{(i,j)\in A} \{p_i - a_{ij}\}$$

y si $j \neq t$, contraer a $R$ (es decir, borrar el nodo inicial $j$ de $R$) si no, ir a la siguiente iteración.

**Paso 2. (Extender la ruta).** Extender a $R$ por el nodo $i_j$, (es decir, hacer $i_j$ el nodo inicial de $R$, precedido por $j$), donde

$$i_j = \arg \max_{(i,j)\in A} \{p_i - a_{ij}\}$$

Si $i_j$ es el origen 1, parar; $R$ es la ruta más corta deseada. De otra manera, ir a la siguiente iteración.

El algoritmo hacia atrás es igual que el algoritmo hacia adelante aplicado a un problema hacia atrás de ruta más corta, por lo tanto es válido y termina con una ruta más corta, si por lo menos existe una ruta de 1 a $t$.

**Ejemplo 6.**

Ilustración del algoritmo hacia atrás. Encontrar la ruta más corta del siguiente problema:

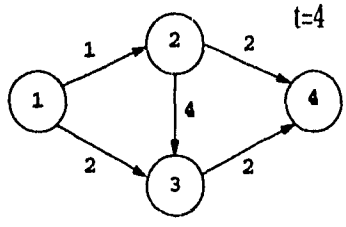



**Fig. 7.** *Problema de ruta más corta*

se comienza con:
$R = (4)$, el nodo inicial de la ruta es $j = 4$
$p = (0,0,0,0)$ como aún no se llega al origen y

$$p_4 = 0 > \max\{0-2, 0-2\} = -2$$

se contrae a 4 y se actualiza $p_4$

ahora se tiene:
$R = (4)$, el nodo inicial de la ruta es $j = 4$

$p = (0, 0, 0, -2)$ como aún no se llega al origen y

$$p_4 = -2 \not> \max\{0 - 2, 0 - 2\} = -2$$

se extiende la ruta al nodo 3, ya que,

$$3 = \arg\max\{0 - 2, 0 - 2\}$$

ahora se tiene:
$R = (3, 4)$, el nodo inicial de la ruta es $j = 3$
$p = (0, 0, 0, -2)$ como aún no se llega al origen, y

$$p_3 = 0 > \max\{0 - 4, 0 - 2\} = -2$$

se contrae a 4 y se actualiza $p_3$

$$p_3 = \max\{0 - 4, 0 - 2\} = -2$$

ahora se tiene:
$R = (4)$, el nodo inicial de la ruta es $j = 4$
$p = (0, 0, -2, -2)$ como aún no se llega al origen, y

$$p_4 = -2 \not> \max\{0 - 2, -2 - 2\} = -2$$

se extiende la ruta al nodo 2, ya que,

$$2 = \arg\max\{0 - 2, -2 - 2\}$$

ahora se tiene:
$R = (2, 4)$, el nodo inicial de la ruta es $j = 2$
$p = (0, 0, -2, -2)$ como aún no se llega al origen y

$$p_2 = 0 > \max\{0 - 1\} = -1$$

se contrae la ruta al nodo 4 y se actualiza $p_2$

$$p_2 = \max\{0 - 1\} = -1$$

ahora se tiene:
$R = (4)$, el nodo inicial de la ruta es $j = 4$
$p = (0, -1, -2, -2)$ como aún no se llega al origen y

$$p_4 = -2 > \max\{-1 - 2, -2 - 2\} = -3$$

se contrae a 4 y se actualiza $p_4$

ahora se tiene:
$R = (4)$, el nodo inicial de la ruta es $j = 4$
$p = (0, -1, -2, -3)$ como aún no se llega al origen y

$$p_4 = -3 \not> \max\{-3, -4\} = -3$$

se extiende la ruta al nodo 2, ya que,

$$2 = \arg\max\{-3, -4\}$$

ahora se tiene:
$R = (2, 4)$, el nodo inicial de la ruta es $j = 2$
$p = (0, -1, -2, -3)$ como aún no se llega al origen y

$$p_2 = -1 \not> \max\{0 - 1\} = -1$$

se extiende la ruta al nodo 1, ya que,

$$1 = \arg\max\{0 - 1\}$$

ahora se tiene:
$R = (1, 2, 4)$, el nodo inicial de la ruta es $j = 1$
$p = (0, -1, -2, -3)$

FIN, se ha encontrado la ruta más corta $P = (1, 2, 4)$ de 1 a 4, con longitud igual a $p_1 - p_4 = 3$

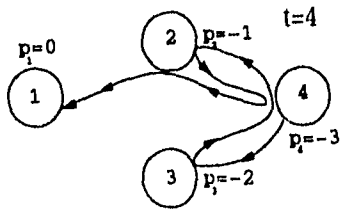



**Fig. 8.** *Trayectoria para encontrar la ruta más corta*

## 4.3 Combinación de los algoritmos hacia adelante y hacia atrás

Ahora se considerará combinar el algoritmo hacia adelante y hacia atrás en uno solo. Para esto se requiere que inicialmente se tenga un vector de precios $p$ y dos rutas $P$ y $R$, las cuales satisfagan HC junto con $p$, donde $P$ inicia en el nodo origen y $R$ en el destino. Las rutas $P$ y $R$ son extendidas y contraídas de acuerdo a las reglas del algoritmo hacia adelante y hacia atrás respectivamente y el algoritmo combinado termina cuando $P$ y $R$ tienen un nodo común. Como $P$ y $R$ satisfacen HC junto con $p$ durante todo el algoritmo, cuando $P$ y $R$ se encuentran, se dice que apartir del nodo $i$ se compone la ruta que consiste de la porción de $P$ que es de 1 a $i$ y de la porción de $R$ que es de $i$ a $t$ y esta será la ruta más corta de 1 a $t$.

### Combinación de los algoritmos

**Paso 1. (Ejecutar el algoritmo hacia adelante).** Ejecutar varias iteraciones del algoritmo hacia adelante (sujeto a la condición de terminación), hasta que al menos $p_1$ incremente su precio original. Ir al paso 2.

**Paso 2. (Ejecutar el algoritmo hacia atrás).** Ejecutar varias iteraciones del algoritmo hacia atrás (sujeta a la condición de terminación) hasta que al menos $p_t$ decremente su precio original. Ir al paso 1.

### Ejemplo 7.

Ilustración del algoritmo combinado. Encontrar la ruta más corta del siguiente problema:

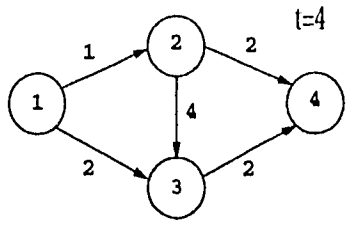



**Fig. 9.** *Problema de ruta más corta*

Método hacia adelante.

se comienza con:
$P = (1)$, el nodo final de la ruta es $i = 1$
$p = (0,0,0,0)$ como aún no se llega al destino y

$$p_1 = 0 < \min\{1+0, 2+0\} = 1$$

se contrae a 1 y se actualiza $p_1$

ahora se tiene:
$P = (1)$, el nodo final de la ruta $P$ es $i = 1$
$p = (1,0,0,0)$ como aún no se llega al destino y

$$p_1 = 1 \not< \min\{1+0, 2+0\} = 1$$

se extiende la ruta $P$ al nodo 2, ya que,

$$2 = \arg\min\{1+0, 2+0\}$$

ahora se tiene:
$P = (1,2)$, el nodo final de la ruta $P$ es $i = 2$
$p = (1,0,0,0)$

Método hacia atrás

se comienza con:
$R = (4)$, el nodo inicial de la ruta $R$ es $j = 4$
$p = (1,0,0,0)$ como aún no se llega al origen y

$$p_4 = 0 > \max\{0-2, 0-2\} = -2$$

se contrae a $t$ y se actualiza $p_t$

ahora se tiene:
$R = (4)$, el nodo inicial de la ruta $R$ es $j = 4$
$p = (1, 0, 0, -2)$ como aún no se llega al origen y

$$p_4 = -2 \not> \max\{0 - 2, 0 - 2\} = -2$$

se extiende la ruta $R$ al nodo 2, ya que,

$$2 = \arg\max\{0 - 2, 0 - 2\}$$

ahora se tiene:
$R = (2, 4)$, el nodo inicial de la ruta $R$ es $j = 2$
$p = (1, 0, 0, -2)$ como el nodo final de la ruta $P$ es 2
igual al nodo inicial de la ruta $R$.

FIN, se ha encontrado la ruta más corta uniendo $P = (1)$ y $R = (2, 4)$
de 1 a 4 con longitud igual a $p_1 - p_4 = 3$

Para justificar el algoritmo combinado, nótese que $p_1$ puede únicamente incrementar su valor y $p_t$ sólo puede decrementar su valor durante en el transcurso del algoritmo, y que la diferencia de $p_1 - p_t$ no puede ser mayor a la distancia más corta de 1 a $t$. Supóngase que las longitudes de los arcos y los precios inicilales son enteros, y que al menos una ruta de 1 a $t$ existe. Entonces $p_1$ y $p_t$ sólo pueden cambiar por cantidades enteras, y además $p_1 - p_t$ es acotado. Aquí $p_1$ y $p_t$ pueden cambiar sólo un número finito de veces, ya que los pasos 1 y 2 deben contener únicamente un número finito de iteraciones del algoritmo hacia adelante y del algoritmo hacia atrás, respectivamente. Por lo que se ve que el algoritmo debe terminar. Nótese que este argumento requiere que $p_1$ se incremente por lo menos una vez en el paso 1 y que $p_t$ se decremente por lo menos una vez en el paso 2. Sin este requerimiento se pueden construir ejemplos que muestren que el algoritmo combinado puede nunca terminar.

En la práctica, se ve que el algoritmo combinado es mucho más rápido que si se trabajara únicamente con el algoritmo hacia adelante o hacia atrás.

## 4.4 Relación del algoritmo de subasta y el ascenso dual

Se explicará como el algoritmo hacia adelante con un sólo origen y un sólo destino, se puede ver como una aplicación del algoritmo ingenuo de subasta para el problema de asignación.

El algoritmo ingenuo de subasta fué descrito para maximizar el problema de asignación, donde se quiere maximizar el beneficio de relacionar $n$-personas y $n$-objetos uno a uno. Conviene reformular el problema y el algoritmo en términos de minimización guardando los signos de los coeficientes de costos y los precios y cambiando únicamente maximización por minimización. En particular, se supone que hay un costo $c_{ij}$ por asignar la persona $i$ con el objeto $j$ y se quiere asignar a las personas con los objetos de tal manera que se minimice el costo. Matemáticamente se quiere encontrar una asignación factible que minimice el costo total $\sum_{i=1}^{n} c_{ij}$, donde una asignación factible significa un conjunto de parejas persona-objeto $(1, j_1), ..., (n, j_n)$ tal que los objetos $j_1, ..., j_n$ sean todos diferentes y $(i, j_i) \in A$ para toda $i$.

El algoritmo de subasta ingenuo procede en iteraciones y genera una secuencia de vectores de precios $p$ y asignaciones (parciales). Al principio de cada iteración la condición de holgura complementaria

$$c_{ij_i} + p_{j_i} = \min_{(i,j)\in A} \{c_{ij} + p_j\} \tag{4.10}$$

se satisface para todas las parejas $(i, j_i)$ de la asignación. El vector de precios y la asignación inicial requiere satisfacer la condición de la ec. 4.10. Si todas las personas son asignadas, entonces el algoritmo termina. Si alguna persona no es asignada se dice que $i$ encuentra un objeto $j_i$, el cual es el mejor en el sentido de que

$$j_i = \arg \min_{(i,j)\in A} \{c_{ij} + p_j\}$$

y entonces:

(a) Se obtiene una asignación para el mejor objeto $j_i$; la persona que fué asignada a $j_i$ al principio de la iteración, si existe, se desasigna.

(b) Se fija el precio de $j_i$ como el nivel en el cual le es indiferente entre $j_i$ y el segundo mejor objeto, es decir:

$$p_{j_i} + \omega_i - v_i$$

donde $v_i$ es el costo de adquirir el mejor objeto (incluyendo el pago del precio correspondiente),

$$v_i = \min_{(i,j)\in A} \{c_{ij} + p_j\}$$

y $\omega_i$ es el costo de adquirir el segundo mejor objeto

$$\omega_i = \min_{(i,j)\in A,\ j\neq j_i} \{c_{ij} + p_j\}$$

Este proceso se repite hasta que todas las personas son asignadas a un objeto.

La diferencia entre el algoritmo ingenuo de subasta y el algoritmo de subasta es la manera de escoger el incremento de los precios. En el algoritmo de subasta el precio $p_{j_i}$ se incrementa por $\omega_i - v_i + \epsilon$, donde $\epsilon$ es una constante positiva. El algoritmo ingenuo de subasta es el mismo que el algorirmo de subasta excepto por que $\epsilon = 0$. El algoritmo de subasta garantiza que termina si al menos existe una asignación factible, el algoritmo ingenuo de subasta puede llegar a tener un ciclo infinito. Sin embargo, si el algoritmo ingenuo de subasta termina, la asignación obtenida al final es óptima.

## 4.5 Formulación del problema de ruta más corta a un problema de asignación

Dado un problema de ruta más corta con el nodo 1 como origen y el nodo $t$ como destino, se puede llegar a un problema de asignación.

Sea $2,...,N$ los nodos "objetos", para cada nodo $i \neq t$ se introduce un nodo "persona" $i'$. Para cada arco $(i,j)$ del problema de ruta más corta con $i \neq t$ y $j \neq 1$, se introduce un arco $(i',j)$ con costo $a_{ij}$ en el problema de asignación. También se introduce un arco $(i',i)$ de costo cero para cada $i \neq 1,t$. El

ejemplo 7 ilustra el problema de asignación y muestra como dada una asignación parcial que asigna al objeto $i$ con la persona $i'$ para $i \neq 1, t$ la ruta de 1 a $t$ se puede asociar con la ruta creciente que comienza en $1'$ y que termina en $t$.

Aplicando el algoritmo de subasta ingenuo iniciando con un vector de precios $p = (0, 0, ..., 0)$ y una asignación parcial que satisfaga la condición HC que es,

$$p_i \leq a_{ij} + p_j, \qquad \forall \quad (i,j) \in A \tag{4.11}$$

se obtiene la ruta más corta.

Se pone una regla adicional para romper empates en el algoritmo ingenuo de subasta. Si en alguna iteración se involucra a una persona no asignada $i'$ y su mejor arco es $(i', i)$ el cual es igualmente deseable que algún otro arco $(i', j_i)$, entonces el segundo arco es preferido, esto es, $(i', j_i)$ es agregado a la asignación. Además se introduce otra modificación al algoritmo de subasta ingenuo para tener en cuenta de manera especial, una contracción al origen, en el algoritmo de ruta más corta. La modificación se tiene cuando se calcula la oferta de $1'$ que ahora consistirá en encontrar un objeto $j_i$ que consiga el mínimo en:

$$\min_{(1,j)\in A} \{a_{1j} + p_j\}$$

y se asignará $j_1$ a $1'$ y se desasignará la persona asignada a $j_1$ (en el caso de que $j_1 \neq t$), pero no se cambiará el precio de $p_{j_1}$

Se puede mostrar ahora que el algoritmo de subasta ingenuo con las modificaciones anteriores es equivalente al algoritmo de rutas más cortas. En particular se verificará por inducción.

(a) La condición HC 4.11 se conserva por el algoritmo ingenuo de subasta

(b) Cada asignación generada por el algoritmo ingenuo de subasta consiste de una secuencia de la forma

$$(1', i_1), (i'_1, i_2), ..., (i'_{k-1}, i_k),$$

junto con los arcos adicionales

$$(i', i) \text{ para } i \neq i_1, ..., i_k, t;$$

esta secuencia corresponde a la ruta $P = (1, i_1, ..., i_k)$ generada por el algoritmo de ruta más corta. La (única) persona no asignada en el algoritmo ingenuo de subasta $i'_k$ corresponde al nodo final de la ruta. Cuando $i_k \neq t$ y da por resultado una asignación factible, el algoritmo ingenuo de subasta termina, conforme al criterio de terminación del algoritmo de ruta de más corta.

(c) En una iteración correspondiente a una persona no asignada $i'$ con $i \neq 1$, el arco $(i', i)$ siempre es el mejor arco, esto es una consecuencia de la condición de holgura complementaria 4.11. Además hay 3 posibilidades:

(1) $(i', i)$ es el único mejor arco, en este caso $(i', i)$ es agregado a la asignación, y el precio $p_i$ se incrementa en:

$$\min_{(i,j)\in A} \{c_{ij} + p_j\} - p_i$$

esto corresponde a una contracción de la ruta actual por el nodo final $i$

(2) Hay un arco $(i', j_i)$ con $j_i \neq t$ el cual se prefiere de igual manera que el arco $(i', i)$, esto es

$$p_i = a_{ij_i} + p_{j_i},$$

en este caso, como se vió en la regla que rompe el empate, $(i', j_i)$ se agrega a la asignación y el precio $p_{j_i}$ se mantiene igual. Además, el objeto $j_i$ debe ser asignado a $j'_i$ al principio de la iteración, así agregar $(i', j_i)$ a la asignación (y borrar $(j'_i, j_i)$) corresponde a extender la ruta actual por el nodo $j_i$.

(3) El arco $(i', t)$ se prefiere de igual manera que el arco $(i', i)$ en este caso el objeto $t$ no asignado es asignado a $i'$, así termina el algoritmo ingenuo de subasta, esto corresponde a que el destino $t$ llega al nodo final de la ruta actual, y así también se termina el algoritmo de ruta más corta.

De esta manera se tiene que el algoritmo de ruta más corta se puede ver como una caso particular del algoritmo ingenuo de subasta. Finalmente nótese que la versión combinada del algoritmo de ruta más corta es equivalente a la versión del algoritmo ingenuo combinado de subasta, con las modificaciones descritas.

**Ejemplo 8.**

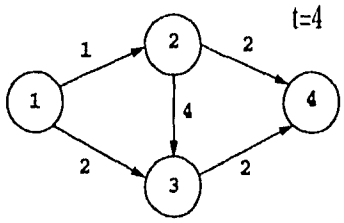



**Fig. 10.** *Problema de ruta más corta*

Resolver el siguiente problema de ruta más corta, por medio del algoritmo de subasta ingenuo (el origen es 1, el destino es $t = 4$). Las longitudes de los arcos y los costos de asignación son mostrados al lado de los arcos. Para cada nodo $i \neq 1$ se introduce un nodo objeto $i$, y para cada nodo $i \neq t$ se introduce una persona de nodo $i'$. Para cada arco $(i, j)$ del problema de ruta más corta con $i \neq t$ y $j \neq 1$ se introduce un arco $(i', j)$ con costo $a_{ij}$ en el problema de asignación. También se introduce el arco $(i', i)$ de costo cero para cada $i \neq 1, t$. Una ruta más corta de 1 a $t$, puede ser asociada con una ruta creciente óptima que empieza en $1'$ y termina en $t = 4$.

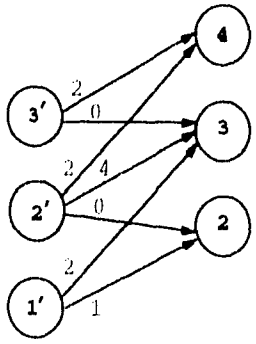



**Fig. 11.** *Problema de asignación*

Se representa el problema de asignación en forma matricial, (Nótese que, para cada arco que no exista en la gráfica se le pone un valor de $\infty$ para asegurar

que no será considerado en la asignación).

$$\begin{array}{cc} & \text{objetos} \\ & 2 \quad 3 \quad 4 \\ \text{personas} \begin{array}{c} 1' \\ 2' \\ 3' \end{array} & \begin{pmatrix} 1 & 2 & \infty \\ 0 & 4 & 2 \\ \infty & 0 & 2 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (0,0,0)$$

Si se empieza con la siguiente asignación parcial $S = \{(2,2'),(3,3')\}$ y con el vector de precios igual a cero los cuales satisfacen la condición HC. Esto corresponde a una contracción al nodo 1 y se tiene que $P = (1)$. La representación matricial se muestra acontinuación.

$$\begin{array}{cc} & \text{objetos} \\ & 2 \quad 3 \quad 4 \\ \text{personas} \begin{array}{c} 1' \\ 2' \\ 3' \end{array} & \begin{pmatrix} 1 & 2 & \infty \\ 0^* & 4 & 2 \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (0,0,0)$$

La persona $1'$ no asignada busca su mejor objeto que es 2, el cual se le asigna sin modificar el precio del objeto. Lo cual corresponde a una extensión al nodo 2 y se tiene que $P = (1,2)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{cc} & \text{objetos} \\ & 2 \quad 3 \quad 4 \\ \text{personas} \begin{array}{c} 1' \\ 2' \\ 3' \end{array} & \begin{pmatrix} 1^* & 2 & \infty \\ 0 & 4 & 2 \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \quad \begin{array}{l} \nu_1' = 1 \\ \omega_1' = 2 \end{array} \quad \text{Vector de precios} \quad (0,0,0)$$

La persona $2'$ no asignada, hace una oferta $\gamma_2' = 2$ por el objeto 2 y al hacer esta oferta la persona $1'$ que tenía el objeto 2 ahora queda desasignada. Lo cual corresponde a una contracción al nodo 1 y se tiene que $P = (1)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{cc} & \text{objetos} \\ & 2 \quad 3 \quad 4 \\ \text{personas} \begin{array}{c} 1' \\ 2' \\ 3' \end{array} & \begin{pmatrix} 1 & 2 & \infty \\ 0^* & 4 & 2 \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \quad \begin{array}{l} \nu_2' = 0 \\ \omega_2' = 2 \\ \gamma_2' = 2 \end{array} \quad \text{Vector de precios} \quad (2,0,0)$$

La persona $1'$ no asignada, busca su mejor objeto que es 3, pero no modifica su precio. Esto corresponde a una extensión al nodo 3 y se tiene que $P = (1, 3)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{ccc} \text{objetos} & & \\ 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 2^* & \infty \\ 0^* & 4 & 2 \\ \infty & 0 & 2 \end{pmatrix} \end{array} \begin{array}{l} 1' \\ 2' \\ 3' \end{array} \quad \begin{array}{rcl} \nu_1' & = & 2 \\ \omega_1' & = & 3 \end{array} \qquad \text{Vector de precios} \quad (2, 0, 0)$$

La persona $3'$ no asignada, hace una oferta $\gamma_3' = 2$ por el objeto 3 y al hacer esta oferta la persona $1'$ que tenía el objeto 3 ahora queda desasignada. Lo cual corresponde a una contracción al nodo 1 y se tiene que $P = (1)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{ccc} \text{objetos} & & \\ 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 2 & \infty \\ 0^* & 4 & 2 \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \begin{array}{l} 1' \\ 2' \\ 3' \end{array} \quad \begin{array}{rcl} \nu_3' & = & 0 \\ \omega_3' & = & 2 \\ \gamma_3' & = & 2 \end{array} \qquad \text{Vector de precios} \quad (2, 2, 0)$$

La persona $1'$ no asignada, busca su mejor objeto que es 2, pero no modifica su precio. Esto corresponde a una extensión al nodo 2 y se tiene que $P = (1, 2)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{ccc} \text{objetos} & & \\ 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1^* & 2 & \infty \\ 0 & 4 & 2 \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \begin{array}{l} 1' \\ 2' \\ 3' \end{array} \quad \begin{array}{rcl} \nu_1' & = & 3 \\ \omega_1' & = & 4 \end{array} \qquad \text{Vector de precios} \quad (2, 2, 0)$$

La persona $2'$ no asignada, hace una oferta $\gamma_2' = 0$, por el objeto 4. FIN, todas las personas están asignadas a objetos diferentes. Lo cual corresponde a una extensión al nodo 4 = nodo final y se tiene que $P = (1, 2, 4)$. La representación matricial queda de la siguiente manera:

$$\begin{array}{c} \\ \\ \text{personas} \end{array} \begin{array}{c} \begin{array}{ccc} \text{objetos} & & \\ 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1^* & 2 & \infty \\ 0 & 4 & 2^* \\ \infty & 0^* & 2 \end{pmatrix} \end{array} \begin{array}{l} 1' \\ 2' \\ 3' \end{array} \quad \begin{array}{rcl} \nu_2' & = & 2 \\ \omega_2' & = & 2 \\ \gamma_2' & = & 0 \end{array} \qquad \text{Vector de precios} \quad (2, 2, 0)$$

La asignación total

$$S = \{(1',2),(2',4),(3',3)\}$$

corresponde a la ruta más corta $P = (1,2,4)$ que va de 1 a 4 con longitud igual a $\sum_{(i,j)\in S} a_{ij} = 3$

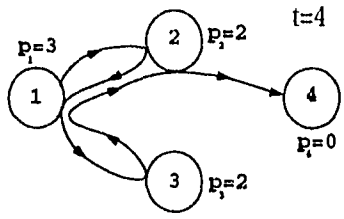



**Fig. 12.** *Trayectoria para encontrar la ruta más corta*

# Capítulo 5

# Problema de transporte

Ahora se considerará la extensión del algoritmo de subasta al problema de transporte. Se tiene un gráfica bipartita con $m$ orígenes y $n$ destinos. El problema es el mismo que el problema de asignación excepto que la ofertas y las demandas no necesitan ser 1 o $-1$. El planteamiento de programación lineal para el problema de transporte, es el siguiente:

$$\text{Maximizar} \quad \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}x_{ij}$$

sujeto a

$$\sum_{j=1}^{n} x_{ij} = \alpha_i, \qquad \forall \quad i = 1,...,m \tag{5.1}$$

$$\sum_{i=1}^{m} x_{ij} = \beta_j, \qquad \forall \quad j = 1,...,n$$

$$x_{ij} \geq 0, \qquad \forall \quad \begin{matrix} i = 1,...,m \\ j = 1,...,n \end{matrix}$$

donde, $\alpha_i$ y $\beta_j$ son enteros positivos. Para que el problema sea factible se debe satisfacer:

$$\sum_{i=1}^{m} \alpha_i = \sum_{j=1}^{n} \beta_j$$

**Proposición 12:** Una condición necesaria y suficiente para que la estructura de transporte 5.1 tenga solución, es que la oferta total sea igual a la demanda total, es decir,

$$\sum_{i=1}^{m} \alpha_i = \sum_{j=1}^{n} \beta_j$$

**Demostración:** De la formulación 5.1 se tiene:

$$\sum_{j=1}^{n} x_{ij} = \alpha_i, \qquad i = 1, ..., m$$

Si se suma sobre todos los orígenes, no se afecta la igualdad, es decir

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = \sum_{i=1}^{m} \alpha_i$$

Por otro lado de la formulación 5.1 se tiene:

$$\sum_{i=1}^{m} x_{ij} = \beta_j, \qquad j = 1, ..., n$$

Sumando sobre todos los destinos,

$$\sum_{j=1}^{n} \sum_{i=1}^{m} x_{ij} = \sum_{j=1}^{n} \beta_j$$

Lo cual establece la prueba, Q.E.D.

Si la oferta total excede a la demanda total, entonces se puede introducir un destino ficticio o artificial con demanda $b_{n+1} = \sum_i \alpha_i - \sum_j \beta_j$ y $a_{i,n+1} = 0$ para $i = 1, ..., m$. Cuando un problema de transporte real no es simétrico, añadiendo ya sea orígenes o destinos artificiales, se hace simétrico para que el problema tenga solución.

## 5.1 Extensión del problema de asignación al problema de transporte

Se puede convertir este problema a un problema de asignación reemplazando cada nodo origen (o destino) por una colección de personas (u objetos) "duplicados". En particular, un nodo origen con oferta $\alpha_i$ es reemplazado por $\alpha_i$

personas, y un nodo destino con demanda $\beta_j$ es reemplazado por $\beta_j$ objetos. Además, para cada arco $(i, j)$ se debe crear un arco de beneficio $a_{ij}$ que una cada persona $i$ con cada objeto $j$ correspondiente.

Ilustración de la conversión de un problema de transporte (Fig. 13) a un problema de asignación (Fig. 14).

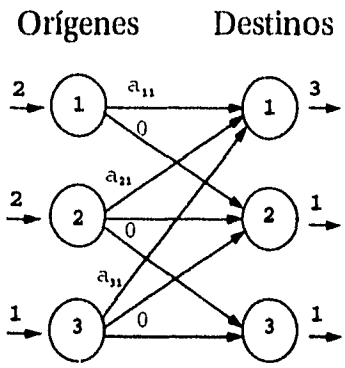



**Fig. 13.** *Problema de Transporte*

Cada origen $i$ se transforma en $\alpha_i$ personas (por ejemplo el origen 1 de la figura 13 es transformado en las personas 1 y 1′ de la figura 14) y cada destino $j$ se transforma en $\beta_j$ objetos (por ejemplo, el destino 1 de la figura 13 se transforma en los objetos 1, 1′ y 1″ de la figura 14). Si $(i, j)$ es un arco de beneficio $a_{ij}$ en el problema de transporte, hay un arco de beneficio $a_{ij}$ asignado que une cada persona $i$ correspondiente al origen $i$ con cada objeto $j$ correspondiente al destino $j$.

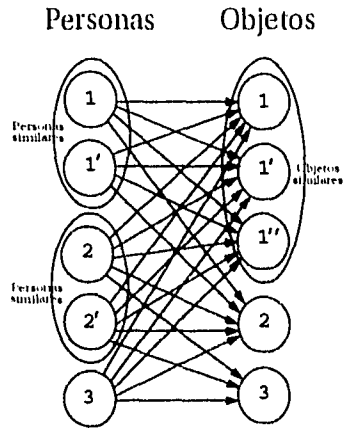



**Fig. 14.** *Problema de asignación*

Es posible resolver el problema equivalente de asignación por medio del algoritmo de subasta, pero hay dos inconvenientes:

1. La dimensión del problema crece grandemente (el números de personas y objetos crece hasta $\sum_{i=1}^{m} \alpha_i$).

2. La estructura del problema de asignación equivalente alarga la guerra de precios y es inevitable, debido a que se crean objetos y personas duplicadas por la transformación.

Se pueden evitar estas dificultades si se modifica el algoritmo de subasta aprovechando la estructura especial de la transformación del problema de transporte al problema de asignación y el tener objetos y personas duplicadas. En particular, dado un problema de asignación, se dice que dos objetos $j$ y $j'$ son semejantes (y se escribe $j \sim j'$), si se pueden asignar con las mismas personas y con los mismos valores, de igual manera, se dice que dos personas $i$ e $i'$ son semejantes (y se escribe $i \sim i'$), si se pueden asignar con los mismos objetos y con los mismos valores.

A la colección de todos los objetos semejantes del objeto $j$ se le llama *la clase de semejanza de* $j$ y se denota por $M(j)$, y a la colección de todas las personas semejantes a la persona $i$ se le llama *la clase de semejanza de* $i$ y se denota por $M(i)$.

## 5.2 Algoritmo de subasta con objetos semejantes

Se puede implementar fácilmente el *algoritmo de subasta con objetos semejantes*. Para cada objeto $j$, se mantiene un umbral de contención $\hat{p}_j$. Si $j$ no es asignado, el umbral de contención $\hat{p}_j$ es igual al precio inicial $p_j$; cada vez que $j$ sea asignado a una nueva persona $i$, el umbral de contención $\hat{p}_j$ se actualiza al mínimo nivel de la clase de semejanza. Los precios son determinados por los umbrales de contención. En particular, el precio de un objeto $j$ es el mínimo umbral de contención sobre los objetos de la clase de semejanza $M(j)$ de $j$.

$$p_j = \min_{k \in M(j)} \hat{p}_k \tag{5.2}$$

por lo tanto, todos los objetos en una clase de semejanza tienen el mismo precio pero posiblemente diferente umbral de contención.

El algoritmo de subasta con objetos semejantes es el mismo que el algoritmo de subasta excepto por una diferencia. En la fase de oferta, una persona no asignada $i$ encuentra su mejor objeto $j_i$ correspondiente al mejor valor del umbral de contención

$$j_i = \arg\max_j \{a_{ij} - \hat{p}_j\}, \qquad \nu_i = \max_j \{a_{ij} - \hat{p}_j\} \tag{5.3}$$

Entonces $i$ es asignado a $j_i$ y se actualiza el umbral de contención $\hat{p}_{j_i}$ como $p_{j_i}$ más un incremento $\nu_i - \omega_i + \epsilon$, lo cual está basado en el segundo mejor objeto de una diferente clase de semejanza, $\omega_i$ se define ahora como:

$$\omega_i = \max_{j, j \notin M(j_i)} \{a_{ij} - \hat{p}_j\} \tag{5.4}$$

(en lugar de $\omega_i = \max_{j \neq j_i} \{a_{ij} - p_j\}$). Cuando el umbral de contención de todos objetos de una clase de semejanza se incrementa, el precio de la clase también se incrementa por el umbral mínimo de contención según 5.2. Nótese que por la relación $p_j = \min_{k \in M(j)} \hat{p}_k$ entre los precios y el umbral de contención, $\hat{p}_j$ se puede reemplazar por $p_j$ en 5.3. Así, una persona no asignada ofrece por el mejor objeto de la mejor clase de semejanza, pero el incremento ahora esta basado en los umbrales de contención y puede ser significativamente más grande que el incremento de la oferta basada en los precios.

Para entender cómo se puede asignar con clases de semejanza, se considera el ejemplo 9. En este ejemplo la persona $i$ ($i = 1, 2, 3$) continuará ofreciendo por los objetos semejantes 1 y $1'$ hasta que los precios $p_1$ y $p_{1'}$ sean por lo menos igual a $a_{i1}$, y el objeto 2 no sea menos atractivo que los objetos 1 y $1'$ para la persona $i$. El umbral de contención de un objeto para alguna clase de semejanza puede ser más grande que el precio del objeto (por las reglas del algoritmo de subasta, se restringe a estar a $\epsilon$ de los precios de cualquier otro objeto de la misma clase). Supóngase ahora que cuando una persona ofrece por un objeto de una clase de semejanza no solo actualiza su precio, sino también actualiza el umbral de contención correspondiente. Supóngase además que los umbrales de contención de todos los objetos de una clase de semejanza incrementan el precio de estos objetos (esencialmente común). Entonces se ve que, sin violar $\epsilon$-HC, se puede incrementar simultáneamente los precios de todos los objetos de la clase al umbral mínimo de contención, de este modo se resuelve la correspondiente guerra de precios.

**Ejemplo 9.**

Resolver el siguiente problema transporte por medio del algoritmo de subasta con objetos semejantes.

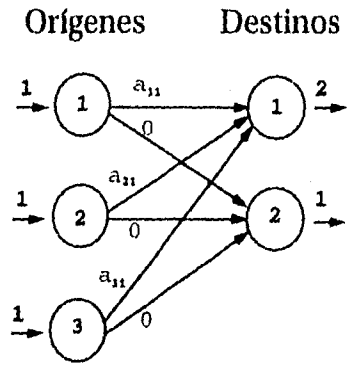



**Fig. 15.** *Problema de Transporte*

En la conversión del problema de transporte al problema de asignación, el destino 1 se convierte en los dos objetos semejantes 1 y $1'$.

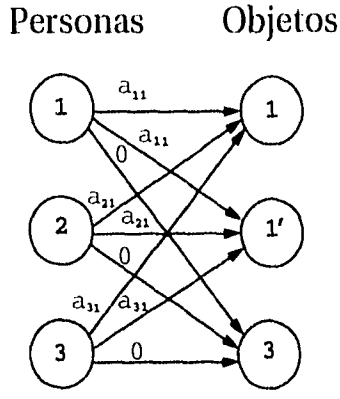



**Fig. 16.** *Problema de Asignación*

Sea $a_{11}$, $a_{21}$ y $a_{31}$ enteros mayores que cero (si $a_{11} = a_{21} = a_{31} = c$, se tendría el mismo problema que en los ejemplos 2 y 3).

La representación matricial del problema de asignación es como sigue:

$$\begin{array}{cc} & \begin{array}{ccc} \text{objetos} & & \\ 1 & 1' & 2 \end{array} \\ \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \text{personas} & \begin{pmatrix} a_{11} & a_{11} & 0 \\ a_{21} & a_{21} & 0 \\ a_{31} & a_{31} & 0 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (0,0,0)$$

La persona 1 ofrece el umbral de contención $\hat{p}_1 = a_{11} + \epsilon$ por el objeto 1 y la persona 2 ofrece el umbral de contención $\hat{p}_{1'} = a_{21} + \epsilon$ por el objeto $1'$, así el precio de la clase de semejanza $M(1)$ se incrementa al $\min\{a_{11}, a_{21}\} + \epsilon$;

$$\begin{array}{cc} & \begin{array}{ccc} \text{objetos} & & \\ 1 & 1' & 2 \end{array} \\ \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \text{personas} & \begin{pmatrix} a_{11}^* & a_{11} & 0 \\ a_{21} & a_{21}^* & 0 \\ a_{31} & a_{31} & 0 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (p_1, p_{1'}, 0)$$

La persona 3 que no fué asignada porque su objeto preferido 1 o $1'$ ya fué asignado a la persona 1 o 2, hace una oferta: si $a_{31} \leq \min\{a_{11}, a_{21}\} + \epsilon$, la oferta de la persona 3 será por el objeto 2 y la subasta termina con la siguiente

asignación.

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 1' \quad 2 \\ \text{personas} \quad \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} a_{11}^* & a_{11} & 0 \\ a_{21} & a_{21}^* & 0 \\ a_{31} & a_{31} & 0^* \end{pmatrix} \end{array}$$

De otra manera la persona 3 ofrecerá por el objeto 1 o $1'$ dependiendo de cual tenga el umbral más pequeño de contención, supóngase que el objeto 1 tiene el umbral más pequeño, entonces la asignación queda de la siguente manera:

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 1' \quad 2 \\ \text{personas} \quad \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} a_{11} & a_{11} & 0 \\ a_{21} & a_{21}^* & 0 \\ a_{31}^* & a_{31} & 0 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (p_1, p_{1'}, 0)$$

Y la persona no asignada; en este caso la persona 1, ofrece por el objeto 2 y la subasta termina.

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 1' \quad 2 \\ \text{personas} \quad \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} a_{11} & a_{11} & 0^* \\ a_{21} & a_{21}^* & 0 \\ a_{31}^* & a_{31} & 0 \end{pmatrix} \end{array}$$

De ésta manera se puede ver que el algoritmo de subasta con objetos semejantes resuelve rápidamente la guerra de precios que ocasionan los objetos duplicados correspondientes a los destinos del problema de transporte.

En cuanto a la validez del algoritmo, se puede demostrar que los precios de los objetos satisfacen $\epsilon$-HC y es escencialmente repetir la prueba de la proposición 4, también se puede ver que el algoritmo termina cuando el problema es factible. Sin embargo, se puede demostrar que para enteros $a_{ij}$, la asignación final será óptima si $\epsilon < 1/m$, donde $m$ es el número de clases de semejanza del objeto, aunque es suficiente que $\epsilon < 1/n$ como en el caso del algoritmo de subasta.

## 5.3 Algoritmo de subasta con personas semejantes

Se puede mejorar la ejecución del algoritmo de subasta teniendo en cuenta la presencia de personas semejantes. La idea es someter una oferta común por todas las personas que pertenecen a la misma clase de semejanza si por lo menos una persona en la clase no está asignada. Las personas que pertenecen a la misma clase de semejanza no compiten entre ellas por los mismos objetos, y las ofertas que hacen son las más altas, así, las personas de la misma clase de semejanza cooperan para evitar la guerra de precios y resuelven el problema en una iteración. Se ilustra esta idea y el correspondiente algoritmo de subasta en el ejemplo 10.

**Ejemplo 10.**

Resolver el siguiente problema de transporte por medio del algoritmo de subasta con personas semejantes.

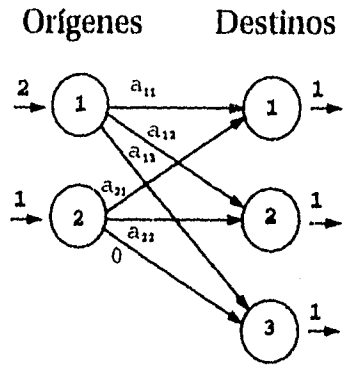



**Fig. 17.** *Problema de Transporte*

De la conversión del problema de transporte al problema de asignación el origen 1 se convierte en las personas semejantes 1 y 1'. En éste algoritmo, todas las personas de una clase de semejanza con $k$ personas, someten ofertas por los $k$ mejores objetos. Los niveles de las ofertas son tales que una vez ofrecidos son aceptados.

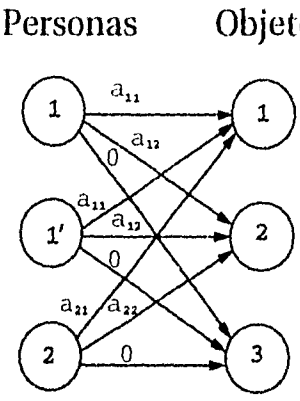



**Fig. 18.** *Problema de asignación*

La representación matricial del problema de transporte es como sigue:

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (0,0,0)$$

Se considera primero el caso donde la asignación inicial es vacía, todos los precios iniciales son 0, y $a_{11} = a_{12} = a_{21} = a_{22} = c > 0$. Entonces el algoritmo regular de subasta operará como en el ejemplo 3, (involucrando a una larga guerra de precios, si $\epsilon$ es relativamente menor que $c$).

En la primera iteración del algoritmo de subasta con personas semejantes, las personas 1 y 1′ someterán una oferta común de $c+\epsilon$ por los dos objetos mejores 1 y 2.

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c^* & 0 \\ c & c & 0 \end{pmatrix} \end{array} \qquad \text{Vector de precios} \quad (c+\epsilon, c+\epsilon, 0)$$

Entonces la persona 2 que no está asignada someterá una oferta por el objeto

3 y el algoritmo termina con la siguiente asignación.

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} c^* & c & 0 \\ c & c^* & 0 \\ c & c & 0^* \end{pmatrix} \end{array}$$

Si en lugar de eso, se tiene que $a_{11} > a_{12} > 0$, en la iteración inicial las personas 1 y $1'$ someterán una oferta de $\hat{p}_1 = a_{11} + \epsilon$ por el objeto 1 y una oferta de $\hat{p}_2 = a_{12} + \epsilon$ por el objeto 2,

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} a_{11}^* & a_{12} & 0 \\ a_{11} & a_{12}^* & 0 \\ a_{21} & a_{22} & 0 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (\hat{p}_1, \hat{p}_2, 0)$$

Si $a_{12} - \epsilon > a_{22} > a_{21} > 0$, entonces la persona 2 someterá una oferta por el objeto 3 en la próxima iteración y el algoritmo terminará.

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} a_{11}^* & a_{12} & 0 \\ a_{11} & a_{12}^* & 0 \\ a_{21} & a_{22} & 0_* \end{pmatrix} \end{array}$$

Si $a_{22} > a_{12} - \epsilon$ y $a_{22} > a_{21} > 0$, la persona 2 someterá una oferta por el objeto 2 e incrementará el precio $\hat{p}_2$;

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} a_{11}^* & a_{12} & 0 \\ a_{11} & a_{12} & 0 \\ a_{21} & a_{22}^* & 0 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (\hat{p}_1, \hat{p}_2, 0)$$

las personas 1 y $1'$ entonces someterán otra oferta común por los objetos 1 y 3 en la tercera iteración, y el algoritmo terminará con la siguiente asignación:

$$\begin{array}{cc} & \text{objetos} \\ & 1 \quad 2 \quad 3 \\ \text{personas} \quad \begin{array}{c} 1 \\ 1' \\ 2 \end{array} & \begin{pmatrix} a_{11}^* & a_{12} & 0 \\ a_{11} & a_{12} & 0^* \\ a_{21} & a_{22}^* & 0 \end{pmatrix} \end{array}$$

De esta manera se puede ver que el algoritmo de subasta con personas semejantes resuelve rápidamente la guerra de precios que ocasionan las personas duplicadas correspondientes a los orígenes del problema de transporte.

## 5.4 Algoritmo de subasta con objetos y personas semejantes

La idea de una oferta común por una persona de una misma clase de semejanza se puede combinar con la idea anterior del umbral de contención y el correspondiente algoritmo de subasta para objetos semejantes. En particular, la oferta cooperativa de una persona en una clase de semejanza está basada en los umbrales de contención de los objetos. Con un cálculo apropiado, se obtiene un algoritmo para el problema de transporte 5.1 el cual, en efecto, es el algoritmo de subasta para el correspondiente problema equivalente de asignación (ver las figuras 13 y 14) pero con personas semejantes y objetos propiamente tomados en cuenta.

En este algoritmo de subasta/transporte, cada destino $j$ tiene un precio $p_j$ y cada arco $(i,j)$ que lleva flujo positivo $x_{ij}$ tiene un umbral de contención $y_{ij}$ asociado. El precio $p_j$ es igual al precio original del destino $j$ si la demanda del destino no ha sido aún agotada ($\sum_i x_{ij} < \beta_j$) y es igual al umbral mínimo de contención de los arcos que llegan y que llevan flujo positivo ($\min_{i,\ x_{ij}\geq 0}\ y_{ij}$). Los orígenes ofrecen por los destinos para incrementar el umbral de contención de los arcos correspondientes, un destino incrementa su precio una vez que toda su demanda es agotada y los umbrales de contención asociados se incrementan. En el ejemplo 11 se ilustra este algoritmo.

**Ejemplo 11.**

Resolver el siguiente problema transporte por medio del algoritmo de subasta con personas y objetos semejantes.

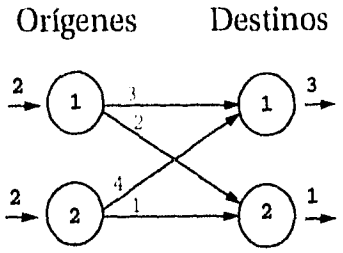



**Fig. 19.** *Problema de transporte*

De la conversión del problema de transporte al problema de asignación, el origen 1 se convierte en las dos personas semejantes 1 y 1′, el origen 2 se convierte en las dos personas semejantes 2 y 2′, el destino 1 se convierte en los objetos semejates 1, 1′ y 1″.

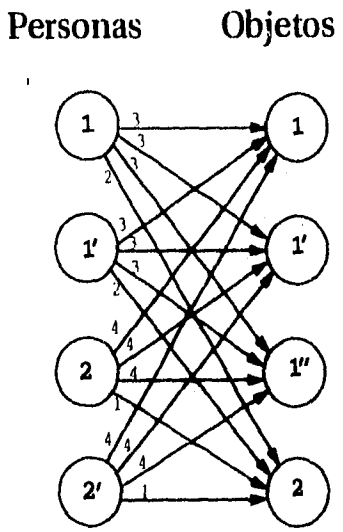



**Fig. 20.** *Problema de asignación*

La representación matricial del problema de transporte es como sigue:

$$\begin{array}{c} \\ \text{personas} \end{array} \begin{array}{c} \text{objetos} \\ \begin{array}{cccc} 1 & 1' & 1'' & 2 \end{array} \\ \begin{array}{c} 1 \\ 1' \\ 2 \\ 2' \end{array} \begin{pmatrix} 3 & 3 & 3 & 2 \\ 3 & 3 & 3 & 2 \\ 4 & 4 & 4 & 1 \\ 4 & 4 & 4 & 1 \end{pmatrix} \end{array} \quad \text{Vector de precios} \quad (0,0,0,0)$$

Para cada una de las personas se toma el $\max\{a_{ij} - p_j\}$, a continuación se muestra la representación matricial de la primera asignación parcial:

$$\begin{array}{c} \\ \text{personas} \end{array} \begin{array}{c} \text{objetos} \\ \begin{array}{cccc} 1 & 1' & 1'' & 2 \end{array} \\ \begin{array}{c} 1 \\ 1' \\ 2 \\ 2' \end{array} \begin{pmatrix} 3^* & 3 & 3 & 2 \\ 3 & 3^* & 3 & 2 \\ 4 & 4 & 4 & 1 \\ 4 & 4 & 4 & 1 \end{pmatrix} \end{array} \quad \begin{array}{llll} \nu_2 = 4 & & \nu_{2'} = 3-\epsilon \\ \omega_2 = 1 & & \omega_{2'} = 1 \\ \gamma_2 = 3+\epsilon & & \gamma_{2'} = 2 \end{array}$$

$$\text{Vector de precios} \quad (1+\epsilon, 1+\epsilon, 0, 0)$$

La persona 2 y $2'$ no estan asignadas, por lo tanto, hacen una oferta $\gamma_2 = 3+\epsilon$ y $\gamma_{2'} = 2$ por los objetos $1''$ y 1 respectivamente, la asignación parcial se muestra a continuación.

$$\begin{array}{c} \\ \text{personas} \end{array} \begin{array}{c} \text{objetos} \\ \begin{array}{cccc} 1 & 1' & 1'' & 2 \end{array} \\ \begin{array}{c} 1 \\ 1' \\ 2 \\ 2' \end{array} \begin{pmatrix} 3 & 3 & 3 & 2 \\ 3 & 3 & 3 & 2 \\ 4 & 4 & 4^* & 1 \\ 4^* & 4 & 4 & 1 \end{pmatrix} \end{array} \quad \begin{array}{llll} \nu_1 = 2 & & \nu_{1'} = 2-\epsilon \\ \omega_1 = 2-\epsilon & & \omega_{1'} = -\epsilon \\ \gamma_1 = 2\epsilon & & \gamma_{1'} = 2+\epsilon \end{array}$$

$$\text{Vector de precios} \quad (3+\epsilon, 1+\epsilon, 3+\epsilon, 0)$$

Las personas 1 y $1'$ hacen una oferta $\gamma_1 = 2\epsilon$ y $\gamma_{1'} = 2+\epsilon$ por los objetos 2 y $1'$ respectivamente, la asignación parcial se muestra a continuación. Fin, todas las personas tienen un objeto asignado.

$$\begin{array}{c} \\ \text{personas} \end{array} \begin{array}{c} \text{objetos} \\ \begin{array}{cccc} 1 & 1' & 1'' & 2 \end{array} \\ \begin{array}{c} 1 \\ 1' \\ 2 \\ 2' \end{array} \begin{pmatrix} 3 & 3 & 3 & 2^* \\ 3 & 3^* & 3 & 2 \\ 4 & 4 & 4^* & 1 \\ 4^* & 4 & 4 & 1 \end{pmatrix} \end{array}$$

La asignación total

$$S = \{(1,2),(1',1'),(2,1''),(2',1)\}$$

y el vector de precios

$$p = (3+\epsilon, 3, 3+\epsilon, 2\epsilon,)$$

satisfacen la condición de holgura complementaria,

$$\sum_{(i,j)\in S} a_{ij} = \sum_{i=1}^{n} \max_{j}\{a_{ij} - p_j\} + \sum_{j=1}^{n} p_j$$

$$\begin{aligned} 2+3+4+4 &= (1-\epsilon+0+1-\epsilon+2-2\epsilon)+(3+\epsilon+3+3+\epsilon+2\epsilon) \\ 13 &= 13 \end{aligned}$$

por lo tanto, $S$ es una asignacion óptima.

La solución del problema de transporte de acuerdo con la asignación que se obtuvo es:

$$\begin{aligned} x_{11} &= 1 \\ x_{12} &= 1 \\ x_{21} &= 2 \\ x_{22} &= 0 \end{aligned}$$

# Capítulo 6

# Aspectos computacionales

Siempre se ha tenido la necesidad de encontrar soluciones a problemas computacionales muy grandes, los cuales dependen de las capacidades de los procesadores y del tiempo que se tardan los procesadores en dar los resultados. Actualmente el cómputo paralelo es una área de intensa y activa investigación para encontar la solución de tales problemas de una manera más eficiente. Además, gracias a la habilidad y al poder de las computadoras paralelas, generalmente se encuentran nuevas e interesantes soluciones a problemas resueltos en el pasado. Aprovechando este hecho, se mostrará el uso del paralelismo en el algoritmo de subasta presentado en los capítulos anteriores.

El propósito del paralelismo es poder ejecutar más rápido una tarea que tradicionalmente corre en un sólo procesador utilizando más procesadores, es decir, hacer la misma tarea en diferentes procesadores al mismo tiempo, para esto existen las máquinas masivamente paralelas conocidas como MPP's, que son de las más poderosas que hay para algoritmos paralelos. Estas máquinas tienen cientos de CPUs iguales, interconectados y coordinados. También existe el cómputo distribuido, que no es más que correr un proceso en un conjunto de máquinas. Una computadora con sistema distribuido, es un conjunto de computadoras, posiblemente de distinto tipo, conectadas entre sí mediante una red.

En cualquier algoritmo paralelo es necesario coordinar algunas actividades de los diferentes procesadores. Esta coordinación muchas veces se implementa dividiendo el algoritmo en "fases". Durante cada fase, cada procesador debe ejecutar un número de operaciones que dependen de los resultados de los cálculos de otros procesadores en fases anteriores; sin embargo, el tiempo del cálculo

de cualquier procesador durante la fase puede ser independiente del tiempo de cálculo de otro procesador durante la misma fase; es decir, dentro de una fase cada procesador no interactúa con otro procesador hasta que el algoritmo lo pida, entonces los procesadores deben esperar en un punto predeterminado a que los demás procesadores terminen su fase. A estos algoritmos se les llama *síncronos.*

Los algoritmos que no requieren estar divididos en fases, ni requieren de una coordinación computacional tan estricta de los diferentes procesadores, es decir, no requieren esperar en puntos predeterminados se les llama *asíncronos.*

El cómputo serial se caracteriza por tener un sólo lugar de control que determina la siguiente instrucción a calcular. Así sólo una instrucción se calculará cada vez; hay que notar que los accesos a la memoria y las instrucciones de entrada y de salida pueden ocacionar que los cálculos computacionales sean más lentos, para evitar estos retardos se han diseñado varios métodos, como son "cache memories" y "pipelining".

Las primeras supercomputadoras se desarrollaron en base a estos avances, con la idea de organizar inteligentemente la memoria y el uso del "pipelining" y de esta manera se ha logrado que las supercomputadoras sean capaces de ejecutar operaciones vectoriales, es decir, se pueden realizar varias operaciones a la vez, en el caso de la Cray-YMP se pueden ejecutar 64 operaciones en una iteración, que a diferencia de las operaciones escalares los primeros 64 resultados se tendrian hasta despues de 64 iteraciones.

El algoritmo de subasta se puede paralelizar perfectamente ya que todas las ofertas se pueden calcular simultáneamentente, entonces se le puede asignar a cada procesador el cálculo de una oferta diferente, a este método se le conoce con el nombre de *Jacobi.* Otra alternativa es el método de *Gauss-Seidel*, el cual calcula las ofertas una por una y usa las ofertas más recientes; esto hace que el algoritmo realice menos iteraciones que el método de Jacobi. Pero una iteración Gauss-Seidel no se puede paralelizar, ya que el cálculo de la oferta $\gamma_i$ depende de todos los precios.

## 6.1 Implementación paralela del algoritmo de subasta

A continuación se muestran algunos aspectos computacionales del algoritmo de subasta, para el caso del problema de asignación. El algoritmo de subasta es por lo menos competidor y muchas veces mejor que los métodos clásicos ya que los algoritmos de subasta son muy adecuados para la computación en paralelo.

Parte de este trabajo fue programar el algoritmo se subasta en una máquina paralela vectorial con memoria compartida (Cray Y-MP). A continuación se explica qué fue lo que se tomó en cuenta para paralelizar el programa.

Las dos fases del algoritmo de subasta, la de oferta y la de asignación son altamente paralelizables. En particular, las ofertas se pueden calcular simultaneamente es decir, en paralelo para todas las personas que participan en la subasta.

Como se necesita una estricta separación temporal entre la fase del cálculo de la oferta y la fase de asignación, se dice que la implementación es síncrona. Para tal implementación, se utilizo el método de Jacobi para paralelizar la fase de oferta del conjunto de personas $I$ no asignadas que someten una oferta simultáneamente.

Los cálculos implicados en la oferta de cada persona $i \in I$ son realizados por cada procesador simultáneamente. Sea $m$ el número de procesadores disponibles, si el número de personas en $I$, es $|I|$, y excede al número de procesadores $m$, entonces algunos procesadores ejecutarán más cálculos que otros. Si $|I| < m$ entonces $m - |I|$ procesadores estarán desocupados durante la fase de la oferta.

Una vez que se termina el cálculo de la fase de oferta de una iteración, se necesita una sincronización para ejecutar la fase de asignación. Esta fase se puede ejecutar por un solo procesador. También se puede considerar usar más de un procesador para ejecutar la fase de asignación en paralelo, pero la ganancia potencial de paralelización puede ser modesta mientras que el tiempo de sincronización de los procesadores (overhead) puede ser muy grande como para compensar esta ganancia, es por esto que se usa sólo un procesador en la

fase de asignación.

A causa de la típica y prolongada "guerra de precios" del algoritmo de subasta, cuando queda un pequeño porcentaje de personas no asignadas, la velocidad que se puede obtener mediante el método de *Jacobi* es relativamente modesta. Sin embargo la velocidad para el método de *Gauss-Seidel* es potencialmente más grande, particularmente cuando el problema es denso, y también cuando se tiene hardware con capacidades vectoriales.

## 6.2 Velocidad y eficiencia

Se describirán algunos conceptos que son útiles para comparar algoritmos seriales y paralelos. Supóngase que se ha escogido un modelo particular paralelizable, sea $P$ el problema computacional dado y $n$ el tamaño del problema, supóngase además que se tiene un algoritmo $A$ paralelo que resuelve el problema $P$ en un tiempo $T_p(n)$ en una computadora paralela con $p$ procesadores ($p$ puede depender de $n$). Sea $T^*(n)$ el tiempo que se tarda en resolver el problema $P$ de la mejor manera serial. La velocidad conseguida por $A$ se define por:

$$S_p(n) = \frac{T^*(n)}{T_p(n)}$$

Claramente, $S_p(n)$ es la medida del factor de velocidad obtenido por el algoritmo $A$ cuando se tienen $p$ procesadores disponibles. Lo ideal sería que $S_p(n) = p$, en realidad hay muchos factores que lo impiden, por ejemplo la comunicación y el "overhead" que se tiene al sincronizar la actividad de varios procesadores y el control del sistema.

La eficiencia del algoritmo $A$ se denota por:

$$E_p(n) = \frac{T_1(n)}{pT_p(n)}$$

esto indica qué tan eficientemente se utilizan los $p$ procesadores. Si el valor de $E_p(n)$ es aproximadamente igual a 1, para alguna $p$, indica que aproximadamente el algorimo $A$ corre $p$ veces mas rápido usando $p$ procesadores que si se usara sólo un procesador. Esto indica que cada procesador está haciendo "eficientemente su trabajo". Para que esto ocurra, el algoritimo paralelo debe ser tal que ningún procesador permanezca inactivo o bien que no se realice

ningún trabajo innecesario. Esta situación ideal es prácticamente inalcanzable. Un objetivo más realista es tratar de tener una eficiencia que esté cerca de 1 cuando $n$ y $p$ crecen.

Para la siguiente gráfica se corrio el programa de subasta para el problema de asignación de dimensión 500, con valores aleatorios, la gráfica (figura 21) muestra el rendimiento del programa en paralelo y que tan paralelizable es, también se muestran los tiempos que se tendrían si se contara con más procesadores.

Las curvas indican las velocidades esperadas conforme el número de procesadores aumenta.

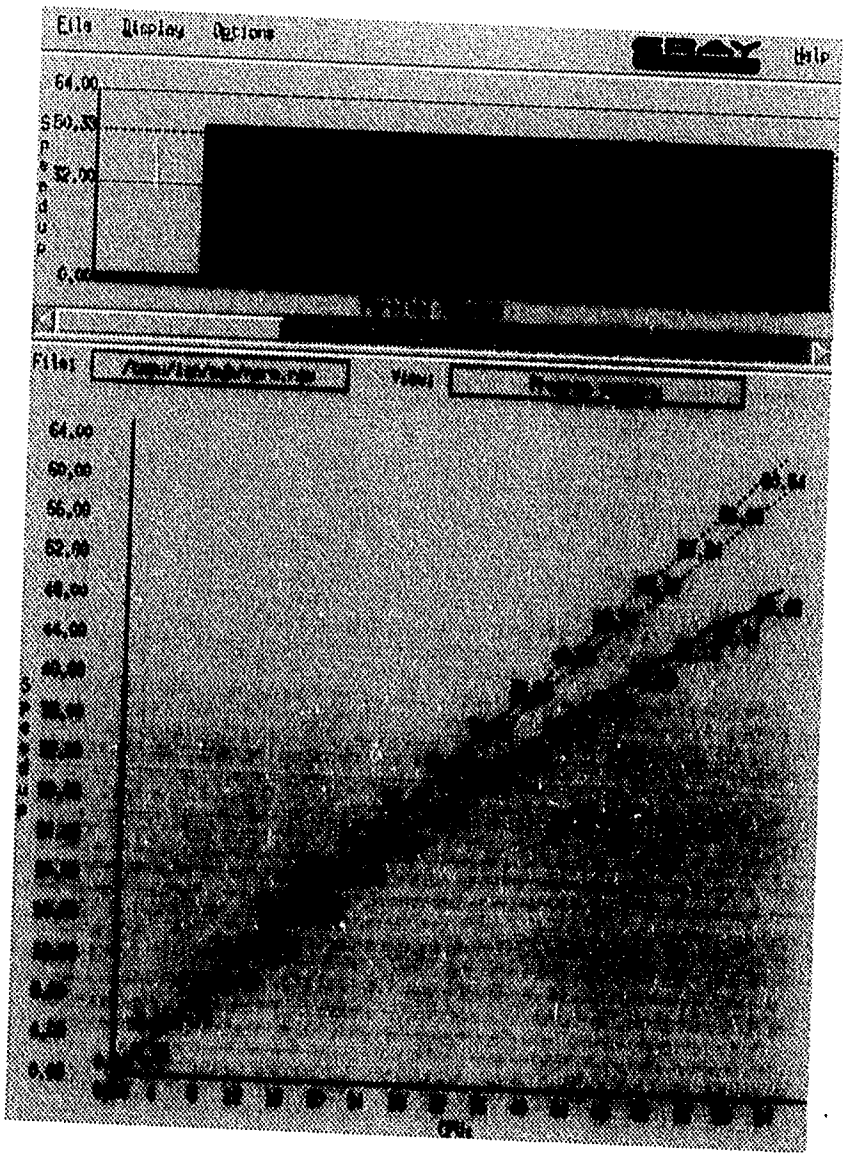



**Fig. 21.** *Rendimiento del programa en paralelo*

- La primera curva muestra la velocidad lineal que el programa tendría si la porción del programa que se está analizando fuera 100% paralelizable y sin "overhead".

- La curva de enmedio muestra la ley de *Amdahl*, y representa la velocidad ideal esperada considerando el tiempo que se requiere en las partes seriales, pero se supone que no hubo ningún "overhead"sociado a la parte paralela.

- La tercera curva muestra la velocidad esperada en un sistema dedicado: aquí se toma en cuenta el "overhead"que hubo en el programa y también se considera el tiempo utilizado en la parte serial del programa. Los escalones indican que el trabajo a realizar en paralelo no es estable, lo cual se debe a que el número de personas no asignadas va disminuyendo, por lo que cada vez se requieren cálcular menos ofertas

Estas tres líneas dividen la ventana en tres áreas, las cuales indican lo siguiente:

- La primera que va desde abajo hasta la tercera curva, muestra el efecto del trabajo realizado en paralelo. Se espera que esta área sea lo más grande posible.

- La segunda área representa el "overhead"que es el tiempo de sincronización de los procesadores.

- La tercera área que está entre la curva de la ley de Amdahl y la línea recta, muestra el efecto del código serial sobre el rendimiento general. Para este programa el código serial consta de la lectura de los datos, la fase de asignación y la escritura de los resultados. El código paralelo es la fase de oferta.

De este modo se espera que las dos últimas áreas sean lo más pequeñas posibles.

A continuación se mostrarán algunos resultados del algoritmo de subasta con sus variantes. Se implementarón para el método de Jacobi la subasta hacia adelante (forward), hacia atrás y combinada, y con el método de Gauss sólo se uso la subasta hacia adelante. La subasta hacia atrás no se muestra en las siguientes gráficas pues el tiempo de ejecución es muy parecido al de la subasta hacia adelante. Las matrices que se usarón tienen una distribición uniforme, el comportamiento de los programas puede ser diferite si se utiliza otro tipo de matrices.

En la siguiente gráfica se muestra como disminuye el tiempo real, cuando se utilizan 4 CPU's en vez de uno. El método utilizado es el Jacobi, de subasta hacia adelante.

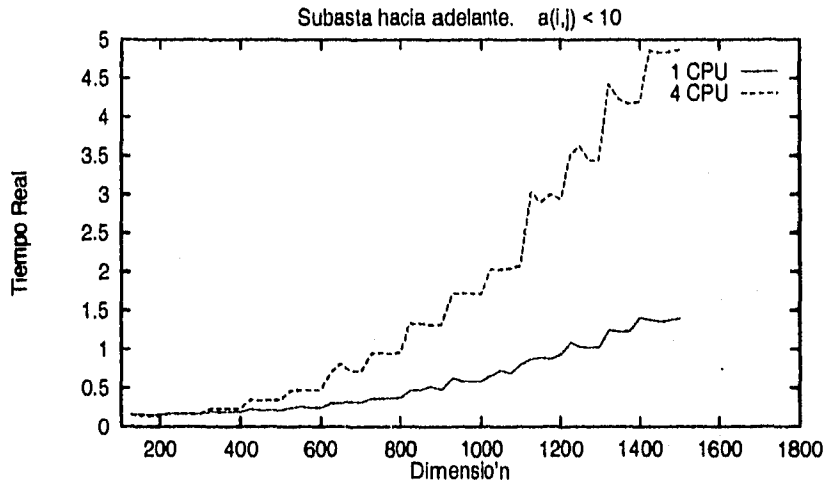



**Fig. 22.** *Tiempo real con diferente número de CPU's*

En la figura 23 se muestran los tiempos de CPU para el método de Jacobi con subasta hacia adelante y combinada y para el método de Gauss. Las entradas de las matrices son menores que 10, es decir, se tiene poca dispersión en los datos. Esto significa, que se tienen muchos objetos con igual beneficio, y además poca diferencia entre los que son diferentes. Esto provoca que varias personas quieran ser asignadas a un mismo objeto. Después de algunas iteraciones en que las personas compiten por ese objeto, el precio del objeto se eleva tanto que existirá un objeto con menor beneficio pero menor precio. Es decir, se genera una guerra de precios. Esta situación se presenta con mayor frecuencia para el método de Jacobi (sólo cuando se tiene poca dispersión) y no para el método de Gauss, ya que en el método de Gauss cada vez que una persona realiza una oferta, actualiza el vector de precios, por lo que la siguiente persona que también quería ese objeto ahora tomara encuenta su nuevo precio lo cual puede ocasionar que prefiera a otro y sea asignada. En cambio, en el método de Jacobi todas las personas no asignadas realizan sus ofertas al mismo tiempo, y sólo una de ellas se quedará con ese objeto. Es por esto, que el método de Gauss llega más rápido a la soulución, como se muestra en la gráfica.

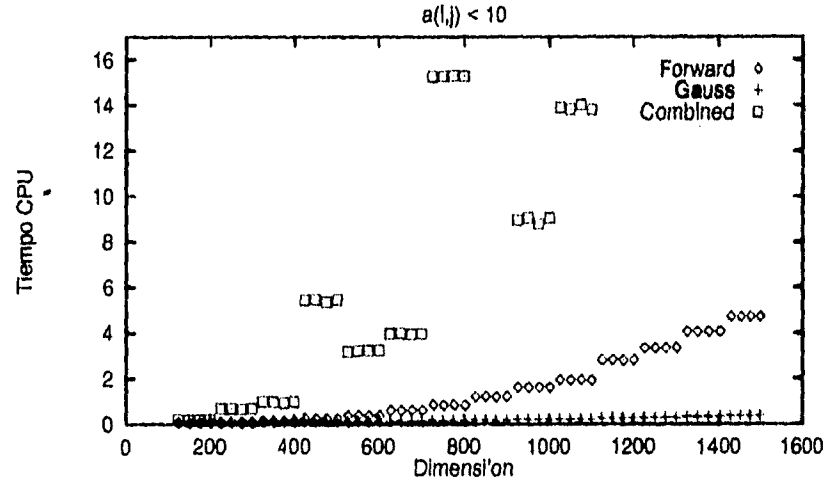



**Fig. 23.** *Comparación de tiempos*

En la figura 24. se muestran los tiempos de CPU usando ahora valores entre 0 y 10000. es decir. se tiene una gran dispesión en los datos. Ahora el método combinado es el que requiere menor tiempo de CPU. Para entender porque la subasta combinada es más rápida. se considera el problema de ruta más corta. El algoritmo trabaja con dos rutas $P$ y $R$. donde $P$ inicia en el nodo origen y $R$ en el destino. aplicando la subasta hacia adelante para la ruta $P$ y la subasta hacia atrás para la ruta $R$. hasta que $P$ y $R$ tienen un nodo en común. Entonces en vez de recorrer toda la gráfica ya sea con la subasta hacia adelante o hacia atrás. se encuentra la ruta más con las dos subastas.

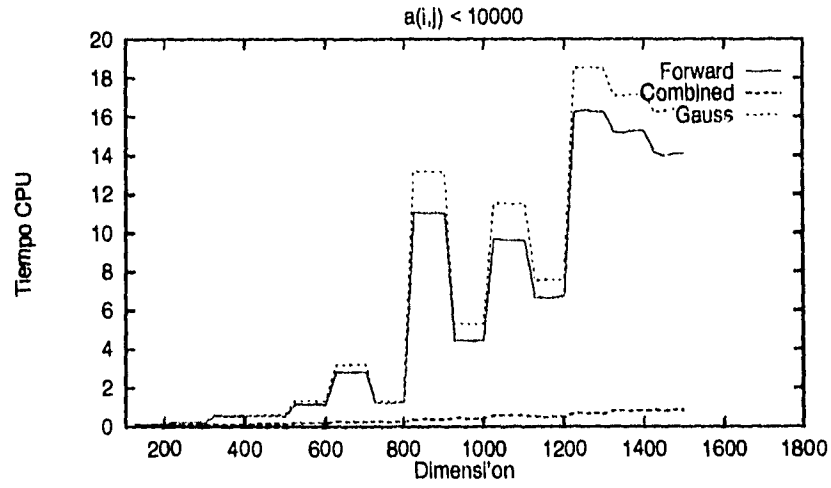



**Fig. 24.** *Comparación de tiempos*

En la figura 25, se muestra el número de ofertas que necesitarón los programas para encontrar una asignación óptima. El valor del eje "y", se encuentra sumando en cada iteración el número de personas no asignadas, es decir, el número de ofertas que se realizan en esa iteración. El trabajo necesario para cálcular una oferta es lo más pesado del programa, por eso el tiempo de CPU esta directamente relacionado con este valor, esto hace que esta gráfica y la anterior sean tan parecidas.

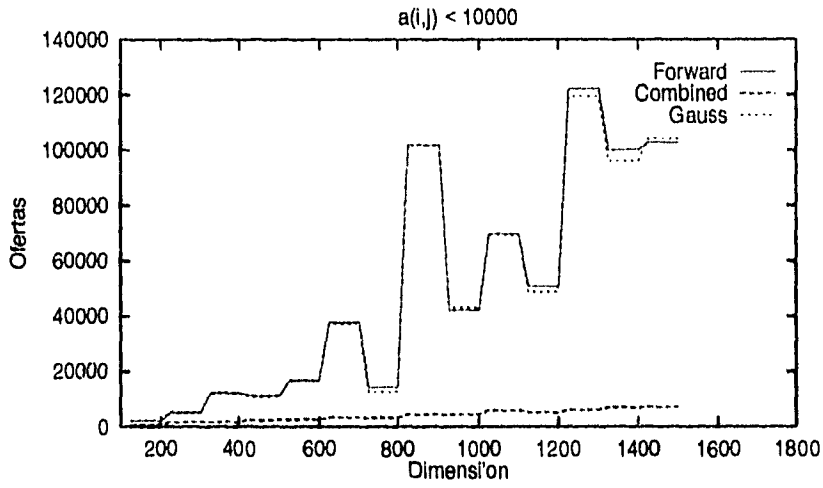



**Fig. 25.** *Número de ofertas efectuadas*

# Conclusiones

El algoritmo de subasta es muy fácil de entender ya que se asemeja mucho a una subasta verdadera. Es novedoso pues fue pensado para ser resuelto en paralelo, esto es en distintos procesadores a la vez. Además, permite resolver los problemas de asignación, de rutas más cortas y el de transporte, entre otros. A pesar de que estos problemas ya se han resuelto mediante los métodos tradicionales, el algoritmo de subasta suele ser más rápido debido a que se puede paralelizar.

Los resultados que se obtuvieron con la implementación de este algoritmo muestran que generalmente se necesitan menos iteraciones si se utiliza el método de *Gauss-Seidel* el cual es serial, pero resulta ser más rápido si se usa el método de *Jacobi*, esto se debe a que diferentes procesadores realizan el cálculo de la oferta al mismo tiempo, pero se actualiza el vector de precios hasta que se tienen todas las ofertas esto es lo que provoca que realice más iteraciones paralelizado. Una desventaja de la implementación del programa es que requiere memoria del orden del cuadrado de la dimensión del problema.

Este trabajo puede servir como apoyo a los estudiantes del tema, ya que contiene algunos ejemplos que hacen más fácil la comprensión del algoritmo. También se puede utilizar como un inicio al estudio de los métodos duales ascendentes o bien para entender un poco más las ventajas del cómputo paralelo.

# Bibliografía


Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali. Linear Programming and Network Flows (2nd Edition). John Wiley and Sons, New York, 1990.

Bazaraa, M. S., C. M. Shetty. Nonlinear Programming: Theory and Algorithms. John Wiley and Sons, Inc., New York, NY, 1979.

Bertsekas, D. P., Linear Network Optimization. Mit, Cambridge, Massachusetts, 1991.

Bertsekas, D. P., "The Auction Algorithm for Assignment and Other Network Flow Problems: A Tutorial", pp 133-149, July-Augnst 1991.

Bertsekas, D. P., and Castanôn, D. A., "The Auction Algorithm for Transportation Problems", Annals of Operations Research, Vol. 20, pp 67-96, 1989.

Bertsekas, D. P., "A New Algorithm for the Assignment Problem", Math. Programming, Vol. 21, pp 157-171, 1981.

Bertsekas, D. P., "An Auction Algorithm for Shortest Paths", Lab. for Information and Decision Systems Report P-2000, M. I. T., Cambrige, MA., SIAM on Optimization, 1990.

Bertsekas, D. P., and Tsitsiklis, J. N. Parallel and Distributed Computation: Numerical Methods, Prentice Hall, Englewood Cliffs, N. J. 1989.

Bruce, P. L., The Art of Parallel Programming, Prentice Hall, Englewood Cliffs, N. J. 1993.