Capítulo 1 Marco Teórico del PAV

1.1 Teoría de Grafos

1.1.1 Grafos No Dirigidos

En un sentido amplio, un *grafo G* es un diagrama que, si se interpreta en forma adecuada proporciona información. Cuando los grafos no tienen dirección alguna, se dice que son *grafos no dirigidos,* por lo que en esta sección se discutirá sobre ellos mencionándolos simplemente como grafos. Lo más importante de un grafo son sus objetos y líneas. Los objetos, representados por puntos, se llaman *vértices* o *nodos* y las líneas que los unen *aristas.* Se denotan los vértices de G como v_1, v_2,….., v_v y las aristas por e_1, e_2,….., e_e, siendo el subíndice v de v_v el número total de vértices y el subíndice e de e_e el número total de aristas, ver el ejemplo en la figura 1.1.a. Pero algunos autores hacen otro tipo de notación, es decir denotan a los vértices con una de las últimas letras del alfabeto, y a las aristas las denotan con números o bien también con letras, de preferencia las primeras letras del alfabeto, ver figura 1.1.c. El grafo de la figura No. 1.1a y 1.1b tienen 5 vértices y 7 aristas, mientras que la figura 1.1.c tiene 4 vértices y 6 aristas. En la figura 1.1.c, las aristas que unen al vértice v con el w se llaman *aristas múltiples* o *paralelas*. Una de las aristas conecta al vértice w consigo mismo, a ésta se le denomina *lazo*. [KEN90].

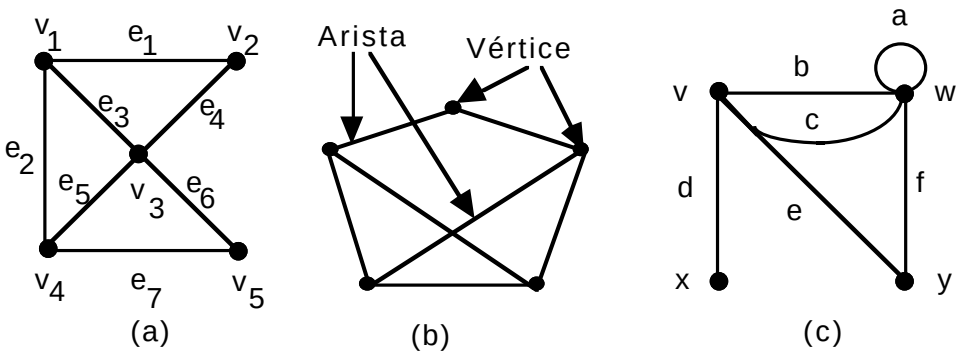

Figura 1.1 Vértices y Aristas

Una manera más formal de definir un grafo es :

Definición 1 : *Se denota al conjunto de todos los vértices de un grafo G por V_G y, al conjunto de aristas por E_G* [LOO85].

Por ejemplo, en el grafo G de la figura 1.1.c, V_G = {v, w, x, y} y E_G = {a, b, c, d, e, f}. El número de elementos de V_G es llamado el *orden* del grafo G, por lo que en el ejemplo de la figura 1.1.c el orden del grafo G es 4. Un grafo nulo es un grafo con orden cero. Una arista está determinada por los nodos que conecta. La arista *c* de la figura 1.1.c, por ejemplo conecta los nodos *v* y *w* y se dice que es de la forma (v, w). Un grafo está completamente definido por sus conjuntos de vértices y aristas [LOO85].

En la teoría de grafos nos interesan las aristas que se unen para formar un camino. En los caminos se permite la repetición de aristas, b-a-b-e-f-a-a-b es un camino. La *longitud de un camino* es la cantidad de aristas que tiene. En b-a-b-e-f-a-a-b tiene longitud 8, ver la figura 1.1.c. Las aristas adyacentes en un camino tienen un vértice común. Por lo tanto, un *camino* determina una sucesión de vértices. Las sucesiones de vértices correspondientes a los caminos de la figura 1.1.c están representados en la siguiente tabla [KEN90] :

El camino	Su Sucesión de vértices
d-b-f-e	x-v-w-y-v
c-f-e-b-a	v-w-y-v-w-w
b-a-f-e-c	v-w-w-y-v-w
c-a-f-e-b	v-w-w-y-v-w
b-a-b-e-f-a-a-b	v-w-w-v-y-w-w-w-v

Tabla 1.1 Caminos de la figura 1.1.c

Es interesante detectar ciertos detalles como que el número de vértices en una sucesión de vértices supera en uno al de las aristas en un camino. Cuando un grafo no tiene aristas paralelas ni lazos, entonces la sucesión de vértices determina el camino de manera única [KEN90]. Un *camino* es *cerrado* si el primero y el último vértice de la sucesión de vértices son el mismo. Un *ciclo* es un camino cerrado eficiente en el sentido de que no repite aristas y en la sucesión de vértices todos son distintos, excepto el primero y el último. Un *grafo* es *acíclico* si no tiene ciclos. Un *camino* es *acíclico* si el subgrafo que contiene las aristas y los vértices del camino es acíclico. El grafo de la figura 1.2.a no es acíclico ya que v-u-y-x-w es un ciclo, mientras que el grafo de la figura 1.2.b es acíclico ya que no contiene ciclos [KEN90]. Es importante hacer notar que el lenguaje de teoría de grafos no se ha estandarizado : varios autores utilizan "circuito" y "lazo" para lo que otros llaman ciclo, y "ciclo" se utiliza a veces para un camino cerrado [KEN90]. Cabe aclarar que en este proyecto se usará la notación de [KEN90] para denotar un circuito, es decir circuito es equivalente a ciclo.

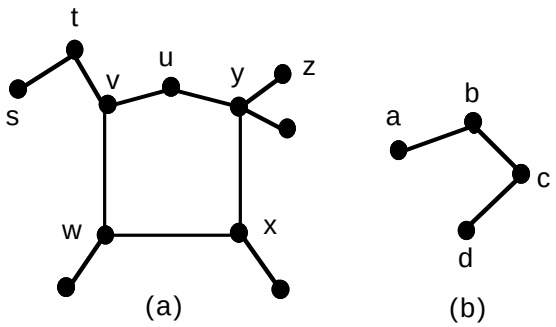

Figura 1.2 (a) Grafo No acíclico; (b) Grafo acíclico

Definición 2 : *Un grafo no dirigido es conexo si todos sus pares de vértices distintos están conectados por un camino en el grafo* [KEN90].

Ver figura 1.3.a donde se muestra un grafo conexo y en la figura 1.3.b un grafo no conexo [AHO74]. Un subgrafo conexo de un grafo conexo G que no está contenido propiamente en ningún otro subgrafo conexo de G se llama un *componente de G* y el *número de componentes de G* es denotado por $\omega(G)$. El componente que contiene a un vértice dado *v* consta de *v* y de todos los vértices y aristas de los caminos que empiezan en *v* [KEN90]. En la figura 1.3.a hay un componente mientras que en la figura 1.3.b hay dos componentes en el grafo.

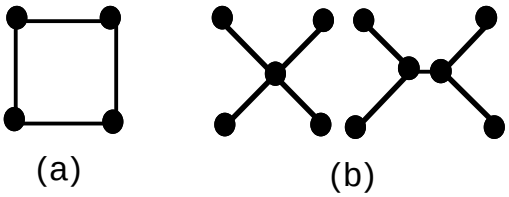

Figura 1.3 (a) Grafo Conexo; (b) Grafo no conexo

La *valencia* de un vértice *en un grafo no dirigido* es el número de aristas que confluyen en él, por ejemplo la valencia del vértice c de la figura 1.2.b es de 2, mientras que la valencia del vértice x en la figura 1.2.a es de 3 [KEN90].

Cada grafo puede ser representado con estructuras matemáticas, una de estas estructuras consiste en que cada grafo puede tener una matriz de incidencia y una matriz de adyacencia, así para cualquier grafo G donde corresponde una matriz *v* x *e* llamada *matriz de incidencia de G*. Se denotan los vértices de G como $v_1, v_2, \ldots, v_v$ y las aristas por $e_1, e_2, \ldots, e_e$. Entonces la matriz de incidencia de G es la matriz $M(G) = [m_{ij}]$, donde m_{ij} es el número de veces (0, 1 o 2) que v_i y e_j son incidentes. La matriz incidente de un grafo es justo un camino diferente de especificar al grafo. Otra matriz asociada con G es la *matriz de adyacencia*, esta es la matriz *v* x *v* $A(G) = [a_{ij}]$, en donde a_{ij} es el número de aristas que unen a v_i con v_j. Se muestra a continuación en la figura No. 1.4 un ejemplo de un grafo, en la tabla 1.2 su matriz de incidencia, y en la tabla 1.3 su matriz de adyacencia [BON76].

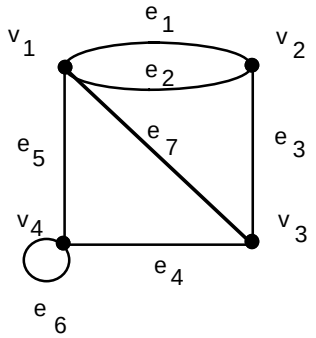

Figura 1.4 Grafo (G)

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
V_1	1	1	0	0	1	0	1
V_2	1	1	1	0	0	0	0
V_3	0	0	1	1	0	0	1
V_4	0	0	0	1	1	2	0

Tabla 1.2 Matriz de incidencia M(G)

	V_1	V_2	V_3	V_4
V_1	0	2	1	1
V_2	2	0	1	0
V_3	1	1	0	1
V_4	1	0	1	1

Tabla 1.3 Matriz de Adyacencia A(G)

La matriz de adyacencia de un grafo es generalmente considerada más pequeña que su matriz de incidencia, y es en esta forma que un grafo se almacena comúnmente en las computadoras [BON76].

Otra forma de representar grafos es por medio de listas enlazadas. Cuando se procesan conjuntos de datos cuyo espacio de almacenamiento no se puede predecir a priori (en tiempo de compilación) y además la actividad de los mismos (inserciones y borrados) es frecuente, las estructuras de datos estáticas (los arreglos) no son adecuadas para su implementación. Por lo que una *lista enlazada* es una secuencia de nodos que se encuentran enlazados cada uno con el siguiente mediante un enlace o puntero. Cada elemento (nodo) de una lista enlazada debe tener dos campos: un campo (info) que contiene el valor de ese elemento y un campo (enlace o puntero) que indica la posición del siguiente elemento. En la figura 1.5 se muestra un ejemplo de un nodo [JOY98].

Matemáticamente, una lista es una secuencia de cero o más elementos de un tipo determinado (que por lo general se denominará tipo_elemento). A menudo se representa una lista como una sucesión de elementos separados por comas $a_1, a_2, \ldots, a_n$ donde $n \geq 0$ y cada a_i es del tipo tipo_elemento. Al número n de elementos se le *llama longitud de la lista*. Al suponer que $n \geq 1$, se dice que a_1 es el primer elemento y a_n es el último elemento. Si n = 0, se tiene una lista vacía, es decir, que no tiene elementos [AHO74].

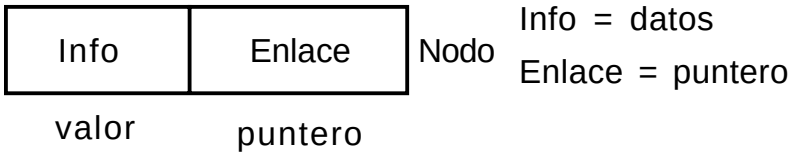

Figura 1.5 Nodo de una lista ligada o lista enlazada

Las listas enlazadas son estructuras de datos lineales que si bien ofrecen una gran flexibilidad de diseño, determinados problemas son difíciles o, en su caso, lentos de resolver, por ejemplo, el caso de la búsqueda de un determinado elemento que exige, un recorrido secuencial que ralentiza el proceso. En este caso son mejor los árboles que son estructuras que permiten representar datos que tienen enlaces jerárquicos entre ellos [JOY98].

Desde el punto de vista de grafos, recordando que un grafo acíclico es aquél que no contiene ciclos, entonces :

Definición 3 : Un *árbol* es un grafo acíclico conexo [BON76].

Ver la figura 1.6 donde se muestra un ejemplo de árboles de 6 vértices, donde se puede observar que todos los árboles de seis vértices tienen 5 aristas.

Teorema 1 : *En un árbol, dos vértices cualesquiera son conectados por un camino único* [BON76].

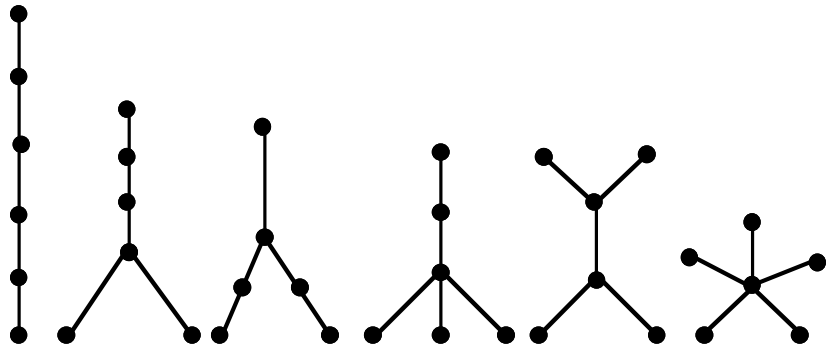

Figura 1.6 Arboles de seis vértices

Supóngase que V' es un subconjunto no vacío de V(vértices), hay grafos que se manipulan quitándole vértices, si v es un vértice y G es un grafo se puede escribir G - v para denotar al grafo que se obtiene al eliminar el vértice v. Esto se hace en lugar de escribir G - V', donde es el subgrafo obtenido de borrar los vértices en V' junto con sus aristas incidentes, Si V' = {v} se escribe *G - v* en un abuso de notación en lugar de G - {v}. Por otro lado, es conveniente muchas veces manipular grafos quitando o agregando aristas, si *e* es una arista y *G* es un grafo escribimos *G - e* para denotar el grafo que se obtiene al eliminar la arista *e*. Siendo E' un subconjunto no vacío de E, cabe aclarar que al grafo G se le borran las aristas en E' que es denotado por G + E' . Si E' = {e} se escribe *G - e* y *G + e* en un abuso de notación en lugar de G - {e} y G + {e} [BON76].

En general puede decirse que :

Teorema 2 *: Sea ε el número de aristas de un grafo, y v el número de vértices de un grafo, entonces si G es un árbol, se cumple que ε = v -1* [BON76].

Para ejemplificar el teorema 2 ver cada uno de los árboles de la figura 1.6 con v = 6 vértices y ε = v - 1 = 6 - 1 = 5 aristas.

Definición 4 : *Sea G un grafo y usando a ω(G) para denotar el número de componentes de un grafo. Una arista cortada (Cut edge) de G es una arista e tal que ω(G-e) > ω(G)* [BON76].

De la definición 4, ω(G-e) > ω(G) puede ser ejemplificado en la figura 1.7 donde se muestra un grafo con las aristas *a* y *b* (figura 1.7.a); si se quita en la figura 1.7.b la arista *a* entonces ω(G) sigue siendo el mismo número de componentes igual a 1 por lo que *a* no es una arista cortada; sin embargo si se quita la arista b en la figura 1.7.c el número de componentes es ahora de 2, lo que demuestra que *b* es una arista cortada.

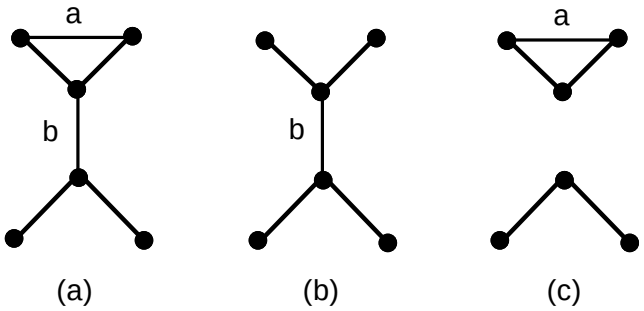

Figura 1.7 (a) ω(G) = 1; (b) ω(G) = 1; (c) ω(G) = 2

Teorema 3 : *Una arista e de G es una arista cortada de G si y solo si e no está contenida en un ciclo de G* [BON76].

El grafo de la figura 1.8 muestra un ejemplo de aristas cortadas usando los conceptos del Teorema 3.

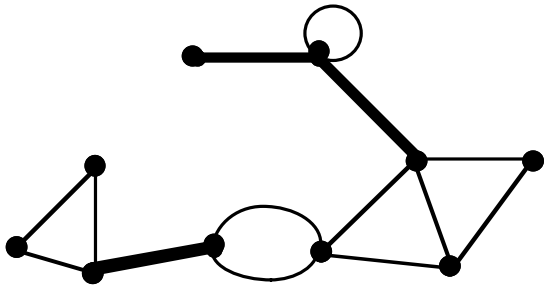

Figura 1.8 Aristas cortadas de un grafo

El siguiente teorema con la ayuda del concepto de aristas cortadas consolida el detectar con firmeza lo que es un árbol :

Teorema 4 : *Un grafo conexo es un árbol si y sólo si cada arista es una arista cortada* [BON76].

Definición 5 : *Un árbol de expansión o bien árbol abarcador, es un árbol que contiene todos los nodos de un grafo y ningún otro más* [AHO74].

1.1.1.1 Grafos Pesados

Un grafo pesado puede ser definido como :

Definición 6 : *Sea G = (V, A) un grafo conexo en donde cada arista (u,v) de A tiene un costo asociado c(u,v). Un árbol abarcador para G es un árbol que conecta todos los vértices de V* [AHO74].

De la definición 6 nace la inquietud de conocer cual es el costo de un árbol abarcador; donde el *costo* es la suma de los costos de las aristas del árbol. Y con ello se cuestiona también si habrá varios costos debido a que hay diferentes combinaciones en los caminos de un árbol abarcador y de entre todos estos costos que exista un costo mínimo [AHO74].

Corolario 1 : *Cada grafo conexo contiene un árbol abarcador* [BON76].

Teorema 5 : *Sea T un árbol abarcador de un grafo G conexo y sea e una arista de G que no esta en T. Entonces T + e contiene un ciclo único* [BON76].

Muchas aplicaciones exigen la identificación de un árbol abarcador para un grafo conexo, pero una aplicación típica de los árboles abarcadores de costo mínimo tienen lugar en el diseño de redes de comunicación, donde los vértices del grafo representan ciudades, y las aristas, las posibles líneas de comunicación entre ellas [AHO74]. Las figuras 1.10 y 1.11 muestran dos de los muchos árboles abarcadores correspondientes al grafo de la figura 1.9. Cada árbol abarcador muestra una manera de planear las rutas de transporte de tal forma que cada ciudad pueda ser atendida [LOO85].

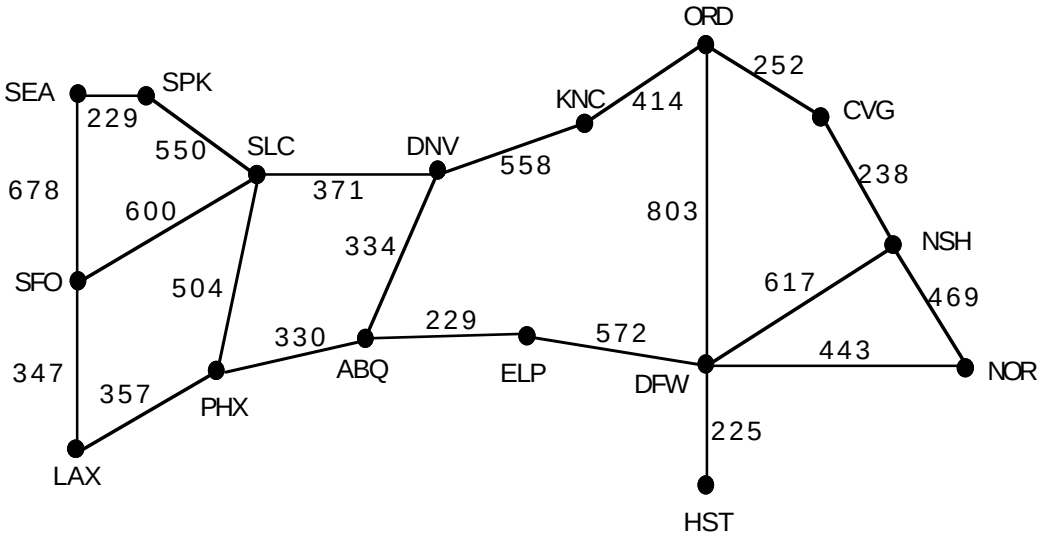

Figura 1.9 Ejemplo de grafo que muestra las distancias entre ciudades

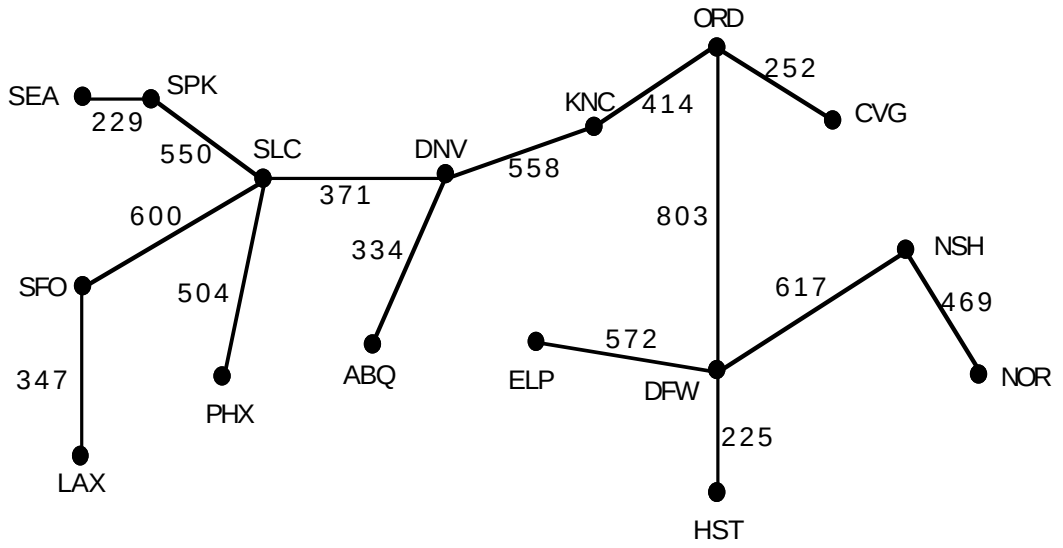

Figura 1.10 Arbol abarcador de la figura 1.9

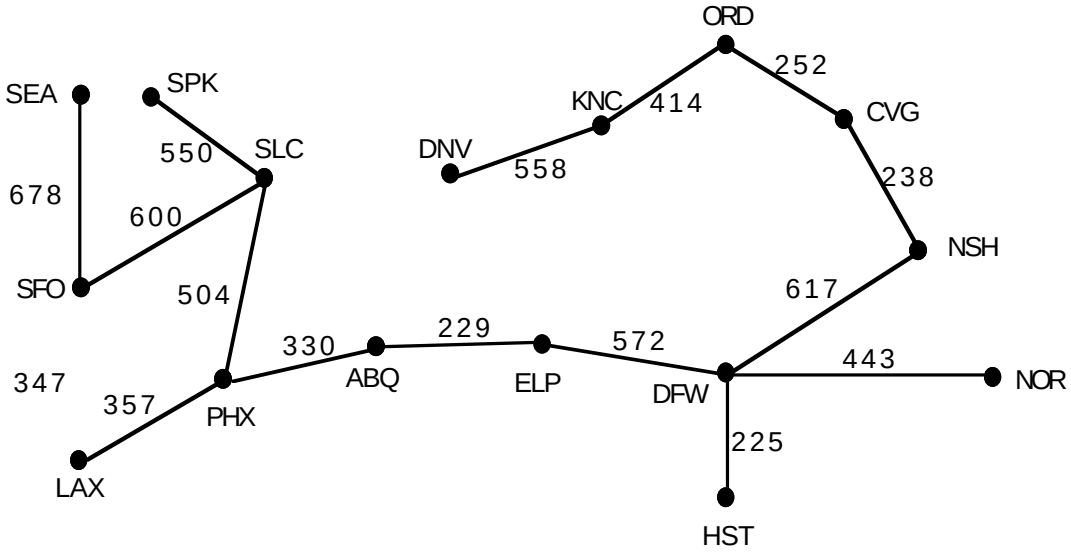

Figura 1.11 Otro árbol abarcador de la figura 1.9

Es de interés primordial identificar un árbol de expansión mínima o árbol abarcador minimal para un grafo. Recordando que el costo de un árbol de expansión es la suma de los pesos de sus aristas. El costo del árbol abarcador de la figura 1.10 es de 6845 millas; el costo del árbol abarcador de la figura 1.11 es de 6567 millas. Ninguno de estos costos parece corresponder al árbol abarcador con el menor costo posible [LOO85].

Se puede calcular el árbol abarcador minimal con un método desarrollado por Kruskal, y el árbol de expansión mínima o árbol abarcador minimal es el que corresponde a la figura 1.12 para el árbol de la figura 1.9, donde se incluyen todos los nodos y el árbol abarcador minimal está completo con un costo de 5479.

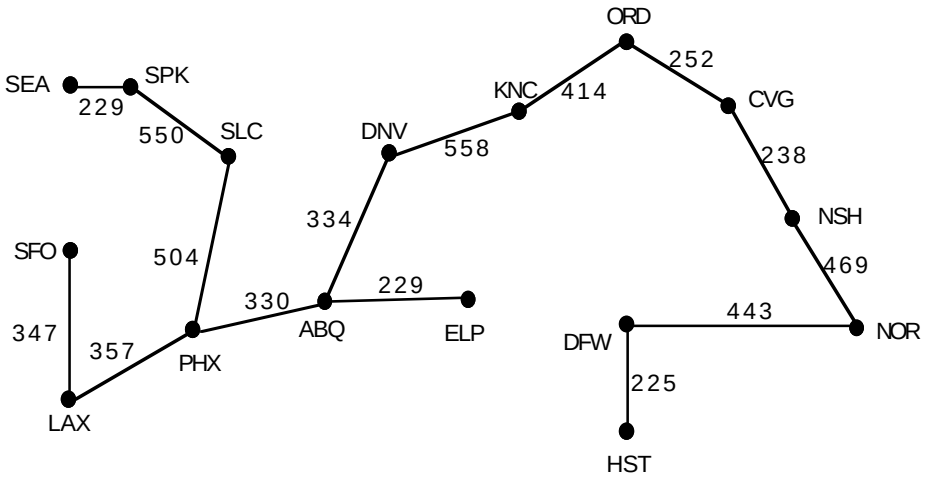

Figura 1.12 Arbol de expansión mínima o árbol abarcador minimal

1.1.2 Grafos dirigidos o Digrafos

Otro caso especial de la estructura de grafos general es *el grafo dirigido o* para abreviar *digrafo*, donde las partes esenciales del digrafo son sus objetos y sus líneas dirigidas. Las líneas dirigidas nacen de asignarle dirección a las aristas del grafo; la figura 1.13 es un ejemplo donde cada arista del grafo dirigido incluye una flecha para indicar la dirección [LOO85]. Puede definirse como :

Definición 7 :*Un grafo dirigido o digrafo es el conjunto finito y no vacío V (de vértices) junto con un conjunto E (disjunto de V) de pares ordenados de distintos elementos de V. En este caso, se refiere a los elementos de E como arcos* [BEH81].

El digrafo D de la figura 1.13.b con V(D) = {u, v, w} y E(D) = { (u, v), (u, w), (w, u)} muestra que (u, v) es un arco de D, mientras que (v, u) no lo es. Pero puede ocurrir en general que para cada arco (u, v) del digrafo D, (v, u) ∈ E(D), entonces los dos arcos (u, v) y (v, u) de D pueden ser denotados juntos como {u, v}, de esta manera un grafo es un *digrafo asimétrico* , pero la teoría de digrafos es esencialmente distinta a la de grafos [BEH81].

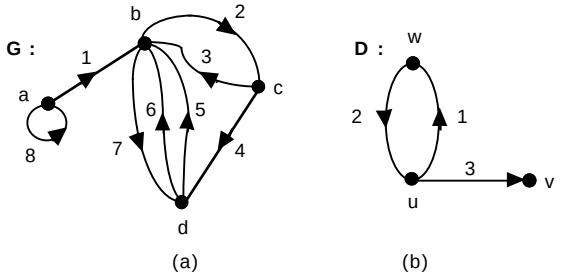

Figura 1.13 Grafo dirigido o digrafo

Otra definición de digrafo importante es la de [KEN90] :

Definición 8 : *Un digrafo G consiste de dos conjuntos, el conjunto no vacío V(G) de vértices de G y el conjunto A(G) de aristas de G, junto con una función γ (gamma minúscula) de A(G) en V(G)* x *V(G). Si e es una arista de G y γ(e) = <p, q>, entonces p se llama vértice inicial de e y q el vértice terminal de e y decimos que e va de p a q* [KEN90].

Esta definición tiene sentido si V(G) o A(G) son infinitos, pero como nuestras aplicaciones son en conjuntos finitos supondremos que A(G) y V(G) son finitos [KEN90].

Un dibujo del digrafo G es un diagrama que consta de puntos correspondientes a los elementos de V(G) y a flechas correspondientes a los elementos de A(G) tales que si γ(e) = <p, q> entonces la flecha correspondiente a e va del punto etiquetado como p al punto etiquetado q [KEN90].

Ejemplo : *Considérese el digrafo G con V(G) ={w, x, y, z}, A(G) = {a, b, c, d, e, f, g, h} y γ dada por la tabla 1.4. Los diagramas de las figuras 1.14.a y 1.14.b son los dos dibujos de G. En la figura 1.14.a se etiquetaron las flechas para una plena correspondencia con A(G) . En la figura 1.14.b simplemente se etiquetaron los puntos y se dejó que las flechas se hagan cargo ellas mismas. Esto no causa confusión porque, en este caso, no hay aristas paralelas, es decir, hay a lo más una arista con un vértice inicial y uno terminal dados. En otras palabras, la función γ es uno a uno. Note también que se omitió la flecha en la arista d porque z es evidentemente vértice inicial y terminal* [KEN90].

e	**γ(e)**
a	<w, z>
b	<w, x>
c	<x, z>
d	<z, z>
e	<z, x>
f	<z, y>
g	<y, w>
h	<y, x>

Tabla 1.4 γ(e) del digrafo 1.14.a

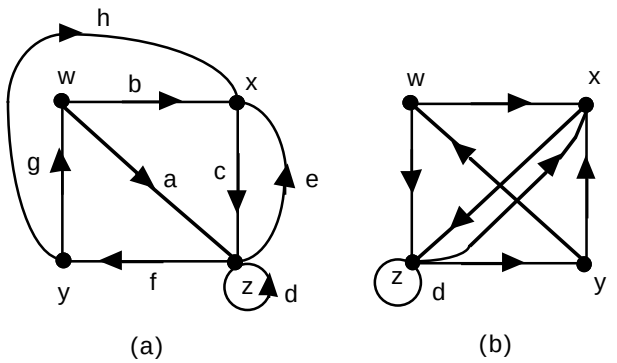

Figura 1.14 Digrafo de la tabla 1.4

Si $\gamma : A(G) \rightarrow V(G) \times V(G)$ es uno a uno, entonces podemos identificar la arista e con sus imágenes $\gamma(e)$ en V(G) x V(G) y considerar a A(G) como un subconjunto de V(G) x V(G). De hecho algunas personas definen a los digrafos para tener $A(G) \subseteq V(G) \times V(G)$ y llaman a los digrafos más generales multigrafos dirigidos como es el caso de la figura 1.13.a que puede ser considerada como un multigrafo dirigido por tener la arista 5 y 6 dirigida hacia b [KEN90].

El *grado interno* de un vértice o también llamada *invalencia* de un nodo en un grafo dirigido es el número de aristas que terminan en ese nodo; el *grado externo* de un nodo o también denominada *exvalencia* de un nodo, es el número de aristas que salen de este nodo. El *grado* de un nodo o bien *valencia* de un nodo es la suma de sus grados internos y externos o bien la suma de su invalencia y exvalencia. Por ejemplo, en la figura 1.13.a los valores son [LOO85] :

grado-interno(a) = 1	grado-externo (a) = 2	grado(a) = 3
grado-interno(b) = 4	grado-externo (b) = 2	grado(b) = 6
grado-interno(c) = 1	grado-externo (c) = 2	grado(c) = 3
grado-interno(d) = 2	grado-externo (d) = 2	grado(d) = 4

Muchos de los problemas importantes que tienen conexión con digrafos pueden enunciarse en términos de sucesiones de aristas que van de un vértice a otro. Un *camino en un digrafo G* es una sucesión de aristas tales que el vértice terminal de una arista es el vértice inicial de la siguiente. Así, si $e_1, e_2, \ldots, e_n$ están en A(G), entonces si $e_1, e_2, \ldots, e_n$ es un camino si hay vértices $x_1, x_2, \ldots, x_n, x_{n+1}$ tales que $\gamma(e_i) = \langle x_i, x_{i+1} \rangle$ para i = 1, 2, ….., n. Se dice para un digrafo que $e_1, e_2, \ldots, e_n$ es un *camino de longitud n* de x_1 a x_{n+1} . Y que el *camino* es *cerrado* si $x_1 = x_{n+1}$ [KEN90].

Un camino cerrado de longitud al menos 1, con sucesión de vértices $x_1, x_2, \ldots, x_n, x_1$ se llama *ciclo* si $x_1, x_2, \ldots, x_n$ son todos diferentes. Aquí lo mismo que en grafos no dirigidos se hace la aclaración de que varios autores utilizan "circuito" y "lazo" para lo que [KEN90] llama ciclo, y "ciclo" se utiliza a veces para un camino cerrado. Un

digrafo sin ciclos se llama acíclico. Un camino es acíclico si el digrafo formado por los vértices y aristas del camino es acíclico [KEN90].

Teorema 6: *Si u y v son vértices de un digrafo G, y si hay un camino en G de u a v, entonces hay un camino acíclico de u a v* [KEN90].

Corolario 3 : *Si hay un camino cerrado de v a v entonces hay un ciclo de v a v* [KEN90].

Corolario 4 : *Un camino es acíclico si y sólo si todos sus vértices son distintos* [KEN90].

Se le llamará *pozo* a un vértice de un digrafo si no es el vértice inicial de una arista. Los pozos corresponden a puntos que no tienen flechas saliendo de ellos.

Lema 1 : *Todo digrafo acíclico finito tiene al menos un pozo* [KEN90].

Los pozos corresponden a puntos de donde no sale ninguna flecha. Se llama a un vértice de un digrafo una *fuente* si no es vértice terminal de ninguna arista.

Lema 2 : *Todo grafo acíclico finito tiene al menos una fuente* [KEN90].

Un digrafo que va de un punto a otro (sin pasar por un punto intermedio) se llama trayecto o camino de longitud 1. Con cada digrafo se puede asociar una matriz llamada *matriz de adyacencia* o de vértices, para determinar cuántos caminos de longitud 1 existen de un punto a otro. Ver la figura 1.15 donde se muestra el digrafo y en la tabla 1.5 su matriz de adyacencia [GER90].

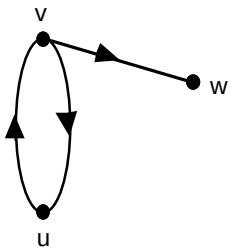

Figura 1.15 digrafo

	u	**v**	**w**
u	0	1	0
v	1	0	1
w	0	0	0

Tabla 1.5 Matriz de adyacencia del digrafo anterior

El número de caminos de longitud 2 de u a w es el elemento del renglón u-ésimo y la columna w-ésima de A^2, el cual es 1. No existen caminos de longitud 2 de w a cualquier otro punto, ya que el renglón w-ésimo de A^2 solamente hay ceros [GER90] .

$$A^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{matrix} \\ u \\ v \\ w \end{matrix} \begin{bmatrix} u & v & w \\ 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

De esta forma puede obtenerse también la matriz de caminos de longitud 3, etc.

1.2 NP-Completitud

Se conocen un sin número de problemas intratables, por ejemplo el problema del agente viajero, el del circuito hamiltoniano (HC del inglés Hamiltonian Circuit), el de la mochila 0-1 (ZOK del inglés Zero-One Knapsack,) el problema de satisfactibilidad, etc. De otra manera cada uno de estos problemas, al menos en su versión de problema de decisión, tiene un algoritmo simple no-determinístico en tiempo polinomial para su solución. Por ejemplo, en el problema ZOK con n objetos solamente n suposiciones son necesarias - una suposición acerca de si o no incluir cada objeto. El problema PAV y el problema HC involucran seleccionar una permutación de los vértices, así se requiere solo seleccionar el k-ésimo elemento como k rangos de 1 a n. Una solución al problema de satisfactibilidad booleana requiere solamente suponer una tarea para cada variable [SMI89].

Note que una vez que las suposiciones han sido hechas para el HC, se puede checar fácilmente que todos los vértices seleccionados sean distintos, sea o no cada vértice adyacente al primero. De hecho, esta verificación puede ser realizada en tiempo O(n) si el grafo es representado como una lista adyacente. Consecuentemente, el problema del circuito hamiltoniano tiene un algoritmo no-determinístico en tiempo polinomial para su solución [SMI89].

Análogamente puede verse que el PAV puede resolverse por medio de un algoritmo polinomial no determinístico. Similarmente para ZOK, se puede verificar que la capacidad de la mochila no sea excedida y compara la ganancia total con la ganancia objetivo.

Definición 9 : *Un problema de decisión está en la clase P si y sólo si es resuelto por un algoritmo determinístico de complejidad en tiempo polinomial. Un problema de decisión esta en el conjunto NP si y sólo si este es resuelto por un algoritmo no-determinístico de complejidad en tiempo polinomial* [SMI89].

¡Nótese que!, "NP" es para "polinomios no-determinísticos" (del inglés Nondeterministic Polynomial) y no es "no-polinomial" (del inglés NonPolynomial). También debe notarse que cualquier problema en P está automáticamente en NP, ya que se puede considerar que un algoritmo determinístico es un algoritmo no-determinístico con una entrada de suposiciones vacía. Así $P \subseteq NP$ [SMI89].

Intuitivamente se piensa que P es la clase de problemas buenos, a pesar de que una complejidad en tiempo de por ejemplo $O(n^{1000})$ puede no parecer muy bueno. En la práctica, sin embargo, problemas interesantes en P tienden a tener exponentes de orden muy bajo.

Se ha visto que ZOK, PAV, HC, y satisfactibilidad booleana están en NP, a pesar de que ninguno se sepa que está en P. De hecho, no es conocido que haya algún problema que este en NP pero que no este en P. Hay muchos problemas conocidos que están en NP, pero no se sabe que estén en P. La pregunta de si P = NP es quizás el problema sin resolver de la ciencia de la computación. En el resto de esta sección se tratará de explicar porque es tan importante esta pregunta.

El beneficio obvio de análisis de algoritmos en general es que se pueden encontrar buenos algoritmos para resolver algunos problemas. Un aún mejor resultado debería ser, encontrar una clase de buenos algoritmos. Para resolver un gran número de problemas de esta clase. Un menos exitoso pero aún valioso resultado debería ser el encontrar que algoritmos no tan buenos sean posibles para uno o mas problemas. Recalcando que se está pensando en que P es la clase de problemas buenos (por ejemplo problemas con una buena solución). Una estrategia para obtener resultados como los previamente sugeridos debería ser encontrar una clase de problemas conocidos que estén en P y para los cuales la membresia en la clase sea fácil de determinar. La clase NP califica en ambos tipos de problemas. Los problemas fáciles en NP, entre estos los que no es conocido que estén en P, podrían ser candidatos para problemas que pueden ser presentados ser no muy buenos [SMI89].

Esta estrategia asume que se tiene un camino de decir si un problema es mas fácil o mas difícil que otro. Un camino simple es decir que un problema es mas fácil que otro si una solución al segundo problema da una solución al primero. En otras palabras se dice que un problema es mas fácil que otro si el segundo problema es mas general que el primero, o si el primer problema reduce al segundo. Se debe ser muy cuidadoso en este proceso de reducción. El algoritmo de reducción no debe ser tan complicado que sea tan difícil como uno de los problemas a reducir. Por ejemplo, se desea decir que la multiplicación es más fácil que la suma porque se puede calcular a x b como $((a+b)^2 - a^2 - b^2) / 2$. Aquí el algoritmo de reducción está involucrando elevar al cuadrado, división entre 2 y restas.

Afortunadamente se dispone de una prueba fácil de simplificar la reducción de un algoritmo. Se asume que la reducción del algoritmo debe estar en P. Ya que se esta tratando solamente con problemas de decisión, la complejidad de la reducción del algoritmo puede lidiar solamente en transformar la instancia del problema fácil dentro de una instancia del problema difícil. La salida del problema difícil es simplemente "si" o "no" , esta respuesta no debe de ser interpretada. Por lo que se requiere de la siguiente definición [SMI89].

Definición 10 : *Un problema A es reducible polinomialmente a un problema B (notación*

A α B) si hay un algoritmo en tiempo polinomial que transforme instancias de A en instancias de B en tal forma que las instancias de A y las instancias transformadas siempre tengan la misma solución [SMI89].

No es difícil ver que si A α B y si B α C, entonces A α C. En otras palabras, la relación α es transitiva.

1.3 Problemas de Decisión

Un problema de decisión es aquel problema que pregunta si la respuesta es "si" o "no", por ejemplo ¿Puede este grafo ser 5-coloreado?. ¿Hay un camino de longitud < 15 millas? ¿Hay un conjunto de 67 vértices independientes? [WIL86].

Muchos de los problemas que estudiamos pueden ser descritos como un problema de decisión o como problemas de optimización; por ejemplo, ¿Cuál es el número más pequeño de colores con el cual G puede ser coloreado?, ¿Cuál es la longitud del camino mas corto de estas ciudades?, ¿Cuál es el tamaño del conjunto independiente mas largo de vértices en G? [WIL86].

La teoría de NP-completitud es un intento, con un éxito parcial, para proveer que ciertos problemas son intrínsecamente difíciles de resolver. Por lo que se dará una breve, e informal introducción a esta área teórica tan importante de la ciencia de la computación. Así, un problema de decisión es especificado por describir una instancia típica del problema, y haciendo la pregunta de ¿cuál es… ? y ser contestada con un es "si" o "no". Cualquier problema de optimización causa en forma natural una relación con un problema de decisión. Para un problema de minimización, se puede preguntar ¿es la solución óptima ≤ a un valor X especificado ?. El valor de X debe ser especificado como parte de las instancias del problema de decisión. Para un problema de maximización, se debe reemplazar el "≤" por "≥". Como un ejemplo, se describirá un problema de decisión basado en la mochila 0-1 [STI87].

Problema de decisión : Mochila 0-1 [STI87].

Instancia : Una lista de ganancias $P = (p_1, …., p_n)$, una lista de pesos $W = (w_1, …., w_n)$, una capacidad M, y una ganancia hipotética Q.

Pregunta : ¿Es la ganancia óptima ≥ Q ?

Un algoritmo para "resolver" el problema de decisión de la mochila 0-1 es requerido solamente para contestar "si" o "no". No se requiere exhibir una mochila particular la cual tiene una ganancia ≥ Q. Si se escribe un programa de computo el cual regresa la respuesta correcta "si" o "no" en un tiempo finito, se dice que se tiene un algoritmo determinístico para resolver un problema de decisión. Parece ser restrictivo considerar solamente problemas de decisión, pero puede haber otro tipo de problemas [STI87].

1.4 Problemas de Optimización

Usualmente, si se encuentra un algoritmo rápido para un problema de decisión entonces con un poco más de trabajo se estará capacitado para resolver el correspondiente problema de optimización. Por ejemplo suponga que se tiene un algoritmo que resuelve el problema de decisión para colorear un grafo, y que se desea que esta sea la solución del problema de optimización (el número cromático) [WIL86].

Sea un grafo G dado, de 100 vértices

Pregunta : ¿Puede el grafo ser 50-coloreado? De esta forma el número cromático se encuentra entre 1 y 50. Entonces se pregunta si puede ser coloreado en 25 colores. Si no, entonces el número cromático esta entre 26 y 50.

Continuando en esta forma, usando bisección de los intervalos que están conocidos para contener el número cromático. Después de O(log n) pasos se ha encontrado el número cromático de un grafo de n vértices. El factor extra multiplicativo de log n no alterará la corrida en tiempo polinomial contra no-polinomial. Ya que si hay un camino rápido para hacer la decisión del problema entonces hay un camino rápido para hacer la optimización del problema, la convergencia es entonces obvia [WIL86].

1.4.1 Optimización local

La búsqueda local u optimización local, es una de las formas primitivas de aplicar optimización continua en un espacio de búsqueda discreto. Esta fue una de las primeras técnicas propuestas para enfrentarse a la intratabilidad computacional de muchos de los problemas de optimización [GUJ94].

Algunas veces, la siguiente estrategia producirá una solución óptima para un problema [AHO74].

1) Empezar con una solución aleatoria.
2) Aplicar a la solución actual una transformación de un conjunto dado de transformaciones para mejorar la solución, de manera que la mejora es la nueva solución "actual".
3) Repetir hasta que ninguna transformación del conjunto pueda mejorar la solución actual.

La solución resultante puede ser óptima o no. En principio, si "el conjunto de transformaciones dado" incluye todas las que toman una solución y la reemplazan por otra, entonces no se parará hasta alcanzar la solución óptima. Pero, en ese caso, el tiempo para aplicar (2) es el mismo que el necesario para examinar todas las soluciones, y todo el enfoque no tiene mucho sentido. El método tiene sentido cuando se puede restringir el conjunto de transformaciones a un conjunto pequeño, de modo que en poco tiempo se puedan considerar todas las transformaciones posibles tal vez deben permitirse las $O(n^2)$ u $O(n^3)$ transformaciones cuando el problema es de "tamaño" n. Si el conjunto de transformaciones es pequeño, es natural considerar las soluciones, que pueden ser

transformadas entre sí en un paso como "cercanas". Las transformaciones se llaman "transformaciones locales", y el método se conoce como búsqueda local [AHO74].

1.4.1.1 Algoritmos de aproximación por búsqueda local

Los algoritmos de búsqueda local han tenido gran efectividad como métodos heurísticos para la solución de problemas cuyas soluciones exactas requieren tiempo exponencial. Un método común de búsqueda es empezar con un número de soluciones aleatorias y aplicar las transformaciones locales a cada una, hasta alcanzar una solución localmente óptima, que ninguna transformación pueda mejorar. Con frecuencia se alcanzarán diferentes soluciones localmente óptimas, a partir de la mayor parte o de todas las soluciones iniciales aleatorias, como se sugiere en la figura 1.16. Si se tiene suerte, una de ellas será globalmente óptima, esto es, tan buena como cualquiera otra solución [AHO74].

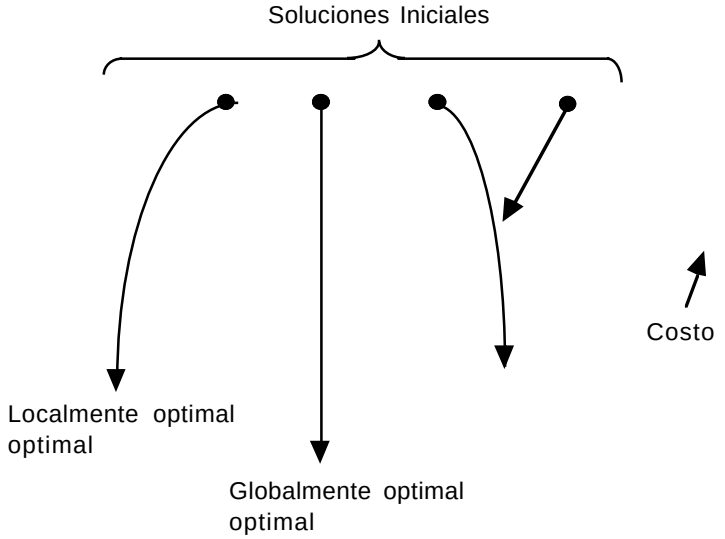

Figura 1.16 Búsqueda local en el espacio de soluciones

En la práctica, tal vez no se pueda encontrar una solución globalmente óptima como la que se sugiere en la figura 1.16, ya que el número de soluciones localmente óptimas puede ser enorme. Sin embargo, es posible escoger por lo menos aquella solución localmente óptima que tenga el menor costo entre todas las que se encontraron [AHO74].

1.4.1.2 Algoritmos de Aproximación

Dado un problema de optimización donde se sabe que se trata de un problema NP-Completo por ejemplo el problema de la mochila 0-1 o el problema del agente viajero, a menudo es posible encontrar un algoritmo en tiempo polinomial cuya solución es aproximadamente óptima. De hecho, algunas veces hay una familia entera de algoritmos de aproximación para un problema dado, en el cual las mejores aproximaciones requieren más tiempo.

Se está consciente del error cometido por los algoritmos de aproximación solamente como una fracción de la solución óptima. Si la solución óptima es grande, un error absoluto grande puede ser tolerado tan grande como el error relativo es pequeño. Para poder manejar problemas de maximización y minimización en el mismo formulismo, se da la siguiente definición [SMI89].

Definición 11 : Un algoritmo A es un algoritmo de aproximación-ε para un problema P si y solo si hay un número ε tal que para cada instancia I de P, la solución aproximada $S_A(I)$ dada por A está relacionada para la solución exacta en $S_P(I)$ por :

$$| (S_P(I) - S_A(I) / S_P(I)) | < \varepsilon$$

Este es un camino formal de decir que el error relativo siempre está acotado por ε.