Universidad de las Ciencias Informáticas

Facultad 3

Título: Algoritmos para la asignación de estudiantes a proyectos productivos.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yuniet Rodríguez Suárez

Tutor: Ing. Miguel Ángel Martínez

Ciudad Habana, Junio 2007

"Año 49 de la Revolución"

<<La verdadera investigación consiste en buscar a oscuras el interruptor de la luz.

Cuando la luz se enciende, todo el mundo lo ve muy claro. >>

-----Anónimo

Dedicado a mis padres,

Novia y a toda mi familia.

AGRADECIMIENTOS

Quiero agradecer a mis padres, hermanos, novia y seres queridos, por su constante apoyo y por las incontables enseñanzas y valores trasmitidos.

Les agradezco a todas las personas que de una forma u otra, aportaron su granito de arena.

A Manuel Vázquez por su ayuda.

A Merlyn por tener siempre su ayuda.

A Daira Pérez por su valioso aporte a la ciencia.

A Eugenia Muñiz y Francisco Cano por la enseñanza trasmitida.

A Arnaldo Utría y a Carlos Yasmany por guiarme por buen camino.

A tantas personas, que en el quehacer de cada día ven ahora el fruto cumplido.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los __ días del mes de _____ del año ___.

_____	_____
Yuniet Rodríguez Suárez	Ing. Miguel Ángel Martínez
Autor	Tutor

DATOS DE CONTACTO

Síntesis del Tutor Ing. Miguel Ángel Martínez
Profesión: Ingeniero informático
Años de graduado: 1

AVAL

OPINIÓN DEL TUTOR

RESUMEN.

La asignación de personal a empresas, proyectos es una tarea que consume mucho tiempo y de dicha asignación depende el futuro desarrollo de la misma, producto de la competitividad existente en el mundo. Con el devenir del tiempo se han agregado características como aptitudes, actitudes y relaciones interpersonales que deben de cumplir las personas. La presente tesis de grado proporciona dos algoritmos que pueden utilizarse para la ayuda a la toma de decisiones por parte de los directivos de la Universidad de las Ciencias Informáticas. Estos algoritmos tienen en cuenta los gustos, las preferencias de las personas y las compatibilidades entre estos, para conformar equipos de desarrollo, siempre teniendo en cuenta las capacidades y conocimientos para una buena selección.

PALABRAS CLAVE

Algoritmo, Selección de personas y toma de decisiones,

ABSTRACT.

The assignment of personal to companies, projects are a task that consumes a lot of time and of this assignment the future development of the same one, product of the existent competitiveness in the world depends. With becoming of the time they have been added characteristic as aptitudes, attitudes and interpersonal relationships that people should complete. The present grade thesis provides two algorithms that can be used for helping to take decisions on the part of the directive of the Information Sciences University. These algorithms keep in mind the likes, the preferences of people and the compatibilities among these, to conform development teams, always keeping in mind the capacities and knowledge for a good selection.

KEY WORDS

Algorithms, selection of person

TABLA DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

Introducción

Dado el perfeccionamiento empresarial que han tenido las empresas en el mundo, todas persiguen dentro de sus objetivos ser las mejores en su campo de acción. Para alcanzar esta meta, se debe tener muy en cuenta la gestión de recursos humanos. Es por ello, que la selección de estos tiene un valor muy importante, y del cual depende el futuro desarrollo de la empresa. Por lo que se ha comprobado, como en estas empresas se hacen grandes esfuerzos para que este indicador se realice de forma óptima, siempre tratando de contratar a las personas más capacitadas.

En el proceso de contratación de recursos humanos, para un proyecto empresarial, es importante tener en cuenta la idoneidad de las personas. Debido a que el trabajo se realiza en equipo, no solo hay que medir cuan bueno se es en la realización de una tarea sino también, hay que conocer de las habilidades, destrezas y relación interpersonales. Garantizando con esto, la satisfacción interna de los trabajadores, evitando situaciones de conflicto entre los miembros de la misma. Muchas veces se hace muy difícil elegir al candidato indicado para que realice una tarea y que esta, se haga en el tiempo establecido y con los resultados esperados.

Un proceso de selección eficiente, que permita cubrir los puestos de trabajo de manera que la organización pueda ser operada de manera competente tanto en el presente, como en el futuro, puede ser la clave para el cumplimiento de la misión organizacional. Para esto es necesario contar con un algoritmo que asigne el personal de forma correcta atendiendo a las características de cada individuo y sea eficiente dicha selección.

Por lo que este trabajo trata de buscar un algoritmo que cumpla con todo lo antes planteado para que sea utilizado en los proyectos productivos de la facultad 3.

Situación Problemática

En la facultad 3 no existe un procedimiento automatizado para lograr una correcta asignación del personal a los distintos proyectos productivos que se formalizan en la misma, por lo que la selección se realiza sin seguir ningún criterio siendo un factor decisivo para cumplir con el tiempo establecido para su entrega y calidad.

Problema a resolver

¿Cómo resolver de forma eficiente la asignación de personal a los proyectos productivos mediante un algoritmo empleando técnicas de ayuda a la toma de decisión?

Objeto de Estudio

Algoritmos de ayuda a la toma de decisiones aplicados a la asignación de recursos.

Campo de acción

Algoritmos de ayuda a la toma de decisión aplicados a la asignación de personal a proyectos productivos de gestión en la Facultad 3.

Objetivo General

Elaborar dos algoritmos de ayuda a la toma de decisión para una correcta asignación de personal a los proyectos productivos de la Facultad 3.

Hipótesis

Si se diseña un algoritmo para la conformación de equipos de software, en la facultad 3 se podrá asignar estudiantes de forma eficiente a los proyectos productivos.

Tareas de Investigación

- Realizar un estudio sobre las principales técnicas de diseño de algoritmos.
- Realizar un estudio sobre el estado del arte en el mundo acerca de los diferentes algoritmos utilizados para asignar recursos.

Métodos a utilizar

Entre las estrategias de investigación que se utilizan están la exploratoria y la explicativa.

Se exploran diferentes técnicas o algoritmos de asignación y se brinda una explicación de los diferentes algoritmos realizados.

- *Métodos del nivel teórico*: Histórico-lógico, Hipotético deductivo, Sistémico, entre otros.

  Se enfoca la problemática de la asignación y la toma de decisiones en general desde un enfoque histórico lógico, en la primera parte de nuestra investigación se realiza el desarrollo de un estudio del estado del arte de la problemática analizada, además de las formulaciones matemáticas a utilizar. La investigación sigue además un método hipotético deductivo porque a partir del problema concreto se plantean objetivos e hipótesis que en el transcurso de la investigación son resueltas siguiendo el método científico idóneo.

- *De los métodos del nivel empíricos:* se desarrollan desde una vinculación dialéctica, en la que partimos de los métodos del procesamiento de datos, sustentados en la utilización *estadística - descriptiva* para la determinación de los índices de idoneidad que facilitan la interpretación del objeto. En este sentido todo este proceso se sustrae al método de la medición, que fundamenta el diseño investigativo, finalmente se combinaron estos paradigmas, para arribar a los análisis de contenido que facilitaron las conclusiones y recomendaciones.

Estructura de la tesis

La tesis consta de 2 capítulos. En el primer capítulo se dará a conocer todo lo referente al estado del arte, brindando una serie de definiciones con respecto a la temática planteada y algunos métodos utilizados en el mundo para realizar asignación o selección de personal.

En el segundo capítulo se abordará sobre los diferentes algoritmos a utilizar en este trabajo para la asignación de personal y se darán varias definiciones para un mejor entendimiento de lo que se esta tratando y evaluar los resultados de los algoritmos planteados.

Capítulo 1. Fundamentación teórica.

Introducción

En este capítulo se dará a conocer la evolución que han tenido los algoritmos y una breve panorámica del estado actual de los algoritmos que se utilizan para la asignación de recursos de forma general (bienes, dinero, personas). Además de conocer como es la asignación en la Universidad de las Ciencias Informáticas del personal a proyectos de software, así como conocer que propuesta dan metodologías enfocadas a la ingeniería de software.

1.1 Proceso actual en la Universidad de las Ciencias Informáticas

Mediante una investigación realizada en la facultad 3 sobre como se realiza la selección de estudiantes a los disímiles proyectos que en esta se encuentran, se pudo constatar que la selección de personal para la entrada en proyectos productivos de la Universidad de las Ciencias Informáticas, en particular en la facultad 3, se realiza acorde a lo establecido por el vicedecanato de producción. En este, existe un procedimiento por el cual se rige la facultad para asignar estudiantes a proyectos. Los estudiantes interesados en participar en proyectos se deben de presentar en el vicedecanato con el objetivo de plasmar el deseo de entrada en proyecto mediante una planilla que han de llenar y posteriormente entregarla en dicho lugar, esta planilla se guardará para que los jefes de proyecto en caso de necesitar personal para el proyecto en el que se encuentran, piden las planillas y de estas escogen a los estudiantes para integrar el proyecto. Si es un proyecto de prioridad, además de llenar la planilla, el jefe de proyecto les realizará una prueba a los interesados en pertenecer al proyecto según el rol por el que opten y se le pedirá un aval de la juventud.

Toda esta selección se realiza de forma manual y las personas que no se hayan destacado o estén desinformadas nunca entrarán en los proyectos que verdaderamente merecen según sus conocimientos o gustos. Por lo que no hay un sistema que de forma automática asigne los respectivos roles a los estudiantes y lo haga de una manera eficiente. Y que tenga toda la información correspondiente para dicha asignación a los proyectos de forma actualizada. Producto de esta selección ha habido problemas con la entrega del producto y la calidad del mismo. La facultad ha empezado a tomar medidas con respecto a la asignación de estudiantes a proyectos. Por todos los problemas presentados, la facultad ha puesto en marcha un proyecto para la construcción de un sistema que realice esta tarea de trascendental importancia para el buen funcionamiento y selección de los proyectos en la Facultad 3. Por lo que surge la necesidad de crear o proponer un algoritmo que haga la selección de forma óptima para que forme parte esencial del software en cuestión.

Es aquí la vital importancia que tiene la elaboración de un algoritmo que realice dicha asignación de forma eficiente.

Primeramente se dará a conocer la evolución que han tenido los algoritmos y algunas definiciones y técnicas utilizadas en el diseño de algoritmos que han tenido buenos resultados en la actualidad.

1.2 Evolución de los algoritmos

Un Algoritmo es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea y/o resolver un problema. De un modo más formal, un algoritmo es una secuencia finita de operaciones realizables, no ambiguas, cuya ejecución da una solución de un problema en un tiempo finito.

También podría definirse un algoritmo como un conjunto de pasos claramente definidos que a partir de una cierta entrada (input) produce una determinada salida (output).

Algunas de las técnicas más utilizadas en el diseño de algoritmos son las siguientes:

- Dividir para conquistar
- Algoritmos aleatorizados
- Programación dinámica
- Algoritmos golosos (Greedy)
- Algoritmos basados en heurísticas
- Reducción a otro problema conocido
- Uso de estructuras de datos que solucionen el problema

El término “algoritmo” no está exclusivamente relacionado con la matemática, ciencias de la computación o informática. En realidad, en la vida cotidiana se emplean algoritmos en multitud de ocasiones para resolver diversos problemas. Algunos ejemplos son el uso de una lavadora (se siguen las instrucciones), pero no la preparación de una comida (porque no están perfectamente definidos los pasos). También existen ejemplos de índole matemática, como el algoritmo de la división para calcular el cociente de dos números, el algoritmo de Euclides para calcular el máximo común divisor de dos enteros positivos, o incluso el método de Gauss para resolver Sistema lineal de ecuaciones.(TORRES)

En el siglo XIX, se produjo el primer algoritmo escrito para un computador. La autora fue Ada Byron, en cuyos escritos en 1842 se detallaba la máquina analítica. Es por ello que es considerada por muchos como la primera programadora.

La falta de rigor matemático en la definición de "procedimiento bien definido" para los algoritmos trajo algunas dificultades a los matemáticos y lógicos del siglo XIX y comienzos de XX. Este problema fue en gran parte resuelto con la descripción de la máquina de Turing, un modelo abstracto de computadora formulado por Alan Turing, y la demostración de que cualquier método anticipado por otros matemáticos que pueda encontrarse para describir "procedimientos bien definidos" puede ser emulado en una máquina de Turing.

En la actualidad, el criterio formal para definir un algoritmo es que se trata de un proceso que puede implementarse en una máquina de Turing completamente especificada, o en alguno de los formalismos equivalentes. El interés original de Turing era el problema de la detención: decidir cuándo un algoritmo describe un procedimiento de terminación. En términos prácticos importa más la teoría de la complejidad computacional, que incluye los problemas llamados NP-completos, es decir aquellos, sobre los que generalmente se presume que requerirán tiempo más que polinómico para cualquier algoritmo (determinista). NP denota la clase de los problemas de decisión que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

1.3 Estructuras de Datos

Una estructura de datos es un conjunto de variables de un determinado tipo agrupadas y organizadas de alguna manera para representar un comportamiento. Lo que se pretende con las estructuras de datos es facilitar un esquema lógico para manipular los datos en función del problema que haya que tratar y el algoritmo para resolverlo. En algunos casos la dificultad para resolver un problema radica en escoger la estructura de datos adecuada.

Entre las estructuras de datos más ampliamente conocidas se encuentran:

1.3.1 Pilas

Una pila es una estructura dinámica en la cual la entrada y la salida de elementos se hace en orden LIFO (último en entrar, primero en salir) según las siglas en ingles (Last-in-first-out).

También pudiese definirse como: Una pila es una colección ordenada de elementos en la que pueden insertarse y suprimirse por un extremo, llamado tope, nuevos elementos.(TENENBAUM *et al.*, 1999)

Una pila (stack) soporta las siguientes operaciones.

- Crear_pila ( ): Crea una nueva pila vacía. O (1).
- Pila_vacía ( ): Indica si la pila esta vacía O (1).

- Tope ( ): Devuelve el elemento en el tope de la pila sin sacarlo de ella. O (1).
- Push (e): Apila un elemento e. O (1).
- Pop ( ): Saca un elemento de la pila y lo devuelve. O (1).

1.3.2 Colas

Una cola es una estructura dinámica en la cual la entrada y salida de elementos se hace en orden FIFO (First-in-first-out).

Según (TENENBAUM *et al.*, 1999) la define como: Una colección ordenada de elementos en un extremo (llamado el frente de la cola). O insertarlos en el otro (llamado final de la cola).

Una cola soporta las siguientes operaciones.

- Crear_cola (): Crea una nueva cola vacía. O (1).
- Enqueue (e): Agrega un elemento e a la cola. O (1).
- Dequeue (): Extrae un elemento de la cola. O (1).

1.3.3 Listas

Una lista es una estructura de datos dinámica. Donde el número de nodos puede variar drásticamente cuando se insertan o eliminan elementos. (TENENBAUM *et al.*, 1999)

Una lista enlazada es una colección dinámica de objetos de propósito general, en su uso más básico una lista sirve para mantener una relación entre una cantidad variable de objetos, pero hay muchas variantes de listas especializadas en otro tipo de aplicaciones. Una lista enlazada simple provee:

- Crear_lista: Crea una lista enlazada vacía. O (1).
- Insertar (e): Agrega un elemento e a la lista. O (1).
- Buscar (e): Busca un elemento e en la lista. O(n).

- Eliminar (e): Elimina el elemento e de la lista. O(n).

Algunas variantes de las listas son las listas doblemente enlazadas, las listas ordenadas (en donde se inserta en forma ordenada), las listas circulares, listas dobles circulares, etc.

Algunos tipos de estas listas serán utilizados para el trabajo con los algoritmos en el segundo capítulo.

1.3.4 Árboles

Un árbol es una estructura de datos ampliamente usada que emula la forma de un árbol (un conjunto de nodos conectados). Un nodo es la unidad sobre la que se construye el árbol y puede tener cero o más nodos hijos conectados a él. Se dice que un nodo *a* es padre de un nodo *b* si existe un enlace desde *a* hasta *b* (en ese caso, también decimos que *b* es hijo de *a*). Sólo puede haber un único nodo sin padres, que llamaremos raíz. Un nodo que no tiene hijos se conoce como hoja.

Formalmente, se puede definir un árbol de la siguiente forma recursiva:

Caso base: un árbol con sólo un nodo (es a la vez raíz del árbol y hoja).

Un nuevo árbol a partir de un nodo n_r y *k* árboles $A_1, A_2, ..., A_k$ de raíces $n_1, n_2, ..., n_k$ con $N_1, N_2, ..., N_k$ elementos cada uno, puede construirse estableciendo una relación padre-hijo entre n_r y cada una de las raíces de los *k* árboles. El árbol resultante de $N = 1 + N_1 + ... + N_k$ nodos tiene como raíz el nodo n_r, los nodos $n_1, n_2, ..., n_k$ son los hijos de n_r y el conjunto de nodos hoja está formado por la unión de los *k* conjuntos hojas iniciales. A cada uno de los árboles A_j se les denota ahora subárboles de la raíz. Para una mejor explicación ver la figura 1.

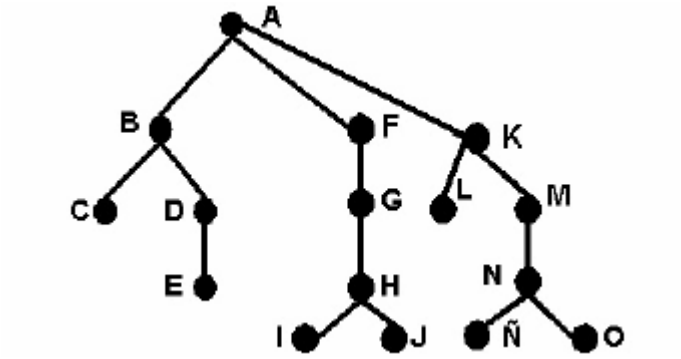

Figura 1 Ejemplo de árbol donde A es el nodo raíz y B, F, K son sus hijos y así consecutivamente hasta llegar a C, E, I, J, O con sus las hojas del árbol presentado.

Este tipo de estructura no se utiliza en la tesis producto que no hay que crear ningún tipo de estructura que se asemeje a un árbol.

1.3.5 Grafos

Un grafo es un objeto matemático que se utiliza para representar circuitos, redes, etc. Los grafos son muy utilizados en computación, ya que permiten resolver problemas muy complejos. Siendo útil para calcular complejidad por la forma de red que con esta se puede conformar.(HEILEMAN, 2003)

Un grafo está compuesto de vértices y aristas, en la figura 2 se muestra un ejemplo de grafo en que cada arista tiene un peso. Una arista no es más que la relación entre dos nodos. Para una mejor compresión se profundiza en el tema en el segundo capítulo pues es objeto de estudio para llegar a una solución algorítmica.

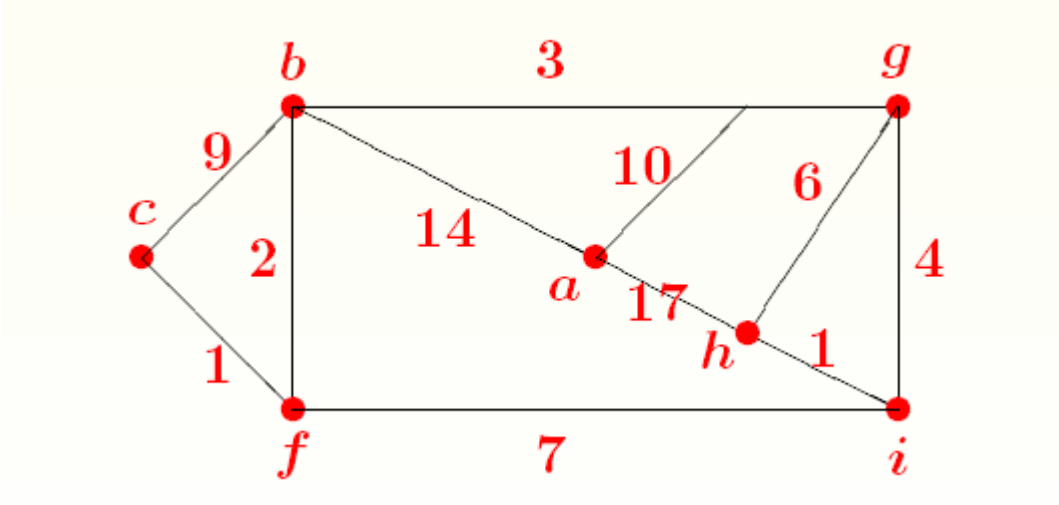

Figura 2 Ejemplo de grafo pesado.

1.4 Técnicas de diseño de algoritmos

1.4.1 Divide y Vencerás

Esta es una técnica de diseño de algoritmos que consiste en resolver un problema a partir de la solución de subproblemas del mismo tipo, pero de menor tamaño. Si los subproblemas son todavía relativamente grandes se aplicará de nuevo esta técnica hasta alcanzar subproblemas lo suficientemente pequeños para ser solucionados directamente. Ello naturalmente sugiere el uso de la recursión en las implementaciones de estos algoritmos.(HEILEMAN, 2003)

1.4.2 Programación Dinámica

Existe una serie de problemas cuyas soluciones pueden ser expresadas recursivamente en términos matemáticos, y posiblemente la manera más natural de resolverlos es mediante un algoritmo recursivo. Sin embargo, el tiempo de ejecución de la solución recursiva, normalmente de orden exponencial y por tanto impracticable, puede mejorarse substancialmente mediante la Programación Dinámica**.**

Según (HEILEMAN, 2003) es una técnica ascendente que comienza habitualmente resolviendo los subproblemas más pequeños, guardando estos resultados, y entonces reutilizándolos para resolver subproblemas cada vez mayores hasta que se obtiene la solución al problema original.

La eficiencia de esta técnica consiste en resolver los subproblemas una sola vez. Guardando sus soluciones en una tabla para su futura utilización.

La Programación Dinámica no sólo tiene sentido aplicarla por razones de eficiencia, sino porque además presenta un método capaz de resolver de manera eficiente problemas cuya solución ha sido abordada por otras técnicas y han fracasado.

La mayor aplicación de la Programación Dinámica es en la resolución de problemas de optimización.

La solución de problemas mediante esta técnica se basa en el llamado "principio del óptimo" enunciado por Bellman en 1957 y que dice: "*En una secuencia de decisiones óptima toda subsecuencia ha de ser también óptima*".

La técnica de la Programación Dinámica tiene grandes ventajas, y una de ellas es la de ofrecer un diseño adecuado y eficiente a todos los problemas que puedan plantearse de forma recursiva y cumplan el principio del óptimo.

Así, es posible plantear por ejemplo el algoritmo de Dijkstra en términos de la Programación Dinámica, y de esta forma aprovechar el método de diseño y las ventajas que esta técnica ofrece.

1.4.3 Algoritmos de Backtracking

Dentro de las técnicas de diseño de algoritmos, también se encuentra el método de Vuelta Atrás (del inglés *Backtracking*) es uno de los de más amplia utilización, en el sentido de que puede aplicarse en la resolución de un gran número de problemas, muy, especialmente en aquellos de optimización.

1.4.4 Algoritmos ávidos

El método que produce algoritmos ávidos es un método muy sencillo y que puede ser aplicado a numerosos problemas, especialmente los de optimización.

Dado un problema con *n* entradas el método consiste en obtener un subconjunto de éstas, que satisfaga una determinada restricción definida para el problema. Si cada subconjunto cumple con las restricciones se dice que son soluciones *prometedoras*. Una solución prometedora que maximice o minimice una función objetivo se denomina solución óptima.

Los algoritmos ávidos constan de los siguientes pasos:

1. Para resolver el problema, un algoritmo ávido tratará de encontrar un subconjunto de candidatos tales que, cumpliendo las restricciones del problema, constituya la solución óptima.

2. Para ello trabajará por etapas, tomando en cada una de ellas la decisión que le parece la mejor, sin considerar las consecuencias futuras, y por tanto escogerá de entre todos los candidatos el que produce un óptimo local para esa etapa, suponiendo que será a su vez óptimo global para el problema.

3. Antes de añadir un candidato a la solución que está construyendo comprobará si es prometedora al añadirlo. En caso afirmativo lo incluirá en ella y en caso contrario descartará este candidato para siempre y no volverá a considerarlo.

4. Cada vez que se incluye un candidato comprobará si el conjunto obtenido es solución.

Los algoritmos ávidos construyen la solución en etapas sucesivas, tratando siempre de tomar la decisión óptima para cada etapa.

El nombre de algoritmos ávidos, también conocidos como voraces (su nombre original proviene del término inglés "*greedy*") se debe a su comportamiento: en cada etapa "toman lo que pueden" sin analizar consecuencias, es decir, son glotones por naturaleza.

En "la asignación de tareas" los algoritmos ávidos son muy utilizados. Si se quieren asignar tareas, para esto existen dos estrategias distintas: asignar cada trabajador la mejor tarea posible, o bien asignar cada tarea al mejor trabajador disponible. Para estos tipos han creado varios algoritmos pero

no cumplen con la primera condición del que se pretende que es: asignar al más adecuado a la mejor tarea.

1.4.5 Algoritmos básicos de búsqueda en grafos

Existen dos técnicas básicas para recorrer los vértices de un grafo, la búsqueda por profundidad (DFS) y la búsqueda por anchura (BFS).

Recorrer un grafo consiste en “visitar” cada uno de los nodos a través de las aristas del mismo. Se trata de realizar recorridos de grafos de manera eficiente. Para ello, se pondrá una marca en un nodo en el momento en que es visitado, de tal manera que, inicialmente, no está marcado ningún nodo del grafo.

- **Recorrido en anchura** (BFS)**:** El recorrido en anchura supone recorrer el grafo, a partir de un nodo dado, en niveles, es decir, primero los que están a una distancia de un arco del nodo de salida, después los que están a dos arcos de distancia, y así sucesivamente hasta alcanzar todos los nodos a los que se pudiese llegar desde el nodo salida.

- **Recorrido en profundidad** (DFS)**:** El recorrido en profundidad trata de buscar los caminos que parten desde el nodo de salida hasta que ya no es posible avanzar más. Cuando ya no puede avanzarse más sobre el camino elegido, se vuelve atrás en busca de caminos alternativos, que no se visitaron previamente.

La idea del procedimiento *DFS* (*G; v*) es la siguiente:

- Se marca el nodo v.
- Si todos los nodos adyacentes a v están marcados, entonces TERMINAR; si no, se elige un nodo, w, adyacente a v que no esté marcado.
- Se ejecuta el proceso DFS (G; w).

Si *G* es conexo (es decir, si dos vértices cualesquiera de *G* siempre están conectados por un camino), entonces *DFS* (*G; v*) visita todos los nodos y aristas del grafo.

La búsqueda por anchura se usa para aquellos algoritmos en donde resulta critico elegir el mejor camino posible en cada momento como sucede en Dijkstra**.** Es importante hacer notar que los recorridos por anchura son útiles en aquella aplicaciones en las queremos encontrar el camino más corto entre cualquier par de vértices y es por ello que forman la base de dichos algoritmos. El recorrido por profundidad sirve por otro lado para averiguar si un par de grafos están conectados.

Existen una serie de algoritmos basados en una técnica de programación ávida que cumplen con dicho requisito, nos enfocaremos particularmente en dos de ellos, el algoritmo de Kruskal y en el de Prim.

El algoritmo de Prim comienza por un vértice y escoge en cada etapa el arco de menor peso que verifique que uno de sus vértices se encuentre en el conjunto de vértices ya seleccionados y el otro no. Al incluir un nuevo arco a la solución, se añaden sus dos vértices al conjunto de vértices seleccionados.

El algoritmo de Kruskal basa su funcionamiento en la elección de las aristas de menor peso que no forman ciclos, para poder elegir dichas aristas es necesario usar un método de almacenamiento que las ordene de menor a mayor peso.

Elige la arista más económica y si la misma no produce un ciclo entonces la agrega al AGM. Cuando no quedan aristas por agregar (no hay más o bien no se puede agregar ninguna) queda formado el árbol generador mínimo.

El seudo-código del Kruskal se muestra a continuación:

MST-KRUSKAL (G, w)

1. A es el conjunto vacío
2. para cada vértice v en G
    - make-set(v)
3. ordenar las aristas de menor a mayor peso
4. para cada arista (u, v) en G, en tomadas en orden creciente
    - si find-set(u) es diferente de find-set(v)
        - A es la union de A con (u ,v)
        - union(u, v)
5. Devolver A

El resultado del algoritmo es el árbol de expansión representado por el conjunto de aristas incluidas en A.

Uno de los ejemplos de su uso es EL VIAJANTE DE COMERCIO

En donde se conocen las distancias entre un cierto número de ciudades. Un viajante debe, a partir de una de ellas, visitar cada ciudad exactamente una vez y regresar al punto de partida habiendo recorrido en total la menor distancia posible.

Este algoritmo no funciona, producto que al elegir por compatibilidades se pueden eliminar personas convertidas en nodos mientras que el Kruskal recorre todos los nodos.

El Problema de Asignación Óptima es un ejemplo de asignación que consiste en asignar cada trabajador a un trabajo distinto, de forma que el coste total de las p asignaciones sea mínimo.

Definimos un grafo G, bipartido y completo, con bipartición (X, Y), donde X= $\{x_1, x_2,..., x_p\}$, e Y= $\{y_1, y_2,..., y_p\}$, y la arista (x_i, y_j) tiene coste w_{ij}. El problema de Asignación Óptima es equivalente a encontrar en G el acoplamiento perfecto de coste mínimo. El algoritmo de Kuhn-Munkres resuelve este tipo de problema con una complejidad de O (n^4) (BENAVENT). Este algoritmo más conocido como el método húngaro requiere tener una matriz de costos de asignación de "m" personas para ser asignadas a "m" tareas, donde todos los costos son no negativos.

1.4.6 Algoritmos aleatorizados

Los algoritmos aleatorizados son aquellos que en algún momento de su ejecución utilizan la generación de un número aleatorio para tomar una determinada decisión. Sorprendentemente aleatorizar un algoritmo haciéndolo tomar decisiones al azar a menudo sirve para que el algoritmo sea más eficiente. Los algoritmos aleatorizados se dividen en dos grandes familias los algoritmos:

- Algoritmos tipo Montecarlo
- Algoritmos tipo Las Vegas.

<u>Algoritmos tipo Las Vegas</u>

Los algoritmos de tipo Las Vegas siempre resuelven correctamente un problema, sin embargo pueden con baja probabilidad de ocurrencia demorar mucho más tiempo que en el caso medio para encontrar la respuesta.

Algoritmos tipo Montecarlo

Los algoritmos de tipo Montecarlo siempre son eficientes en cuanto a la cantidad de tiempo necesaria para encontrar la respuesta. Sin embargo en algunos casos con baja probabilidad de ocurrencia, pueden producir una respuesta no correcta al problema.

1.4.7 Algoritmos de ordenamiento (Sort)

Si bien existen distintos criterios para clasificar a los algoritmos de ordenación, una posibilidad es atendiendo a su eficiencia. De esta forma, en función de la complejidad que presentan en el caso medio, se puede establecer la siguiente clasificación:

- O (n^2): Burbuja, Inserción, Selección.
- O ($n \log n$): Mezcla, Montículos, Quicksort.
- Otros: Incrementos O ($n^{1.25)}$, Cubetas O(n), Residuos O(n).

El propósito es revisar algunos algoritmos de ordenamiento, para conocer cuales utilizar en el diseño de los algoritmos propuestos. Primeramente se dará algunas definiciones que serán útiles más adelante.

Estabilidad: Se dice que un algoritmo de sort es estable si para elementos que tienen la misma clave en el vector original mantiene el orden en el vector resultado. La estabilidad es una propiedad deseable en la mayoría de los algoritmos de sort y se vuelve necesaria cuando estamos ordenando por más de una clave.

In-situ: Un algoritmo de sort funciona (in-situ) cuando no requiere de espacio de memoria adicional (a excepción de un cierto espacio constante) para realizar el ordenamiento.

Los algoritmos pueden dividirse en:

Algoritmos no recursivos

Los algoritmos de sort no recursivos son los más simples para programar y entender, hay muchos métodos pero tres de ellos son los mas conocidos el Burbujeo, la selección y la inserción.

- Burbujeo

Es un algoritmo "in-situ", puede implementarse como un algoritmo estable y es de orden cuadrático O (n^2). Es un algoritmo que en general resulta demasiado lento ya que realiza una gran cantidad de intercambios entre los elementos del vector.

- Selección

Este algoritmo es de tipo "in-situ". Lamentablemente no puede implementarse en forma estable sin grandes modificaciones. Es de orden cuadrático al igual que el burbujeo O (n^2). En general es más eficiente que el burbujeo porque realiza menor cantidad de intercambios entre los elementos del vector.

- Inserción

El insert-sort es un algoritmo "in-situ" que puede implementarse en forma estable, es también de orden cuadrático O (n^2) pero en su mejor caso es de orden lineal O(n). Pese a ser considerado un algoritmo lento es en general mejor que el burbujeo y la selección.

Algoritmos recursivos

Los algoritmos recursivos se basan en general en el paradigma "Divide & Vencerás". Los algoritmos de este tipo más populares son el QuickSort, el MergeSort y el HeapSort.

- Quick Sort

El algoritmo de Quick Sort es uno de los algoritmos de sort más conocidos, es un algoritmo de tipo Divide & Vencerás cuyo tiempo es $O(n \log n)$ pero cuyo peor caso es O (n^2). Quick Sort es un algoritmo in-situ, es decir que a diferencia del algoritmo de MergeSort no necesita de espacio

adicional en memoria para ordenar el vector ya que el proceso se lleva a cabo sobre el vector mismo.

En la figura 3 se muestran los tres tipos de casos de los algoritmos mencionados

Algoritmo	Peor Caso	Caso Medio	Mejor Caso	Estable	In-Situ
Burbujeo	$O(n^2)$	$O(n^2)$	$O(n^2)$	Si	Si
Seleccion	$O(n^2)$	$O(n^2)$	$O(n^2)$	No	Si
Insercion	$O(n^2)$	$O(n^2)$	$O(n)$	Si	Si
MergeSort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	Si	No
QuickSort	$O(n^2)$	$O(n \log n)$	$O(n \log n)$	No	Si
HeapSort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	No	Si
CountingSort	$O(n)$	$O(n)$	$O(n)$	Si	No
RadixSort	$O(n)$	$O(n)$	$O(n)$	Si	No

Figura 3 Complejidad de algoritmos de ordenamiento

1.5 Métodos de asignación mediante Investigación de Operaciones

La investigación de operaciones es una ciencia que modela problemas complejos haciendo uso de las matemáticas y la lógica, permitiendo por otra parte el análisis de la toma de decisiones teniendo en cuenta la escasez de recursos, para poder determinar como se pueden maximizar o minimizar los recursos.

Para poder escoger el personal, y que cumpla con una serie de requisitos, no se puede obtener con una simple cuenta matemática por lo que se hace necesario aplicar métodos más sofisticados.

En este sentido se han aplicado Métodos de ordenación, técnicas basadas en la comparación con un candidato ideal, modelos de operadores OWA y técnicas de agregación basada en la eficiencia.

A continuación se presentan algunos métodos utilizados por la diversidad que presentan en cuanto a su funcionamiento.

1.5.1 Media Ponderada Ordenada (OWA)

Los operadores de agregación son objetos matemáticos cuya función es reducir un conjunto de números a un único número representativo.

La familia de operadores OWA permite ajustar fácilmente el grado de conjunción y disyunción implícito en una agregación.

Los operadores OWA son capaces de tener en cuenta una amplia variedad de agregadores, su naturaleza es definida por un vector de ponderaciones y no por un único parámetro. A esto hay que añadir que el concepto de *orness*, "parecido a", y su medida generan un operador muy útil basado en los valores del vector de pesos. Yager propone que la medida de *orness* sea interpretada como una medida de optimismo en la toma de decisiones, mientras que la medida de *andness* es una medida de pesimismo. La idea básica de la técnica OWA es asociar pesos con una posición ordenada más que con un elemento particular.

Un operador OWA de dimensión *n* es una función de tipo $F : R^n \rightarrow R$, que tiene asociado un vector de ponderaciones W de longitud n.

$W = [w_1, w_2, ..., w_n]^T$, donde $w_i \in [0,1]$, y $\sum_{i-1}^{n} w_i = 1 \quad 1 \leq i \leq n$

F ($x_1, x_2, ..., x_n$)= $\sum_{k=1}^{n} w_k * x_{jk}$ donde x_{jk} es el k-ésimo valor mayor de la colección $x_1, x_2, ..., x_n$

andness = 1 – orness

orness (F)= $\frac{1}{n-1} \sum_{1}^{n} [(n-i) w_i]$

Un aspecto fundamental de los operadores OWA es el paso de la reordenación. Un agregado x_i no está asociado con un peso particular w_j, sino que un peso está asociado con una posición ordenada *j* particular de los argumentos.

Los operadores OWA proporcionan una gran flexibilidad para modelar una amplia variedad de agregadores, pues su naturaleza es definida por un vector de ponderaciones, y no por un único parámetro. De este modo las necesidades de los directivos que deben tomar una decisión son plasmadas en el modelo más correctamente y con más fidelidad a la realidad. Además, estos operadores permiten los intercambios entre objetivos en conflicto.

1.5.2 Ordenación por pares

Con este método lo que se pretende es comparar un candidato con otro, en caso en que sean varios candidatos y se consideren equivalentes aparecerán en el mismo grupo o nivel, teniendo el mismo valor de ordenación para esto ver el siguiente ejemplo Tabla 1.

Ordenación por Pares	
P3	1
P2	2
P1	2

Tabla 1 Ejemplo de resultados de la ordenación por pares de tres personas, en el que la persona P3 tiene menor nivel de ordenación que los demás candidatos por lo tanto se considera con mejores condiciones para ser seleccionado.

Establecimiento de una ordenación por pares

1. Se determina el conjunto de candidatos aspirantes a un puesto vacante.

2. Se determina el conjunto de competencias deseadas.

3. Para cada competencia se establece una comparación entre candidatos en el sentido de determinar si uno tiene un nivel más o menos adecuado que otro.

Para ello se indicará con un 1 en la casilla correspondiente a la fila i y la columna j cuando el candidato de la fila i tenga un nivel más adecuado que el de la columna j. En caso de ser ambos equivalente se pondrá un 1 en ambas casillas simétricas correspondientes. El resultado es una matriz booleana cuyos elementos son 0 ó 1.

4. Se pondera cada matriz booleana por el factor correspondiente. Siendo la suma de todos los pesos igual a 1.

Se resume toda la información en una sola matriz, por agregación.

6. Se realiza un α-corte con el fin de obtener nuevamente una matriz booleana, donde se asignará un 1 únicamente a las casillas con un valor mayor que el α determinado.

Establecimiento de grupos de candidatos equivalentes o indiferentes

1. Se parte de una matriz booleana que relaciona a los candidatos considerados dos a dos, es decir, de la matriz booleana resultante del apartado anterior, una vez aplicado el α-corte.

2. Se escoge arbitrariamente un candidato cualquiera i y se obtiene su cierre transitivo (conjunto de candidatos con un nivel menos adecuado que el candidato i) y su cierre transitivo inverso (conjunto de candidatos con un nivel más adecuado que el candidato i).

3. Se realiza la intersección entre el cierre transitivo y el cierre transitivo inverso, hallándose como resultado el conjunto de candidatos que junto con el escogido i forman una clase de equivalencia.

4. Se eliminan de la matriz booleana que relaciona candidatos por pares, las filas y columnas correspondientes a la clase ya obtenida. Se obtiene una matriz de orden inferior.

5. De esta matriz de orden inferior se escoge, de nuevo arbitrariamente, un candidato j, obteniendo su cierre transitivo y su cierre transitivo inverso.

6. Se continúa el proceso a partir del punto 3 y así sucesivamente hasta el agotamiento de la matriz.

7. Habremos obtenido, entonces, todas las clases de equivalencia, surgidas sin orden alguno, como consecuencia de la elección arbitraria de los candidatos.

1.6 Competencias que aportan a la buena selección de personal

Para comenzar a hablar acerca de algoritmos sobre asignación de personal hay que tener muy en cuenta, como se está moviendo el mundo en este sentido, y no dejar de mencionar que para una buena selección, hay que aplicar competencias laborales, que se definen según:

CONOCER (México, 1997): capacidad productiva de un individuo que se define y mide en términos de desempeño en un determinado contexto laboral, y no solamente de conocimientos, habilidades, destrezas y actitudes; estas son necesarias pero no suficientes por sí mismas para un desempeño efectivo.

Leonard Mertens[1] (2000), la conceptualiza así: “Aptitud de un individuo para desempeñar una misma función productiva en diferentes contextos y con base en los requerimientos de calidad esperados por el sector productivo. Esta aptitud se logra con la adquisición y desarrollo de conocimientos, habilidades y capacidades que son expresados en el saber, el hacer y el saber hacer”.

(URRUTIA BADILLO y WONG JAUMA, Junio 2004) la define como: “Un sistema de conocimientos, habilidades, actitudes, valores, motivos, aptitudes y capacidades que debe poseer el individuo para el desempeño satisfactorio de su actividad laboral”

[1] Mertens, Leonard. Competencia laboral: sistemas, surgimiento y modelos. Cinterfor. 1996.

Muchas empresas en Estados Unidos, Europa y recientemente en América han incorporado la Gestión de Recursos Humanos basada en competencia laboral como una herramienta para mejorar la productividad y mantener un clima positivo en las relaciones con sus colaboradores. La justificación de estos esfuerzos se encuentra en el intento de mejorar los niveles de productividad y competitividad mediante la movilización del conocimiento y de la capacidad de aprender de la organización. Se hace evidente así, la tendencia de revalorización del aporte humano a la competitividad organizacional.

Por otra parte en Cuba la gestión por competencias es un enfoque novedoso para el mundo empresarial, los modelos existentes y aplicados en países diferentes al nuestro no son aplicables de manera exacta al contexto de Cuba y en especial a las universidades de nuestro país.

Para formar equipos de profesionales ya tan solo no se les mide su conocimiento sino también relaciones entre compañeros y que se sientan a gusto con el trabajo que desempeñan logrando con esto una mayor productividad. Por lo que, la selección del personal es un punto determinante para el buen funcionamiento de una empresa.

1.7 Aplicación en la ingeniería de software

La Ingeniería de software es la rama de la ingeniería que crea y mantiene las aplicaciones de software aplicando tecnologías y prácticas de las ciencias computacionales, manejo de proyectos, ingeniería, el ámbito de la aplicación, y otros campos.

Según Alan Davis la define como "la aplicación inteligente de principios probados, técnicas, lenguajes y herramientas para la creación y mantenimiento, dentro de un coste razonable, de software que satisfaga las necesidades de los usuarios"...

En la informática existen metodologías como RUP, XP, EXTREME, TSP que definen los roles en cada una de sus etapas. En la universidad la más utilizada es el Proceso Unificado de Desarrollo (RUP).

RUP es un proceso interactivo e incremental de Ingeniería de Software la cual designa tareas y responsabilidades. Que asegura la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con un costo y un calendario predecible. Es guiado por Casos de Uso y centrado en arquitectura:

- Casos de uso: Reemplazan la antigua especificación racional tradicional.
- Iterativo e Incremental: Divide al proceso en cuatro fases dentro de las cuales se realiza varias iteraciones.

Cada rol es descrito por un conjunto de características y competencias. Esta metodología no cuenta con un método o algoritmo para asignar estos roles de forma que logre asignar de la forma más justa y eficiente posible. Esta brinda un compendio de características por roles que son definidas por sus creadores como las más importantes que debe de tener una persona para ocupar un rol específico en un proyecto, pero no brinda un procedimiento de cómo realizar dicha asignación de forma óptima a un determinado rol.

Roles: Define el comportamiento y responsabilidades del individuo, o de un grupo de individuos trabajando como equipo.

Otras metodologías como XP tampoco proponen un método para asignar los respectivos roles que proponen para un buen funcionamiento del proyecto que utilice este tipo de metodologías.

Conclusiones

Con la investigación realizada se pudo conocer como ha sido la evolución y técnicas de diseño de los algoritmos. La importancia que tienen, si se pretende un algoritmo que sea eficiente.

Se brinda una serie de definiciones acerca de la temática planteada, y lo más actual en cuento a conformación de equipos de trabajo. Tanto Kruskal como Prim debido a recorrido por los grafos no recorren los caminos de la forma que se quiere debido a que para realizar compatibilidades, pueden quedar vértices fuera de la lista por lo que no cumplen con los requisitos. En cuanto a el recorrido en profundidad se puede utilizar en la conformación del nuevo algoritmo pero aplicándole variaciones. Debido a que el recorrido se hace de forma ordenada de manera ascendente.

Los métodos analizados en cuanto a investigación de operaciones para asignar personas no son de gran utilidad debido a que dan la misma importancia a todas las competencias por lo que no es adecuado para el caso específico en la Facultad 3 debido a que las competencias no tienen la misma importancia.

Capítulo 2 Descripción de los algoritmos

Introducción

En este capítulo se hará una descripción detallada de los algoritmos propuestos con el objetivo que se puedan comparar para llegar una conclusión de cual es el más adecuado para ser utilizado en el software que esta creando la Facultad 3, este se realizará con el objetivo de gestionar de forma correcta la asignación de estudiantes a proyectos productivos. Siendo de gran ayuda a la toma de decisiones.

2.1 Precondiciones para conformar los algoritmos.

Para empezar hay que tener en consideración que los algoritmos devolverán un equipo de personas, y que estos algoritmos están diseñados para proyectos de gestión, aunque también puedan utilizarse en proyectos de otra índole, en el que todos los estudiantes seleccionados tengan las competencias requeridas para cumplir la tarea y que por supuesto, esta sea, con calidad.

Primeramente se debe de saber que un equipo idóneo no es más que un colectivo de personas que interactúan entre sí, en el que cada persona ocupa el puesto en el que se encuentra más capacitado.

Para conformar este equipo idóneo, se tendrá en cuenta los gustos y las preferencias de cada estudiante evitando de esta manera que el estudiante pueda sentirse mal con la tarea asignada. Teniéndose en cuenta que las relaciones interpersonales entre todo el equipo de trabajo sea la más adecuada posible, en donde prime la armonía entre cada uno de los integrantes.

Para lograr que ocurra lo antes planteado, cada estudiante debe de llenar un currículo donde se le pedirán:

- Los proyectos en que ha participado
- Rol ocupado en cada proyecto en el que se ha vinculado
- Datos personales (nombre, grupo, etc.)
- Experiencia en proyecto.

Acompañado de este currículo también el estudiante llenará una encuesta. Mediante la encuesta se podrán tener en cuenta sus gustos, preferencias y conocimientos.

Se le pedirá que marquen el rol o los roles en los que le gustaría trabajar, para la posible entrada en los proyectos productivos.

También se le realizarán pruebas para medir conocimientos y habilidades, estas pruebas brindarán información de que tan bueno se es en un determinado rol. Para evaluar la compatibilidad de los estudiantes para la conformación del equipo se le realizará otra encuesta (ver Anexo 3). A grupos escogidos de forma aleatoria se le aplicará una encuesta grupal con el objetivo de saber de estas personas con quien le gustaría trabajar y con quien no, como otra variable a medir.

Los resultados de las encuestas se tomarán en consideración para conformar también los equipos.

Todos estos datos obtenidos a partir del procesamiento del currículo y de las encuestas son utilizados y convertidos automáticamente en entradas para los algoritmos que se implementen.

Mediante esta forma se podrán dar a conocer estudiantes que no habían sido reconocidos por diferentes motivos con grandes posibilidades de vincularse a proyectos. Siempre teniendo en cuenta sus conocimientos, intereses y aspiraciones.

Para que no exista problemas de asignación, producto de la creación de los nuevos proyectos, los equipos se escogerán según el tipo de proyecto que se conforme, clasificándose según su tamaño en grande, medio o chico el tamaño guiándose por un grupo de prototipos ya creados. Por lo que el especialista podrá cambiar el valor de los parámetros a comparar en el caso que se estime conveniente teniendo en cuenta la complejidad que tenga el nuevo proyecto. Evitando con esto, que el proyecto que se cree de primero no se lleve a los mejores en cada uno de los roles sino a los que verdaderamente necesita, dándole oportunidad a personas no reconocidas, de forma tal que todos los estudiantes podrán tener un proyecto atendiendo claramente a su nivel.

Previendo con esto inconformidades por parte de los estudiantes y que la facultad esté segura que las personas asignadas cuentan con las capacidades para la realización del proyecto.

Todos los datos, en la medida que las personas los vayan entrando, serán almacenados para que después, el algoritmo pueda utilizarlos para gestionar la información que necesita para su correcto funcionamiento. Todas las personas en el momento de entrada se guardarán con un índice "*i*" en una lista de personas por orden de llegada, esta lista de personas se hará según lo planteado anteriormente.

Una persona puede estar propuesta para varios roles al mismo tiempo, solamente se le asignará el rol para el que se encuentre más apto. Para esto se tendrá en cuenta los resultados del currículo, la encuesta y las pruebas para el rol y que tenga excelentes relaciones con el resto del grupo que se esta conformando. En todos los casos se empezará a seleccionar el personal por la jerarquía de roles que predefine el prototipo que se escoge para la conformación del equipo, siendo el primero el líder de proyecto y así consecutivamente los diferentes roles con los que contará el nuevo proyecto que se esta creando.

Estos algoritmos si no encuentran una solución óptima, serán capaces de entregar soluciones factibles y razonablemente buenas.

2.2 Primer algoritmo

Para realizar este algoritmo lo primero que se debe hacer es, del listado de personas, llevar los datos de cada estudiante a números reales para un mejor trabajo; estos datos no son más que competencias donde cada una de las características significativas se convertirán en valores numéricos para el futuro trabajo del algoritmo que se esta planteando.

Los valores numéricos conforman un vector compuesto por c_i competencias de la forma $v(c_1, c_2, ..., c_n)$ por las que compiten las personas para optar por el rol que desea.

Los roles y cantidad de personas por roles entrarán a través de un prototipo. Para tener una noción de lo que es, solamente basta conocer que este prototipo contendrá el nombre del proyecto y la cantidad de estudiantes requeridas por rol y las características de un ideal por cada rol que predeterminado, será entrado por unos expertos en una encuesta realizada (Ver Anexo 2).

Este ideal no es más que un vector $v(c_1, c_2, ..., c_n)$ predefinido tomado de una encuesta realizada a varios expertos.

Los expertos no trabajan físicamente juntos, sino que cada uno de ellos opina por escrito, de forma libre, sin que ninguno de los otros participantes conozca sus opiniones personales. Con ello se persigue recibir información de un conjunto de especialistas en un ambiente de anonimato que facilita su libertad de expresión. Además, debido a la forma en que se realiza, cualquier participante puede cambiar de opinión a lo largo del proceso gracias a los datos que haya ido recibiendo, sin que este cambio quede individualmente reflejado hacia el exterior.

Para elegir a los expertos se realizó una encuesta (ver Anexo 1) para ver si las personas seleccionadas reunían los requisitos de experto.

La cantidad de encuestados fueron 8 personas de las cuales 6 el ellas según el procesamiento de la encuesta resultaron adecuados como para ser expertos.

Posteriormente se realiza otra encuesta solamente a los expertos seleccionados hallando el porcentaje de acuerdo a las respuestas dadas por los expertos aplicando un método porcentual . A la conclusión que se llegó después de haber procesado las encuestas es que:

El 100% de los expertos consideran que la Experiencia es muy adecuada medirla, coincidiendo con que el 83 % de estos expertos señalan que es la experiencia el más importante para desempeñar el rol de líder. Para el rol de analista, la experiencia sigue siendo uno de los aspectos más importantes a medir, en un 100 %. Para el rol de desarrollador, la experiencia no es evaluada de la misma forma, ya que solo el 50% coincide en señalar que la experiencia es un aspecto importante a medir y solo un 33% dice que es muy importante.

El 67% de los expertos consideran que la Prueba es bastante adecuado medirlo. El 67 % esta de acuerdo que entre la experiencia, las pruebas y las notas a fines según el rol dada en la carrera, la Prueba es el segundo más importante mientras un 33% dice que es el más importante para el rol de líder. Para el rol de analista el 33% consideran que las pruebas tienen igual importancia que la experiencia y un 67% cree que es menos importante que la experiencia. Para el rol de desarrollador la prueba según el 50 % de los expertos opinan que es lo más importante de los aspectos, el 33% dice que es lo segundo importante y el resto es menos importante para el rol de desarrollador.

El 50% de los expertos consideran que las asignaturas a fines por el plan de estudio es bastante adecuado tenerlo en cuenta mientras que el 33% considera que es adecuado. Para el rol de líder el 50 % dice que es, lo menos importante de los aspectos antes nombrados, el 33% dice que es lo mas importante de todos y un 17 % que solo es importante. Para el rol de analista el 50 % opina que es importante medirlo, el 33% dice que es lo menos importante y un 17 % que es lo más importante. El 67 % esta de acuerdo que entre los aspectos es menos importante, mientras un 33% dice que es importante para el rol de desarrollador.

Las características del vector ideal para un rol determinado, puede ser modificable según el tipo de proyecto que se vaya a crear. Hay que tener muy en cuenta que todos los proyectos no tienen la misma complejidad. Este vector ideal puede ser cambiado según los parámetros que se crean que

son más importantes a medir para seleccionar a la persona capacitada en un determinado puesto. Siempre teniendo en consideración que este vector tiene la forma $v(c_1, c_2, ..., c_n)$, donde los valores de c estará dada entre [0,1].

Este cambio debe hacerse por alguna persona con los conocimientos necesarios puesto que es fundamental, para seleccionar el nuevo grupo de proyecto. Se recomienda que sea una persona con vasta experiencia y conocimientos en el desarrollo de proyectos productivos.

Primeramente se introducen los vectores anteriormente mencionados en cada uno de los nodos creados por la entrada del prototipo. Estos prototipos están conformados por una lista de objetos que contendrán el nombre del rol, la cantidad de personas necesarias y una lista de vectores además del vector ideal sacado por los expertos. La asignación estará dada por sus preferencias y gustos, pudiéndose repetir en varios roles. Posterior a esto, se realizará la asignación en orden jerárquico evitando que personas con vastos conocimientos sean asignados inadecuadamente.

2.2.1 Pasos para hacer la selección o asignación.

Algoritmo en lenguaje natural:

1. Entrada del algoritmo:
    - Prototipo, este contendrá la lista de roles de forma jerárquica que se quiere tomar para el proyecto, donde cada rol tiene como datos: el nombre del rol, la cantidad de estudiantes necesarios, el vector ideal y la lista de personas.
    - lista de personas que se encuentran sin proyecto.
2. Salida: el prototipo con todas las personas que se necesitan para el nuevo proyecto, en caso de que este se pueda formar
3. Primeramente hay verificar que la cantidad de estudiantes total sea igual o mayor que la cantidad de personal necesitado.

4. si la cantidad total es menor retornará que no se puede conformar el equipo por falta de personal.

5. En caso contrario, según el nombre de cada uno de los roles; este nombre se buscará en la lista de preferencias de cada persona, en la lista original. Creándose una lista nueva de personas, que dentro tendrá las competencias y el índice en la lista original.

6. Esta nueva lista será ordenada ascendentemente por las distancias, producto que a menor distancia mayor será la semejanza con el ideal.

7. Si al comparar la lista creada con la cantidad que se quiere, en el rol en el que se esta trabajando, la lista es mayor. Se tomará la lista del prototipo hasta la cantidad prevista y se eliminarán de la lista original

8. Si es menor o vacío se recorrerá la lista general aplicando un método de búsqueda hasta encontrar al que mas se asemeje al ideal.

9. Se realizará la repetición hasta que la lista del prototipo en la posición, donde se esta sea igual a la cantidad que se pedía.

10. Al recorrer todos los roles del prototipo devolverá un equipo idóneo.

Entradas: Prototipo, lista de personas

Salida: equipo ideal

Buscar_Ideal (Prototipo, lista personas)

---

1. Si personas .cantidad () < Prototipo.cantidad
    - Mensaje (“No puede devolverse el equipo por falta de personal”)
1. Sino
    - **Para** i desde 1 **hasta** prototipo. cantidad **hacer**
        - **Llenar lista** (i)
2. **Retornar** Prototipo

// Este método coge las listas de las personas por roles y las compara con cada ideal que tiene el prototipo y las entra en forma ascendente al prototipo.

Llenar lista (i)

---

1. nombre_posición $\Leftarrow$ Prototipo$[i]$. Nombre
2. X $\Leftarrow$ 0
3. Si Prototipo $[i]$.cantidad==1
    - Si lista _ personas (nombre_posición).cantidad==0

- o X⇐ lista original .buscar mejor ()
- o Prototipo $[i]$.Lista. Adicionar (X)
- o Lista original .Eliminar (X)

- Sino
  - o Prototipo $[i]$.Lista. Adicionar (lista _ personas (nombre_posición).buscar mejor ())
  - o Lista original .Eliminar (Lista(nombre_posición).buscar mejor ().índex)

4. Sino

- Si lista _ personas (nombre_posición).cantidad < Prototipo$[i]$.cantidad
  - o **Mientras** lista _ personas(nombre_posición).cantidad <> Prototipo$[i]$.cantidad **hacer**
    - lista _ personas (nombre_posición). Adicionar (lista original .buscar mejor ())
- lista _ personas (nombre_posición). Ordenar Ascendentemente ()
- Prototipo $[i]$.Lista. Adicionar (lista _ personas (nombre_posición), Prototipo $[i]$.cantidad)
- Lista original .Eliminar (lista _ personas, Prototipo $[i]$.cantidad)

Este algoritmo no tiene en cuenta las compatibilidades entre las personas.

El seudo-código de este algoritmo esta implementado en C++ (ver Anexo 4).

A continuación mostraremos las funciones y las tablas de los roles más importantes por los que nos regimos para darle un peso a cada uno de los atributos que se medirán en dicho algoritmo.

Experiencia: será el resultado de lo que el estudiante llene en el currículo y datos específicos que se recogerán en la encuesta a realizar (tener en cuenta que no es objetivo de esta tesis) solamente conocer que se entrarán como datos y la formula que se utilizará será:

$$\text{Experiencia} = \frac{\text{Estancia}}{2}$$

La experiencia es el resultado de medir el periodo que un estudiante a estado vinculado a un proyecto con un determinado rol, este valor será Estancia entrado mediante la encuesta que se le realizará antes mencionada y tendrá valores entre 0 y 2 (ver tabla 2)

Cantidad de tiempo (año)	**Valores a tomar (estancia)**
Si el tiempo es ninguno o menor que 1 año	0
De 1 a 2 años	1
3 años o más	2

Tabla 2 Valores asignados a la cantidad de tiempo en proyectos productivos de una persona

Prueba: Son las pruebas que se le hacen para medir sus conocimientos con respecto al rol, estas pruebas van a tener un valor desde 0 a 5 puntos.

Aprovechamiento docente: Promedio de notas según el plan de estudios se saben que asignaturas son afines al rol a desempeñar.

a) Si un estudiante esta en primer año las asignaturas que darán peso a los diferentes roles son:

- Introducción a la programación.
- Programación 1.
- Practica Profesional1.

b) Si el estudiante esta en segundo año las asignaturas serán:

- Programación 2.
- Bases de Datos.

c) Si esta en tercer año las asignaturas serán:

- Programación 3.
- Ingeniería de Software I y II
- Programación 4.
- Administración de Empresas.

d) Si está en cuarto año:

- Gestión de proyecto.

Por el plan de estudio y el rol que quiera desempeñar se le dará una cantidad de puntos que oscilará siempre entre 0 y 1 atendiendo al año que este cursando.

Se debe de tener en cuenta que el máximo valor a coger en cualquiera de los roles es 1.

De las asignaturas antes mencionadas:

Tributan a programación:

1. Introducción a la programación.
2. Programación 1.
3. Programación 2.
4. Programación 3.
5. Programación 4.
6. Bases de Datos.

Tributan a analistas:

1. Practica Profesional1
2. Ingeniería de Software I y II
3. Gestión de proyecto.

Tributan a líderes:

1. Administración de Empresas.

Tanto la prueba por rol, como el aprovechamiento docente estarán representados por la Figura 4.

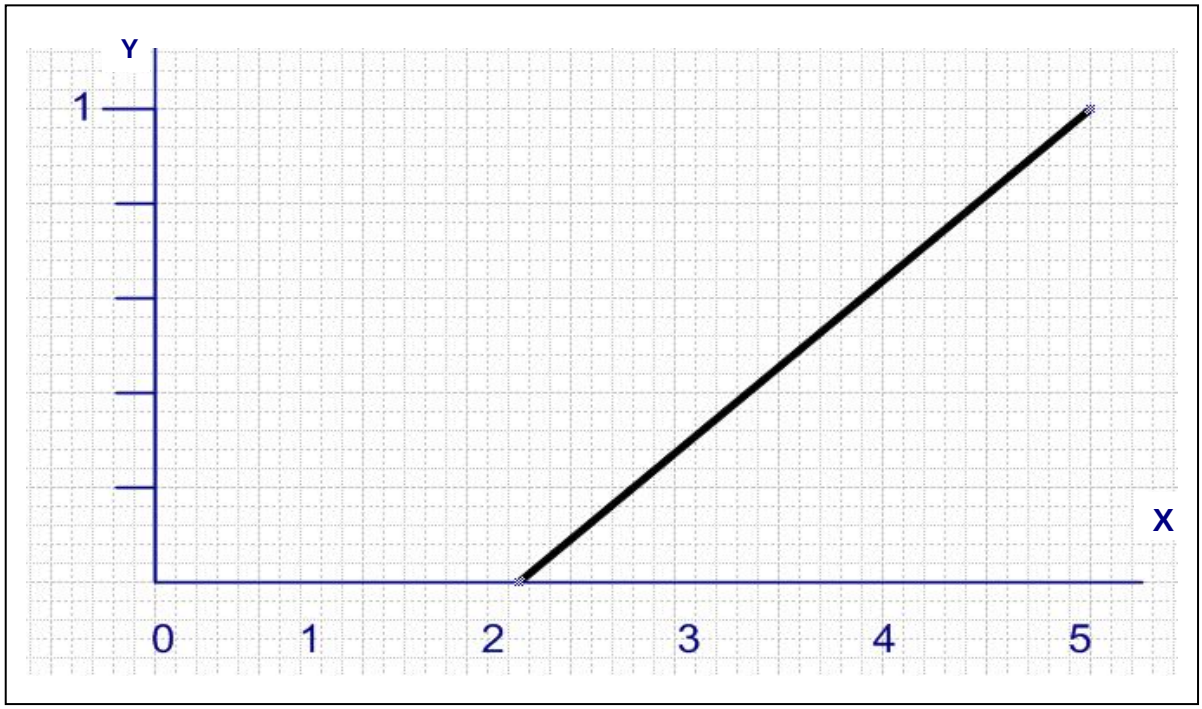

Figura 4 En el eje X se señalarán las notas promedio según el año en que se encuentre el estudiante de acuerdo al plan de clases y la nota que se obtenga de la prueba hecha para optar por el rol.

La escala del eje Y contendrá un valor entre 0 y 1. En el eje X se representará la nota del estudiante. Un estudiante que tenga 2 puntos o menos su aprovechamiento docente tendrá un valor nulo.

$$promedio = \sum notas / cantidad \qquad (1)$$

$$aprovechamiento\_.docente = \left(\ promedio - 2 \ \right) / 3 \qquad (2)$$

La formula de la distancia euclidiana para el cálculo de la distancia con el ideal estará dada en la siguiente fórmula.

$$Dist(A,B) = \sqrt{\left(c_{1A} - c_{1B}\right)^2 + \ldots + \left(c_{2A} - c_{2B}\right)^2}$$

Donde A es una persona *x* y B el vector ideal

Producto que las competencias antes mencionadas según los expertos no tienen el mismo peso para los diferentes roles, la formula de distancia que se ha utilizado es la euclidiana pesada que se define de la siguiente forma:

$$Dist(A,B) = \sqrt{w_0\left(c_{1A} - c_{1B}\right)^2 + \ldots + w_f\left(c_{2A} - c_{2B}\right)^2}$$

Donde w_i es el peso que tiene la competencia c_j y esta en el rango entre [0,1]

En donde para cada rol esta ponderación podrá ser ajustada según las necesidades y características de los proyectos que se pretendan crear. Según los expertos ellos consideran que:

- Para el rol de líder: $w_1 > w_2 > w_3$
- Para los desarrolladores $w_1 < w_2$ y $w_3 < w_1$
- Para los analistas: $w_1 > w_2 > w_3$

Estos valores fueron tomados de la encuesta realizada a los expertos w_1 es el valor que mide la experiencia en proyecto, w_2 mide las pruebas realizadas al rol y w_3 el aprovechamiento docente del estudiante.

2.3 Segundo algoritmo propuesto

Para la realización de este algoritmo primeramente hay que conocer una serie de definiciones para un mejor entendimiento.

- Un grafo es conexo si todo par de vértices está unido por un camino.

Vértices adyacentes

Un vértice w es adyacente a un vértice v si existe una arista de v hacia w. En un grafo no dirigido decimos que una arista es incidente en un vértice si el vértice es punto de llegada o partida de una arista. En un grafo dirigido decimos que una arista entra o sale de un vértice por lo que hablamos de aristas entrantes o salientes.

Grafo pesado

Un grafo es pesado si sus aristas están rotuladas con valores numéricos llamados pesos. El significado del peso depende de la aplicación, por ejemplo puede ser la distancia entre los vértices, la capacidad de la arista.

La modificación traerá consigo que todas las personas sean compatibles. Y vendrá dada debido a que las aristas de mayor peso se les asignará el mismo valor que a todas las demás aristas que compartan el vértice previniendo con esto que si una persona no se lleva con alguien, no puede ser seleccionado en el equipo producto de poderse llevar con otra persona que se lleve con el y que no se empieza por el orden de las aristas sino por un vértice determinado, para ver esto ver la siguiente Figura 5.

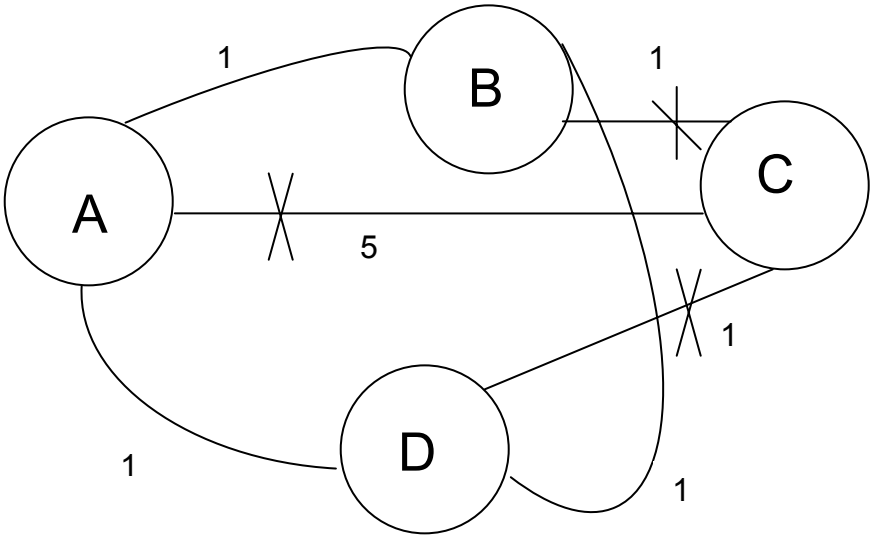

Figura 5 Ejemplo del funcionamiento del algoritmo implementado

Si se empieza desde A y la compatibilidad es de 1 con B y D, y con C es superior a 3 todos las aristas que se relacionan con C tomarán valor 0. Por lo que no se tendrá en cuenta conformándose el equipo ideal.

2.4 Algoritmo de grafos

Sea G (E, v) un grafo no dirigido con aristas pesadas. Si (u, V) $\in$ E es una arista de G entonces el peso de la arista es W (u, v). Supongamos por el momento que las aristas siempre tienen pesos.

Dado un camino $\pi = (u_0, u_1, ..., uk)$ el costo del camino es la suma de los pesos de las aristas.

Se creará el grafo donde cada uno de los vértices será una persona y la distancia entre dos vértice será una arista que estará dada por la compatibilidad que tendrá con los demás estudiantes, para esto se les aplicará el perfil cibernético a todos los estudiantes. Cada persona o vértice contendrá dos banderas, la primera band1 será 1 si fue visitado de forma aleatoria en caso contrario 0 y la segunda: band2 =1 si fue visitada por su compatibilidad con un determinado vértice sino tendrá predefinido valor 0.

Se tiene una lista de persona con sus respectivas relaciones, para una mejor comprensión (ver Tabla 3)

Persona	1	1	2	3
Relacionado con persona	2	3	4	4
Peso (w)	1	1	3	1

Tabla 3 Ejemplo de relación entre vértices con su determinado peso.

El peso (*w*) se determina de la siguiente manera:

1. En el perfil cibernético se mide la compatibilidad de dos personas.
2. Con la encuesta se podrá conocer con quien le gustaría trabajar una X persona.
3. La encuesta grupal se le aplica a los grupos docentes de forma manual e independiente y proporciona la siguiente información: con quien le gustaría trabajar, con quien no y quien del grupo tiene características de líder.

Con todos estos datos, los valores se obtienen de la siguiente forma:

- Si en el resultado al aplicar el perfil cibernético da que son compatibles y en la encuesta las dos personas ponen que quieren trabajar juntas, entonces se le asigna valor 1 a w_i.
- En caso que sean compatibles y en encuesta no se relacionen es 2.
- En caso que sean compatibles y en grupal no se relacionen (deben ser del mismo grupo) es 5.
- En caso que no sean compatibles y en grupal se relacionen es 3.
- En caso que no sean compatibles y en encuesta ambos se relacionen es 1.

- En caso que no sean compatibles y en encuesta no se relacionen es 5.

La relación es la compatibilidad existente entre dos personas o más y el peso es el valor que se le asigna a la distancia entre cada nodo.

Se empezará a crear un grafo tomando al nodo que haya escogido como líder y que cumpla con las competencias exigidas e insertándose en el principio de la lista, de no existir esa persona se busca alguien que cumpla con los requisitos.

2.4.1 Pasos para la selección del segundo algoritmo

- En el algoritmo **eliminación por compatibilidad** se toma la lista original de personas sin proyecto y se crea una matriz simétrica donde los valores matriz [i, j] representa la compatibilidad entre cada uno de candidatos.

- Se escoge un líder de la lista de líderes formada por las preferencias de las personas y como base para la interrelación se aplica la compatibilidad y se buscan todas las personas que son compatibles con él.

- Si la cantidad de compatibles es menor a la deseada, se escoge otro líder y se realiza de nuevo el mismo proceso de búsqueda hasta que se hayan escogido todas las personas, o el umbral establecido por los especialistas sea menor o igual que la distancia euclidiana con el ideal.

- Se busca la persona que más compatibilidades alcanzó y se le agregan las incompatibilidades en orden creciente hasta alcanzar un número superior de personas a las que se pedían.

- Devuelve una lista de personas, donde todos se llevan según diferentes criterios de compatibilidad.

- El algoritmo **repartir por preferencia** aplica la distancia euclidiana para realizar la selección del personal, teniendo en cuanta las preferencias de las personas, solamente del subconjunto formado.

- **Repartir _roles** (Prototipo pro), se entrará el mismo prototipo que en el primer algoritmo explicado

- Se genera el grafo donde la compatibilidad es el peso de las aristas y las personas los diferentes nodos.

Entrada: listado de personas, cant =personas que se necesitan y un umbral según decidan los especialista para hallar la semejanza con el líder que se pretende buscar.

Salida: conjunto de personas compatibles

---

eliminación_ por_ compatibilidad (lista <Persona> *p, cantidad, umbral)

---

1. contador ⇐0
2. posición ⇐1
3. ordenar ascendentemente p (prototipo [1]. Nombre)
4. Crear grafo (p) //genera una matriz donde la relación matriz [i, j] es la compatibilidad entre i y j
5. Nueva lista. Adicionar (p [posición])
6. band1⇐1
7. Band2⇐1

8. Incrementar contador
9. **para** i desde 1 **hasta** cantidad de caminos **hacer**
    - Si posición != i
        - Si matriz [posición, i]> 3 ó vértice adyacente Band2==2
            - matriz [posición, i] ⇐ 0 //Eliminar camino de la relación.
            - Vértice adyacente i Band2=2
        - Sino
            - Incrementar contador
10. Ordenar lista de caminos ascendentemente por pesos.
11. **para** i desde 1 **hasta** cantidad de nodos **hacer**
    - Nueva lista. Adicionar (p. lista [i])
    - **para** j desde 1 **hasta** cantidad de nodos -1 **hacer**
    - Si matriz [lista [i], j] > 3 o vértice adyacente Band2==2
        - matriz [lista [i], j] ⇐ 0. Eliminar camino de la relación.
        - Band2=2
        - Decrementar contador
        - Eliminar (lista [j]).
12. Si contador < cantidad

- Si umbral < líder. distancia
    - Incrementar posición
    - Nodo. Adicionar_ lista(Nueva lista)
    - Nueva lista = NULL
    - Escoger _ vértice líder
- Sino
    - Retorna Nueva lista. Adicionar_ lista (nodo)

13. Sino

- Retornar Nueva lista

Entrada: Prototipo de equipo

Salida: equipo ideal

Repartir _roles (Prototipo pro)

---

1. si p. cant >total de personas
    - mensaje (“no se puede conformar equipo”);
2. sino
    - si p.cant==total de personas
        i. Mylista ⇐ listado de personas
    - sino
        i. Mylista ⇐ **eliminación por compatibilidad** (lista <Persona> *p, cant, umbral)
    - **Repartir por preferencia** (Mylista, pro)
3. Retornar pro

Entrada: lista de personas, prototipo

Salida: prototipo lleno

Repartir por preferencia (lista, prototipo)

---

1. **para** i desde 2 **hasta** pro. cantidad **hacer**
2. pro. Lista. Adicionar (Mylista[1])
3. P $\Leftarrow$ Prototipo$[i]$. Nombre
4. X $\Leftarrow$ 0
5. Si prototipo $[i]$.cantidad==1
   - Si lista(P).cantidad==0
     - X $\Leftarrow$ lista original .buscar mejor ()
     - Prototipo $[i]$.Lista. Adicionar (X)
     - Lista original .Eliminar (X)
   - Sino
     - Prototipo $[i]$.Lista. Adicionar (Lista(P).buscar mejor ())
     - Lista original .Eliminar (Lista(P).buscar mejor ().índex)
6. Sino
   - Si Lista (P).cantidad < Prototipo$[i]$.cantidad

- **Mientras** Lista (P).cantidad != Prototipo$[i]$.cantidad **hacer**
  - Lista (P). Adicionar (lista original .buscar mejor ())

- Lista (P). Ordenar Ascendentemente ()
- Prototipo $[i]$.Lista. Adicionar (Lista (P), Prototipo $[i]$.cantidad)
- Lista original .Eliminar (lista hasta lista. Prototipo $[i]$.cantidad);

Para sacar al líder se realiza una matriz. Donde la primera columna serán los nombres de los estudiantes y las demás las competencias que se miden. Comparándolos con un ideal devolverá el más adecuado por lo que se tomará la menor distancia.

En caso de que no tenga experiencia pasan a ser el rol de programador según criterio de expertos (sacado de encuesta realizada a expertos).

Cada persona tiene un identificador y que nunca se repite. La relación es la compatibilidad existente entre dos personas o mas y el peso es el valor que se le asigna a la distancia entre cada nodo.

Se empezará a crear un grafo tomando al nodo que haya optado como líder y que cumpla con las competencias exigidas e insertándose en el principio de la lista, de no existir esa persona se busca alguien que cumpla con los requisitos. Posteriormente se le pregunta todas sus relaciones y se empiezan a agregar al nuevo grafo que se esta construyendo si el peso es mayor que 1 ese vértice no se agrega por lo que solamente se crearán nodos con sus respectivas aristas igual 1. Al nodo que no se agrega por ser diferente su peso se busca y se cambia a visitado para que no se pase mas por el nodo mencionado. Por lo que se seguirá recorriendo la lista y sumando las aristas. En caso de que la suma de la cantidad que se pide sea menor se vuelve al principio escogiendo otro nodo y de ahí sus respectivas relaciones. Conformando un grafo en donde todos se lleven.

Para la conformación del algoritmo se tiene primeramente una matriz cuadrada simétrica de $(n \times n)$ en la cual la cantidad de filas será la misma que la cantidad de columnas. Donde la lista de personas serán los valores k_i de los índices de cada persona mencionada y la relación entre ellos estará dada por $K[i, j]$.

Cada valor de la relación de i y j estará dado por la compatibilidad entre las personas. Cumpliéndose que si i=j entonces el valor de $K[i, j]$=0 para que no se contemple como una relación entre una misma persona. Si la compatibilidad es 1 significa que son compatibles y si es 5, es que no son compatibles entre si.

2.5 Análisis algorítmico

Para realizar este análisis se dará algunas definiciones y de que forma se podrá saber de los algoritmos planteados cual es el más eficiente para ser propuesto a la facultad para formar parte del software que se encargara de asignar a los estudiantes a los diferentes proyectos.

Análisis de algoritmos

El análisis de algoritmos mediante el uso de herramientas como por ejemplo la evaluación de costos intenta determinar que tan eficiente es un algoritmo para resolver un determinado problema. En general el aspecto más interesante a analizar de un algoritmo es su costo.

Costo de tiempo

Suele ser el más importante e indica cuanto tiempo consume un determinado algoritmo para encontrar la solución a un problema se mide en función de la cantidad o del tamaño de los datos de entrada.

Costo de espacio

Mide cuanta memoria (espacio) necesita el algoritmo para funcionar correctamente.

Para ambos se halla la complejidad para saber la eficiencia. Los métodos Buscar de ambos algoritmos son de orden O(n) y el ordenamiento utilizando quick-sort que es de O(n log n) y los ciclos de O (n). El resultado final doy que la complejidad algorítmica de ambos algoritmos es O (n^3).

2.6 Resultados Preliminares

Producto que esta tesis es parte de un proyecto, y este fue dividido en varias tesis, es imposible entregar una aplicación como resultado ya que depende por completo de los resultados de otros factores. Por lo que se realizó una evaluación por especialista. Para esto se seleccionaron 5 personas como especialista. Para dar una valoración si los algoritmos que se presentan en la tesis ayudan a la toma de decisiones para crear proyectos productivos de gestión.

Todos los especialistas concordaron que este algoritmo sirve para ser utilizado como ayuda a la toma de decisiones. Los especialistas serán nombrados a continuación:

Nombre y Apellidos: Eugenia G. Muñiz Lodos

Título: Lic. Matemática, Msc. Sistemas Digitales, Investigador Auxiliar del ICID, Profesor Titular Adjunto.

Años de experiencia: 34

Argumentación:

En una gran Industria de Software es importante que la conformación de los equipos de trabajo se realice en forma automática. El algoritmo expuesto plantea una interesante aproximación a este problema. Pienso que debe probarse y si es necesario ajustarse.

Nombre y Apellidos: Arturo Arias Orizondo

Título: Ing. Informático

Años de experiencia: 4 años

En una gran Industria de Software es importante que la conformación de los equipos de trabajo. El algoritmo expuesto plantea una interesante y lógica aproximación a este problema. El primer algoritmo merece una fundamentación teórica más fuerte y una comprobación práctica.

El segundo algoritmo podría ser validado por un psicólogo. Se están utilizando herramientas matemáticas (ciencias exacta) para medir aspectos del comportamiento humano (que no es exacto), de allí la necesidad de buscar referencias en estudios similares. Sin embargo, la propuesta sigue una lógica que podría ser usada como punto de partida para la experimentación lo que permitiría validar el método en la práctica.

Nombre y Apellidos: Yudenia Ramírez Mastrapa

Título: Ing. Informático

Años de experiencia: 4 años

La facultad 3, que es el ámbito donde se imbrica este proyecto o tesis de grado, y mas aun, la uci, necesitan realmente el uso y aplicación de algoritmos para la selección y asignación de los estudiantes, profesores, e incluso muchas veces hasta de los recursos materiales, a los proyectos. Es común encontrar recursos humanos asignados a roles de los que no se tiene dominio, o en los que no se encuentra conforme por preferir otras tares, así como grupos en los que las personas que lo conforman no son capaces de trabajar como un equipo unido.

De ahí la mayor importancia en que estos algoritmos sean desarrollados, aplicados y validados con premura de tiempo.

Considero que deberían añadirse y aplicarse en este proyecto el uso de técnicas estadísticas, de simulación y de inteligencia artificial para lograr mejores resultados, aunque creo que como una primera aproximación al asunto es totalmente valido.

Nombre y Apellidos: José Kadir Febrer Hernández

Título: Lic. Ciencia de la Computación, Profesor Asistente Adjunto.

Años de experiencia: 9 años

Siempre que se va a realizar un nuevo proyecto informático tiene mucha importancia el grupo de trabajo que lo desarrollará. Por esto, es vital para el cumplimiento de la tarea, el proceso de selección de este. Ahora, si se puede lograr que esta selección se realice de forma automática sería de gran ayuda.

El algoritmo expuesto en el trabajo plantea una aproximación a este problema. Soy del criterio que se debería poner a prueba y en caso necesario ajustarlo.

Nombre y Apellidos: José Luis León Rosquet

Título: Ing. Automático

Años de experiencia: 4 años

Cuando surge un proyecto, uno de los principales problemas que surge es a la hora de seleccionar las personas y ubicarlas por roles, es por esta razón que manifiesto después de haber realizado un estudio minucioso de las dos variantes que se plantean en la presente tesis, que estos algoritmos pueden ser una forma muy eficaz y rápida de solucionar este problema, además de ayudar a impulsar el avance del proyecto reduciendo al mínimo al posibilidad de error en el transcurso del mismo.

2.6.1 Ventajas y desventajas de los algoritmos

Con los algoritmos propuestos una de las principales ventajas es que cada persona según sus conocimientos podrá ser incluido en la conformación de un nuevo proyecto. Con este algoritmo se podrán dar a conocer personas que nunca habían ocupado un destacado rol en un proyecto. Tratando de unir aspectos sociales que son imprescindibles para una buena convivencia. Logrando con esto que las personas que se escojan sean adecuadas para desempeñar dicha actividad.

Con el primer algoritmo se pueden crear proyectos en donde se tenga en cuenta las preferencias y gustos de las personas, como factor importante para una correcta asignación. Logrando con esto, aspectos que anteriormente no se tenían en cuenta para la conformación del equipo, tratando que cada persona pueda cumplir con la tarea asignada y que el producto que se entregue, tenga la calidad requerida y que cumpla con el tiempo estimado para su terminación. Como desventaja se tiene que el grupo conformado puede o no tener las mejores relaciones. Por lo que depende del vicedecano de producción o especialista que este encargado de crear los grupos de proyecto.

El segundo algoritmo es una variante del primer algoritmo, se podrá utilizar para la creación de proyectos en donde primeramente se tenga en cuenta la compatibilidad entre cada uno de los integrantes. Como principal desventaja tiene que del conjunto de personas sin proyectos, solamente seleccionará un subconjunto de estos, para asignarles los respectivos roles en el nuevo proyecto.

Conclusiones

A las conclusiones que se pueden llegar es que el primer algoritmo propuesto devuelve un equipo de trabajo atendiendo a las preferencias y gustos de las personas, logrando con esto de que la asignación realizada tenga una cierta aceptación por parte del individuo al que se le fue asignada. Este algoritmo según el criterio de varios expertos seria muy aconsejable aplicarlo para de esta forma poder lograr un grupo de trabajo donde cada uno de los integrantes tenga los conocimientos imprescindibles para realizar dicha tarea.

La propuesta del segundo algoritmo es si se pretende conformar un equipo donde se mida la compatibilidad con las personas, aplicando criterios tanto de índole sicológica como matemática llegamos a la conformación de un equipo lo más factible posible, como desventaja tiene que al ser eliminado un conjunto de competidores puede ser que entre las personas que devuelve el subconjunto no se encuentre por lo que si vamos a hablar de equipo mas idóneo es aconsejable utilizar el primer algoritmo.

Conclusiones

Después de realizado un análisis del estado del arte sobre diferentes técnicas de diseño de algoritmos y de conocer como se mueve el mundo en cuanto a la asignación de personal teniendo en cuenta criterios de personalidad, preferencia y relaciones interpersonales.

A partir de este análisis se propuso objetivos que fueron cumplidos y se arriba a las siguientes conclusiones:

Se desarrolló un algoritmo de ayuda a la toma de decisiones, para asignar personal a los proyectos productivos en la facultad 3, lográndose con esto, que se puedan seleccionar estudiantes de una forma automatizada a los proyectos y que dicha captación más rápida. Teniendo en cuenta las preferencias y gustos de las personas.

Este algoritmo será capaz de devolver personas antes no conocidas que son buenos para realizar una *x* tarea. Brindando un elevado grado de justicia en el proceso de selección. Si no devuelve un equipo óptimo para realizar dicha tarea escogerá una solución los más factible posible.

La propuesta del segundo algoritmo es para proyectos que se quieran conformar teniendo en cuenta la compatibilidad entre las personas, siendo muy útil si se quiere tener en cuenta las relaciones entre los integrantes del equipo.

Recomendaciones

Después de haberse realizado esta tesis y los resultados alcanzados se plantea una serie de recomendación a tener en cuenta.

Las recomendaciones son:

1. Realizar un estudio después de la puesta en funcionamiento de forma practica para un ajuste de las variables que ponderan cada vector.

2. Que se archive toda la información pertinente a la asignación de personal y que se recojan datos de cuan buena fue la asignación.

3. Culminados los proyectos a los que se le aplicará dicho algoritmo, se le haga una encuesta al jefe de proyecto si esta satisfecho con la asignación de personal realizada, para tener en cuenta el desempeño de los estudiantes en futuras asignaciones.

4. Aplicar al algoritmo inteligencia artificial basada en aprendizaje con la utilización de los datos estadísticos que se generarán de las recomendaciones anteriores.

5. Implementar los algoritmos propuestos como parte de un software que automatice la asignación de personal para ser aplicado en la facultad 3.

6. Proponer después de probado su implantación en toda la Universidad

Bibliografía

Algoritmos y Estructuras de Datos III.htm. 2007, Disponible en: http://www2.dc.uba.ar/materias/aed3/2007-01/Documents/practicas200701.ps.

Capítulo 4 Algoritmos sobre grafos. [Notas de clases]. 2004, Disponible en: http://aulavirtual.uji.es/file.php/586/teoria-semestre1/tocho4.pdf.

Grafos Disponible en: http://www.denega.com.ar/facu/ED3/.

Logica difusa (Fuzzy Logic) Disponible en: http://answermath.com/Panels/fuzzy/esp-fuzzy1.htm.

LÓGICA DIFUSA. ¿Una concepción infinitesimal de la verdad? Disponible en: http://personal.telefonica.terra.es/web/mir/ferran/kosko.htm.

OPTIHPER Asignación Optimizada de Horarios al Personal Disponible en: http://www.dsic.upv.es/users/ia/gps/optihper/Doptihper.html.

Ordenación interna [Consultado el: 2 de abril de 2007]. Disponible en: http://www.algoritmia.net/articles.php?id=31.

Quicksort Disponible en: http://es.wikipedia.org/wiki/Quicksort

Sort_algorithms [Consultado el: 3 de abril de 2007]. Disponible en: http://en.wikipedia.org/wiki/Category:Sort_algorithms.

Tema 4. Grafos. 2003-2004,

ALBA, E.; LAGUNA, M., et al. Métodos Evolutivos. 2003, nº Disponible en: http://www.uv.es/~rmarti/paper/docs/heur3.pdf.

ALONSO JIMENEZ, J. A.; GRACIANI DIAZ, C., et al. Representacion de problemas como espacio de estados. En 2005.

BAIER ARANDA, J. Grafos. PUC:

BARÓN, J. H. y RIVERA, S. S. Matriz de vulnerabilidad difusa. En Iberian Latin-American Congress on Computational Methods in Engineering – XX CILAMCE. San Pablo, Brasil. 1999.

BEJAR, J. Resolución de problemas. Algoritmos de búsqueda. 2006, nº Disponible en: http://www.lsi.upc.es/~bejar/ia/material/teoria/busqueda06071q.pdf

BENAVENT, E. Teoría de Grafos. Notas de clases. 52 p.

BERRIZBEITIA, P. Algoritmos Deterministas de Primalidad. Departamento de Matemáticas Puras y Aplicadas. Universidad Simón Bolívar, 2004.

CAMPOS, J. Algoritmos voraces Disponible en: http://lsi.upc.es/iea/transpas/2_voraces/sld001.htm.

CANÓS DARÓS, L.; CAÑO ALEGRE, C., et al. La Ordenación de candidatos en la selección de personal. Rect@, 2006, vol. Actas_14, nº 1, Disponible en: http://www.doaj.org/doaj?func=abstract&id=178274&toc=y. ISSN 1575605X.

CORMEN, T. H.; LEISERSON, C. E., et al. Introduction to Algorithms. Second Edition ed. The MIT Press, 2001. ISBN 0-07-013151-1.

CORREA HENAO, G. J.; ANAYA MARTÍNEZ , E., et al. Implementación Computacional para el Apoyo a la Toma de Decisiones, Utilizando Metodologías

Difusas. Facultad de Minas. Grupo de Investigación en Sistemas e Informática. UNIVERSIDAD NACIONAL DE COLOMBIA. , 2004.

COTO, E. Algoritmos Básicos de Grafos. Universidad Central de Venezuela, 2003.

CRESPO, J. Fuzzy Open Regression Tool. 2004, Disponible en: http://www.fdi.ucm.es/profesor/jcrespo/web/index.html.

CRESPO YÁÑEZ, F. J. Un Modelo Paramétrico Matemático Difuso para la Estimación del Esfuerzo de Desarrollo del Software. Tesis Doctoral, Universidad de Alcala,

CUNQUERO, R. M. Algoritmos Heurísticos en Optimización Combinatoria. nº

DÍEZ, J. L.; NAVARRO, J. L., et al. Algoritmos de agrupamiento en la identificación de modelos borrosos. 2004, nº

DÍEZ , J. L.; NAVARRO, J. L., et al. Algoritmos de clustering en la identificación de modelos borrosos. nº

GÁLVEZ LIO, D. Sistemas Basados en el Conocimiento. Especialización de Inteligencia Artificial. Departamento de Ciencia de la Computación, Facultad de Matemática, Física y Computación. Universidad Central “Martha Abreu” de Las Villas, 1998.

GARCÍA BORROTO, M. y FUENTES CORRALES, A. Implementación de un Shell para la construcción de Sistemas Basados en el Conocimiento borrosos. Tesis de grado, Facultad Matemática Física Computación. Universidad Central de las Villas, 2000.

GARCÍA NIETO, J. M. Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos. septiembre de 2006, nº

GOIC, M. y CABALLERO, C. Aplicación de algoritmos geneticos para el mejoramiento del proceso de programación del rodaje en la industria del cine independiente. Dpto de Ingeniería industrial. Universidad de Chile, 2005.

GÓMEZ, J. G. Conjuntos y Sistemas Difusos (Lógica Difusa y Aplicaciones). Departamento de Lenguajes y Ciencias de la Computación. Universidad de Málaga,

GONZÁLEZ, E. L.; CUERVO, C. M., et al. La Selección del personal con un algoritmo genético borroso. nº

GRACIA, O. R. Herramientas de cribado virtual aplicadas a inhibidores de tirosina. Constribucion al desarrollo del programa Pralins para el diseño de quimiotecas combinatorias. Tesis Doctoral, Universitat Ramon Llull,

GUEREQUETA, R. y VALLECILLO, A. Técnicas de diseño de algoritmos. Universidad de Málaga, 1998. ISBN 84-7496-666-3.

GUTIÉRREZ MARTÍNEZ, I. Un Modelo para la Toma de Decisiones usando Razonamiento Basado en Casos en condiciones de Incertidumbre. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas, Departamento de Ciencias de la Computación. Universidad Central Marta Abreu,
2003.

HEILEMAN, G. L. Estructura de datos, algoritmos, y programación orientada a objetos La Habana: Félix Varela, 2003. ISBN 0-07-027893-8.

HERNÁNDEZ VALADEZ, E. Algoritmo de clustering basado en entropía para descubrir grupos en atributos de tipo mixto. Tesis de Maestría, Dpto de ingeniería eléctrica sección de computación. Centro de Investigación y de estudios avanzados del Instituto politécnico nacional, Agosto de 2006.

IGLESIAS FERNÁNDEZ, C. Á. Introducción a la Inteligencia Artificial Distribuida. Noviembre 1999, nº

J. CAVUTO, D. An exploration and development of current Artificial Neural Network theory and applications with emphasis on artificial life. Master of Engineering, The cooper union Albert Nerken School of Engineering, 1997.

JIMÉNEZ, M.; RODRÍGUEZ, M. V., et al. Aplicación de la programación compromiso a la resolución de un programa multiobjeto posibilístico. nº

LLAURÓ FÁBREGAS, X. Estudio de la aplicacion de sistemas basados en el conocimiento a la operacion de una planta de tratamiento de residuos solidos urbanos por valorizacion energetica. Doctoral, Universidad de Girona, 1999.

LÓPEZ, D. y RIERA, E. Implementación del algoritmo de extracción de cálculosinvariantes en ICTÍNEO Disponible en: http://www.ac.upc.edu/pub/reports/DAC/1995/UPC-DAC-1995-24.ps.Z.

LOURENCO, H. R. D.; DE LA FIGUERA, D. S., et al. Métodos de solución de problemas de asignación de recursos sanitarios. Fundación BBVA, 2004.

MARENTES CUBILLOS, L. A. Aproximación con algoritmos evolutivos para el problema de selección de proyectos con restricciones lineales y variables binarias bajo ambiente de incertidumbre. nº Disponible en: http://triton.uniandes.edu.co:5050/dspace/handle/1992/361.

MARTÍ, R. Procedimientos Metaheurí¸sticos en Optimización Combinatoria. 2003, nº p. 60. Disponible en: http://www.uv.es/~rmarti/paper/docs/heur1.pdf.

MARTÍNEZ, L.; MATA, F., et al. Un Modelo Lingüístico para la Resolución de Problemas de Toma de Decisión en Grupo para el Sector Turístico. En IV Congreso “Turismo y Tecnologías de la Información y las Comunicaciones”. TuriTec. 2002.

MENDAÑA CUERVO, C. y LÓPEZ GONZÁLEZ, E. La Gestión Presupuestaria de Distribución con un Algoritmo Genético Borroso. 2005, Disponible en: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642005000300007&lng=es&nrm=iso.

MORALES, E. Busqueda, Optimizacion y Aprendizaje.

MUÑOZ GONZÁLEZ, O. L. Procedimiento de Asignación Óptima en un Entorno Difuso. (Aplicación en la Asignación de Responsabilidades en un Grupo de Desarrollo de Software ante un Proyecto). Tesis de Maestría,

NORIEGA, A.; AGUADO, A., et al. Aplicación de Técnicas Neuro-Difusas para el Diseño de un Controlador. 2005, Disponible en: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642005000300006&lng=es&nrm=iso. ISBN 0718-0764.

PEÑA, M.; SCIASCIO, F. D., et al. Identificación de la estructura de un modelo borroso del tipo TAKAGI-SUGENO. 2002,

PERAZA RODRIGUEZ, F. Inteligencia Artificial Disponible en: http://www.monografias.com/trabajos10/trabajos/trabajos.shtml.

PIÑERO PÉREZ, P. Y. GACom biblioteca de componentes de Algoritmos Genéticos. Tesis de grado, Facultad de Matemática, Física y Computación. Universidad Central "Marta Abreu" de Las Villas, 2000.

---. Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing. UNIVERSIDAD CENTRAL “MARTA ABREU” DE LAS VILLAS, 2005.

PINOCHET OLAVE, R. Los Sistemas Informáticos expertos de toma de decisiones y la voluntad como elemento de validez del negocio jurídico. 2003. vol. 9, 161-184 p. ISBN 0718-0012.

RODRÍGUEZ, J. O.; SUBIAS, A. C., et al. Modelos de programación matemática para la asignación de horarios de trabajadores en un centro de servicios. nº

RODRÍGUEZ VILLALOBOS, A. Grafos. 2003, nº Disponible en: http://ttt.upv.es/%7Earodrigu/Grafos/index.htm.

SEDGEWICK, R. Algorithms. Addison-Wesley, 1983. ISBN O-201 -06672-6.

---. Algoritmos en C++. Ediciones Díaz de Santos, 1995, Disponible en: http://books.google.com/books?id=8OBlquzq83oC&pg=PA4&lpg=PA1&ots=M08OTW0OPW&dq=algoritmos+utilizados+para+la+asignar+personas&hl=es&sig=L__B9VeB_6JJ4KFcmX_5Fv-58sI. ISBN 0201625741.

TENENBAUM, A. M.; LANGSAM, Y., et al. Estructuras de datos en C. 1999, Disponible en: http://bibliodoc.uci.cu/pdf/9688802565.pdf. ISBN 968-880-256-5.

TORRES, L. M. PPP. Quito, Ecuador: 1999, 233 p. Disponible en: http://www.math.epn.edu.ec/%7Eltorres/algo_all.ps.

URRUTIA BADILLO, Y. y WONG JAUMA, K. L. El enfoque de Competencia Laboral en la Gestión de los Recursos Humanos. Una aplicación práctica en la empresa de Seguro Estatal Nacional. Tesis para obtener el título de Licenciado en Economía., Junio 2004.

VALDERRUTEN, A. Algoritmos: Algoritmos voraces Disponible en: www.lfcia.org/alg.

YOLIS, E. Algoritmos genéticos aplicados a la categorización automatica de documentos. Tesis de grado, Facultad de Ingeniería. Universidad de Buenos Aires, 2003.