El problema de Asignación de Tareas en los Sistemas Distribuidos

Jose Lisandro Aguilar Castro
Dpto. de Computación, Fac. de Ingeniería, Universidad de los Andes
Av. Tulio Febres Cordero. 5101. Mérida. Venezuela
Telf: (58.74) 440002 Fax: (58.74) 402872
email: aguilar©ing.ula.ve

RESUMEN.

En los Sistemas Distribuidos, la asignación de tareas es uno de los problemas mas importantes a considerar. En este articulo, nosotros definimos varias funciones de costo a partir de las cuales se pueden obtener asignaciones optimas de las tareas. Las funciones de costo toman en consideración el costo de comunicación entre los procesadores, los tiempos de ejecución de las tareas, el costo debido a la interferencia entre tareas paralelas al residir en un mismo sitio, el costo por repartir la carga de trabajo y el costo debido al acceso a datos guardados en otros sitios. Finalmente, presentamos dos algoritmos de resolución, uno basado en el Temple Simulado y el otro en los Algoritmos Genéticos, y presentamos ejemplos de utilización de esas funciones usando esos algoritmos.

PALABRAS CLAVES: Asignación de Tareas, Sistemas Distribuidos, Evaluación de Rendimiento, Algoritmos Genéticos, Temple Simulado.

1. Introducción

La posibilidad de realizar procesamiento paralelo sobre Sistemas Distribuidos, involucra la resolución de nuevos problemas antes de poder explotar esa potencialidad [1]. Uno de los principales problemas es la degradación del rendimiento. En un ambiente ideal el rendimiento debería crecer linealmente, al crecer el número de procesadores. Esto no sucede en la realidad, ya que a partir de un cierto número de procesadores los excesivos intercambios de mensajes de control y de transferencia de datos entre los procesadores, provocan un efecto de saturación en el sistema, lo que genera automáticamente una degradación del rendimiento. Parece entonces evidente la relación entre la optimización del rendimiento y la minimización de los intercambios entre los procesadores. Existen otros aspectos que juegan un papel importante para optimizar el rendimiento, tales como el equilibrio de la carga de trabajo. En este articulo, nosotros estudiamos los diferentes factores ligados al problema de asignación de tareas, para después definir una función de costo, y proponemos dos algoritmos para resolver este problema, basados en los Algoritmos Genéticos y en el Temple Simulado.

El resto del articulo esta estructurado de la siguiente manera: una definición del problema de asignación de tareas es presentada en la sección 2. Además, son definidos los diferentes criterios de optimización a considerar durante la asignación de tareas. En la sección 3, presentamos dos algoritmos para resolver el problema de asignación de tareas, uno basado en los Algoritmos Genéticos y el otro en el Temple Simulado. Además, es presentado un estudio sobre el uso de los diferentes criterios.

2. Presentación del Problema

La asignación de un programa compuesto por n tareas sobre una arquitectura distribuida compuesta por P procesadores, es equivalente a la proyección F: T->Π según un objetivo predefinido, donde T es el conjunto de tareas y Π es el conjunto de procesadores [1, 3, 5, 6]. Existen P^n posibles asignaciones. Para realizar esta selección varios criterios pueden ser utilizados, tales como equilibrar la carga de trabajo entre los procesadores, optimizar el grado de paralelismo, minimizar el costos de comunicación entre los procesadores, minimizar el costo de ejecución de las tareas, etc. La asignación optima debe buscar un buen compromiso entre estos criterios [7, 8, 11]. Las arquitecturas distribuidas donde este problema es crucial, son las arquitecturas del tipo MIMD con memoria distribuida.

La búsqueda de una asignación optima es extremadamente compleja (NP-completo) [1, 3, 5, 6]. Esta complejidad esta ligada a las características del sistema y de los programas. Si utilizamos una función de costo, la asignación optima será la asignación que minimice dicha función. En esta sección, nosotros describimos los diferentes costos que intervienen en el problema de asignación de tareas. Los costos describen las características de cada uno de los elementos del sistema, así como de las tareas que componen los programas. Estos costos constituyen los criterios que deben ser optimizados por la asignación final y determinan el rendimiento del sistema. Nosotros suponemos un sistema con las siguientes características:

- No hay conexión total entre los procesadores,
- Los procesadores pueden ser heterogéneos,
- Los enlaces de comunicación son fijos,

Costo de Ejecución: Este costo depende de la complejidad de la tarea a ejecutar y de la capacidad de calculo del procesador donde la tarea reside. El costo total de ejecución de un programa es definido, como la suma de los costos de ejecución de cada tarea del programa sobre los procesadores del sistema donde residen. Este costo es definido como:

$$C_E = \Sigma_p \Sigma_i e_i U_p X_{ip} \quad (1)$$

donde,

e_i es la complejidad de la tarea *i*. En nuestro caso, nosotros lo definimos como el número de instrucciones de la tarea *i*. Además, suponemos el mismo grado de complejidad para todas las instrucciones.

U_p es el costo por usar el procesador *p*. Lo definimos como el tiempo promedio de ejecución de una instrucción en el procesador *p*.

X_{ip} es una variable de estado la cual es igual a 1 si la tarea *i* reside en el procesador *p*, de lo contrario es 0.

Costo de Comunicación: Este costo es considerado como el criterio mas importante a optimizar en un Sistema Distribuido [1, 5, 6]. El depende de la cantidad de información a intercambiar entre las tareas, de la topología del sistema y de las características de los canales de comunicación. Este costo existe a partir del momento en que dos tareas que se encuentran en procesadores diferentes, desean comunicarse. Nosotros lo definimos como:

$$C_C = \Sigma_p \Sigma_{q \neq p} \Sigma_i \Sigma_{j \& j \neq i} D_{pq} C_{ij} X_{ip} X_{jq} \quad (2)$$

C_{ij} es la cantidad de información a transferir entre las tareas *i* y *j*.

D_{pq} es el costo por usar los canales de comunicación. Lo definimos como el tiempo promedio para transmitir una información, desde el procesador *p* hasta el procesador *q*.

$$D_{pq} = CCP_{piqj} X_{ip} X_{jq} + CCP_{qjpi} X_{ip} X_{jq} + r_{pq}$$

CCP_{piqj} (respectivamente CCP_{qjpi}) es el costo por usar el procesador *p* (respectivamente *q*) para realizar la comunicación entre las tareas *i* y *j*, a sabiendas que *i* reside en *p* y *j* en *q*. Matemáticamente, puede ser expresado como:

$$CCP_{piqj} = Pr_{piqj} + AT_{pi}$$

Pr_{piqj} es el tiempo promedio que el procesador *p* necesita para preparar el ambiente de comunicación (negociación del protocolo de comunicación, preparación de los datos a transmitir, etc.).

AT_{pi} es el tiempo promedio que debe esperar la tarea *i* en el procesador *p*, antes de poder usar los canales de comunicación. Depende del trafico en los canales de comunicación.

r_{pq} es el tiempo promedio que se dura en los enlaces de comunicación que conectan a *p* y *q*. Depende de la velocidad de los enlaces (canales). Este costo debe reflejar el problema de enrutamiento de la información. Es definido como:

$$r_{pq} = \min_{s \in ch_{pq}} \left(\Sigma_{o,g \in s} \, 1/VT_{og} + \Sigma_{o \in s-(p,q)} AT_{oi} \right) \qquad s = \{ p, ..., o, g, ..., q \},$$

donde,

ch_{pq} es el conjunto de todos los caminos posibles entre *p* y *q*,

s es el conjunto de los procesadores por donde se debe pasar para enlazar a *p* y *q* , en el camino escogido,

VT_{og} es la velocidad del canal de comunicación que une directamente a dos procesadores *o* y *g*.

Los detalles de la formula propuesta provienen del protocolo de comunicación usado, así como de la topología de interconexión del sistema. Esta función puede ser definida de una manera mas simple:

$$C_C = \Sigma_p \Sigma_{q \neq p} \Sigma_i \Sigma_{j \& j \neq i} C_{ij} X_{ip} X_{jq}$$

Costo de Referencia a un Archivo: Es el costo que se origina cuando una tarea *i* referencia a datos contenidos en un archivo *f*, el cual reside en un procesador diferente a donde reside la tarea *i*. Este costo esta íntimamente ligado al problema de asignación de datos [3, 6]. Nosotros lo definimos como:

$$C_F = \Sigma_p \Sigma_{q \neq p} \Sigma_i \Sigma_f D_{pq} V_{if} X_{ip} Y_{fq} \qquad (3)$$

V_{if} es la cantidad de datos promedio referenciados por la tarea *i*, del archivo *f*. Depende del tamaño del archivo, del tipo de organización (secuencial, etc.) y del tipo de acceso (directo, etc.).

Y_{fq} es una variable de estado igual a 1 si el archivo *f* reside en el procesador *q*, de lo contrario es 0.

Costo de Interferencia: Es el costo en que se incurre cuando dos o mas tareas, las cuales deben ser ejecutadas en paralelo, residen en el mismo procesador por lo que deben compartir los recursos. También es conocido como el costo por perdida de paralelismo. Puede ser atribuido a dos factores:

- La competencia por usar los componentes del procesador: CPU, memoria, etc. (Ie)
- La competencia por usar los servicios de comunicación del procesador (Ic).

$$C_I = \Sigma_p \Sigma_i \Sigma_{j \neq i} (I_{pij}/2)\, X_{ip} X_{jp} \quad (4)$$

donde, I_{pij} es el costo de interferencia entre las tareas *i* y *j*, si residen en el procesador *p*.

$$I_{pij} = I^e_{pij} + I^c_{pij}$$

$$I^e_{pij} = (e_i + e_j)U_p$$

$$I^c_{pij} = \Sigma_{q \neq p} \Sigma_{\iota \neq i \& \iota \neq j} [CCP_{piqt} C_{it} + CCP_{pjqt} C_{jt}] X_{tq}$$

Costo por el Desequilibrio de la Carga de Trabajo: Una buena asignación debe asegurar el uso mínimo, pero a su misma vez equitativo, de los recursos del sistema [3, 5, 7, 11]. Como una buena asignación debe considerar los dos objetivos, nosotros debemos definir un costo que asegure una distribución equilibrada de la carga de trabajo entre los procesadores del sistema. Este costo depende de la complejidad de las tareas a ejecutarse y de las capacidades de procesamiento de los procesadores. Nosotros lo definimos como la varianza de la carga de trabajo entre los procesadores.

$$C_B = \Sigma_p ([\Sigma_i e_i U_p X_{ip}] - [\Sigma_p \Sigma_i e_i U_p X_{ip}] / P)^2 / P \quad (5)$$

3. Diferentes Técnicas de Resolución

El problema de asignación de tareas, por ser NP-completo, es difícil de resolver usando métodos exactos [1, 3, 5]. Los métodos exactos permiten siempre encontrar una solución optima, pero con tiempos de calculo muy grandes. Para parámetros grandes de este problema (por ejemplo, para n=17 o P>2) la resolución por estos tipos de métodos se hace imposible [1]. Así, los estudios realizados hasta ahora han sido orientados hacia estrategias que permiten obtener buenas asignaciones (soluciones suboptimas), a veces optimas, en un tiempo de calculo reducido. Estos métodos son conocidos como métodos heurísticos o aproximativos.

En esta sección, presentamos dos ejemplos de aplicación de métodos heurísticos en la resolución del problema de asignación de tareas. Estos métodos son el Temple Simulado y los Algoritmos Genéticos. Suponemos un sistema a bus compartido con *K* procesadores homogéneos. Las tareas son idénticas a nivel de tiempo de ejecución y de comunicación. Igualmente, el costo de acceso a archivos distante es el mismo para todas las tareas.

3.1 Temple Simulado

Este método utiliza los conceptos físicos de *temperatura* y de *energía* para representar y resolver el problema de optimización [3, 13]. La función de costo del problema constituye la energía del sistema, mientras que la temperatura es introducida para hacer una búsqueda aleatoria de la solución.

El método parte de una solución inicial completa, donde la escogencia puede ser aleatoria, e intenta mejorarla a través de cambios locales. Si la solución obtenida es mejor que la precedente, ella es aceptada, sino es aceptada según cierta probabilidad. La operación es repetida hasta un valor mínimo de temperatura. Cuando se llegue al valor minimo, se ha llegado a un suboptimo local. Para mas detalles del metodo ver [3].

3.2 Algoritmos Genéticos:

Este método esta basado en los principios de la biología evolutiva [3, 14]. El método sigue un proceso de evolución inteligente utilizando operadores evolutivos (mutación, cruzamiento, etc.). La idea general consiste en buscar un optimo local partiendo de un conjunto de soluciones. Para esto, uno aplica sucesivamente los operadores evolutivos sobre las soluciones iniciales e intermediarias, de manera a generar nuevas y mejores soluciones.

En nuestro caso, uno define un espacio de soluciones que esta compuesto por n individuos. Cada individuo es un vector con n elementos. Cada elemento representa una tarea del programa y tiene un valor entre 1 y K para indicar a cual procesador ella esta asignada. Para cada individuo, uno calcula su costo usando la función de costo del problema. La población inicial es definida aleatoriamente. Como la población es constante, los peores individuos son reemplazados por los mejores individuos en cada generación. El procedimiento es detenido cuando uno llega a un número predefinido de generaciones, o cuando la solución obtenida no puede ser mejorada. Nosotros, en este trabajo utilizamos dos operadores genéticos, el operador de cruzamiento que permite la combinación de dos individuos y el operador de mutación que cambia aleatoriamente una parte de un individuo. Para mas detalles del metodo ver [3]. El algoritmo general es:

```
Generar la población inicial
Repetir hasta la convergencia del sistema
        Evaluar a cada individuo
        Seleccionar los mejores individuos para reproducir
        Reproducir los nuevos individuos usando los operadores evolutivos
        Sustituir los peores individuos por los mejores nuevos individuos
```

3.3 Análisis de los Resultados

Usamos grafos serie-paralelos para representar los programas paralelos, los cuales son definidos por el número de tareas (n) y el número máximo de sucesores por tarea (d). El número de sucesores para cada tarea es uniformemente generado según el intervalo [1, d]. La ejecución es medida en segundos. Nuestro principal objetivo consiste en observar la manera como son asignadas las tareas según diferentes funciones de costo y según las dos técnicas de asignación descritas previamente. Además, mientras sea posible las comparamos con las soluciones exactas. Las diferentes funciones de costo utilizadas son:

1. $F_1 = C_C + C_B$
2. $F_2 = C_C + C_B + C_E$
3. $F_3 = C_C + C_B + C_I$
4. $F_4 = C_C + C_B + C_E + C_I + C_F$
5. $F_5 = C_C + C_F$

La búsqueda de la asignación optima para cada una de las funciones de costo, usando el Temple Simulado (TS) y los Algoritmos Genéticos (AG), nos a permitido constatar que la escogencia del algoritmo de optimización no debe ser arbitraria, sino que depende de la función de costo a optimizar. Así, los mejores resultados para las funciones F_1, F_2 y F_3 son obtenidos con el Temple Simulado. La Figura 1 muestra los costos obtenidos con la función F_1 para programas con 50 o menos tareas y para 5 procesadores. El comportamiento de las curvas es igual para las otras dos funciones F_2 y F_3.

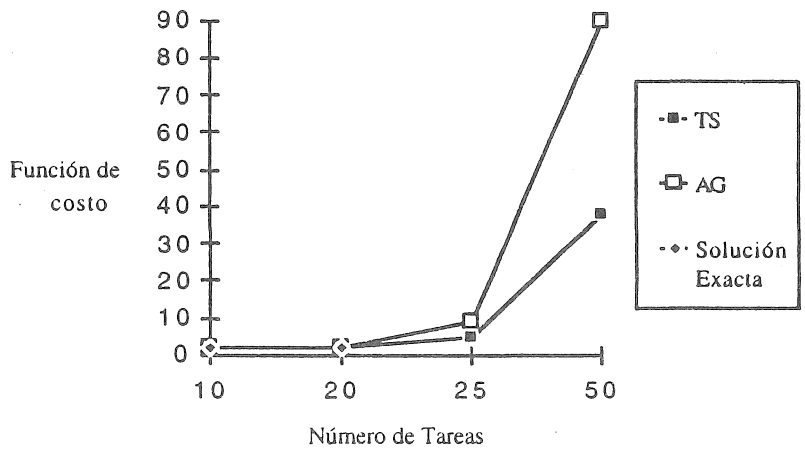

Figura 1 Optimización de la Función de Costo F1

En cuanto a la optimización de la función de costo F_4, la escogencia del método no tiene ningún impacto sobre los resultados obtenidos (ver Figura 2). El comportamiento de las dos curvas es casi idéntica.

Los Algoritmos Genéticos son mas interesantes para la función de costo F_5, ya que la calidad de los resultados obtenidos con este método es mejor que la del Temple Simulado (Figura 3). En general, para grafos pequeños ($n > 20$ y $K > 5$) la diferencia entre las soluciones exactas y las soluciones de los métodos heurísticos no es grande. Igualmente, los tiempos de ejecución son similares.

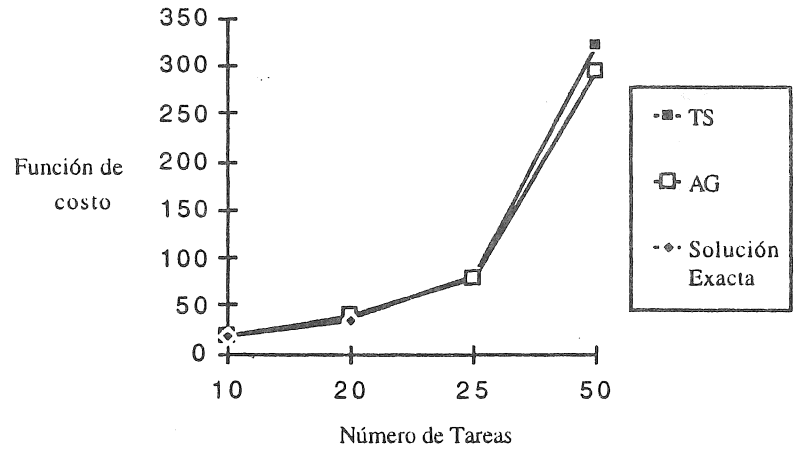

Figura 2. Optimización de la función de costo F4

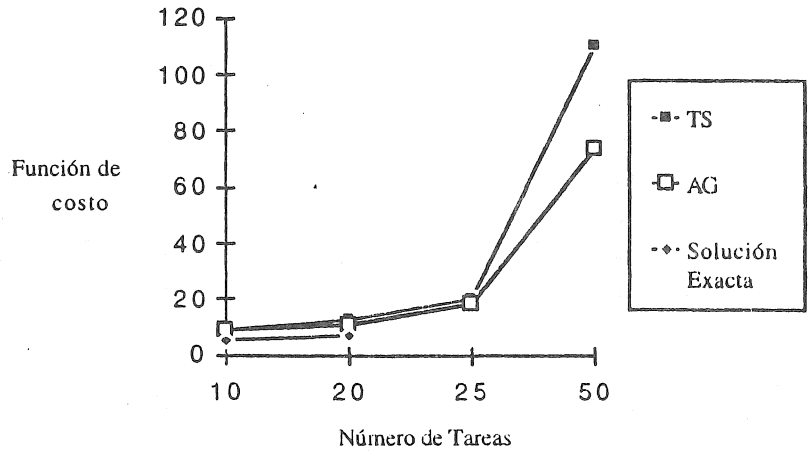

Figura 3. Optimización de la Función de Costo F5

Las asignaciones obtenidas usando las funciones de costo F_1 y F_5 son completamente diferentes (ver Tabla 1). Estos resultados traducen la contradicción total que existe entre los costos C_B y C_C, si consideramos que el comportamiento del costo de referencia a los archivos C_F es similar al costo de comunicación C_C.

Tabla 1. Asignación de tareas usando las funciones de costo F_1, F_5 y los Algoritmos Genéticos

Funciones \ Procesadores	1	2	3	4	5
F1	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F5	1,2,3,5,7	4		9,15,16,17	6,8,10,11 12,13,14

Con las funciones de costo F_1, F_2 y F_3 han sido obtenidas asignaciones de tareas similares (Tabla 2). Por otro lado, los costos de ejecución (C_E) y de interferencia (C_I) no son contradictorios con los costos C_C y C_B. Ciertamente, C_I trata de repartir las tareas que deben ser ejecutadas en paralelo entre los procesadores del sistema, por lo que no habrá comunicación entre ellas.

Tabla 2. Asignación de las tareas usando las funciones de costo F_1, F_2, F_3 y los Algoritmos Genéticos

Funciones \ Procesadores	1	2	3	4	5
F1	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F2	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8
F3	7,12	6,13,15,17	3,10,11,14,16	4,9	1,2,5,8

4. Conclusiones

La asignación de tareas sobre un sistema tiene una influencia directa sobre su rendimiento. En este articulo, ha sido definido una función de costo para el problema de asignación de tareas. La función de costo describe de manera detallada las características mas importantes de estos sistemas, tales como la comunicación (velocidad de los enlaces, etc.), la ejecución de los procesos (costo por el uso de los procesadores, etc.), etc. Nosotros hemos extendido el modelo de manera a incluir factores que uno no encuentra frecuentemente en la literatura, tales como la carga sobre los enlaces o la referencia a datos distantes.

Este problema es NP-completo a nivel de la complejidad de resolución, por consecuencia solo los métodos aproximativos permiten obtener soluciones suboptimas, a veces optimas, en tiempos de ejecución aceptables. Para escoger la técnica de resolución, es necesario llegar a un compromiso entre la calidad de los resultados que se quieren y los tiempos de respuesta deseados para obtenerlos. La búsqueda de la mejor técnica para el problema de asignación de tareas es un problema abierto. En este trabajo, nosotros hemos encontrados soluciones buenas, a veces optimas, en tiempos de ejecución cortos.

El problema de asignación de tareas no puede ser aislado de los problemas de descomposición de programas, de asignación de archivos, de tolerancia a fallas, de migración de procesos, de replica de tareas, etc. Los futuros métodos de optimización en los Sistemas Distribuidos deberán considerar todos estos problemas juntos.

Bibliografía

[1] Billonnet M., Costa M. y Sutter A.: «Les problèmes de placement dans les Systèmes Distribués». Technique et Science Informatique, Vol 8, No. 4 (1989), 307-337.

[2] Agrawall R. y Jaladish H.: «Partitioning techniques for Large-Grained Parallelism». IEEE Trans. on Computers, Vol 37, No. 12 (1988), 1627-1633.

[3] Aguilar J.: «L'allocation de tâches, l'équilibrage de la charge et l'optimisation combinatoire». Tesis de Ph.D de la Universidad Rene Descartes, Paris, 1995.

[4] Aguilar J.: «Parallel Programs: Task Graph generation and Task Allocation». Actas Científicas de la 10th International Conference on Mathematical and Computer Modelling and Scientific Computing, Boston, 1995.

[5] Lo V.: «Heuristic Algorithms for Task Assignment in Distributed Systems». IEEE Trans. on Computers, Vol 37, No. 11 (1988), 1384-1397.

[6] Muntean T. y Talbi G.: «Méthodes de placement statique des processus sur architectures parallèles». Technique et Science Informatique, Vol. 10, No. 5 (1991), 355-373.

[7] Aguilar J.: Heuristic algorithms for Task Assignment of Parallel Programs. En: L. Dekker (ed), Massively Processing Applications and Development, Elsevier Science, Holland (1994), 146-153.

[8] Andre F. y Pazat J.: «Le placement de tâches sur des architectures parallèles», Technique et Science Informatique, Vol 7, No. 4 (1988), 385-401.

[9] Bollinger W. y Midkiff S.: «Heuristic Technique for Processor and Link Assignment in Multicomputers». IEEE Trans. on Computers, Vol 40, No. 3 (1991), 325-333.

[10] Bowen N., Nikolaou C. y Ghafoor A.: «On the Assignment Problem of Arbytrary Process Systems to heterogeneous Distributed Computer Systems». IEEE Trans. on Computers, Vol 41, No. 3 (1992), 257-273.

[11] Aguilar J.: «Maximizing the Speed-Up of Parallel Programs in Distributed Computing Environment». Actas Científicas de la IASTED International Conference on Intelligent Information Management and Systems, Washington, 1996.

[12] Aguilar J. y Hernandez M.: Dynamic Task Assignment on Distributed System with Failures. En: G. Sechi (ed), Massively Parallel Computing Systems, IEEE Computer Society, USA (1996).

[13] Kirkpatric S., Gelatt C. y Vecchi M.: «Optimization by Simulated Annealing», Sciences, Vol 220 (1983), 671-685.

[14] Golberg D., «Genetic Algorithms in search, optimization and machine learning», Addison-Wesley, New York, 1991.