UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

# DISEÑO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA COMBINACIÓN BALANCEADA DE NUTRIENTES UTILIZANDO LÓGICA DIFUSA Y BÚSQUEDAS HEURÍSTICAS

Luisa Mercedes Colón Salazar

Barquisimeto, 2005

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

# DISEÑO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA COMBINACIÓN BALANCEADA DE NUTRIENTES UTILIZANDO LÓGICA DIFUSA Y BÚSQUEDAS HEURÍSTICAS

Trabajo presentado para optar al grado de
Magíster Scientiarum

Por: Luisa Mercedes Colón Salazar

Barquisimeto, 2005

## DEDICATORIA

*A mi adorada hija Luisa Paola,*
*amor que me inspira a luchar para alcanzar*
*mis metas y superarme cada día más.*
*A mi esposo Frank, que siempre ha apoyado mis ideas y*
*ha sido mi sustento en todo momento.*

# AGRADECIMIENTOS.

A mi tutora Lic. Belkis López de Lameda, por incentivarme a adquirir conocimientos sobre lógica difusa, los cuales fueron el basamento para iniciar esta investigación.

Al Ing. Carlos Lameda, por haber estado siempre dispuesto a colaborar con ideas y conocimientos valiosos que impulsaron el desarrollo de esta investigación.

A los Ing. Wilmer Ozal y Manuel Picón, con quienes compartí ideas, conocimientos y esfuerzos para el logro de objetivos relacionados a la maestría cursada.

A la profesora Omaira Peña, por prestarme la ayuda necesaria para desarrollar ideas y convertirlas en una realidad.

A la Lic. Mariana Tierno, por aportar principios y conocimientos importantes, que permitieron integrar el área de nutrición con las técnicas de inteligencia artificial utilizadas a lo largo de la investigación.

Al Ing. José L. Rojas, por aportar ideas en el diseño y desarrollo de la herramienta computacional.

A todos los profesores con los que compartí ideas y conocimientos a la largo de la maestría, especialmente, a la Lic. Belkis López de Lameda y a la Ing. Margarita Pereira, ya que sus consejos marcaron una pauta en mi forma de ver y resolver situaciones que se presentaron en el camino transitado para llegar a esta meta.

# ÍNDICE

# ÍNDICE DE TABLAS

**PÁG.**

# ÍNDICE DE FIGURAS

**PÁG.**

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

# DISEÑO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA COMBINACIÓN BALANCEADA DE NUTRIENTES UTILIZANDO LÓGICA DIFUSA Y BÚSQUEDAS HEURÍSTICAS

**AUTOR:** Ingº Luisa Mercedes Colón Salazar
**TUTOR:** Lic. Belkis López de Lameda
**AÑO :** 2005

## RESUMEN

El presente trabajo de grado consiste en el diseño e implementación de una herramienta computacional para obtener compuestos alimenticios que tengan una combinación balanceada de nutrientes, utilizando técnicas de inteligencia artificial. Esta herramienta fue aplicada para evaluar la cantidad de nutrientes en los alimentos de una comida, con el objeto de adaptarla a determinados requerimientos nutricionales. Tales alimentos se caracterizaron por poseer imprecisión y difusidad, tanto en sus cantidades como en su composición. Por medio de la técnica de emparejamiento de patrones difusos ponderada, se evaluaron compuestos alimenticios calculando índices de compatibilidad en términos de posibilidad y necesidad, los cuales permitieron realizar el balance de sus nutrientes, y donde las cantidades de los mismos fueron representadas por medio de números difusos trapezoidales, estableciéndose una relación entre éstas y normas de nutrición previamente establecidas. El modelo matemático para realizar los cálculos fue fundamentado en aritmética de números difusos, la cual permitió modelar la entrada y el conocimiento en la base de datos, además de permitir el cálculo de las cantidades difusas totales de todos los nutrientes requeridos para una combinación balanceada. Definidos los problemas de combinación presentes en el requerimiento nutricional de un usuario venezolano, esta herramienta es capaz de corregir o transformar los alimentos especificados en la entrada de datos utilizando un algoritmo de búsqueda heurística, el cual permitió encontrar un compuesto que se adaptó de forma adecuada a dicho requerimiento, aplicando el conjunto más pequeño de cambios aceptables.

**Palabras Claves:** Lógica Difusa, Aritmética Difusa, Incertidumbre, Emparejamiento de Patrón Difuso, Búsqueda Heurística, Nutriente.

# INTRODUCCIÓN

La lógica difusa es un supra conjunto de la lógica (Bivalente) convencional, la cual ha sido extendida para manejar el concepto de verdad parcial, es decir, valores de verdad entre "completamente verdadero" y "completamente falso". Fue introducida por L. A. Zadeh en los 60's como un medio para modelar la incertidumbre del lenguaje natural. Por lo tanto, es una forma de razonamiento que incorpora criterios múltiples para tomar decisiones y valores múltiples para evaluar posibilidades.

En la lógica clásica bivalente, se espera derivar una solución decidiendo por sí o por no, si cada una de las restricciones o parámetros es verdadera o falsa; en cambio en la lógica difusa, es admisible usar escalas de condiciones (restricciones) y matices (flexibilidad) en los valores numéricos. En el intervalo [0,1] puede caber cualquier valor de verdad sin necesitar ser un número entero.

Básicamente, la lógica difusa tiene cuatro facetas principales: (a) la faceta lógica, L; (b) la faceta de teoría de conjuntos, S; (c) la faceta relacional, R; y (d) la faceta epistémica, E. [Yen y Langari (1999)]

La faceta lógica, es un sistema lógico o, más generalmente, una colección de sistemas lógicos, los cuales incluyen como caso especial los sistemas bivalentes y multivalentes. Una característica importante de esta faceta, es que las reglas de inferencia sirven como reglas, las cuales gobiernan la propagación de restricciones difusas. Además, juega un papel fundamental en aplicaciones de representación de conocimiento e inferencia de información la cual es imprecisa, incompleta, incierta o parcialmente verdadera.

La faceta de teoría de conjuntos, está involucrada con clases o conjuntos cuyos límites no están definidos rigurosamente. El trabajo inicial sobre lógica difusa fue

enfocado en esta faceta. La mayoría de aplicaciones de lógica difusa en matemáticas, han sido y continuarán siendo relacionadas a este enfoque.

La faceta relacional, está relacionada con representación y manipulación de relaciones y funciones definidas imprecisamente.

Por último, la faceta epistémica de la lógica difusa está unida a la faceta lógica y está enfocada en aplicaciones de representación de conocimiento, sistemas de información, bases de datos difusas, y en las teorías de posibilidad y necesidad. En esencia esta faceta está relacionada con significado, conocimiento y decisión.

Las facetas de teoría de conjuntos y epistémica son una base para el desarrollo de la técnica de emparejamiento de patrones difusos, ya que ésta aplica conceptos de la teoría de conjuntos difusos y de la teoría de la posibilidad, con la finalidad de tomar en cuenta la imprecisión y la incertidumbre que poseen determinados valores, los cuales tienen que ser comparados en un proceso de emparejamiento. Básicamente, un proceso de emparejamiento de patrón involucra un patrón para describir algunos requerimientos y una base de conocimiento donde la data está organizada, es decir, los datos que son idénticos a la especificación expresada en el patrón son buscados y recuperados. [Dubois y otros (1988)]

El presente trabajo de investigación consiste en el diseño e implementación de una herramienta computacional para obtener compuestos alimenticios adaptados a los requerimientos nutricionales del venezolano, que tengan una combinación balanceada de nutrientes aplicando técnicas de inteligencia artificial, tales como, lógica difusa y búsquedas heurísticas. Con esta herramienta se evalúa la cantidad de nutrientes en los alimentos de una comida, para sugerir una modificación en la composición de la misma, y de esta forma satisfacer un requerimiento nutricional, el cual cumpla a su vez, con normas de nutrición previamente establecidas. Pretendiéndose además con la herramienta, el manejo y representación adecuada de las cantidades vagas o imprecisas de los valores de entrada y del conocimiento almacenado en la base de datos relacionadas a tales alimentos y a su composición nutricional.

La herramienta permite evaluar el contenido nutricional de una comida por medio de un emparejamiento difuso de patrones entre nutrientes y normas de nutrición

previamente establecidas, calculando índices de compatibilidad en términos de posibilidad y necesidad, los cuales permiten determinar si existe una combinación balanceada de nutrientes. El modelo matemático para realizar los cálculos está fundamentado en operaciones de números difusos trapezoidales, las cuales permiten modelar las difusidades e imprecisiones en las entradas de datos y en el conocimiento almacenado. Un compuesto alimenticio adaptado a determinados requerimientos nutricionales, se obtiene añadiendo o removiendo cantidades de alimentos o cambiando el peso de los mismos por medio de un algoritmo de búsqueda heurística, el cual permite realizar el balanceo del compuesto aplicando un número mínimo de operaciones de balanceo.

Por lo tanto, el problema de investigación se basó en el diseño e implementación de una herramienta computacional para obtener compuestos alimenticios que tengan una combinación balanceada de nutrientes, con la finalidad de lograr de manera eficaz una combinación o compuesto final, adaptado a los requerimientos nutricionales de un usuario venezolano y a normas de nutrición previamente establecidas y que además, permita manejar y representar de forma adecuada cantidades vagas o imprecisas, utilizando técnicas de inteligencia artificial.

# CAPÍTULO I
# EL PROBLEMA

## Planteamiento del Problema

Existen ocasiones en las que es necesario combinar un cierto número de componentes, por ejemplo, nutrientes, minerales, vitaminas, sustancias líquidas, sustancias sólidas, entre otros; para obtener un compuesto o producto final que tenga una combinación balanceada de esas sustancias o ingredientes.

En el caso de la combinación de alimentos para obtener una comida adaptada a ciertos requerimientos nutricionales, puede presentarse el caso, en que los valores de algunos nutrientes existentes en esos alimentos sean conocidos de forma imprecisa y algunas veces completamente desconocidos, es decir, que no se tiene un conocimiento exacto de sus valores. Por ejemplo, la cantidad de glúcidos en 100 grs. de durazno no es constante, ya que ésta depende de su procedencia, estado de madurez, etc. También existe otra fuente de imprecisión relacionada al conocimiento exacto de la persona que va a realizar la combinación, en cuanto a, la cantidad específica que se necesita de tales componentes. En muchos casos, al realizar estas combinaciones se utilizan unidades de medida poco precisas, expresadas por ejemplo, en un número de porciones (cucharadas, tazas, rebanadas, trozos, vasos, etc.) o también en gramos.

Por lo tanto, de lo anteriormente expuesto se presenta un problema de combinación o mezcla de componentes, más específicamente, combinación de alimentos para alcanzar un balance o equilibrio adecuado de las cantidades de sus nutrientes, y de esta manera lograr una combinación final deseada que satisfaga un

requerimiento nutricional y que a su vez cumpla con normas de nutrición previamente establecidas.

Bien es sabido, que este procedimiento de combinación puede hacerse manualmente mediante ensayo y error añadiendo o removiendo alimentos. Pero esto es muy difícil de llevarse a cabo hasta el final, ya que al modificar el peso de un alimento se producen desequilibrios en las cantidades de los nutrientes que intervienen en el proceso, lo cual conduce a un balance al mismo tiempo de los mismos. Siendo esto una tarea tediosa que toma mucho tiempo y cuyos resultados pueden ser aún imprecisos o difusos.

Por lo tanto, se hace necesario contar con una herramienta adecuada para realizar este tipo de proceso, la cual permita manejar la vaguedad o imprecisión presente en las cantidades y en la composición nutricional de los alimentos. Es decir, que permita obtener la combinación balanceada de nutrientes en un cierto número de alimentos, para lograr un compuesto alimenticio adaptado a los requerimientos nutricionales de un usuario y que al mismo tiempo cumpla con normas de nutrición previamente establecidas, utilizando técnicas de inteligencia artificial, tales como, lógica difusa y búsquedas heurísticas.

Por consiguiente, se puede resumir el problema de investigación como la dificultad para obtener compuestos alimenticios adaptados a los requerimientos nutricionales de los venezolanos con una combinación balanceada de nutrientes, y la dificultad para manejar la vaguedad o imprecisión de las cantidades de los alimentos (tazas, vasos, rebanadas, cucharadas, trozos, etc.) y de su composición nutricional. Se vislumbra que el uso de técnicas de inteligencia artificial integradas e implementadas en una herramienta computacional permiten resolver este tipo de problema.

Por lo tanto, el problema de investigación se orientará en diseñar e implementar una herramienta computacional para obtener compuestos alimenticios que tengan una combinación balanceada de nutrientes, con la finalidad de lograr de manera eficaz una combinación o compuesto final adaptado a los requerimientos nutricionales de un usuario y a normas de nutrición previamente establecidas y que permita además,

manejar y representar de forma adecuada cantidades vagas o imprecisas, utilizando técnicas de inteligencia artificial.

## Objetivo General.

Diseñar e implementar una herramienta computacional para obtener compuestos alimenticios adaptados a los requerimientos nutricionales de los venezolanos, los cuales tengan una combinación balanceada de nutrientes utilizando lógica difusa y búsquedas heurísticas.

## Objetivos Específicos.

1. Analizar los requerimientos nutricionales de los venezolanos para definir los valores de entrada y el dominio difuso del conocimiento a almacenar.
2. Establecer criterios para la representación de los valores de entrada.
3. Establecer criterios para el diseño de la base de datos de compuestos alimenticios.
4. Establecer criterios para la interacción herramienta-usuario.
5. Seleccionar la técnica de emparejamiento de patrones difusos que permita evaluar el balance nutricional de un compuesto alimenticio.
6. Analizar algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios, con la finalidad de comparar los métodos empleados en la búsqueda de soluciones.
7. Desarrollar la herramienta implementando las técnicas de lógica difusa y búsquedas heurísticas.
8. Evaluar la herramienta para determinar cuál de los algoritmos heurísticos de balanceo implementados es más eficiente para lograr una combinación balanceada de nutrientes.

## Justificación e Importancia.

La importancia del diseño y desarrollo de una herramienta computacional que integre técnicas de inteligencia artificial, específicamente, técnicas de lógica difusa y búsquedas heurísticas, para obtener compuestos alimenticios que tengan una combinación balanceada de nutrientes, consiste en que ayudará a manejar y a representar de una forma más adecuada la vaguedad o imprecisión presente tanto en las cantidades como en la composición nutricional de sus componentes. Permitiendo automatizar procedimientos de combinación de alimentos, que realizados por una persona manualmente agregando o removiendo sus cantidades por ensayo y error, serían costosos en tiempo y recursos; y cuyos resultados podrían seguir siendo imprecisos o difusos, es decir, compuestos desbalanceados en cuanto a tales cantidades.

Por el contrario, si se tiene una herramienta de software que no integre técnicas de inteligencia artificial para realizar tales combinaciones, el proceso sería más rápido, pero los resultados no serían adecuados o ajustados a la realidad, ya que el conocimiento almacenado acerca del contenido nutricional de los alimentos y las cantidades de los mismos en la captación de los datos de entrada, deben ser representados en forma vaga o imprecisa, siendo la utilización de técnicas de inteligencia artificial la mejor forma de realizar éstas.

Una herramienta computacional con tales características podría servir de apoyo en establecimientos, como por ejemplo, unidades de nutrición de clínicas u hospitales, comedores universitarios, entre otros, donde se requiera un conocimiento para el manejo de cantidades vagas o imprecisas de alimentos, con la finalidad de lograr mayor precisión en el balanceo del contenido nutricional de los mismos, para adaptarlos de forma adecuada a las necesidades nutricionales de cada usuario.

Esta herramienta también podría servir de apoyo a una persona que quiera por si misma controlar su consumo calórico diario y evaluar la composición nutricional de los alimentos escogidos para una comida, corrigiendo sus cantidades difusas o

imprecisas y adaptándolas a sus requerimientos nutricionales sin necesidad de visitar frecuentemente a un nutricionista .

## Alcance y Limitaciones.

### Alcance.

La herramienta computacional implementa la técnica de emparejamiento de patrones difusos ponderada para evaluar la composición nutricional de un compuesto alimenticio requerido por un usuario. Con esta técnica se calculan por medio de la intersección de líneas rectas de números difusos, índices de compatibilidad en términos de posibilidad y necesidad, que permiten determinar el grado de balance nutricional del compuesto, y de esta forma, definir los posibles problemas de combinación (si los hay) en sus componentes, indicando los nutrientes responsables del desbalance.

El modelo matemático para realizar los cálculos está fundamentado en operaciones de números difusos trapezoidales, las cuales además, permiten modelar la entrada y la representación del conocimiento en la base de datos de compuestos alimenticios. El uso de estas operaciones permite además, calcular las cantidades totales de todos los componentes requeridos para una combinación balanceada de nutrientes, estableciéndose una relación entre éstas y normas de nutrición previamente establecidas. Por ejemplo, la norma: “el total de grasas en un compuesto alimenticio debe ser considerado entre 20% y 30% del consumo calórico total” puede ser trasladada a la representación trapezoidal (20,30,5,5) si se supone que los valores límites tienen un máximo de imprecisión del 5%.

La herramienta da la facilidad al usuario de describir todos los componentes alimenticios para obtener una combinación balanceada de los mismos tomando en cuenta sus nutrientes básicos. Por lo tanto, éste debe especificar cuantas comidas se desean consumir en un día, con la finalidad de repartir las necesidades energéticas de

las mismas, haciéndose esto en base a su edad, estatura y peso. Luego el usuario escoge los tipos de componentes y sus cantidades para obtener la combinación balanceada. Es decir, podría indicar que quiere comer para el almuerzo un bistec mediano a la plancha, una taza de ensalada de hortalizas y una porción grande de papas fritas.

Los intervalos difusos trapezoidales son modelados por una 4 - tupla (a, b, α, β), la cual es una distribución de posibilidad que permite modelar valores precisos, intervalos ordinarios y números difusos. En esta representación los valores a y b representan el núcleo del número difuso, y los valores α y β representan el soporte derecho e izquierdo respectivamente. Por otra parte, el modelado de la imprecisión del conocimiento almacenado consiste en la obtención de distribuciones de posibilidad trapezoidal, a partir de las indicaciones del usuario en la interfaz de captación de datos. Por lo tanto, la definición de criterios para la representación de los valores de entrada consiste en si son nítidos o difusos.

Se establecen categorías lingüísticas para los desbalances, es decir, conjuntos difusos que representan cantidades de componentes tales como: baja, normal y alta. Si los indicadores de compatibilidad ( $\Pi$ , para la posibilidad y **N,** para la necesidad) para un componente no clasifican en la categoría "normal" para indicar que se ha obtenido una combinación balanceada de nutrientes, se debe realizar un cálculo de compatibilidad con las otras dos categorías (baja y alta) en las cuales es necesario que $\Pi$ clasifique, esto con la finalidad de obtener una información detallada de los problemas de combinación de nutrientes presentes en el compuesto requerido por el usuario.

Por lo tanto, definidos los problemas de combinación en el requerimiento nutricional, esta herramienta es capaz de sugerir una modificación en la composición de la comida con la finalidad de balancearla, de modo que se satisfaga de forma adecuada dicho requerimiento. La modificación de un componente implica la modificación simultánea de otros nutrientes que intervienen en la combinación. Es decir, las cantidades de los componentes pueden ser modificadas utilizando un algoritmo de búsqueda heurística, donde se asigna un costo a estas acciones de

transformación. Tales acciones sobre los componentes alimenticios son: (a) aumentar su cantidad, (b) disminuir su cantidad y (c) eliminar el componente. Por lo tanto, para el compuesto deseado para almorzar descrito anteriormente, la aplicación del algoritmo de búsqueda heurística podría indicar, por ejemplo, que se disminuya la porción de papas fritas.

**Limitaciones.**

1. El balanceo de compuestos alimenticios se realiza en base a los requerimientos nutricionales básicos diarios.
2. Los nutrientes básicos considerados en el balance son: carbohidratos, proteínas y grasas.
3. El tipo de usuario considerado consiste en una persona sana que quiere consumir alimentos de forma balanceada. Por lo tanto, no se consideran enfermedades que conduzcan a la evaluación de requerimientos nutricionales especiales.
4. Las operaciones sobre los componentes alimenticios consisten en: (a) aumentar su cantidad, (b) disminuir su cantidad y (c) eliminar el componente.
5. Los cálculos con números difusos toman un tiempo que no es percibido por el usuario, solamente las operaciones de balanceo de compuestos alimenticios pueden provocar retardos.
6. El cálculo del índice de masa corporal (IMC) fue en base a un procedimiento general para personas masculinas o femeninas mayores de 18 años, con requerimientos calóricos normales.

# CAPÍTULO II
# MARCO TEÓRICO

## Antecedentes

En el emparejamiento patrones difusos la noción de compatibilidad (b. Dubois y Prade, 1985) de un conjunto difuso con respecto a otro juega un papel fundamental. Este concepto introducido por Zadeh (1978), intenta representar el valor de verdad de un predicado difuso con respecto a otro considerado como hecho actual. Un predicado difuso puede ser visto como un conjunto difuso A sobre algún universo U, el cual puede ser usado para calificar algún objeto x, a través de una afirmación de la forma "x es A", por ejemplo, "John es alto". El valor de verdad (relativo) de "x es A" con respecto a la declaración (verdad garantizada) "x es F" es definido como un conjunto difuso en el intervalo [0,1], el cual es denotado por ι, con función miembro denotada por: $\forall t \in [0,1]$, $\mu_\iota(t) = \sup\{ \mu_F(u) \setminus t = \mu_A(u), u \in U\}$.

Claramente en esta definición el principio de extensión es aplicado a la función miembro $\mu_A(u)$, la cual es vista como un conductor desde U al intervalo [0,1] del conjunto difuso *F*, es decir, el grado de membresía de *F* en *A*. En cambio, dado un conjunto difuso en [0,1], donde ι es visto como un valor de verdad difuso, entonces la declaración calificada verdadera "*X* es *A* es ι" es considerada como semánticamente equivalente a la declaración absoluta "*x es F*", donde $\mu_F = \mu_\iota \circ \mu_A$.

Cuando $F = A$ entonces; $\mu_\iota(t) = t$ si existe un *u*, tal que $t = \mu_A(u)$, en el caso contrario $\mu_\iota(t) = 0$. En este caso ι puede ser interpretado como "verdadero". En un contexto difuso se define el valor de verdad difuso por la función miembro: $\forall t \in [0,1]$, $\mu_{true}(t) = t$, lo cual conduce que $\forall A$, $\mu_{true} \circ \mu_A = \mu_A$. Es claro que si *F* es un

conjunto nítido, entonces, $\mu_A(F) = \{\mu_A(u) \setminus u \in F\}$, por lo tanto, $\sup\mu_A(F) = \sup_{u \in F} \mu_A(u)$ y $\inf\mu_A(F) = \inf_{u \in F} \mu_A(u)$.

En términos de medidas de posibilidad de eventos difusos (Zadeh, 1978), $\sup\mu_A(F) = \Pi(A \setminus F)$ y $\inf\mu_A(F) = 1 - \Pi(\bar{A} \setminus F) = N(A \setminus F)$, donde $\Pi(\cdot \setminus F)$ es una medida de posibilidad nítida basada en $F$. Más generalmente cuando $F$ es difuso la compatibilidad $\mu_A(F)$, está relacionada a la posibilidad $\Pi(A \setminus F)$ y a la necesidad $N(A \setminus F)$ por las siguientes ecuaciones:

1. $\Pi(A \setminus F) = \Pi(\text{true} \setminus \mu_A(F))$.
2. $N(A \setminus F) = N(\text{true} \setminus \mu_A(F))$.

Las medidas atómicas de posibilidad y necesidad (Dubois y otros, 1988) son agregadas separadamente para obtener dos medidas globales entre el patrón entero y el dato entero. Cuando el patrón expresa una conjunción de requerimientos elementales "$P_1 \wedge \ldots\ldots P_n$" esta agregación es ejecutada usando la operación "min" o "mínimo" y preserva las semánticas respectivas de las medidas de posibilidad y necesidad, de esto se obtiene que:

1. $\Pi(P_1 \times \ldots \times P_n; D_1 \times \ldots\ldots \times D_n) = \min_{i = 1,..n} \Pi(P_i ; D_i)$
2. $N(P_1 \times \ldots \times P_n; D_1 \times \ldots\ldots \times D_n) = \min_{i = 1,..n} N(P_i ; D_i)$

donde $P_i$ y $D_i$ son definidos sobre el mismo dominio $U_i$, y donde × denota el producto cartesiano para dos conjuntos difusos $F_i$ y $F_j$ , el cual es definido por: $\forall u_i \in U_i$, $\forall u_j \in U_j$ , $\mu_{FixFj}(u_i, u_j) = \min(\mu_{Fi}(u_i), \mu_{Fj}(u_j))$.

Una igualdad aproximada puede ser convenientemente modelada por medio de una relación difusa R, la cual es reflexiva ($\forall u \in U$, $\mu_R(u, u) = 1$) y simétrica ($\forall u_1 \in U$, $\forall u_2 \in U$, $\mu_R(u_1, u_2) = \mu_R(u_2, u_1)$). Las más cercanas son $u_1$ y $u_2$, entonces, la más cercana a 1 debe ser $\mu_R(u_1, u_2)$. Por lo tanto, la cantidad $\mu_R(u_1, u_2)$ puede ser vista como un grado de emparejamiento de $u_1$ con $u_2$, en este caso, R es llamada relación de tolerancia.

El uso de relaciones de tolerancia (Cayrol y otros, 1982) en el emparejamiento de patrones difusos, implica que una relación de tolerancia está relacionada al dominio de una variable, y diferentes de estas relaciones pueden estar involucradas en la evaluación del emparejamiento de un dato con un patrón compuesto. Por lo tanto, una

manera posible para tomar en cuenta la importancia relativa de requerimientos elementales en un patrón compuesto es el uso de relaciones de tolerancias (a. Dubois y Prade, 1985).

Otro método propuesto (Dubois y otros, 1988) para tomar en cuenta la importancia relativa de los requerimientos elementales de un patrón compuesto, fue el emparejamiento de patrones difusos ponderado y el emparejamiento de patrones difusos parcial. En la técnica de emparejamiento de patrones difusos ponderada, un patrón recupera un dato, sí y sólo sí, todos los componentes de dato emparejan sus átomos de patrón homólogos. En la técnica de emparejamiento de patrones difusos parcial un dato puede ser aceptado por el patrón, a pesar de que el emparejamiento haya tenido éxito para solamente parte de los átomos.

Se han desarrollado herramientas de computación en el departamento de diabetes del Hospital Rangueil, en Toulouse, Francia, por más de doce años, las cuales han demostrado que la aplicación de lógica difusa (teoría de la posibilidad y aritmética difusa) y algoritmos de búsquedas heurísticas a software educacionales en nutrición, permiten proporcionar no sólo un mejor modelo matemático, sino también mejoras funcionales significativas para usuarios finales. Un software creado para este propósito fue Diabeto (Farreny y otros, 1985), el cual es un sistema experto que permite a pacientes diabéticos controlar su ingesta diaria de alimentos en forma personalizada, siendo accedido desde la red de videotex francesa Teletel, con la finalidad de servir de ayuda en la toma de decisiones para el tratamiento de la diabetes. El enfoque escogido para tratar con la incertidumbre e imprecisión del conocimiento médico en este sistema está basado en la teoría de la posibilidad, siendo desarrollado en el Laboratorio de Lenguajes y Sistemas Informáticos de la Universidad de Toulouse, Francia. Para evaluar este sistema experto (Turnin y otros, 1992), se escogieron 500 pacientes con diabetes mellitus dependientes y no dependientes de insulina, los cuales fueron divididos de forma aleatoria en dos grupos. Con la ayuda de Diabeto los pacientes controlaban sus dietas y el balance de sus comidas con asesoramiento personalizado. Durante los primeros seis meses, el grupo A usó Diabeto mientras que el grupo B fue sujeto a control; para el segundo

semestre de estudio, el grupo B usó también el sistema. La evaluación estuvo basada en el conocimiento dietético, hábitos de alimentación y equilibrio metabólico de los pacientes. El uso del sistema condujo a una mejora significativa del conocimiento dietético en el grupo A, además de mejorar sus hábitos alimenticios disminuyendo el consumo calórico excedido al inicio del estudio, aumentando el consumo de carbohidratos de 39.7 +/- 0.7 a 42.9 +/- 0.9% en pacientes con un consumo inicial menor a 45% y disminuyendo el consumo de grasas de 41.9 +/- 0.9 a 37.4 +/- 1.1% en pacientes con un consumo inicial mayor a 35%. En un segundo estudio, además de obtener mejoras similares a las observadas en el primer estudio, los pacientes mejoraron varios de sus indicadores fisiológicos, tales como glucosa en sangre, colesterol, entre otros. Por lo tanto, Diabeto se convirtió en una herramienta terapéutica efectiva en el control de enfermedades metabólicas.

El software Nutri-Advice (Buisson, 2002) es usado en Francia en los restaurantes de las empresas para controlar los requerimientos nutricionales diarios, evaluando el contenido nutricional de una comida. Todos los usuarios poseen una tarjeta proporcionada por un grupo médico la cual tiene inscrito en un código de barra sus necesidades nutricionales y posibles problemas médicos. El usuario pasa su tarjeta por un lector de código de barra y tres sugerencias de comidas son impresas en un ticket, el cual es emitido por el software. Una versión diferente existe en los comedores escolares de autoservicio, la cual contiene una pantalla táctil que muestra las fotos de los platos servidos. El estudiante pasa su tarjeta por el lector y selecciona el plato de comida que desea, además, son mostrados platos sustitutos para ampliar el número de opciones de comidas a escoger. Esta versión utiliza emparejamiento de patrones difusos de nutrientes con normas médicas en nutrición, además de representaciones fundadas en aritmética difusa y algoritmos de búsquedas heurísticas para realizar los cálculos, con la finalidad de balancear comidas y así adaptarlas a las necesidades nutricionales de los usuarios.

Nutri-Expert (Buisson, 1999) también es un software desarrollado en Francia, para llevar a cabo la tarea diaria de analizar y corregir las comidas de los pacientes en su casa. Experimentos médicos demostraron que un uso no supervisado en seis meses

de este software mejoró significativamente el conocimiento del paciente en nutrición y sus hábitos de preparación de comidas, además de mejorar varios indicadores fisiológicos, tales como, glucosa en la sangre. Este software permite describir en detalle todos los elementos de una comida real o hipotética, para hacer un balance completo tomando en cuenta todos sus nutrientes. El usuario escoge los alimentos que van a componer su comida apuntando a fotos de platos o seleccionándolos entre un conjunto de categorías y subcategorías, para luego transformar las cantidades y la naturaleza de los alimentos de la comida obteniendo su balance. Contiene un módulo de tipos de menús, el cual proporciona algunos menús para un día, cuyos contenidos y cantidades son adaptadas a las necesidades de los usuarios. Un menú producido por este módulo puede ser fácilmente examinado en el módulo de análisis y ser transformado si es necesario. Además, otro módulo actúa como una enciclopedia en nutrición y da recetas de diferentes platos. Al igual que Nutri-Advice, utiliza técnicas de lógica difusa y algoritmos de búsquedas heurísticas.

La Asociación Española para la lucha contra la hipertensión arterial (Botella, 2001) desarrolló un software en línea que permite consultar bases de datos sobre tablas de alimentos, tablas de platos e incluye una calculadora para realizar cálculos de composición nutricional. La tabla de alimentos sirve a su vez de base de cálculo para determinar la composición nutritiva de diferentes platos. El contenido en fibra vegetal de los alimentos, dato de creciente importancia en la nutrición moderna, se obtuvo a partir de publicaciones del Laboratorio del Servicio de Nutrición de la Clínica Puerta de Hierro, en España. Los alimentos están clasificados en grupos (cereales, lácteos, etc.) de entre 26 posibles, constituyendo los ingredientes básicos de la alimentación, y se identifican con un nombre de 30 caracteres como máximo. Para cada alimento se indica la composición en nutrientes por 100 gramos. Los nutrientes considerados en los cálculos son; carbohidratos, lípidos, proteínas y energía. Asimismo, se da la porción comestible en tanto por uno, de tal manera que 1 significa que no hay parte desechable mientras que, por ejemplo 0.95 expresa que un 5% del alimento no es comestible (cáscara, huesos, etc.). Para aquellos alimentos en los que se ha creído conveniente, se hacen recomendaciones en cuanto a su prohibición o

cautela en el caso de estar sometidos a dietas con restricciones de sodio, potasio, azúcar, o colesterol. No se consideró la inclusión de otros nutrientes, tales como el zinc, selenio, magnesio, etc.

El Calculador de Metabolismo Basal (Botella, 2001) es un programa que calcula el reposo basal y activa los índices calóricos metabólicos. Es una herramienta de enseñanza útil para personas diabéticas, donde el balance de elementos nutricionales es importante. Permite controlar el metabolismo basal para aumentar o disminuir de peso.

EvaNut2002 (Botella, 2001) es un software de evaluación nutricional múltiple orientado a profesionales de la salud en consulta privada o a pacientes hospitalizados. Evalúa nutricionalmente a niños menores de dos años según normativa Minsal/OMS, mujeres embarazadas según su índice de masa corporal (IMC), adolescentes según grado de desarrollo puberal de Tanner y sin Tanner, además de adultos según su IMC y relación cintura cadera. Genera un completo informe técnico para ser agregado a la ficha clínica o entregado al cliente. Una versión más reciente es WinEvaCurso, el cual es un software técnico para evaluación nutricional colectiva, el cual permite evaluar el estado nutritivo de niños menores de 10 años según la normativa Minsal. Desarrollado en ambiente Windows permite la evaluación de múltiples cursos emitiendo un listado resumen de los niños evaluados e indica aquellos que requieren asistencia nutricional.

Nutripac (Botella, 2001) es el primer software para nutricionistas desarrollado en México para transformar los tediosos y ordinarios análisis dietéticos, en un potente conjunto de herramientas de acceso a la información, apoyo a la decisión y manejo de ideas con gran versatilidad. Permite realizar un análisis total de dietas y menús, indicando los excesos, deficiencias, porcentajes de adecuación, gráficos y cálculo de recomendaciones y requerimientos. Por medio de éste se puede obtener el valor nutritivo de los alimentos de mayor consumo en México, en cuanto a, valores máximos, mínimos, raciones, equivalentes, comparaciones y resultados gráficos, permitiendo además manejar 800 alimentos y 25 nutrientes. Es una poderosa

calculadora de equivalentes, analizando menús con las tablas de alimentos mexicanos.

# BASES TEÓRICAS

## Conjuntos Difusos.

El concepto de conjunto difuso fue introducido por L. A. Zadeh en 1965, para representar matemáticamente la incertidumbre e imprecisión no probabilística presente en datos e información. Es por esto que la teoría de conjuntos difusos parte de la teoría clásica de conjuntos, añadiendo una función de membresía al conjunto, definida ésta como un número real entre 0 y 1. Por lo tanto, un conjunto o subconjunto difuso es asociado a un determinado valor lingüístico definido por una palabra, adjetivo o etiqueta lingüística A. Para cada conjunto o subconjunto difuso se define una función de pertenencia o inclusión $\mu_A(u)$, que indica el grado en que la variable u está incluida en el concepto representado por la etiqueta A.

Formalmente, un conjunto difuso A en X es caracterizado por una función miembro $f_A(x)$, la cual asocia a cada punto en X un número real en el intervalo [0,1], con el valor de $f_A(x)$ en x representando el "grado de pertenencia" de x en A. Siendo X un espacio de puntos con un elemento genérico de X denotado por x.

Por ejemplo, se tiene un conjunto de deportistas y se desea agruparlos de acuerdo a su altura. Para esto, se establecen las funciones de membresía que representarán los conjuntos difusos asociados a los tres valores lingüísticos "Alto", "Mediano" y "Bajo", los cuales son las diferentes clasificaciones que se efectúan a una variable lingüística. Se llamará "variable lingüística", al concepto que se va a calificar en forma difusa, siendo en este caso, la altura. El rango de valores que pueden tomar los elementos que poseen la propiedad expresada por la variable lingüística es conocido como el "Universo de Discurso", que en el caso de la variable altura comprende el conjunto de valores entre 1.5 y 2.3 mts. Para un deportista que mide 1.80 mts. (ver

Figura 1), se pudiera decir que tiene un grado de membresía de 0.2 en el conjunto difuso “Mediano” y un grado de membresía de 0.8 en el conjunto difuso “Alto”.

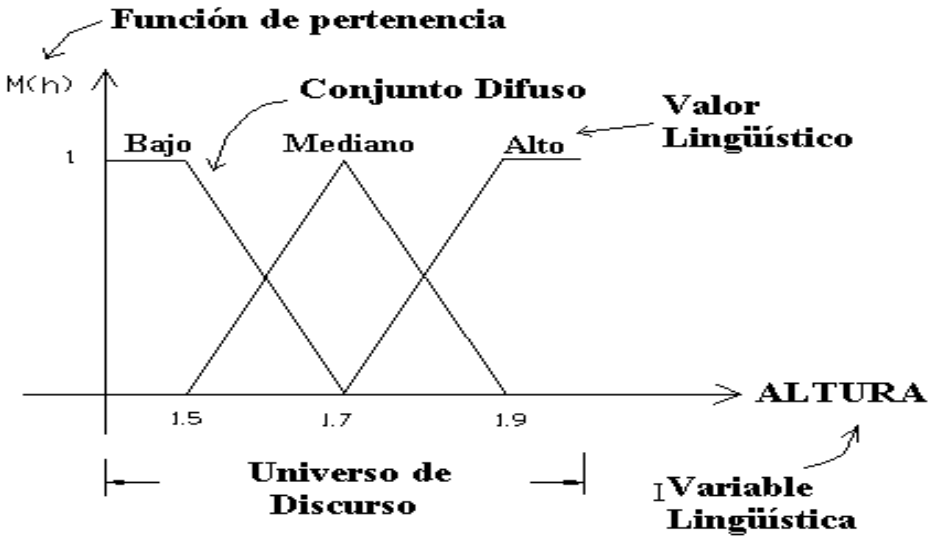



**Figura 1.** Funciones de Membresía para las Alturas de los Deportistas.

Las formas de funciones de membresía que más se usan para la representación de conjuntos difusos son: triangulares, trapezoidales, gaussianas y de campana generalizada, las cuales pueden definirse por medio de las fórmulas matemáticas parametrizadas mostradas en la Tabla 1.

**Tabla 1.** Fórmulas y Parámetros de las Funciones de Membresía.

| ***FORMA*** | ***FÓRMULAS*** | ***PARÁMETROS*** |
|---|---|---|
| ***Triangular*** | Triángulo($x;a,b,c$) = $0$ ; $x \leq a$<br>$(x-a)/(b-a)$ ; $a \leq x \leq b$<br>$(c-x)/(c-b)$ ; $b \leq x \leq c$<br>$0$ ; $c \leq x$ | a, b, c |

**Tabla 1 (Continuación).** Fórmulas y Parámetros de las Funciones de Membresía.

| | | |
|---|---|---|
| ***Trapezoidal*** | Trapezoide($x;a,b,c,d$) = $0$ ; $x \le a$<br>$(x-a)/(b-a)$ ; $a \le x \le b$<br>$1$ ; $b \le x \le c$<br>$(d-x)/(d-c)$ ; $c \le x \le d$<br>$0$ ; $d \le x$ | a, b, c, d |
| ***Gaussiana*** | Gaussiana($x;c,\sigma$) =exp $(-1/2((x-c)/\sigma))^2$ ) | c, σ |
| ***Campana*** | Campana($x;a,b,c$) = $1/(1 + \|(x-c)/a\|^{2b})$ | A, b, c |

## Operaciones de Conjuntos Difusos.

Un conjunto difuso es vacío, sí y sólo sí, su función miembro tiene un valor se 0 en X. Dos conjuntos difusos A y B son iguales, lo que se denota por A = B, sí y sólo sí, $f_A(x) = f_B(x)$ para todo x en X, o lo que es lo mismo, $f_A = f_B$. Además, el complemento de un conjunto difuso A es denotado por A' y es definido por $f_{A'} = 1 - f_A$.

Como en el caso de conjuntos ordinarios, la noción de contención juega un rol fundamental en el caso de conjuntos difusos. A está contenido en B, o equivalentemente, A es un subconjunto de B o A es más pequeño o igual a B, sí y sólo sí, $f_A \le f_B$, es decir, $A \subset B \Leftrightarrow f_A \le f_B$.

La unión de dos conjuntos difusos A y B con funciones miembros respectivas $f_A(x)$ y $f_B(x)$ es un conjunto difuso C, denotado por $C = A \cup B$, cuya función miembro es $f_C(x) = \text{Max}[f_A(x), f_B(x)]$ tal que $x \in X$, o en forma abreviada, $f_C = f_A \vee f_B$. Además, la unión tiene la propiedad asociativa, es decir, $A \cup (B \cup C) = (A \cup B) \cup C$.

La intersección de dos conjuntos difusos A y B con funciones miembros respectivas $f_A(x)$ y $f_B(x)$ es un conjunto difuso C, denotado por $C = A \cap B$, cuya función miembro es $f_C(x) = \text{Min}[f_A(x), f_B(x)]$ tal que $x \in X$, o en forma abreviada, $f_C = f_A \wedge f_B$. La intersección de A y B es el conjunto difuso más grande el cual está contenido tanto en A como en B. Además A y B son disjuntos si $A \cap B$ es vacía, y al igual que la unión, tiene la propiedad asociativa. La intersección y la unión de dos

conjuntos difusos en $R^1$ son ilustradas en la Figura 2, donde la función de membresía de la unión está comprendida por los segmentos de curva 1 y 2; y la función de membresía de la intersección está comprendida por los segmentos de curva 3 y 4. [Zadeh (1965)]

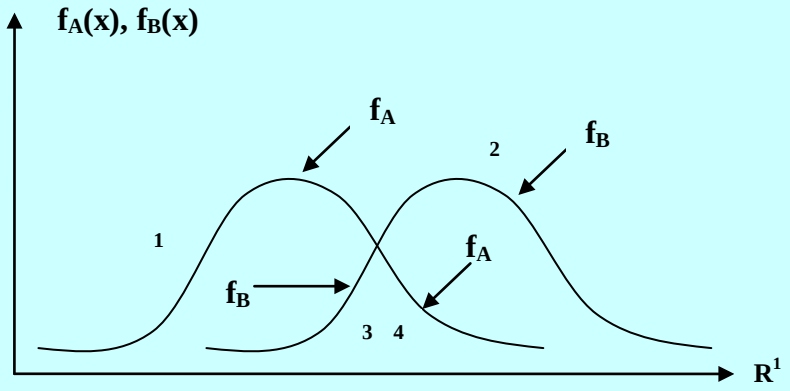



**Figura 2.** Unión e intersección de conjuntos difusos en $R^1$

## El Principio de Extensión.

Uno de los conceptos más importantes de la teoría de conjuntos difusos es el principio de extensión, el cual es usado para generalizar conceptos matemáticos nítidos a conjuntos difusos y el cual fue introducido por Zadeh (1965).

Sea X un producto cartesiano de universos $X = X_1x\ldots\ldots X_r$, y $A_1x\ldots\ldots A_r$ , r conjuntos difusos en $X_1x\ldots\ldots X_r$, respectivamente. Si $f$ en un mapeo desde X al universo Y, $y = f(x_1,\ldots\ldots,x_r)$. Entonces el principio de extensión permite definir un conjunto difuso B en Y por $B = \{(y, \mu_B(y)) \mid y = f(x_1,\ldots\ldots,x_r), (x_1,\ldots\ldots,x_r) \varepsilon X\}$ y donde su función miembro está definida por: [Zimmermann (1996)]

$$\mu_B(y) = \begin{cases} \underset{(x_1,\ldots x_r)\ \varepsilon\ f^{-1}(y)}{\text{Sup}} \min\{\mu_{A1}(x_1),\ldots,\mu_{Ar}(x_r)\}, \text{ si } f^{-1}(y) \neq \phi \\ 0 \qquad \text{en caso contrario} \end{cases}$$

Para r = 1, este principio se reduce a $B = f(A) = \{(y,\mu_B(y)) \setminus y = f(x), x \varepsilon X\}$, donde la función miembro está definida por:

$$\mu_B(y) = \begin{cases} \sup_{x \varepsilon f^{-1}(y)} \mu_A(x), & \text{si } f^{-1}(y) \neq \phi \\ 0 & \text{en caso contrario} \end{cases}$$

## Números Difusos.

Los números difusos son usados en conexión con aplicaciones donde es deseable la representación explícita de la ambigüedad e incertidumbre presente en conjuntos de datos numéricos. Las operaciones de conjuntos difusos tales como la unión y la intersección, así como también las nociones de $\alpha$-cuts, y el principio de extensión, son aplicables a los números difusos. Por lo tanto, un número difuso A, es un conjunto difuso en R (números reales), el cual es "normal" y "convexo" y cuyo soporte está definido por $supp(A)=\{x \in R \setminus \mu_A > 0\}$. La normalidad implica que $\ni x \varepsilon R$ tal que $\mu_A(x)=1$, es decir, que un número difuso es normal cuando tiene una altura igual a uno y es convexo, cuando la forma de su función miembro no sube y baja más de una vez. Hay diferentes clases de números difusos, entre ellos están los números difusos triangulares, los cuales pueden ser definidos por una 3-tupla $(a_1, a_2, a_3)$ con su función miembro definida en la Tabla 1, y los números difusos trapezoidales, los cuales son definidos por una 4-tupla(a,b,c,d) y cuya función miembro fue definida también en la Tabla 1.

Existen dos maneras diferentes de representar números difusos, las cuales son; a través de funciones miembros o a través de sus cortes $\alpha$ o intervalos, lo que proporciona la opción de definir operaciones aritméticas a través del principio de extensión o equivalentemente a través de operaciones de aritmética de intervalo. La función miembro de un número difuso puede ser parametrizada por un parámetro $\alpha$

(el cual es un número en el intervalo [0,1]), en una manera similar a la representación tabular del número 3 mostrada en la Tabla 2. Representando en la fila inferior el universo de discurso o soporte, el cual está formado por los números naturales en [1,9] y en la fila superior el grado de pertenencia de cada elemento del universo, dando como resultado el conjunto difuso o número difuso triangular que representa al numero 3. [Tsoukalas y Uhrig (1997)]

**Tabla 2.** Representación del número difuso 3 a través de sus cortes $\alpha$.

| | **0.3** | **0.7** | **1** | **0.7** | **0.3** | **0** | **0** | **0** | **0** |
|---|---|---|---|---|---|---|---|---|---|
| **$\alpha$=1.0** | | | 1 | | | | | | |
| **$\alpha$=0.9** | | | 1 | | | | | | |
| **$\alpha$=0.8** | | | 1 | | | | | | |
| **$\alpha$=0.7** | | 1 | 1 | 1 | | | | | |
| **$\alpha$=0.6** | | 1 | 1 | 1 | | | | | |
| **$\alpha$=0.5** | | 1 | 1 | 1 | | | | | |
| **$\alpha$=0.4** | | 1 | 1 | 1 | | | | | |
| **$\alpha$=0.3** | 1 | 1 | 1 | 1 | 1 | | | | |
| **$\alpha$=0.2** | 1 | 1 | 1 | 1 | 1 | | | | |
| **$\alpha$=0.1** | 1 | 1 | 1 | 1 | 1 | | | | |
| **$\alpha$=0.0** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |

La parametrización de un número difuso por $\alpha$ ofrece una forma conveniente para cálculos con números difusos, debido a la transformación de operaciones de aritmética difusa en operaciones de aritmética de intervalo. En cada nivel $\alpha$ se tiene un intervalo de la función miembro, el cual es su corte $\alpha$, representado por un intervalo $[a_1,a_2]_\alpha$; por lo tanto, la representación tabular muestra por cada fila la longitud de cada corte $\alpha$, es decir, la longitud de la función miembro en un nivel $\alpha$. Esta representación tiene como requerimiento esencial, que la función miembro del número difuso sea normal y convexa. Por lo tanto, el corte $\alpha$ de un conjunto difuso A denotado por $A_\alpha$, es el conjunto nítido comprendido de todos los elementos x de un universo de discurso X, para los cuales la función de membresía de A es mayor o

igual a $\alpha$ , es decir: $A_\alpha = \{x \in X / \mu_A(x) \geq \alpha\}$; donde $\alpha$ es un parámetro en el rango de $0 < \alpha \leq 1$. Por lo tanto, la colección de intervalos que representan el número difuso triangular A =3 para los diferentes niveles de $\alpha$, están representados a continuación en la Tabla 3.

**Tabla 3.** Cortes $\alpha$ del número difuso triangular 3.

| $\alpha$ | **0.0** | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** | **0.8** | **0.9** | **1.0** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{A_\alpha}$ | [1,9] | [1,5] | [1,5] | [1,5] | [2,4] | [2,4] | [2,4] | [2,4] | [3,3] | [3,3] | [3,3] |

## Aritmética de Números Difusos.

Un conjunto de operaciones muy similares a las operaciones de aritmética tales como; suma, resta, multiplicación y división, pueden ser también definidas por medio del principio de extensión para números difusos, dando la ventaja de que sean usadas en conexión con ecuaciones difusas, permitiendo con esto introducir operaciones de aritmética difusa para reducir ambigüedades en cálculos numéricos. Dados dos números difusos A y B, y la operación aritmética o, tal que A o B = C. Donde C es definido por $C = \{(z,\mu_C(z)) \setminus z = x \circ y \ \ \mu_C(z) = \max_{x \circ y = z} \min [\mu_A(x), \mu_B(y)]$ para todo $x \in A, y \in B\}$.

Debido a las propiedades de proporcionalidad de los triángulos en el caso de números difusos triangulares se puede probar que la suma, resta y la multiplicación (por una constante) de números difusos triangulares da como resultado un número difuso triangular (Ver Figuras 3 y 4). Como se puede observar en la Figura 5, el resultado de multiplicar dos números difusos no es triangular, pero este puede ser aproximado a un número difuso triangular teniendo el mismo valor nominal ($\mu = 1$) y el mismo intervalo como resultado obtenido; solamente el grado de pertenencia de

algunos de los puntos en el intervalo es cambiado, simplificándose con esta aproximación la aritmética difusa.

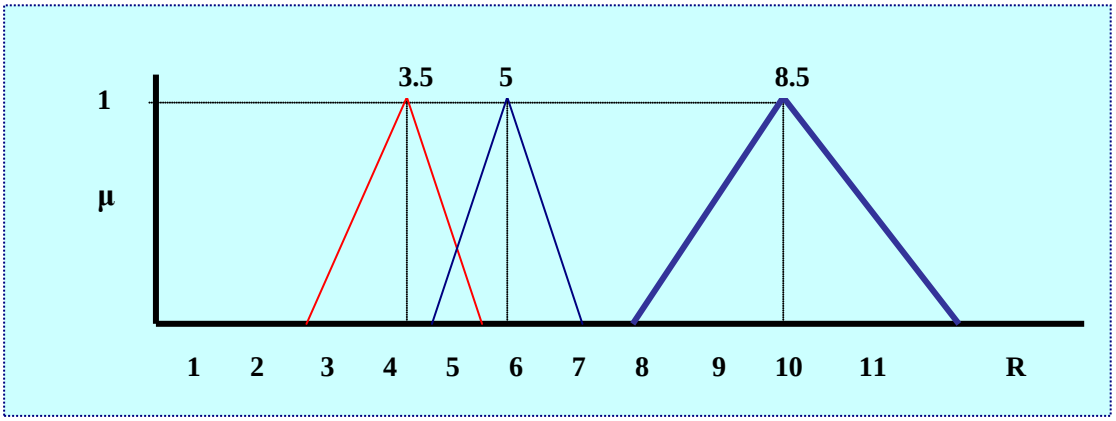



**Figura 3.** Suma difusa de los números 3.5 y 5.

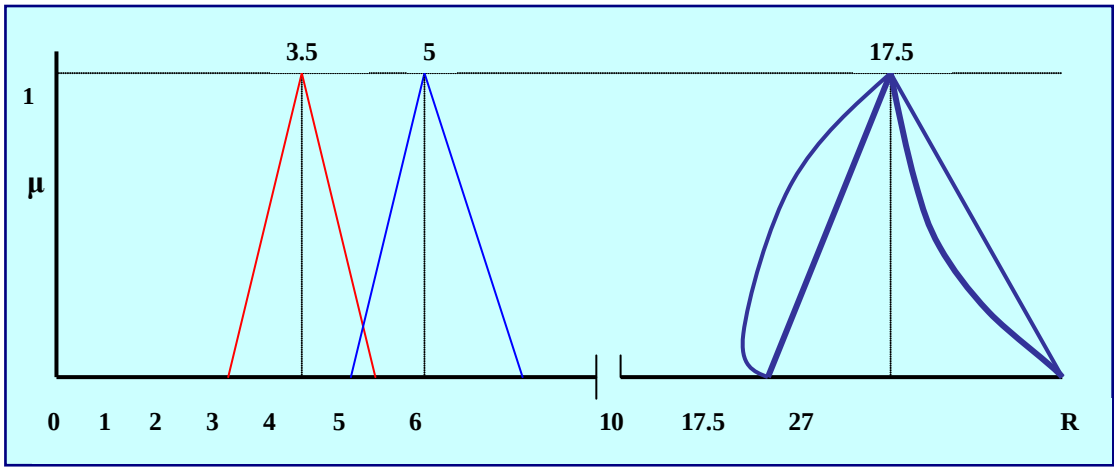



**Figura 4.** Multiplicación difusa de los números 3.5 y 5.

La Tabla 4 muestra un resumen de las operaciones aritméticas difusas más importantes, tanto las definidas con aritmética de intervalo, así como también las definidas por el principio de extensión. [Tsoukalas y Uhrig (1997)]

**Tabla 4.** Operaciones de Aritmética Difusa entre A y B.

| ***Operaciones con Aritmética de Intervalo entre A y B para obtener C.*** |
|---|
| $A_\alpha + B_\alpha = [a_1,a_2]_\alpha + [b_1,b_2]_\alpha = [a_1 + b_1,\ a_2 + b_2]_\alpha = [c_1,c_2]_\alpha$ |
| $A_\alpha - B_\alpha = [a_1,a_2]_\alpha - [b_1,b_2]_\alpha = [a_1 - b_2,\ a_2 - b_1]_\alpha = [c_1,c_2]_\alpha$ |
| $A_\alpha \times B_\alpha = [a_1,a_2]_\alpha \times [b_1,b_2]_\alpha = [a_1b_1,\ a_2b_2]_\alpha = [c_1,c_2]_\alpha$ |
| $A_\alpha / B_\alpha = [a_1,a_2]_\alpha / [b_1,b_2]_\alpha = [a_1/b_2,\ a_2/b_1]_\alpha = [c_1,c_2]_\alpha$; con $b_2$ y $b_1 \neq 0$ |
| $1/ B_\alpha = 1/ [b_1,b_2]_\alpha = [1/b_2,\ 1/b_1]_\alpha = [c_1,c_2]_\alpha$; con $b_2$ y $b_1 \neq 0$ |
| $A_\alpha \pm k = [a_1,a_2]_\alpha \pm k = [a_1 \pm k,\ a_2 \pm k]_\alpha = [c_1,c_2]_\alpha$ |
| $A_\alpha \times k = [a_1,a_2]_\alpha \times k = [ka_1,\ ka_2]_\alpha = [c_1,c_2]_\alpha$ |
| ***Operaciones con el Principio de Extensión entre A y B para obtener C.*** |
| $A + B = \mu_C(z) = \underset{z = x+y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ |
| $A - B = \mu_C(z) = \underset{z = x-y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ |
| $A \times B = \mu_C(z) = \underset{z = x \div y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ $\quad A/B = \mu_C(z) = \underset{z = x \bullet y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ |
| *Mínimo entre A y B* $= \mu_{A \wedge B}(z) = \underset{Z = x \wedge y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ |
| *Máximo* entre A y B $= \mu_{A \vee B}(z) = \underset{Z = x \vee y}{\vee} [\mu_A(x) \wedge \mu_B(y)]$ |

## Comparación de Números Difusos.

Cuando se comparan números difusos dos clases de preguntas pueden surgir: (a) ¿Cuál es el valor difuso del número más grande o más pequeño de una familia de números difusos?, y (b) ¿Qué tan cierta es esta afirmación?: “m es más grande o más

pequeño que n, o en otras palabras, ¿Cuál es la posibilidad que m sea más grande o más pequeño que n?. La primera pregunta es respondida aconsejando el uso de los operadores max o min definidos por medio del principio de extensión. Se puede escribir el valor difuso del número más grande entre m y n como max(m,n), el cual es un número difuso normal convexo como es mostrado en la Figura 5.

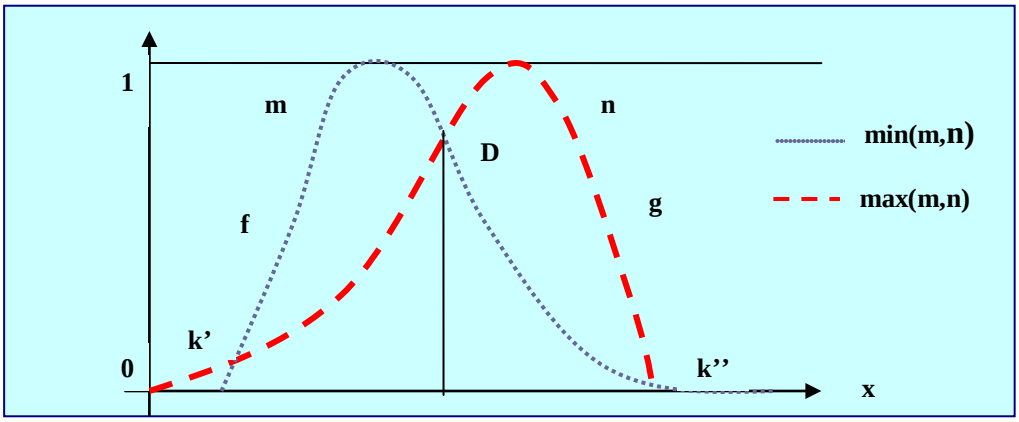



**Figura 5.** Comparación de los números difusos m y n.

En cuanto a la segunda pregunta, el valor de verdad de la afirmación "n es más grande que m", la cual se escribe como n>m, es υ(n>m), definida como $\upsilon(n > m) = \max_{y \geq x} \min(f(x), g(y))$; donde $f$ y g son las funciones miembros respectivas de m y n. En el caso de la figura 5 es fácil chequear que υ(n>m) = 1 y que υ(m>n) = $f$(d) = g(d), lo cual es una evaluación de la separación de m y n (ordenada de D).

Una representación conveniente para m es la 3-tupla(m,α,β) representando los parámetros de su función miembro $f$ definida como:

1. $f(x) = L\left(\frac{m-x}{\alpha}\right); x \leq m$
2. $f(x) = R\left(\frac{x-m}{\beta}\right); x \geq m;$

donde L y R son funciones simétricas con forma de campana, tal que L(0)=R(0)=1. Como m es del tipo L-R y n es del tipo R-L con m = (m,α,β) y n = (n,γ,δ), es fácil chequear para el caso de la relación difusa "<" que:

$$f(d) = g(d) = R\left[\frac{n-m}{\beta+\gamma}\right] \text{ si } m < n.$$

Por lo tanto, se puede definir la diferencia de los números difusos m y n por; m – n = (m,α,β) - (n,γ,δ) = (m-n, α + δ, β + γ), donde la operación m-n es un número difuso del tipo L – R. Entonces, la comparación de m y n es equivalente a la comparación de n - m o m - n con (0,0,0) = 0 y $f$(d) = g(d) = $\mu_{n-m}(0) = \mu_{m-n}(0)$.

Para comparar m y n se necesita ambas, υ(m>n) y υ(n>m). Si por ejemplo υ(n>m)=1, puede significar que m es más grande que n, o que ambos números difusos están demasiado cerca para ser separados. Entonces se puede escoger un umbral y admitir que m es más grande que n en el nivel τ, denotado por $m >_\tau n$ y decir que υ(n>m) ≤ τ. Por lo tanto, para números difusos del tipo L-R se puede decir que; n>m sí y sólo sí n-m ≥ β + γ (τ=R(1)), y que m>n sí y sólo sí m-n ≥α + δ (τ=L(1)). En otros casos se admite que m es aproximadamente igual a n. [Dubois y Prade (1980)]

## Teoría de la Posibilidad.

La teoría de conjuntos difusos proporciona una base fundamental para la teoría de la posibilidad enfocándose principalmente en la imprecisión, la cual se asume que es posibilística en lugar de probabilística. Además el término variable es frecuentemente usado en un sentido lingüístico más que en un sentido estrictamente matemático. Esta es una razón por la cual la terminología y la simbología de la teoría de la posibilidad difieren de algún modo de la empleada en la teoría de conjuntos difusos (Zimmermann, 1996).

Uno de los conceptos más importantes de la teoría de la posibilidad es el de distribución de posibilidad, utilizada para representar una restricción difusa con su función miembro jugando el rol de una función de distribución de posibilidad,

además, una variable difusa es asociada a una distribución de posibilidad de la misma manera como una variable aleatoria es asociada a una distribución de probabilidad. Por lo tanto, la importancia de esta teoría proviene del hecho de que mucha de la información en que están basadas las decisiones humanas es posibilística en lugar de probabilística. Basándose en esta premisa es posible construir un lenguaje universal, en que la traducción de una proposición expresada en un lenguaje natural tome la forma de un procedimiento para calcular la distribución de posibilidad de un conjunto de relaciones difusas en una base de datos. [Zadeh (1978)]

Supóngase la proposición "X es F", donde X es el nombre de un objeto, una variable o una proposición. Por ejemplo, en "X es un entero pequeño" X es el nombre de una variable. En "Juan es joven" Juan es el nombre de un objeto y F(por ejemplo, "entero pequeño" o "Joven") es un conjunto difuso caracterizado por su función de membresía $\mu_{F.}$

Para hablar de distribución de posibilidad primero hay que hablar de la noción de restricción difusa. Sea X una variable la cual toma valores en un universo de discurso U con el elemento genérico de U denotado por u y X = u, u $\in$U. Sea F un subconjunto difuso de U, el cual es caracterizado por una función miembro $\mu_F$. Entonces F es una restricción difusa en X (o asociada con X) si F actúa como una restricción elástica en los valores que pueden ser asignados a X, en el sentido de que la asignación de un valor u a X tiene la forma X = u: $\mu_F(u)$, donde $\mu_F(u)$ es interpretada como el grado al cual la restricción representada por F es satisfecha cuando u es asignada a X. En forma equivalente, 1 - $\mu_F(u)$ es el grado al cual la restricción debe ser estirada para permitir la asignación de los valores u a la variable X.[Zadeh (1978)]

Para ilustrar el concepto de restricción difusa considérese dos maletas: una de tapa dura, en donde el volumen es un número nítido. Mientras que otra es elástica, donde el volumen de su contenido depende en cierto grado de la fuerza usada para estirarla. La variable en este caso sería el volumen de la maleta y los valores que esta variable puede asumir, pueden ser u $\varepsilon$ U y el grado al cual la variable (X) puede asumir diferentes valores de u está expresado por $\mu_F(u)$.

Sea R(X) una restricción difusa asociada con X entonces R(X) = F es llamada ecuación de asignación relacional, la cual asigna el conjunto difuso F (o relación difusa) a la restricción difusa R(X). Ahora asúmase que E(X) es un atributo implicado de la variable X, por ejemplo, E(X)=edad de Alex y F es el conjunto difuso "joven", la expresión "Alex es joven" puede entonces ser expresada como R(A(X)) = F.

Sea p la proposición "Alex es joven", en la cual "joven" es un conjunto difuso del universo U = [0,100] caracterizado por la función miembro $\mu_{joven}(u) = S(u;20;30;40)$, donde u es la edad numérica y la función F es definida por:

$$S(u;\alpha;\beta;\gamma) = \begin{cases} 1;u < \alpha \\ 1 - 2\left(\dfrac{u-\alpha}{\gamma-\alpha}\right)^2;\alpha \le u \le \beta \\ 2\left(\dfrac{u-\gamma}{\gamma-\alpha}\right)^2;\alpha < u \le \gamma \\ 0;u > \gamma \end{cases}$$

En este caso, el atributo implicado E(X) es Edad (Alex) y la traducción de "Alex es joven" tiene la forma Alex es joven → **R**(edad (Alex)) = joven. Por la tanto, Zadeh(1978) relacionó el concepto de restricción difusa al de distribución de posibilidad como sigue: considérese una edad numérica, por ejemplo, u = 28 cuyo grado de membresía en el conjunto difuso joven es aproximadamente 0.7. Primero se interpreta 0.7 como el grado de compatibilidad de 28 con el concepto etiquetado joven. Entonces se postula que la proposición "Alex es joven" convierte el significado de 0.7 de grado de compatibilidad de 28 con joven, al grado de posibilidad de que Alex tenga 28 años dada la proposición "Alex es joven". De aquí se concluye que la compatibilidad de un valor de u con joven se convierte en la posibilidad de este valor de u dado que "Alex es joven".

Sea F un conjunto difuso en un universo U, caracterizado por su función de membresía $\mu_F(u)$, la cual es interpretada como la compatibilidad de u con el concepto etiquetado F. Sea X una variable que toma valores de U y donde F es un conjunto

difuso el cual actúa como una restricción difusa R(X) asociada con X. Entonces la proposición "X es F" la cual se traduce como R(X)=F, asocia una distribución de posibilidad $\pi_x$, con X que es postulada a ser igual a R(X).

La función de distribución de posibilidad $\pi_x(u)$, la cual caracteriza la distribución de posibilidad $\pi_x$, se define numéricamente igual a la función de membresía $\mu_F(u)$ de F, es decir, $\pi_x = = \mu_F(u)$, donde el símbolo "= =" siempre se lee como "denota" o "se define como".

Para ilustrar el concepto de distribución de posibilidad considérese U el universo de enteros positivos y F el conjunto difuso de los enteros pequeños definido por: F = {(1,1),(2,1),(3,0.8),(4,0.6),(5,0.4),(6,0.2)}. Entonces la proposición "X es un entero pequeño" asocia a X la distribución de posibilidad $\pi_x$ = F, en la cual (3,0.8) significa que la posibilidad que X sea 3 dado que X es un entero pequeño, es 0.8.

Para mostrar la diferencia entre posibilidad y probabilidad (Zadeh, 1978) se puede considerar la proposición "Juan comió X huevos en el desayuno", en la cual X = {1,2,3…….}. Una distribución de posibilidad así como una distribución de probabilidad pueden ser asociadas con X. La distribución de posibilidad $\pi_x(u)$ puede ser interpretada como el grado de facilidad con que Juan puede comer u huevos, mientras que la distribución de probabilidad $P_x(u)$ pudiera haber sido determinada observando a Juan durante el desayuno por 100 días. De aquí se puede decir que un grado alto de posibilidad no implica un grado alto de probabilidad. Sin embargo, si un evento no es posible, éste es también improbable. Por lo tanto, de alguna manera la posibilidad es un límite superior para la probabilidad. Los valores de $\pi_x(u)$ y $P_x(u)$ para ilustrar lo anteriormente dicho, son mostrados a continuación en la Tabla 5.

**Tabla 5.** Valores de distribución de posibilidad $\pi_x(u)$ y probabilidad $P_x(u)$**.**

| **U** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|---|---|---|---|---|---|---|---|---|
| $\pi_x(u)$ | 1 | 1 | 1 | 1 | 0.8 | 0.6 | 0.4 | 0.2 |
| $P_x(u)$ | 0.1 | 0.8 | 0.1 | 0 | 0 | 0 | 0 | 0 |

**Medida de Posibilidad.**

Sea A un conjunto difuso en el universo U y sea $\pi_x$ la distribución de posibilidad asociada con la variable X, la cual toma valores en U. La medida de posibilidad $\pi_x(A)$ de A es definida como: Poss{X está en A} = = $\pi(A)$ = = sup min $\{\mu_A(u), \pi_x(u)\}$, u $\varepsilon$ U.

Por ejemplo, considérese la distribución de posibilidad inducida por la proposición "X es un entero pequeño", donde $\pi_x$= {(1,1), (2,1), (3,0.8), (4,0.6), (5,0.4), (6,0.2)} y el conjunto nítido A = {3,4,5}. Entonces, la medida de posibilidad es $\pi(A)$ = max(0.8,0.6,0.4}= 0.8.

Si por otra parte se asume que A es el conjunto difuso "enteros que no son pequeños" definido por A = {(3,0.2), (4,0.4), (5,0.6), (6,0.8), (7,1),…..}. En este caso, la medida de posibilidad es $\pi(A)$ =max(0.2,0.4,0.4,0.2}= 0.4.

Las medidas difusas expresan el grado al cual un cierto subconjunto de un universo $\Omega$, o un evento es posible. Por ello tenemos que g(0) = 0 y g($\Omega$)=1 y como consecuencia de la definición donde se cumple que, $A \subseteq B \Rightarrow \pi(A) \leq \pi(B)$ se obtiene que: $A \subseteq B \Rightarrow g(A) \leq g(B)$. Por lo tanto, $g(A \cup B) \geq \max(g(A),g(B))$ y $g(A \cap B) \leq \min(g(A), g(B))$, donde A y B son subconjuntos de $\Omega$.

Las medidas de posibilidad son definidas para casos límite como: $\pi(A \cup B) = \max(\pi(A), \pi(B))$ y $\pi(A \cap B) = \min(\pi(A), \pi(B))$.

Si A' es el complemento de A en $\pi$, entonces $\pi(A \cup A') = \max(\pi(A), \pi(A'))$, lo cual expresa que tanto A como A' son completamente posibles. [Zimmermann 1996)]

**Medida de Necesidad.**

En la teoría de la posibilidad otra medida adicional es definida, la cual usa la relación conjuntiva y en un sentido es dual a la medida de posibilidad. Entonces, $N(A \cap B) = \min(N(A),N(B))$, donde N es llamada medida de necesidad. Entonces, N(A) = 1 indica que A es necesariamente verdadera (A es segura). Se puede decir que la relación dual de la posibilidad y necesidad requiere que: $\pi(A) = 1 - N(A')$,

$\forall A \subseteq \Omega$. La medida de necesidad satisface la condición min(N(A), N(A')) = 0, por lo tanto, las relaciones de posibilidad y necesidad satisfacen las siguientes condiciones, donde Ω es siempre finito: [Zimmermann (1996)]

1. $\pi(A) \geq N(A), \quad \forall A \subseteq \Omega$
2. $N(A) > 0 \Rightarrow \pi(A) = 1$
3. $\pi(A) < 1 \Rightarrow N(A) = 0$

## Emparejamiento de Patrones Difusos.

La técnica de emparejamiento de patrones difusos ha sido desarrollada en el marco de los conjuntos difusos y la teoría de la posibilidad, con la finalidad de tomar en cuenta la imprecisión y la incertidumbre que poseen determinados valores los cuales tienen que ser comparados en un proceso de emparejamiento. Esta técnica ha probado ser muy útil para implementar patrones de razonamiento aproximado en máquinas de inferencia de sistemas expertos y para diseñar sistemas de recuperación, capaces de manejar bases de datos con información difusa e incompleta, además de consultas vagas. Básicamente, un proceso de emparejamiento de patrón involucra un patrón para describir algunos requerimientos y una base de conocimiento donde la data está organizada.

Cuando la imprecisión e incertidumbre está presente en la data o cuando el patrón incluye especificaciones vagas, el emparejamiento entre el patrón y un dato se vuelve un asunto de grado. En este caso un requerimiento difuso puede ser más o menos satisfecho, es decir, no se puede estar completamente seguro que un requerimiento nítido se mantiene si el dato bajo consideración es difuso.

El emparejamiento de patrones difusos toma en cuenta la representación semántica relacionada a los componentes de la data y los patrones, proporcionando una evaluación de la similitud entre un patrón y un dato en el intervalo [0,1]. Por lo tanto, la idea principal es asociar a cada átomo de un patrón la función de pertenencia de un conjunto difuso, restringiendo los valores más o menos compatibles con el

significado del átomo. Estos valores pertenecen a alguna escala prescrita, correspondiente al rango del atributo al cual el átomo hace referencia. Por ejemplo, el símbolo alto puede hacer referencia a una escala de alturas y pertenece a un subconjunto difuso de esta escala. En lugar de una escala numérica también se puede tener un conjunto discreto de elementos típicos. La data es representada por listas, cuyos componentes son asociados a distribuciones de posibilidad, las cuales permiten modelar la imprecisión presente en la data.

En este enfoque, un patrón difuso representa una clase mal limitada de objetos, mientras un dato difuso representa un objeto mal conocido cuya descripción precisa no está disponible. Sean P y D respectivamente un átomo de patrón y un componente de dato pertenecientes a un mismo atributo, los cuales van a ser comparados y hacen referencia a la misma escala U. Sea $\mu_P$ la función miembro asociada a un átomo P y $\pi_D$ la distribución de posibilidad asociada a D, las cuales hacen referencia a una escala U evaluada en el intervalo [0,1]. Si u es un elemento de U, entonces $\mu_P(u)$ es el grado de compatibilidad entre el valor u y el significado de P, donde P es un conjunto difuso de valores compatibles seguros. Por lo tanto, $\mu_P(u) = 1$ significa certeza total de que u es compatible con P y $\mu_P(u) = 0$ significa certeza total de que u no es compatible con P.

Por otra parte, $\pi_D(u)$ es el grado de posibilidad de que u es el único valor del atributo para describir el objeto modelado por el dato, donde D es un conjunto difuso de valores posibles. Por lo tanto, $\pi_D(u)=1$ significa que u es totalmente posible, mientras que $\pi_D(u)=0$ significa que u es totalmente imposible como un valor de atributo del objeto. Siempre hay un valor el cual es totalmente compatible con P y un valor totalmente posible en el rango de D. [D. Dubois y otros (1988)]

**Índices de Emparejamiento Elementales.**

Sean P y D los intervalos difusos que representan el patrón y la data respectivamente. Dos medidas escalares son usadas para estimar la compatibilidad entre P y D; un grado de posibilidad de emparejamiento denotado por $\Pi(P; D)$ y un

grado de necesidad de emparejamiento N (P; D), los cuales son respectivamente definidas por:

1. $\Pi(P;D) = \underset{u \in U}{Sup} \min\left(\mu_P(u), \pi_D(u)\right)$
2. $N(P;D) = \underset{u \in U}{\inf} \max\left(\mu_P(u), 1 - \pi_D(u)\right)$

La medida Π(P; D) estima a que grado es posible que P y D se refieran al mismo valor u; en otras palabras, Π(P;D) es el grado de solapamiento del conjunto difuso de valores compatibles con P, con el conjunto difuso de valores posibles de D. La medida N(P; D) estima a que grado es necesario que el valor al cual D refiere, está entre los valores compatibles con P; en otras palabras, N(P;D) es un grado de inclusión del conjunto de valores posibles de D en el conjunto de valores compatibles con P. Estas medidas son calculadas por intersección de líneas rectas obtenidas de representaciones de valores. (Ver Figuras 6 y 7)

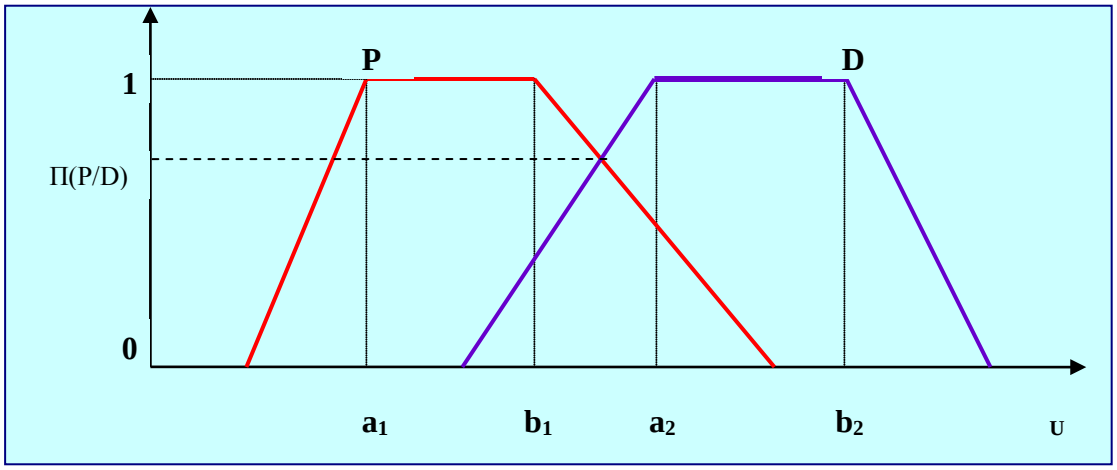



**Figura 6.** Representación gráfica de la medida de posibilidad.

La necesidad de un evento corresponde a la imposibilidad del evento opuesto, la cual es expresada por la siguiente relación: $N(P;D) = 1 - \Pi(\bar{P};D)$, donde $\mu_{\bar{P}} = 1 - \mu_P$ es la función miembro del complemento del conjunto difuso de valores compatibles con P. Evidentemente, siempre se cumple que

$\Pi(P;D) \geq N(P;D)$. Además $N(F,F)=1$ sí y sólo sí $\mu_F$ es la función miembro de un subconjunto ordinario de U; de lo contrario, $N(F,F) \geq 1/2$. Cuando dos constantes idénticas tienen un significado difuso no se puede estar completamente seguros que ellas hacen referencia exactamente al mismo conjunto de valores. En algunos casos se tiene que $N(s(F);F)=1$, donde el soporte de F es $s(F)$ y está definido por $s(F)=\{u \in U, \mu_F(u) > 0\}$.

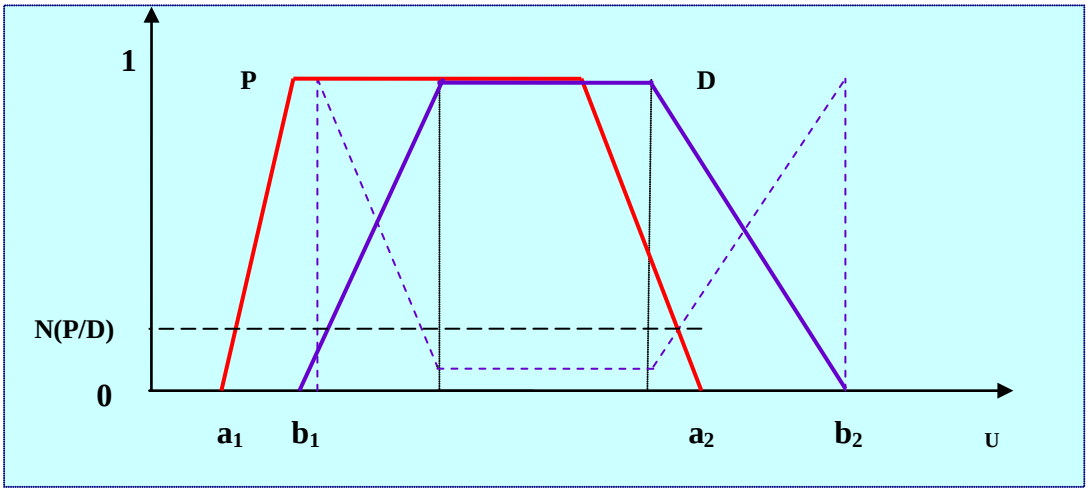



**Figura 7.** Representación gráfica de la medida de necesidad.

Los casos límite donde $\Pi(P;D)$ y $N(P;D)$ toman valores de 0 y 1 son también útiles para caracterizar. Para algún conjunto difuso F en U, sea $F^o=\{u \in U / \mu_F(u)=1\}$ el núcleo de $F$ y $s(F)$ su soporte. Por lo tanto, de las definiciones anteriores se obtienen las siguientes propiedades:

1. $\Pi(P;D)=0$ *sí y sólo sí* $s(P) \cap s(D)=\phi$.
2. $\Pi(P;D)=1$ *sí y sólo sí* $P^o \cap D^o \neq \phi$.
3. $N(P;D)=1$ *sí y sólo sí* $s(D) \subseteq P^o$.

Nótese que la propiedad 3 define una inclusión más fuerte entre conjuntos difusos que la propiedad usual $\pi_D \leq \mu_P$, la cual solamente implica que $N(P;D) \geq 0.5$. La

Tabla 6 resume las respectivas posiciones de $\Pi(P;D)$ y $N(P;D)$ en el intervalo [0,1] cuando P y/o D es preciso, impreciso o difuso.

**Tabla 6.** Posiciones de Π(P;D) y N(P;D) en el intervalo [0,1].

| | **D** | | |
|---|---|---|---|
| | ***Preciso*** | ***Impreciso*** | |
| ***Difuso*** **P** | **D = {d}** | **(No difuso)** | |
| ***Preciso*** ***P = {p}*** | Π = N = 1 si p = d 0 si p ≠ d | N = 0 Π = 1 si p ∈ D 0 si p ∉ D | N = 0 $\Pi = \pi_D(d)$ |
| ***Impreciso*** ***(No difuso)*** | Π = N = 1 si d ∈ p 0 si d ∉ p | Π = N = 1 si D⊆P 0 si D∩P = $\phi$ Π = 1, N = 0 {en caso contrario}. | N > 0 ⇒ Π =1 |
| ***Difuso*** | Π = N = $\mu_p(d)$ | Π ≥ N | Π ≥ N |

Nótese en la Tabla 6 que cuando D es preciso entonces, $\Pi(P;D) = N(P;D) = \mu_P(D)$, el cual es un número real regular en [0,1]. Cuando D es impreciso pero no difuso $\mu_P(D)$ es un subconjunto de [0,1], siendo $\Pi(P;D)$ y $N(P;D)$ respectivamente el límite superior mínimo y el límite inferior máximo del conjunto $\mu_P(D)$. Más generalmente, si $\tau$ es un valor modal de $\mu_P(D)$ ($\mu_{P/D}(\tau)=1$), entonces $\tau$, es un buen indicador escalar de la magnitud del emparejamiento entre P y D, es decir, es un buen representante de la cantidad difusa. Por lo tanto, los pares de

índices $\Pi(P;D)$ y $N(P;D)$ son una razonable aproximación de $\mu_P(D)$, produciendo el resultado de que $\forall P$, D y $\tau$ tal que $\mu_{P/D}(\tau)=1$, la desigualdad $N(P;D) \leq \tau \leq \Pi(P;D)$ se mantiene. [D. Dubois y otros (1988)]

**Tipos de Emparejamiento de Patrones Difusos.**

El emparejamiento difuso de patrones es especialmente útil para el manejo de consultas vagas y también para manejar información difusa e incompleta en bases de datos. Existen tipos de emparejamiento de patrones difusos que están basados en las medidas de posibilidad y necesidad, donde cada componente de dato y cada requerimiento elemental son asociados a una distribución de posibilidad y a un conjunto difuso. Entre estos están el emparejamiento de patrones difusos ponderado y el emparejamiento de patrones difusos parcial. En la técnica ponderada, los índices de emparejamiento de patrón global son definidos vía una agregación conjuntiva de grados elementales de emparejamiento posible o necesario entre átomos y componentes de data, es decir, un patrón recupera un dato, sí y sólo sí, todos los componentes de dato emparejan sus átomos de patrón correspondientes. Sin embargo, en algunas aplicaciones un dato puede ser aceptado por el patrón cuando el emparejamiento tiene éxito para solamente parte de los átomos, ocurriendo un emparejamiento de patrón difuso parcial, el cual es definido por medio de pesos adecuadamente distribuidos. [Cayrol y otros(1982)]

## Estrategias de control no informadas.

Los algoritmos de búsqueda (Rubio y otros, 1998) trabajan con estructuras de datos llamadas nodos que contienen información sobre un estado. La Tabla 7 muestra una primera aproximación de un algoritmo genérico de búsqueda en un espacio de estados (conjunto de estados factibles para un problema junto a la estructura de adyacencia definida implícitamente por los operadores o reglas).

El tipo de búsqueda depende del paso nro. 7 en el algoritmo de la Tabla 7. Se tratará de búsquedas no informadas si el algoritmo no conoce nada del problema en concreto que debe resolver. Esto quiere decir que el paso nro. 7 se realiza con criterios independientes del dominio del problema. Un concepto importante que aparece cuando la estrategia de control reposa sobre la noción de nodo, es el de árbol de búsqueda, el cual es la estructura combinatoria que va siendo construida en el proceso de búsqueda. Sus vértices son nodos y existe una arista dirigida de un nodo a otro, si el segundo nodo es construido a partir del primero en el paso nro. 7 del algoritmo de la Tabla 7. En particular, la estructura combinatoria construida en la ejecución del algoritmo es siempre un árbol, puesto que los nodos en el paso nro. 7 son construidos y no reutilizados. Esto es independiente de que nodos distintos (que han sido construidos en distintos momentos de la ejecución) tengan asociado el mismo estado. Otra noción importante es la de profundidad de un nodo (en el árbol de búsqueda), la cual se trata del número de aristas que tiene el camino que lo une con el nodo raíz (ese camino es único, por tratarse de un árbol).

**Tabla 7.** Algoritmo genérico de búsqueda.

```
PRINCIPIO
1  ABIERTOS:= (nodo-inicial)
   RESUELTO:= FALSO
2  Mientras que ABIERTOS no es vacía  Y  NO  RESUELTO  hacer
3      N:= quitar primer elemento de ABIERTOS;
       E:= estado asociado a N
4      si  E es un estado objetivo
5      entonces RESUELTO:= verdad
       si no para cada operador O hacer
6              si O se puede aplicar a E
7              entonces crear un nodo correspondiente al estado obtenido por aplicación de
                        O a E y añadir ese nodo a ABIERTOS
   si  RESUELTO
8  entonces devuelve el estado objetivo (y si se requiere una explicación, el
                camino por el que se ha llegado a él)
9  si no informa de que el objetivo no puede ser alcanzado
FIN
```

**Búsqueda en Anchura.**

En este tipo de búsqueda el paso nro. 7 del algoritmo de la Tabla 7 tiene la particularidad, de que los nuevos nodos son añadidos al final de la lista Abiertos. Así se consigue que los nodos en dicha lista estén ordenados según su profundidad en orden decreciente, es decir, los menos profundos al principio y los más profundos al final. La lista Abiertos tiene de este modo una forma de acceso que la convierte en una cola (o fila de espera) donde los datos que primero entran son los primeros en salir. Mediante esta estrategia, el árbol de búsqueda se va generando por niveles de profundidad. Hasta que todos los nodos de un nivel no han sido revisados, no se revisa ninguno del siguiente nivel. Por lo tanto este tipo de búsqueda es una estrategia exhaustiva, ya que el recorrido por niveles examina todos los estados que son accesibles desde el estado inicial.

En particular, si un estado objetivo es alcanzable esta búsqueda lo encuentra, en otras palabras, el estado objetivo aparecerá en un cierto nivel de profundidad y, en un número finito de pasos (suponiendo un número finito de operadores y un número finito de posibles aplicaciones de un operador sobre un estado) se llegará a ese nivel. Este mismo razonamiento muestra que la búsqueda en anchura siempre encuentra la secuencia solución más corta (una de las que tienen el mínimo número de aristas). Pero esto no significa en absoluto que realice un número pequeño de operaciones, ya que se trata de una propiedad de la secuencia de la solución encontrada, y no del proceso por el que ha sido construida, por lo tanto, esta búsqueda puede llegar a ser extremadamente ineficaz.

**Búsqueda en profundidad.**

En este tipo de búsqueda el paso nro. 7 del algoritmo de la Tabla 7 tiene la particularidad, de que los nuevos nodos son añadidos al comienzo de la lista Abiertos. Así se consigue que los nodos en dicha lista estén ordenados según su profundidad en orden creciente, es decir, los más profundos al principio y los menos profundos al

final. La lista Abiertos tiene de este modo una forma de acceso que la convierte en una pila, donde los datos que primero entran son los últimos en salir. Al igual que en la búsqueda en anchura, los nodos de igual profundidad se ordenan arbitrariamente, según el orden de las reglas u operadores, siendo más sensible a ese ordenamiento que la búsqueda en anchura.

En cuanto a la comparación de estos dos tipos de búsqueda, una primera observación es que la búsqueda en profundidad necesita menos espacio para ser realizada. En concreto, su complejidad en espacio será O(bd) frente a la complejidad exponencial $O(b^d)$ de la búsqueda en anchura. Recuérdese que b es el factor de ramificación (media del número de nodos generados desde un nodo) y d mide la profundidad del estado objetivo (mínimo del número de aplicaciones de reglas necesarias para llegar del estado inicial a un estado objetivo). En cambio, su complejidad asintótica en tiempo es de orden exponencial $O(b^d)$ al igual que la búsqueda en anchura, pero pese a este resultado teórico, en ocasiones el número de estados examinados por la búsqueda en profundidad puede ser considerablemente menor.

## Estrategias de control heurísticas.

Los métodos de búsqueda en anchura y en profundidad son denominados métodos ciegos de búsqueda (Rubio y otros, 1998). En general, son muy ineficaces y sin aplicación práctica debido al crecimiento exponencial del tiempo y del espacio que requieren (explosión combinatoria). Al contrario, los métodos de búsqueda heurística disponen de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores. Estos métodos no garantizan que se encuentre una solución aunque existan soluciones, pero si encuentran una, no se asegura que ésta tenga buenas propiedades (que sea de longitud mínima o de costo óptimo). Además en algunas ocasiones, encontrarán una solución (aceptablemente buena) en un tiempo razonable.

Se puede decir entonces, que los métodos heurísticos sacrifican la completitud (es posible que algunas soluciones se "pierdan") al incrementar su eficiencia. En general, éstos son preferibles a los métodos no informados en la solución de problemas difíciles, en los que una búsqueda exhaustiva necesite un tiempo demasiado grande. Esto cubre prácticamente la totalidad de los problemas reales que interesan en Inteligencia artificial. En cuanto al aspecto algorítmico, el mismo esquema presentado en la Tabla 7 sirve para las búsquedas heurísticas o informadas. La única diferencia, es que los pasos 6 (sensibilización de las reglas) y 7 (reorganización de la lista Abiertos) se realizan con criterios dependientes del dominio del problema, de modo que el algoritmo tiene cierto conocimiento que utilizará en tiempo de ejecución sobre el problema concreto que debe resolver.

La información del problema concreto que se intenta resolver se suele expresar por medio de heurísticas, las cuales son procesos que pueden resolver un problema dado, pero que no ofrecen ninguna garantía de que lo harán. La idea consiste en concentrar toda la información heurística en una única función que se denomina función de evaluación heurística. Se trata de una función que asocia a cada estado del espacio de estados una cierta cantidad numérica que evalúa de algún modo lo prometedor que es ese estado para acceder a un estado objetivo. Habitualmente, se denota esa función por h(e). La función heurística puede tener dos interpretaciones. Por una parte, puede ser una estimación de la "calidad" de un estado. En este caso, los estados de mayor valor heurístico son los preferidos.

Por otra parte, la función puede ser una estimación de lo próximo que se encuentra el estado de un estado objetivo. Bajo esta perspectiva, los estados de menor valor heurístico son los preferidos. Ambos puntos de vista son complementarios (de hecho, un cambio de signo en los valores permite pasar de una perspectiva a la otra).

## Estrategias no informadas para la búsqueda de una solución óptima.

Este tipo de estrategias (Rubio y otros, 1998) aparece cuando en el espacio de estados cada arista tiene asociado como peso un coste numérico mayor que cero.

Llamaremos coste de un camino en el espacio de estados a la suma de los costes de las aristas que lo constituyen. Como es habitual, la longitud de un camino es su número de aristas. Si el coste de cada arista es 1, coste y longitud coinciden. El problema consiste entonces en encontrar una solución óptima, es decir un camino solución cuyo coste sea mínimo entre los de los caminos solución. Donde a un camino, se le denomina camino solución óptimo y a su coste, coste óptimo.

Por ejemplo, considérese el recorrido de un tren donde se podría tener interés no sólo en encontrar un trayecto entre la ciudad de partida y la de llegada, sino en encontrar uno de duración mínima. En este caso el peso de cada arista debería ser la duración del viaje. Si hubiera interés en minimizar el número de kilómetros, entonces deberíamos disponer para cada arista del número de kilómetros. Por último, si el interés es solo encontrar el trayecto con el mínimo número de trasbordos, se le asignaría a cada arista el peso 1.

Una primera estrategia para resolver el problema planteado consiste en mantener la lista de Abiertos ordenada, pero en lugar de hacerlo respecto a los valores heurísticos, se ordena respecto al coste de los caminos definidos por los nodos (apoyándose en la información relativa al padre). Se denominarán a las estrategias que llevan a cabo esta idea "primero el menos costoso". Las cuales permitirán calcular soluciones de coste óptimo, pero que van a ser tan ineficaces (en tiempo y en espacio) como la búsqueda en anchura. De hecho, si el coste de cada arista es una constante fija k

De aquí que estas estrategias son equivalentes (se examinan y expanden los mismos estados y en el mismo orden) a la búsqueda en anchura. Por otra parte, si el coste es constante y se ordena la lista Abiertos de mayor a menor, obtendremos un comportamiento equivalente al de la búsqueda en profundidad.

## Estrategias Heurísticas de los Algoritmos A.

Considérese de nuevo el problema del recorrido de un tren donde se dispone de las coordenadas de cada una de las ciudades y, para cada trayecto entre dos ciudades

adyacentes del número de kilómetros que realiza el tren en ese trayecto. Situémonos en un momento intermedio del proceso de búsqueda.

Para decidir cuál es el siguiente paso a dar, es decir cuál es la siguiente ciudad a la que viajar, se podría utilizar varios criterios (Rubio y otros, 1998): (a) Elegir una ciudad cualquiera sin tener en cuenta el conocimiento adicional del que se dispone (coordenadas, costes). Esto es esencialmente lo que hacen las búsquedas no informadas en anchura o en profundidad, donde no se garantiza que se vaya a obtener un camino solución óptimo ni tampoco que la búsqueda esté dirigida por ninguna otra consideración. (b) Tener en cuenta la estimación (heurística) de la cercanía dada por la distancia en línea recta a la ciudad objetivo. Esto es lo que hacen las escaladas o "primero el mejor", donde no se considera la distancia previamente recorrida (es decir, la información acerca del coste de las aristas). La búsqueda estará dirigida por cierto conocimiento específico (aunque no totalmente seguro), pero no se garantiza el hallazgo de una solución óptima. (c) Ir calculando el coste de los caminos que van siendo construidos, eligiendo en cada paso el menos costoso. Aquí se aplican las estrategias para encontrar una solución óptima, pero al no tener en cuenta el conocimiento heurístico, pueden resultar tan ineficaces como la búsqueda en anchura.

Por lo tanto en el caso de las estrategias de los algoritmos A, se estudia la posibilidad de integrar las aproximaciones (b) y (c) anteriores, para obtener estrategias que encuentren soluciones óptimas, pero de un modo más eficaz que la estrategia "primero el menos costoso". Esto es lo que se conoce en la literatura con el nombre de algoritmos A, nominación ciertamente poco descriptiva.

Para esta familia de algoritmos los nodos han de disponer (al menos) de cuatro informaciones: estado, padre, g (costo mínimo de transformación desde el nodo inicial a un nodo meta) y h (valor heurístico, que sólo depende del estado y que representa una estimación del costo para transformar un nodo m a sus soluciones más cercanas).

Una instancia del esquema A en la que la heurística sea admisible se denomina algoritmo A*, el cual siempre encontrará una solución óptima si se esta existe.

## Algoritmo de búsqueda A*.

De manera de expandir nodos lo menos posible en la búsqueda de un camino óptimo, un algoritmo de búsqueda (Nilsson y otros, 1968) debe estar constantemente informado para decidir el próximo nodo a expandir, ya que si éste expande nodos que no están en el camino óptimo, se estaría desperdiciando tiempo y espacio. Al contrario, si éste ignora nodos los cuales podrían estar en un camino óptimo, fallará en encontrar tal camino, convirtiéndose en un algoritmo no admisible. Por lo tanto, un algoritmo eficiente necesita de alguna manera evaluar los nodos disponibles para determinar cual será expandido. Supóngase alguna función $\hat{f}(n)$, la cual podría ser calculada para un nodo n. Esta función puede ser definida de tal manera, que un nodo disponible con el valor más pequeño de $\hat{f}$ sea el próximo nodo a expandir.

Por lo tanto, para una apropiada elección de la función de evaluación $\hat{f}$, el algoritmo A* garantiza encontrar un camino óptimo a un nodo meta de *s*, siendo de esta manera un algoritmo admisible.

### Función de Evaluación.

Para algún subgrafo G y algún conjunto meta T, sea $f(n)$ el costo actual de un camino óptimo restringido a ir a través de *n*, desde *s* a un nodo meta de *n*. Por lo tanto, se puede definir $f(n)$ como la suma de dos partes, es decir, $f(n) = g(n) + h(n)$, donde $g(n)$ es el costo actual de un camino óptimo desde *s* a *n* y $h(n)$ es el costo actual de un camino óptimo desde *n* a un nodo meta preferido de *n*. Por otro lado, si se tienen estimaciones de *g* y *h*, éstas se podrían agregar para formar una estimación de $f$. Sea $\hat{g}(n)$ una estimación de $g(n)$, entonces una elección obvia para $\hat{g}(n)$ es el costo del camino desde *s* a *n* con el costo más pequeño hasta ahora

encontrado por el algoritmo. Nótese que esto implica que $\hat{g}(n) \geq g(n)$. También se puede tener una estimación $\hat{h}(n)$ de $h(n)$. En este caso se cuenta con información del dominio del problema.

**Admisibilidad de A*.**

La función de evaluación a ser usada en A* es $\hat{f}(n) = \hat{g}(n) + \hat{h}(n)$, donde $\hat{g}(n)$ es el costo del camino desde *s* a *n* con el costo mínimo hasta ahora encontrado por A* y $\hat{h}(n)$ es alguna estimación del costo de un camino óptimo desde *n* a un nodo meta preferido de *n*. Entonces puede ser definido un algoritmo como se muestra en la Tabla 8.

**Tabla 8.** Algoritmo de Búsqueda A*.

1 Colocar el nodo *s* como "Abierto" y calcular $\hat{f}(s)$.

2 Seleccionar el nodo *n* con el menor valor de $\hat{f}$.
Resolver ataduras arbitrariamente, pero siempre a favor de algún nodo *n* en T.

3 Si $n \in$ T, colocar *n* como "Cerrado" y terminar el algoritmo.

4 De lo contrario, colocar *n* "Cerrado" y aplicar el operador sucesor $\Gamma$ a *n*.
Calcular $\hat{f}$ para cada sucesor de *n* y colocar como abierto cada sucesor no colocado cerrado. Volver a colocar como abierto cualquier nodo cerrado $n_i$, el cual sea un sucesor de *n* para el cual $\hat{f}(n_i)$ tiene un valor más pequeño que cuando $n_i$ fue colocado como cerrado. Ir al paso 2.

## Principios de la nutrición humana.

El organismo obtiene la energía indispensable para su funcionamiento de los sustratos provenientes de los carbohidratos, las grasas y las proteínas; formando estas últimas parte de estructura celular, participando también, en procesos enzimáticos intracelulares, defensa antiinfecciosa, etc. [Villazón y Arenas, 1993]

El primer principio básico de la nutrición (Taylor G., 1981) consiste en que la ingesta diaria de todos los nutrientes básicos debe estar en equilibrio, de forma que no existan ni deficiencias ni excesos de ninguno de ellos. Esto quiere decir, que mientras mayor sea la variedad de los alimentos en una dieta, menor será el peligro de que se produzcan estas deficiencias o excesos. Por lo tanto, una dieta bien equilibrada puede elaborarse mediante un gran número de combinaciones de alimentos, tratando de asegurar que se ingieran cada día las cantidades adecuadas de estos nutrientes esenciales.

### Unidades de Energía.

Con el fin de relacionar la energía química presente en los alimentos con el calor producido por el cuerpo y el trabajo mecánico realizado, es esencial disponer de unidades de medida mediante las cuales se puedan comparar cuantitativamente las diferentes formas de energía. Durante muchos años la unidad empleada por los nutricionistas para este fin era la Kilocaloría (Kcal) pero, debido a un acuerdo internacional ha sido reemplazada por el kilojulio (Kj), el cual es la unidad patrón de energía reconocida por el sistema SIU (Sistema Internacional de Unidades): 1 Kcal equivale a 4,181 kj. Sin embargo la Kcal todavía se utiliza ampliamente.

### Contenido energético de alimentos y dietas.

Hay dos caminos para determinar los valores energéticos de los alimentos (Taylor G., 1981): el primero consiste en consultar tablas sobre la composición de los

alimentos y el segundo, consiste en analizar el alimento en un laboratorio para conocer su contenido real en grasas, proteínas y carbohidratos. Un ejemplo de un cálculo de este tipo se expone en la Tabla 9 para una muestra de pan.

**Tabla 9.** Valor energético del pan.

| | *%* | *Energía disponible por* | | | |
|---|---|---|---|---|---|
| | *Composición* | *g nutriente* | | *100 g de pan* | |
| | | *kj* | *kcal* | *kj* | *kcal* |
| ***Carbohidratos*** | 50 | 16 | 4 | 800 | 200 |
| ***Grasa*** | 2 | 37 | 9 | 74 | 18 |
| ***Proteína*** | 10 | 17 | 4 | 170 | 40 |
| ***Humedad y cenizas*** | 38 | — | — | — | — |
| ***Total*** | 100 | — | — | 1044 | 258 |

Por lo tanto, de la Tabla 9 cabe explicar que: 1 gramo de hidrato de carbono (carbohidrato) aporta 4 kcal, 1 gramo de proteínas aporta 4 kcal y 1 gramo de lípidos o grasas aporta 9 kcal.

## Los nutrientes básicos.

Según los valores de referencia de energía y nutrientes para la población venezolana (Instituto Nacional de Nutrición, Revisión 2000), existen requerimientos mínimos para cada uno de los nutrientes básicos que deben satisfacerse.

### Carbohidratos.

Los hidratos de carbono o carbohidratos (CHO) constituyen la principal fuente de energía alimenticia de la mayor parte de la población mundial. Los alimentos que

contribuyen con los carbohidratos al total de energía disponible son, en orden de importancia, los cereales, los azúcares simples y en menor proporción las raíces, tubérculos, leguminosas y frutas. Por lo tanto, el aporte de carbohidratos debe oscilar entre el 56% y 69% de las calorías totales de la dieta diaria, lo cual se garantiza con un consumo aproximado de 140 a 173 grs. de CHO por cada 1000 Kcal.

**Proteínas.**

Las proteínas de la dieta deben ser suficientes para cumplir con varias funciones, entre ellas; crecimiento, reposición y mantenimiento de tejidos, generación de enzimas, hormonas, anticuerpos, balance de electrolitos, balance ácido-básico, y el transporte de energía para garantizar la respuesta inmunológica, la cicatrización y la conservación de todos los órganos y sistemas corporales.

El aprovechamiento de las proteínas por el organismo depende de la cantidad y la calidad de las mismas. En las dietas de consumo tradicional en Venezuela generalmente se combinan alimentos tales como; cereales con leguminosas, carnes, lácteos y hortalizas. Estos alimentos aportan gran parte de las necesidades de proteínas en cantidad y calidad, es decir, en las preparaciones de tales alimentos la proteína animal complementa los aminoácidos deficientes de la proteína vegetal. Para lograr la cantidad necesaria de proteínas se han establecido valores de referencia que deben aportar los alimentos, los cuales se estiman con base en necesidades tanto individuales como de población. Las recomendaciones van de 11,1% a 12,6% de energía proteica.

**Grasas o lípidos.**

Se refieren a un grupo heterogéneo de compuestos que abarcan las grasas y aceites usuales, ceras y compuestos relacionados que se encuentran en los alimentos y en el cuerpo del hombre. Tienen las propiedades comunes de ser: (a) insolubles en agua; (b) solubles en solventes orgánicos como éter y cloroformo y (c) capaces de ser

utilizadas por organismos vivos. Casi todas las grasas naturales consisten en un 98 a 99% de triglicéridos, que a su vez están constituidos principalmente de ácidos grasos. El 1% o 2% restante incluye monoglicéridos y diglicéridos, ácidos grasos libres, fosfolípidos y sustancias no saponificables que contienen esteroides. En este grupo también se incluyen las vitaminas liposolubles.

Aunque todas las grasas de los alimentos contienen una combinación de ácidos grasos, las de los animales tienden a ser más saturadas, en tanto que las vegetales son primordialmente insaturadas excepto los aceites de plantas tropicales como los de palma y coco. Las principales fuentes de grasas son las llamadas grasas visibles, entre las que se encuentran; los aceites vegetales, margarina, mantequilla, mayonesa y la manteca vegetal; y las grasas invisibles que se encuentran en carnes, vísceras, embutidos, tocino, leche completa, quesos, aguacate, aceitunas, maní, nueces, almendras y avellanas, entre otros.

De acuerdo a las Hojas de Balance de Alimentos de Venezuela para el período 1990-1997 el aporte de las grasas al total de energía disponible se encuentra entre 25% y 27%, siendo alrededor del 60% de éstas de origen vegetal. Estudios epidemiológicos han revelado que existe relación entre la ingesta de dietas con contenido de grasa superiores al 35% de las calorías totales y el desarrollo de enfermedades cardio y cerebrovasculares. Por lo tanto, de acuerdo a las recomendaciones de grasas para la población adulta venezolana, el consumo total de las mismas debe representar de 20% a 30% del total de la energía consumida.

**Requerimiento calórico total.**

Un peso de una persona por encima de los límites considerados normales para la estatura, sexo y edad indicaría que ésta consume más calorías de las que necesita. Por lo tanto, el índice de masa corporal (IMC) es la medida estándar para definir estados de delgadez, sobrepeso u obesidad. Este índice (Taylor G., 1981) posee rangos dependiendo de la edad para determinar en que estado de nutrición se encuentra la

persona, es decir, si tiene un peso normal o acorde a la edad y estatura, un sobrepeso o un bajo peso. La tabla 10 muestra las ecuaciones para el cálculo del IMC.

**Tabla 10.** Rangos de valores para el IMC.

| ***Ecuación para el cálculo del índice de masa corporal.*** |
|---|
| IMC= Peso en kilos/ (altura en metros)$^2$ |
| ***Rangos del IMC para determinar el estado de nutrición.*** |
| **1. Si la edad está entre 19 y 24 años.**<br>• Si el valor calculado del IMC está en el rango [19,24] → Peso normal<br>• Si el valor calculado del IMC < 19 → Bajo peso<br>• Si el valor calculado del IMC > 24 → Sobrepeso<br>**2. Si la edad entre 25 y 34 años.**<br>• El valor del IMC para determinar "peso normal" está en el rango [20,25]<br>**3. Si la edad entre 35 y 44 años.**<br>• El valor del IMC para determinar "peso normal" está en el rango [21,26]<br>**4. Si la edad entre 45 y 54 años.**<br>• El valor del IMC para determinar "peso normal" está en el rango [22,27]<br>**4. Si la edad entre 55 y 64 años.**<br>• El valor del IMC para determinar "peso normal" está en el rango [23,28]<br>**4. Si la edad es mayor a 65 años.**<br>• El valor del IMC para determinar "peso normal" está en el rango [24,29] |

El requerimiento calórico total (Taylor G., 1981) es el número de kcal que debe consumir una persona para satisfacer sus requerimientos nutricionales básicos. Este se calcula en base al IMC y al rango de las calorías por estado de nutrición. La forma para el cálculo del IMC mostrada en la Tabla 10 es un método general para personas adultas sanas mayores de 18 años, las cuales desean consumir alimentos de forma balanceada, que no requieren dietas especiales, y que además, tengan una actividad

física leve, como por ejemplo, ir al trabajo y caminar al lo sumo 30 minutos diarios. Basándose en la Tabla 10 para el cálculo del IMC, la Tabla 11 muestra el cálculo del requerimiento calórico para una persona con las características mencionadas anteriormente.

**Tabla 11.** Cálculo del requerimiento calórico para una persona adulta sana.

| ***Rango de las calorías por estado de nutrición.*** |
|---|
| Peso normal = 25 a 30 kcal x peso. (Se recomienda el promedio)<br>Sobrepeso = 20 kcal x peso.<br>Bajo peso = 35 kcal x peso. |
| ***Cálculo del Requerimiento Calórico Total.*** |
| Edad = 24<br>Peso = 54<br>Estatura = 1.61<br>**Cálculo del IMC.**<br>$IMC = 54/(1.61)^2 \rightarrow IMC = 20{,}85 \rightarrow$ Estado de nutrición corresponde a peso normal.<br>**Cálculo del Requerimiento Calórico Total (RCT).**<br>RCT = Peso * Rango de Kcal x Estado de Nutrición → RCT = 54*28 →<br>RCT= 1512 kcal. |

Luego de calculado el requerimiento calórico total, este debe ser distribuido de forma porcentual entre el número de comidas que se van a tener en el día. La distribución recomendada por el INN (Instituto Nacional de Nutrición, Revisión 2000) es: (a) 25% para el desayuno, (b) 45% para el almuerzo y (c) 30% para la cena.

## Principales variables y procesos manejados por la herramienta.

El problema de investigación se basa en el diseño y desarrollo de una herramienta computacional eficiente para obtener compuestos alimenticios con una combinación balanceada de nutrientes, usando lógica difusa y búsquedas heurísticas. Por lo tanto, se manejan un conjunto de variables que intervienen en determinados procesos con la finalidad de proporcionar una respuesta o salida satisfactoria a determinados requerimientos de nutrición, logrando de esta forma una interfaz adecuada con el usuario. La Tabla 12 resume las variables y procesos más importantes.

**Tabla 12:** Variables y procesos importantes manejados por la herramienta.

| ***Variable*** | ***Tipo*** | ***Valores*** |
|---|---|---|
| *Peso de la importancia del átomo de patrón.* | *Número* | *Toma valores en el intervalo [0,1].* |
| *Compuesto Alimenticio (descripción del compuesto requerido en componentes y cantidades).* | *Objeto* | *Conjunto de atributos cuyos valores ∈ a un dominio D.* |
| *Índice de Posibilidad de Emparejamiento.* | *Número* | *Toma valores en el intervalo [0,1].* |
| *Índice de Necesidad de Emparejamiento.* | *Número* | *Toma valores en el intervalo [0,1].* |
| *Patrón de normas médicas de nutrición y Patrón difuso con cantidad de nutrientes.* | *Objeto* | *Comprende valores difusos representados por números trapezoidales.* |

**Tabla 12 (Cont.):** Variables y procesos importantes manejados por la herramienta.

| | | |
|---|---|---|
| *Componente de Dato. (Almacenado en BD)* | *Valor Lingüístico* | *Valores que ∈ a un dominio D (números difusos trapezoidales que representan valores precisos o difusos).* |
| ***Procesos*** | | |
| ***Especificación de requerimientos nutricionales por parte del usuario.***<br>***Variables relacionadas:*** Componente alimenticio, Cantidad. | | |
| ***Evaluación de patrón con técnica de emparejamiento de patrones difusos.***<br>***Variables relacionadas:*** Patrón difuso que representa la cantidad total de nutrientes en un compuesto alimenticio, Peso de la Importancia del Átomo de Patrón, Índice de Necesidad de Emparejamiento, Índice de Posibilidad de Emparejamiento, Componente de Dato (almacenado en la BD), Patrón que representa las normas médicas de nutrición para cada nutriente (carbohidratos, lípidos y proteínas). | | |
| ***Corrección de patrón para obtener una combinación balanceada de nutrientes.***<br>***Variables relacionadas:*** Compuesto Alimenticio (componentes y patrón con las cantidades difusas totales por nutriente), Componente de Dato (valor almacenado en la base de datos de compuestos alimenticios), Patrón que representa las normas médicas de nutrición o cantidades recomendadas para cada nutriente | | |

# Definición de Términos Básicos

**Algoritmo A:** aplica de forma eficaz estrategias heurísticas guiadas por una función de evaluación para obtener una solución óptima. Cuando la primera solución encontrada tiene el más bajo costo, el algoritmo A se convierte en un algoritmo de búsqueda A*.

**Alimento:** sustancia natural sólida o líquida que estimula el apetito y contiene gran variedad de nutrientes según su composición química.

**Alimentación Balanceada:** alimentación que lleva a cabo un individuo de acuerdo a sus necesidades tanto fisiológicas como de actividad física y de su estado sociocultural, dentro del marco de las leyes de la nutrición.

**Aritmética Difusa:** conjunto de operaciones muy similares a las operaciones clásicas de aritmética de números nítidos, tales como; suma, resta, multiplicación y división, las cuales pueden ser definidas para números difusos.

**Átomo de Patrón:** tiene asociado la función de pertenencia de un conjunto difuso, el cual restringe los valores más o menos compatibles con su significado. Estos valores pertenecen a alguna escala prescrita correspondiente al rango de un atributo, al cual el átomo hace referencia.

**Base de Datos Relacional (BDR):** sistema de almacenamiento en la cual toda la información está almacenada en tablas (relaciones). Cada relación tiene varios atributos(o columnas) y la información está almacenada en las filas de cada relación.

**Caloría:** unidad utilizada para expresar el valor de energía o calor producido por un nutriente al utilizarse por el organismo.

**Carbohidrato:** nutriente compuesto de carbono, hidrógeno y oxígeno, conformado en diferentes combinaciones y enlaces (glucosa, fructosa, galactosa). Aporta 4 calorías por cada gramo de carbohidratos.

**Conjunto Difuso:** conjunto difuso A en X caracterizado por una función miembro $f_A(x)$, la cual asocia a cada punto en X un número real en el intervalo [0,1], con el valor de $f_A(x)$ en x representando el "grado de pertenencia" de x en A. Siendo X un espacio de puntos con un elemento genérico de X denotado por x.

**Compuesto Alimenticio:** formado por la combinación de varios alimentos.

**Componente de dato:** elemento de un conjunto especificado en un patrón.

**Dato Difuso:** representa un objeto mal conocido cuya descripción precisa no está disponible y el cual puede ser representado por un conjunto difuso.

**Dato nítido:** dato que no posee imprecisión o incertidumbre.

**Dieta:** cómo, cuánto y qué clase de alimentos se deben tomar diariamente para satisfacer las necesidades nutritivas del organismo.

**Dominio de un atributo:** conjunto de todos los valores que pueden ser tomados por un atributo.

**Emparejamiento de Patrón Difuso:** involucra un patrón para describir algunos requerimientos y una base de conocimiento donde la data está organizada, proporcionando una evaluación de la similitud o compatibilidad entre el patrón y un dato en el intervalo [0,1].

**Emparejamiento de Patrones Difusos Ponderado:** técnica de emparejamiento donde se le asigna grados de importancia a los átomos del patrón.

**Espacio de Estados:** conjunto de estados factibles para un problema junto a la estructura de adyacencia definida implícitamente por los operadores o reglas.

**Función de Evaluación:** Estimación del costo total de la solución más cercana en el área. Está compuesta por dos términos: g, el cual denota el costo mínimo de transformación desde el nodo inicial a un nodo m y h, el cual denota una estimación del costo para transformar m a sus soluciones más cercanas.

**Grado difuso:** valor que indica el nivel de relevancia respecto a algún significado concreto de un determinado valor o valores (tupla). Normalmente dicho valor será un valor real entre 0 y 1, aunque pueden considerarse otros valores (con otros dominios, distribuciones de posibilidad, etc.). El significado de este grado puede ser variado, aunque el más usual, es el grado de pertenencia.

**Grado de importancia:** grado asignado a un átomo de patrón en el intervalo [0,1] para indicar su importancia en el proceso de emparejamiento.

**Heurística:** se deriva del griego "heuriskein" que significa encontrar o descubrir.

**Índice de masa corporal (IMC):** valor que se calcula en base al peso y estatura de un individuo, y el cual permite determinar su estado de nutrición, es decir, si tiene peso normal, sobrepeso o bajo peso.

**Lípido o grasa:** nutriente compuesto por carbono, hidrógeno, oxígeno y otros elementos como fósforo entre otros (ácidos grasos + glicerol).
Su función principal es ser formador de energía de reserva en el organismo.
Aporta 9 calorías por cada gramo de grasa.

**Medida de Posibilidad ($\Pi$):** grado de solapamiento del conjunto difuso de valores compatibles con P con el conjunto difuso de valores posibles de D.

**Medida de Necesidad (N):** grado de inclusión del conjunto de valores posibles de D en el conjunto de valores compatibles con P.

**Método de Búsqueda Heurístico:** método donde se dispone de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores.

**Nutriente:** sustancia química indispensable que aporta energía; tal como, las proteínas, grasas y carbohidratos.

**Nutrición:** disciplina científica que estudia y analiza los procesos mediante los cuales el organismo utiliza, transforma e incorpora en sus estructuras una serie de sustancias químicas proporcionadas por los alimentos.

**Nutrirse:** proceso por medio del cual el individuo toma los alimentos y utiliza sus nutrientes, para el funcionamiento normal de su organismo.

**Patrón difuso:** representa una clase mal limitada de objetos.

**Proteína:** nutriente compuesto de carbono, nitrógeno, oxígeno e hidrógeno (aminoácidos). Su función principal es construir o reparar tejidos en el organismo. Aporta 4 calorías por cada gramo de proteína.

**Solución Óptima:** camino solución cuyo costo sea mínimo entre los de los caminos solución.

**Técnicas de Emparejamiento de Patrón Difuso:** basadas en medidas de posibilidad y necesidad, donde cada componente de dato y cada requerimiento elemental son asociados a una distribución de posibilidad y a un conjunto difuso.

# CAPÍTULO III
# MARCO METODOLÓGICO

## Tipo de Investigación.

El tipo de investigación utilizada correspondió a la de “proyecto factible” o estudio de proyecto, ya que se trata de una proposición sustentada en una herramienta computacional viable, tendente a satisfacer una necesidad referida a tecnología, métodos y procesos, como lo es el diseño y desarrollo de una solución computacional para obtener compuestos alimenticios con una combinación balanceada de nutrientes utilizando lógica difusa y búsquedas heurísticas.

Este proyecto se apoya además en investigación documental, ya que se revisaron aspectos teórico-prácticos relacionados con lógica difusa y búsquedas heurísticas tales como; la teoría de conjuntos difusos, teoría de la posibilidad, técnicas de emparejamiento de patrones difusos y algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios. Además, se define la investigación como aplicada ya que permitió llevar a la práctica los aspectos teóricos antes mencionados.

El trabajo de investigación desarrollado se enmarca dentro de la Línea de Investigación de Inteligencia Artificial, ubicándose específicamente en el campo de la lógica difusa (aplicada a la teoría de conjuntos difusos, a la teoría de la posibilidad y al emparejamiento de patrones difusos) y búsquedas heurísticas (estrategias heurísticas de los algoritmos A) .

## Fases del Estudio

Para lograr la culminación del proyecto se cumplieron una serie de fases o etapas, contempladas dentro del manual para la presentación del trabajo conducente al grado académico de Magíster Scientiarum en Ciencias de la Computación, mención Inteligencia Artificial. A continuación se explicarán los puntos a tratar en cada una de ellas:

## Fase 1: Diagnóstica.

### Diseño de la investigación o procedimiento.

Posteriormente a la investigación documental se realizaron los siguientes pasos:

1. Se analizaron los requerimientos nutricionales de los venezolanos los cuales permitieron definir los valores de entrada y el dominio difuso del conocimiento a almacenar.
2. Se definieron criterios para la representación de los valores de entrada.
3. Se definieron criterios para el diseño de la BD de compuestos alimenticios.
4. Se definieron criterios para la interacción herramienta-usuario.
5. Se seleccionó la técnica de emparejamiento de patrones difusos a implementar.
6. Se analizaron algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios.
7. Se desarrolló la herramienta implementando las técnicas de lógica difusa y búsquedas heurísticas.
8. Se evaluó la herramienta balanceando un conjunto de compuestos alimenticios, lo cual permitió analizar el comportamiento de los algoritmos heurísticos de balanceo implementados y comparar su eficiencia en la obtención de compuestos alimenticios con una combinación balanceada de nutrientes, con los cuales se satisfagan ciertos requerimientos nutricionales.
9. Se establecieron las conclusiones y recomendaciones de fase diagnóstica.

**Análisis de los requerimientos nutricionales de los venezolanos para definir los valores de entrada y el dominio impreciso del conocimiento a almacenar.**

Estos requerimientos se determinaron en base a las necesidades nutricionales recomendadas para los venezolanos (Instituto Nacional de Nutrición, 2000) en cuanto a; las cantidades de nutrientes básicos (carbohidratos, lípidos o grasas y proteínas) estadísticamente comprobadas que se deben consumir diariamente para satisfacer el requerimiento calórico diario de un individuo en particular.

Se definieron los tipos de alimentos requeridos por los usuarios venezolanos y se investigó en tablas de composición de alimentos suministradas por la División de Investigaciones en Alimentos del INN (Instituto Nacional de Nutrición, 1999) la composición nutricional de los mismos en cuanto a; sus cantidades (gramos) de carbohidratos, grasas y proteínas, además de revisar manuales de nutrición elaborados por nutricionistas (Lic. Millán María del Pilar, 1998), los cuales ofrecen una variedad de menúes, para llevar a cabo un plan de alimentación adecuado a través del número de comidas diarias (desayuno, almuerzo, meriendas y cena) y en los cuales se da información acerca del aporte calórico de cada una de los nutrientes básicos; esto con la finalidad de construir una base de datos lo más representativa, la cual permita proporcionar al usuario una gran variedad de alimentos cuando este requiera satisfacer un requerimiento nutricional.

**Definición de criterios para la representación de los valores de entrada.**

Se definieron los criterios para la representación de los valores de entrada por medio de números difusos trapezoidales modelados por una 4-tupla (a, b, c, d), con la finalidad de representar tanto números nítidos y difusos. En esta representación los valores a y b representan el núcleo del número difuso, y los valores c y d representan el soporte derecho e izquierdo respectivamente. Por lo tanto, se definió una forma o lenguaje para tal representación en base a los siguientes criterios: (a) para los números nítidos o conocidos precisamente, por ejemplo, el número “120”, su

representación trapezoidal es (120,120,0,0) y (b) para la obtención general de números difusos, su representación trapezoidal (a, b, c, d) se obtuvo mediante un porcentaje para representar la amplitud del núcleo (área comprendida entre [a,b]) y un porcentaje para representar el tamaño de la dispersión (área comprendida entre [a-c] y [b+d]). Por ejemplo, el par de valores (10,20) representan la imprecisión del peso del pan cuando es representado por una rebanada. Si la rebanada pesa 25 gramos, el número difuso trapezoidal que representa el peso de la misma es: (25–25*10%, 25+25*10%, 25*20%, 25*25%), es decir, (22.5, 27.5, 5, 5). La amplitud del núcleo está representada por el valor 2,5 y el tamaño de la dispersión está representada por el valor 5, ver Figura 8.

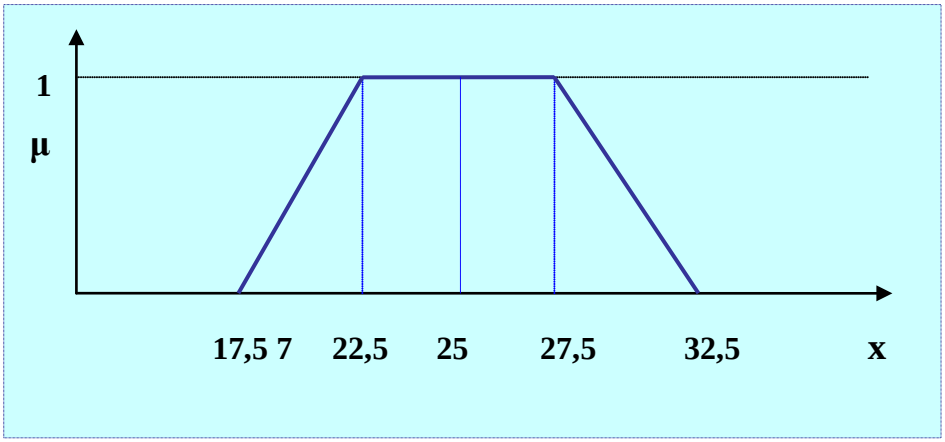



**Figura 8.** Representación trapezoidal del peso de la rebanada de pan.

Los criterios anteriores permiten también representar las cantidades nítidas o difusas de los nutrientes básicos y de sus metas calóricas en los tipos de comidas seleccionados por el usuario (desayuno, almuerzo, meriendas, cena), además de permitir la realización de todos los cálculos de aritmética difusa. Se definió una forma adecuada para mostrar estas cantidades en la interfaz de la herramienta con el usuario; por ejemplo, el peso de la rebanada de pan tiene la representación difusa: (25 ± 2,5 ± 5).

**Definición de criterios para el diseño de la base de datos de compuestos alimenticios.**

Se escogió Microsoft Office Access 2003 como herramienta de software para el diseño y manejo adecuado de las tablas que componen la base de datos y en las cuales se almacenaron todos los datos relacionados a los componentes alimenticios (alimentos), medidas, etc. Esto permitió identificar y describir los atributos que forman parte del dominio del problema y establecer las relaciones entre ellos. En las Tablas 13 a la 21 se da una descripción de los atributos correspondientes a las tablas que forman parte de esta base de datos, donde el símbolo ⌂ significa campo o llave principal.

**Tabla 13:** Descripción de los atributos de la tabla de manejo de Alimentos.

| ***Tabla de Alimentos*** | | |
|---|---|---|
| ***Descripción:*** Tabla que contiene la información referente a los tipos de alimentos y a la clase a la que pertenecen. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código del alimento | Número<br>***Tamaño:*** Entero largo | Código con que se identifica un alimento. |
| Código de la clase | Número<br>***Tamaño:*** Entero largo | Código de la clase a la que pertenece el alimento, por ejemplo; lácteos, carnes blancas, entre otras. |
| Nombre | Texto<br>***Tamaño:*** hasta 50 caract. | Nombre del alimento. |

**Tabla 14.** Descripción de los atributos de la tabla de manejo de medidas.

| ***Tabla de Medidas*** | | |
|---|---|---|
| ***Descripción:*** Tabla que contiene la información referente a las medidas manejadas. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la medida | Numérico<br>***Tamaño:*** Entero largo | Código para identificar un tipo de medida. |
| Nombre | Texto<br>***Tamaño:*** hasta 50 caract. | Nombre de la medida, por ejemplo, taza, trozo, etc. |

**Tabla 15.** Descripción de los atributos de la tabla de manejo de usuarios.

| ***Tabla de usuarios*** | | |
|---|---|---|
| ***Descripción:*** Tabla que contiene la información referente a sus parámetros físicos. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código del usuario | Número<br>***Tamaño:*** Entero largo | Código con que se identifica un usuario. |
| ⌂ Código de dieta actual | Número<br>***Tamaño:*** Entero largo | Código que permite relacionar el usuario con una dieta en particular. |
| Apellidos | Texto<br>***Tamaño:*** hasta 25 caract. | Almacena los apellidos del usuario. |
| Nombres | Texto<br>***Tamaño:*** hasta 25 caract. | Almacena los nombres del usuario. |
| Fecha de nacimiento | Fecha<br>***Tamaño:*** Fecha corta | Permite describir la fecha de nacimiento en cuanto al día, mes y año; la cual permite calcular la edad. |

**Tabla 15 (Cont.).** Descripción de los atributos de la tabla de manejo de usuarios.

| | | |
|---|---|---|
| Peso | Número<br>***Tamaño:*** Número simple | Almacena el peso del usuario, el cual permitirá el cálculo de su requerimiento calórico. |
| Estatura | Número<br>***Tamaño:*** Número simple | Almacena la estatura del usuario, la cual permitirá el cálculo de su requerimiento calórico. |
| Fecha de registro | Fecha<br>***Tamaño:*** Fecha corta | Permite describir la fecha de registro en la tabla de usuarios en día, mes y año. |

**Tabla 16.** Descripción de los atributos de la tabla de medidas por alimento.

| ***Tabla de medidas por alimento.*** | | |
|---|---|---|
| ***Descripción:*** Tabla que contiene la información de la cantidad de carbohidratos, grasas o lípidos y proteínas en la medida de un alimento. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código del alimento | Número<br>***Tamaño:*** Entero largo | Código que identifica un alimento. |
| ⌂ Código de la medida | Número<br>***Tamaño:*** Entero largo | Código que identifica un tipo de medida relacionada al alimento. |
| Cantidad de carbohidratos | Número<br>***Tamaño:*** Simple | Cantidad de carbohidratos contenidos en el alimento. |

**Tabla 16 (Cont).** Descripción de los atributos de la tabla de medidas por alimento.

| | | |
|---|---|---|
| Cantidad de grasas | Número<br>***Tamaño:*** Simple | Cantidad de lípidos o grasas contenidos en el alimento. |
| Cantidad de proteínas | Número<br>***Tamaño:*** Simple | Cantidad de proteínas contenidas en el alimento. |
| Gramos/cc | Texto<br>***Tamaño:*** hasta 2 caract. | Representa los g/cc que ocupa la medida. |
| Porcentaje del núcleo | Número<br>***Tamaño:*** Entero largo | Permite obtener la amplitud del núcleo del número difuso. |
| Porcentaje de dispersión | Número<br>***Tamaño:*** Entero largo | Permite obtener la dispersión del núcleo del número difuso. |
| Gramos Mínimos | Número<br>***Tamaño:*** Simple | Cantidad mínima en g/cc del alimento. |
| Cantidad en mínimo | Número<br>***Tamaño:*** Simple | Cantidad mínima que representa los g/cc del alimento, por ejemplo; 1 Rebanada, ½ Taza, etc. |
| Promedio | Número<br>***Tamaño:*** Simple | Cantidad promedio en g/cc del alimento. |
| Cantidad en promedio | Número<br>***Tamaño:*** Simple | Cantidad promedio que representa los g/cc del alimento. |
| Máximo | Número<br>***Tamaño:*** Simple | Cantidad máxima en g/cc del alimento. |
| Cantidad en máximo | Número<br>***Tamaño:*** Simple | Cantidad máx. que repre-senta los g/cc del alimento. |

**Tabla 17:** Descripción de los atributos de la tabla de alimentos por comidas.

| ***Tabla de alimentos por comida*** | | |
|---|---|---|
| ***Descripción:*** Contiene la información de los alimentos contenidos en una comida.. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la comida | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una comida. |
| ⌂ Código del alimento | Número<br>***Tamaño:*** Entero largo | Código del alimento relacionado a una comida. |
| ⌂ Código de la medida | Número<br>***Tamaño:*** Entero largo | Código de la medida relacionada a un alimento. |
| Cantidad | Número<br>***Tamaño:*** Simple | Cantidad del alimento en una medida. |

**Tabla 18:** Descripción de los atributos de la tabla de clases de alimentos.

| ***Tabla de clases de alimentos.*** | | |
|---|---|---|
| ***Descripción:*** Tabla que contiene la información referente a las clases de alimentos, es decir, contiene la clasificación de los alimentos por tipo (lácteos, carnes rojas, carnes blancas, verduras y hortalizas, frutas, entre otras). Establece una relación con la tabla de alimentos. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la clase | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una clase de alimento. |
| Nombre | Texto<br>***Tamaño:*** hasta 25 caract. | Nombre de la clase. |

**Tabla 19:** Descripción de los atributos de la tabla de manejo de comidas.

| ***Tabla de comidas.*** | | |
|---|---|---|
| ***Descripción:*** Contiene información de las comidas requeridas por un usuario. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la comida | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una comida. |
| Tipo de comida | Número<br>***Tamaño:*** Entero largo | Identifica un tipo de comida (desayuno, merienda, almuerzo, cena). |
| Total carbohidratos | Número<br>***Tamaño:*** simple | Cantidad total de carbohidratos en la comida. |
| Total lípidos o grasas | Número<br>***Tamaño:*** simple | Cantidad total de lípidos en la comida. |
| Total proteínas | Número<br>***Tamaño:*** simple | Cantidad total de proteínas en la comida. |

**Tabla 20:** Descripción de los atributos de la tabla de comidas por dieta.

| ***Tabla de comidas por dieta.*** | | |
|---|---|---|
| ***Descripción:*** Contiene información sobre las comidas de una dieta. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la dieta | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una dieta. |
| ⌂ Código de la comida | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una comida. |

**Tabla 21:** Descripción de los atributos de la tabla de dietas.

| ***Tabla de dietas.*** | | |
|---|---|---|
| ***Descripción:*** Contiene información sobre las dietas correspondientes de un usuario. | | |
| ***Atributos*** | | |
| ***Nombre*** | ***Tipo de dato*** | ***Descripción*** |
| ⌂ Código de la dieta | Número<br>***Tamaño:*** Entero largo | Código con que se identifica una dieta. |
| ⌂ Código del usuario | Número<br>***Tamaño:*** Entero largo | Código con que se identifica un usuario. |
| Fecha | Fecha<br>***Tamaño:*** Fecha corta | Fecha en día, mes y año en que se sugirió la dieta. |

Luego de haber explicado los atributos de las tablas que conforman la base de datos, la Figura 9 muestra el diagrama entidad-relación.

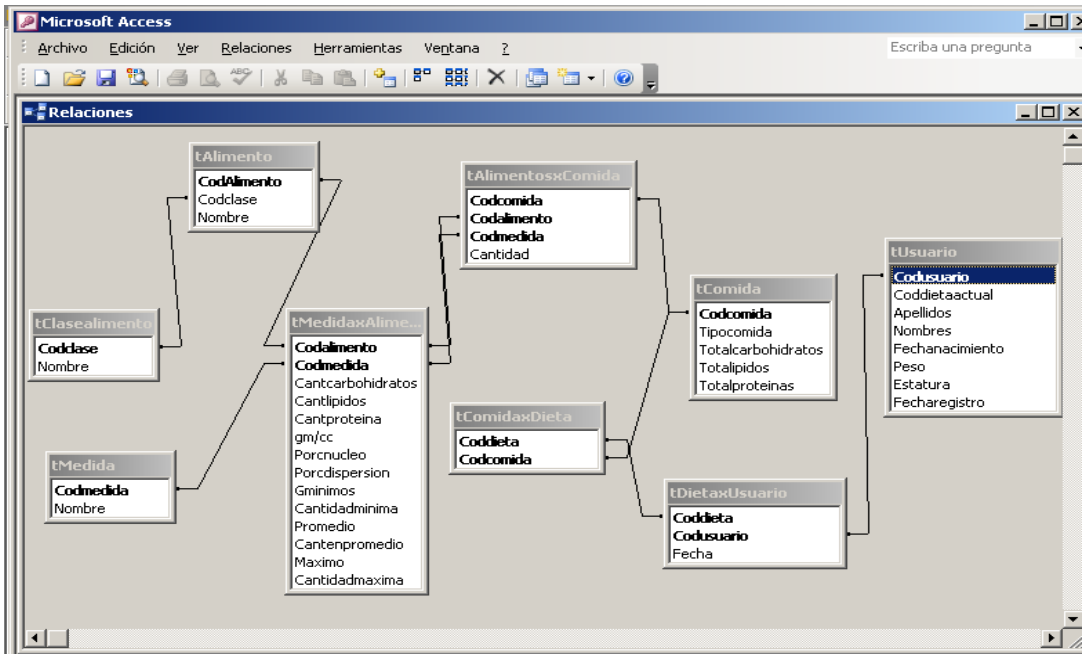



**Figura 9:** Diagrama entidad-relación de la herramienta computacional.

**Definición de criterios para la interacción herramienta-usuario.**

Por medio de una interfaz amigable la herramienta permite al usuario realizar siguientes operaciones:

1. En la interfaz de entrada de datos el usuario puede incluir sus datos personales y parámetros físicos tales como; edad, peso y estatura; obteniendo como respuesta su índice de masa corporal, la calificación de su peso (normal, sobrepeso o bajo peso) y su requerimiento calórico total (calorías que debe consumir de acuerdo a su peso o índice de masa corporal). La Figura 10 muestra esta interfaz o pantalla de entrada de datos.

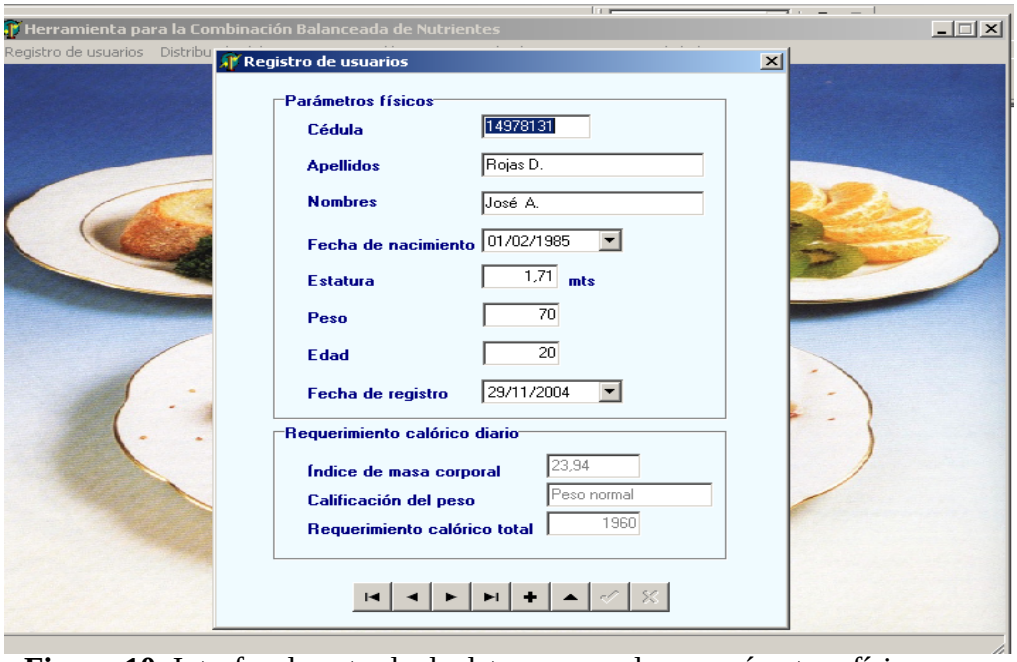



**Figura 10.** Interfaz de entrada de datos personales y parámetros físicos.

2. La herramienta da la facilidad al usuario de seleccionar el número de comidas que desea entre los siguientes tipos: desayuno, merienda mitad mañana, almuerzo, merienda mitad tarde y cena. El Instituto Nacional de Nutrición (INN, Revisión 2000) recomienda que la distribución del requerimiento calórico para las principales comidas, sea 25% para el desayuno, 45% para el almuerzo y 30% para la cena. Si son

seleccionadas las comidas adicionales (meriendas) se les asignaría un 10% a cada una, lo cual resultaría de la disminución de los porcentajes para el desayuno, almuerzo y cena. El usuario debe seleccionar al menos las tres comidas principales, para que la distribución de su requerimiento calórico le permita alimentarse de forma balanceada. Por ejemplo, si el usuario selecciona las tres comidas principales y su requerimiento calórico es 1440 calorías, entonces la Tabla 22 indica el cálculo de los gramos de nutrientes que debe consumir en el desayuno.

**Tabla 22.** Cálculo de los gramos de nutrientes para un desayuno.

| ***Requerimiento Calórico = 1440*** | | |
|---|---|---|
| ***Tipo de Comida*** | ***Porcentaje de Distribución*** | ***Distribución para cada nutriente*** |
| Desayuno | 1440 x 0.25 = 360 calorías.<br><br>- La distribución recomendada por el INN es 25%. | **Carbohidratos**<br>Gramos = 360 x 0.60= 216/ 4 = 54<br>- 1 gramo aporta 4 calorías<br>- La norma nutricional para los carbohidratos es el 60% del total calórico.<br>**Lípidos o Grasas**<br>Gramos = 360 x 0.25= 90/ 9 = 10<br>- 1 gramo aporta 9 calorías<br>- La norma nutricional para los lípidos es el 25% del total calórico.<br>**Proteínas**<br>Gramos = 360 x 0.15= 54/ 4 = 14<br>- 1 gramo aporta 4 calorías<br>- La norma nutricional para las proteínas es el 15 % del total calórico. |

Posterior a la distribución se mostrará en forma gráfica las cantidades de calorías asignadas a los tipos de comidas seleccionados. La figura 11 muestra esta interfaz de distribución del requerimiento calórico.

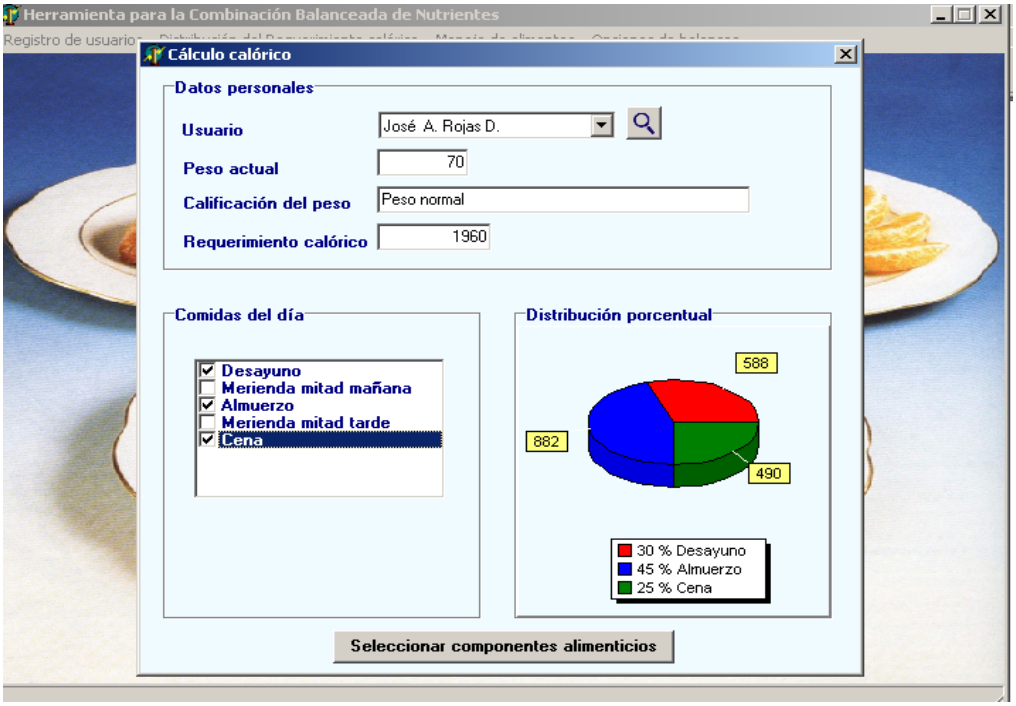



**Figura 11.** Interfaz de distribución del requerimiento calórico.

3. Para un tipo de comida seleccionado la herramienta permite al usuario escoger los componentes alimenticios respectivos, los cuales están clasificados en las siguientes categorías: (a) huevos, lácteos y derivados, (b) carnes rojas y vísceras, (c) carnes blancas, (d) alimentos grasos y embutidos, (e) harinas y cereales, (f) pescados y mariscos, (g) vegetales y hortalizas, (h) frutas (i) leguminosas, (j) alimentos preparados, (k) y bebidas alcohólicas y analcohólicas. Cada categoría ofrece una variedad de componentes alimenticios, en los cuales la herramienta indica para cada selección la unidad de medida en la cual se representa el alimento (taza, trozo, rebanada, cucharada, etc.) y la cantidad (2 rebanadas, 2 cucharadas, trozo pequeño, 2/3 taza, etc.). La Figura 12 muestra la interfaz de selección de componentes alimenticios.

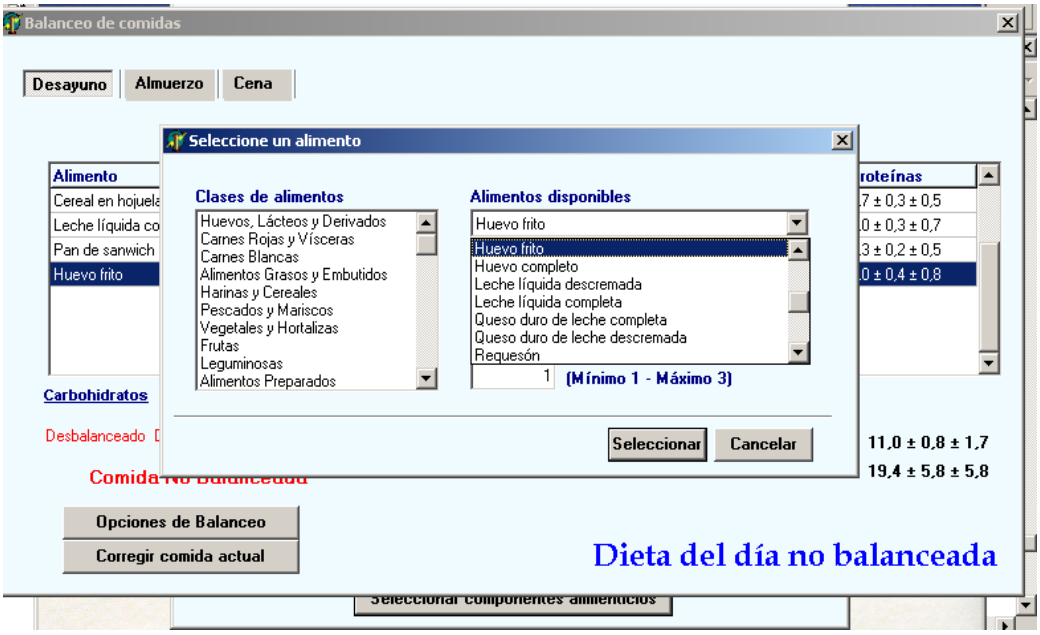



**Figura 12.** Interfaz de selección de componentes alimenticios.

4. Luego de la selección de los componentes alimenticios se muestra la siguiente información: tipo de medida en la cual es expresado el componente, cantidad de la medida y las cantidades difusas de los nutrientes (carbohidratos, lípidos y proteínas) contenidos en cada componente. Se muestran las representaciones de los números difusos de las metas calóricas (gramos máximos para cada nutriente) y las representaciones de los números difusos para indicar el total de gramos de nutrientes en el compuesto alimenticio (comida). Si los índices de emparejamiento (posibilidad y necesidad) indican desbalances nutricionales, el usuario tendrá la opción de corregir el compuesto alimenticio hasta conseguir uno que se aproxime lo más posible a las metas calóricas establecidas para cada nutriente. Si se encuentra una solución se mostrará al usuario las modificaciones que el procedimiento de corrección ejecutó sobre el compuesto alimenticio para balancearlo. Además la herramienta permite al usuario guardar y recuperar compuestos alimenticios. La Figura 13 muestra esta interfaz de usuario y la Tabla 23 muestra en forma resumida el procedimiento general de especificación de requerimientos nutricionales y distribución del requerimiento calórico.

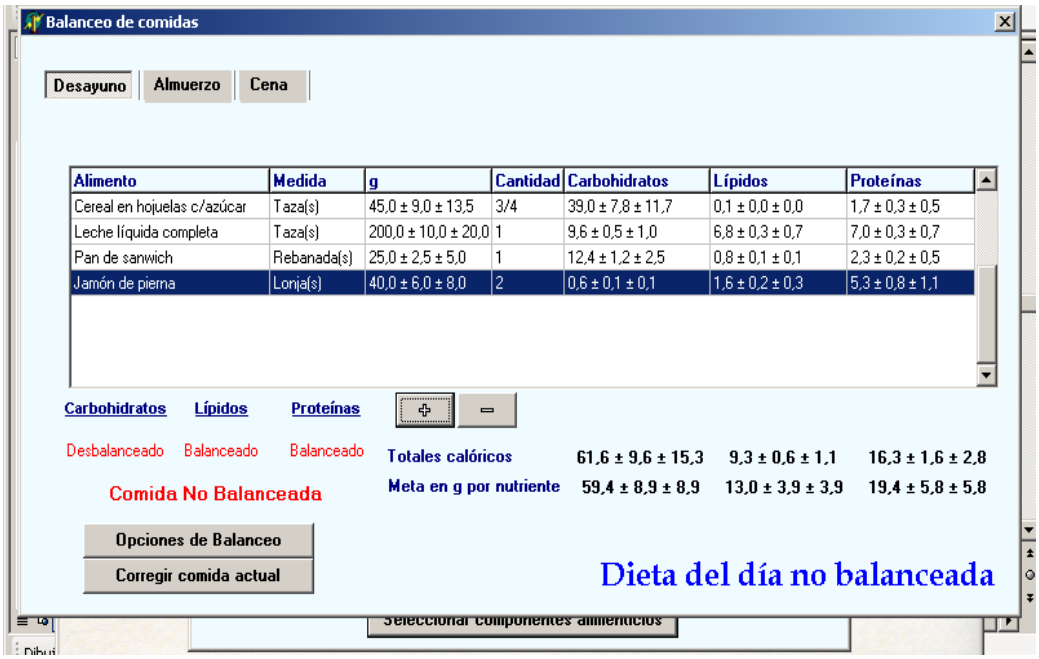



**Figura 13.** Interfaz de usuario para corregir un compuesto alimenticio.

**Tabla 23:** Procedimiento para especificar requerimientos nutricionales.

| |
|---|
| **Procedimiento:** Especificación de requerimientos nutricionales por parte del usuario y distribución del requerimiento calórico.<br>**Variables relacionadas:** Componentes del compuesto alimenticio, Cantidad de Componente, Requerimiento Calórico, Tipo de Comida, Patrón difuso de requerimientos de nutrientes (número difuso que representa la cantidad meta para un nutriente *n*), Número difuso o componente de dato que representa el total de nutriente *n* en una comida. |
| **Comienzo**<br>1 Calcular el requerimiento calórico del usuario (cantidad de calorías que se deben consumir en un día).<br><br>2 Escoger el número de comidas (desayuno, merienda, almuerzo o cena ) deseadas por el usuario.<br><br>3 Distribuir el requerimiento calórico del usuario entre el número de comidas escogidas.<br><br>4 Transformar el requerimiento calórico de cada una de las comidas escogidas en |

**Tabla 23 (Cont.):** Procedimiento para especificar requerimientos nutricionales.

| |
|---|
| las cantidades difusas que representan las metas calóricas de los nutrientes considerados.<br><br>5 Para un compuesto alimenticio (comida) requerido<br>Seleccionar los componentes alimenticios almacenados en base de datos, en los cuales se indicará el tipo de medida y la cantidad.<br><br>6 Mostrar las cantidades difusas de composición nutricional de los carbohidratos, lípidos y proteínas del compuesto seleccionado.<br><br>7 Mostrar las cantidades totales en forma difusa para cada nutriente *n*.<br><br>8 Si se desea, se pueden modificar las cantidades de los componentes.<br>**Fin** |

**Selección de la técnica de emparejamiento de patrones difusos a implementar.**

Se seleccionó la técnica de emparejamiento de patrones difusos ponderada (Dubois y otros, 1988), ya que permite de forma adecuada evaluar el contenido nutricional de un compuesto alimenticio por medio del cálculo de índices de compatibilidad en términos de posibilidad y necesidad, indicándose por medio de éstos el grado de balance nutricional en sus componentes. Un compuesto tiene una combinación balanceada de nutrientes cuando el índice de posibilidad es igual a 1 y el índice de necesidad es mayor o igual a 0.7, esto significa, que las cantidades de nutrientes están emparejadas con sus respectivas metas calóricas o normas de nutrición. A la técnica escogida se le denomina ponderada, ya que en el proceso de emparejamiento se le asigna a los átomos del patrón un grado de importancia en el intervalo [0,1]. Los átomos con grados altos tienen mayor importancia en el emparejamiento que los átomos con grados bajos. Por lo tanto, se le asignó un grado de importancia igual a 1 a todos los nutrientes considerados en el emparejamiento. La Tabla 24 resume las ecuaciones para el cálculo de los índices de posibilidad y necesidad tomando en cuenta la importancia de los átomos de patrón $w_i$.

**Tabla 24.** Ecuaciones para el cálculo de $\Pi$ y N tomando en cuenta la importancia $w_i$.

| | |
|---|---|
| ***Índice de Posibilidad* ($\Pi$)** | $\Pi = \min_{i=1,\ldots,n} \max(1 - w_i, \Pi(P_i; D_i))$ donde<br>$\Pi(P;D) = \underset{u \in U}{Sup} \min\left(\mu_P(u), \pi_D(u)\right)$ |
| ***Índice de Necesidad* (N)** | $N = \min_{i=1,\ldots,n} \max(1 - w_i, N(P_i; D_i))$ donde<br>$N(P;D) = \inf_{u \in U} \max\left(\mu_P(u), 1 - \pi_D(u)\right)$<br>*Para ambos casos, la ecuación de la recta que pasa por dos puntos se describe como:*<br>$y = \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) + y_0$ |

Para realizar la implementación de esta técnica de emparejamiento se calculó las funciones de membresía para un número difuso trapezoidal P, en base a sus coordenadas (a, b, c, d). Estas funciones de membresía se muestran a continuación.

$$\mu_p = \begin{cases} 0 & si \quad u < a_1\text{-}c_1 \\ 1/c_1(u\text{-}a_1)+1 & si \quad a_1\text{-}c_1 < u < a_1 \\ 1 & si \quad a_1 \leq u \leq b_1 \\ 1\text{-}1/d_1(u\text{-}b_1) & si \quad b_1 \leq u \leq b_1 + d_1 \\ 0 & si \quad u > b_1 + d_1 \end{cases}$$

Luego de haber establecido las funciones de membresía, la posibilidad entre dos números difusos trapezoidales P y D se obtiene por medio del cálculo de

intersección de líneas utilizando las ecuaciones del índice de posibilidad y de la recta que pasa por dos puntos descritas en la Tabla 24, es decir, el valor de la posibilidad se obtiene calculando el punto máximo de las rectas donde comienzan a intersectarse o solaparse P y D. Por lo tanto, se implementó un método de emparejamiento de patrones difusos basándose en las posiciones de $\Pi(P,D)$ en el intervalo [0,1] descritas en la Tabla 6, donde el número difuso trapezoidal P representa un átomo de patrón con coordenadas ($a_1$,$b_1$,$c_1$,$d_1$), correspondiente a una norma de nutrición para el nutriente n y el número difuso trapezoidal D representa un componente de dato con coordenadas ($a_2$,$b_2$,$c_2$,$d_2$), correspondiente a la cantidad total del nutriente n en un compuesto alimenticio. La Tabla 25 resume los valores de posibilidad para los números difusos trapezoidales P y D.

**Tabla 25.** Valores de posibilidad para los números difusos trapezoidales P y D.

| $\Pi(P,D)$ | | ***Condición*** |
|---|---|---|
| *0* | *si* | $b_1+d_1 < a_2-c_2$ *o* $b_2+d_2 < a_1-c_1$ |
| *1* | *si* | $a_2 <= b_1$ *y* $b_1 <= b_2$ *o* $a_1 <= b_2$ *y* $b_2 <= b_1$ |
| $1-((a_2-b_1)/(c_2+d_1))$ | *si* | $(b_1 < a_2$ *y* $a_1-c_1 < a_2-c_2)$ *o* $b_1 < a_2$ |
| $1-((a_1-b_2)/(c_1+d_2))$ | *si* | $(b_2 < a_1$ *y* $a_2-c_2 < a_1-c_1)$ *o* $b_2 < a_1$ |

La necesidad entre dos números difusos trapezoidales P y D se obtiene también por medio del cálculo de intersección de líneas, utilizando las ecuaciones de la recta que pasa por dos puntos y del índice de necesidad descritas en la Tabla 24, es decir, el valor de la necesidad se obtiene calculando el punto mínimo (entre $N_1$ y $N_2$, cuando el valor de la necesidad es mayor que 0 y menor que 1) donde la recta del número difuso trapezoidal P es intersectada por la recta complemento del número difuso trapezoidal D. La Tabla 26 resume las ecuaciones generales para el cálculo de los valores de necesidad para los números difusos trapezoidales P y D.

**Tabla 26.** Valores de necesidad para los números difusos trapezoidales P y D.

| $\Pi(P, D)$ | | ***Condición*** |
|---|---|---|
| *0* | *si* | $a_2 < (a_1-c_1)$ *o* $b_2 > (b_1+d_1)$<br>*o* $(a_1 < a_2$ *y* $c_1 <> 0$ *o* $c_2 <> 0)$<br>*o* $(b_1 < b_2$ *y* $d_1 <> 0$ *o* $d_2 <> 0)$<br>*o* $(c_1 = 0$ *y* $c_2 = 0$ *y* $a_1 > a_2)$<br>*o* $(d_1 = 0$ *y* $d_2 = 0$ *y* $b_1 > b_2)$ |
| *1* | *si* | $a_1 <= a_2-c_2$ *y* $b_1 >= b_2+d_2$<br>*o* $(a_1 = a_2$ *y* $b_1 = b_2$ *y* $c_1 = 0$ *y* $c_2 = 0$ *y* $d_1 = 0$ *y* $d_2 = 0)$<br>*o* $(c_1 = 0$ *y* $c_2 = 0$ *y* $a_1 \leq a_2)$<br>*o* $(d_1 = 0$ *y* $d_2 = 0$ *y* $b_1 \leq b_2)$ |
| $N_1 = (a_2-a_1+c_1)/(c_1+c_2)$ | *si* | *no está en los casos anteriores*<br>*y* $(c_1 = 0$ *o* $c_2 = 0)$ |
| $N_2 = (b_1-b_2+d_1)/(d_1+d_2)$ | *si* | *no está en los casos anteriores*<br>*y* $(d_1 = 0$ *o* $d_2 = 0)$ |

Luego de haber detallado el proceso de cálculo de los índices de posibilidad y necesidad, la Tabla 27 muestra el procedimiento en forma generalizada del emparejamiento de patrones difusos con la técnica que permite tomar en cuenta la importancia de los átomos de patrón.

**Tabla 27.** Procedimiento general del emparejamiento de patrones difusos.

| |
|---|
| **Procedimiento:** Evaluación de compuesto alimenticio con técnica de emparejamiento de patrones difusos ponderada. |
| **Variables relacionadas:** Índice de necesidad de emparejamiento, Índice de posibilidad de emparejamiento, Componente de dato que contiene la cantidad difusa |

**Tabla 27 (Cont.).** Procedimiento general del emparejamiento de patrones difusos.

total del nutriente n en una comida, Peso de la importancia del átomo de patrón difuso para el nutriente n, Átomo de patrón que contiene la cantidad difusa de la norma médica de nutrición para el nutriente n.

**Comienzo**

Para cada átomo de patrón que contiene la cantidad difusa de la norma médica para el nutriente *n* en una comida (desayuno, merienda, almuerzo o cena).

**Repetir**

1 Calcular el índice de posibilidad del átomo de patrón difuso para el nutriente *n* y su componente de dato correspondiente tomando en cuenta la importancia.

2 Almacenar el índice de posibilidad del átomo de patrón para el nutriente *n* en un vector que contiene los valores de posibilidad parcial.

3 Calcular el índice de necesidad del átomo de patrón difuso para el nutriente *n* y su componente de dato correspondiente tomando en cuenta la importancia.

4 Almacenar el índice de necesidad del átomo de patrón para el nutriente *n* en un vector que contiene los valores de necesidad parcial.

**Fin de Repetir**

5 Calcular el mínimo de los valores de posibilidad parcial como la posibilidad global.

6 Calcular el mínimo de los valores de necesidad parcial como la necesidad global.

7 Si posibilidad global = 1 y necesidad global >= 0.7
entonces existe un balance de nutrientes con respecto a sus normas de nutrición
de lo contrario dar un mensaje indicando que la comida no está
balanceada, mostrar los nutrientes responsables del desbalance y dar la
opción al usuario de corregir o balancear el compuesto alimenticio.

**Fin**

**Análisis de algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios.**

Se analizaron algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios (Buisson y Garel, 2003) con la finalidad de comparar los métodos empleados en la búsqueda de soluciones. Tales algoritmos emplean las estrategias de búsqueda heurística de los algoritmos A (Rubio y otros,1998), en las cuales se toma en cuenta la estimación heurística de la proximidad a un nodo objetivo y el cálculo de los costos de los caminos que se van construyendo, eligiendo en cada paso el menos costoso. Por lo tanto la idea básica (Nilsson, 2001) es utilizar una función de evaluación heurística, la cual tomará valores pequeños en los nodos más prometedores, ayudando de esta forma a decidir cuál es el mejor nodo a expandir (con el menor valor de la función de evaluación). Se terminará el proceso de búsqueda cuando el nodo a expandir sea un nodo objetivo.

En la función de evaluación $f = g + h$ de los algoritmos de balanceo estudiados, g denota el costo mínimo de transformación desde el compuesto alimenticio inicial a un estado *m* del mismo. Es decir, para el cálculo de g se van a emplear un conjunto de operaciones sobre el contenido nutricional del compuesto teniendo como meta principal realizar el número mínimo de éstas para balancearlo. Un conjunto de tales operaciones se mencionan a continuación: (a) agregar una cantidad *x* a la cantidad de componente, (b) restar o remover una cantidad *x* a la cantidad de componente y (c) reemplazar el componente por la misma cantidad de un equivalente mejor. La aplicación de estas operaciones es realizada con aritmética de números difusos trapezoidales, entre estas operaciones aritméticas están la suma, resta y la multiplicación. La Tabla 29 muestra las ecuaciones de estas operaciones.

Por otro lado, el término h es una estimación del costo para transformar un estado *m* (compuesto) a sus soluciones más cercanas, y el cual se calculará en base a la distancia antes del emparejamiento para un nutriente *n*. Por lo tanto, la solución más cercana está a mayor distancia, es decir, hay que nivelar el nutriente que esté más alejado de alcanzar su meta calórica. Por lo tanto, se analizaron algoritmos con las características descritas anteriormente con diferentes heurísticas para determinar la

eficiencia de los mismos en tiempo (rapidez en encontrar soluciones aplicando un conjunto de operaciones de balanceo) y en espacio (tamaño del espacio de búsqueda generado). La Tabla 28 describe de forma general el procedimiento de un algoritmo heurístico de balanceo.

**Tabla 28:** Procedimiento general de un algoritmo heurístico de balanceo.

**Procedimiento:** Corrección o balanceo de componentes alimenticios.
**Variables relacionadas:** Cantidad nítida o difusa de carbohidratos, lípidos y proteínas de cada componente del compuesto alimenticio, Cantidad difusa de la meta calórica para el nutriente *n*, Índice de posibilidad de emparejamiento, Índice de necesidad de emparejamiento.

**Comienzo**

1 Definir como estado inicial el compuesto alimenticio requerido por el usuario.

**Repetir**

2 Obtener las cantidades difusas totales de carbohidratos, lípidos y proteínas del compuesto alimenticio.

3 Chequear la compatibilidad de las cantidades difusas totales de nutrientes con sus respectivas metas calóricas, utilizando el procedimiento de emparejamiento de patrones difusos ponderado.

4 Colocar el compuesto *m* en "Compuestos Visitados".

5 Generar los compuestos vecinos *m's*, aplicando las posibles operaciones sobre los componentes (por ejemplo, agregar una cantidad *x* a la cantidad de componente, restar o remover una cantidad *x* a la cantidad de componente, o sustituir el componente por la misma cantidad de un equivalente mejor, aplicando operaciones de aritmética difusa), los cuales son colocados en la lista "compuestos alimenticios generados".

6 Escoger un compuesto alimenticio *m* de la lista de compuestos generados con el mejor valor de evaluación.

**Hasta que el chequeo en el paso 3 determine que el compuesto está balanceado.**
(Es muy importante destacar que los algoritmos de balanceo tienen la misma estructura, sólo se diferencian en la definición del término heurístico en la función de evaluación).
**Fin**

**Tabla 29.** Ecuaciones de las operaciones de aritmética difusa.

| | |
|---|---|
| $A = (a_1, b_1, c_1, d_1) \Rightarrow$ Coordenadas de la meta calórica para un nutriente *n*.<br>$B = (a_2, b_2, c_2, d_2) \Rightarrow$ Coordenada de la cantidad total para un nutriente *n*. | |
| ***Suma*** | $A + B = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$ |
| ***Resta*** | $A - B = (a_1 - b_2, b_1 - a_2, c_1 + d_2, d_1 + c_2)$ |
| ***Multiplicación*** | $nxA = (nx\,a_1, nx\,b_1, nx\,c_1, nx\,d_1)$, *donde n es un número real mayor que 0.* |

Para ejemplificar el proceso de balanceo utilizando el algoritmo de la tabla 28, se supondrá que se tiene un compuesto alimenticio cuya cantidad total de carbohidratos en sus componentes es 10 g. y la cantidad de carbohidratos que se deben consumir para una ingesta que cumpla con el requerimiento calórico y el porcentaje de distribución es 19 g. Para un mayor entendimiento se supondrá que la cantidad total y la cantidad meta son nítidas. Se asumirán dos operaciones para aumentar los carbohidratos y emparejarlos a la meta calórica, tales operaciones son: agregar pan y agregar pasta. La Figura 14 muestra el árbol generado por este algoritmo, donde el estado inicial son los componentes alimenticios y la cantidad total de carbohidratos en su composición. La heurística aplicada consistirá en determinar si la cantidad de carbohidratos en un estado m es menor a 19 entonces h = 19 – carbohidratos, de lo contrario ese estado será una solución no factible. La cantidad de carbohidratos en un estado m dependerá de la operación aplicada: agregar pan, la cual añade 5 carbohidratos (flecha gruesa) o agregar pasta, la cual añade 3 carbohidratos (flecha punteada). La numeración indica el orden en que se generan los estados.

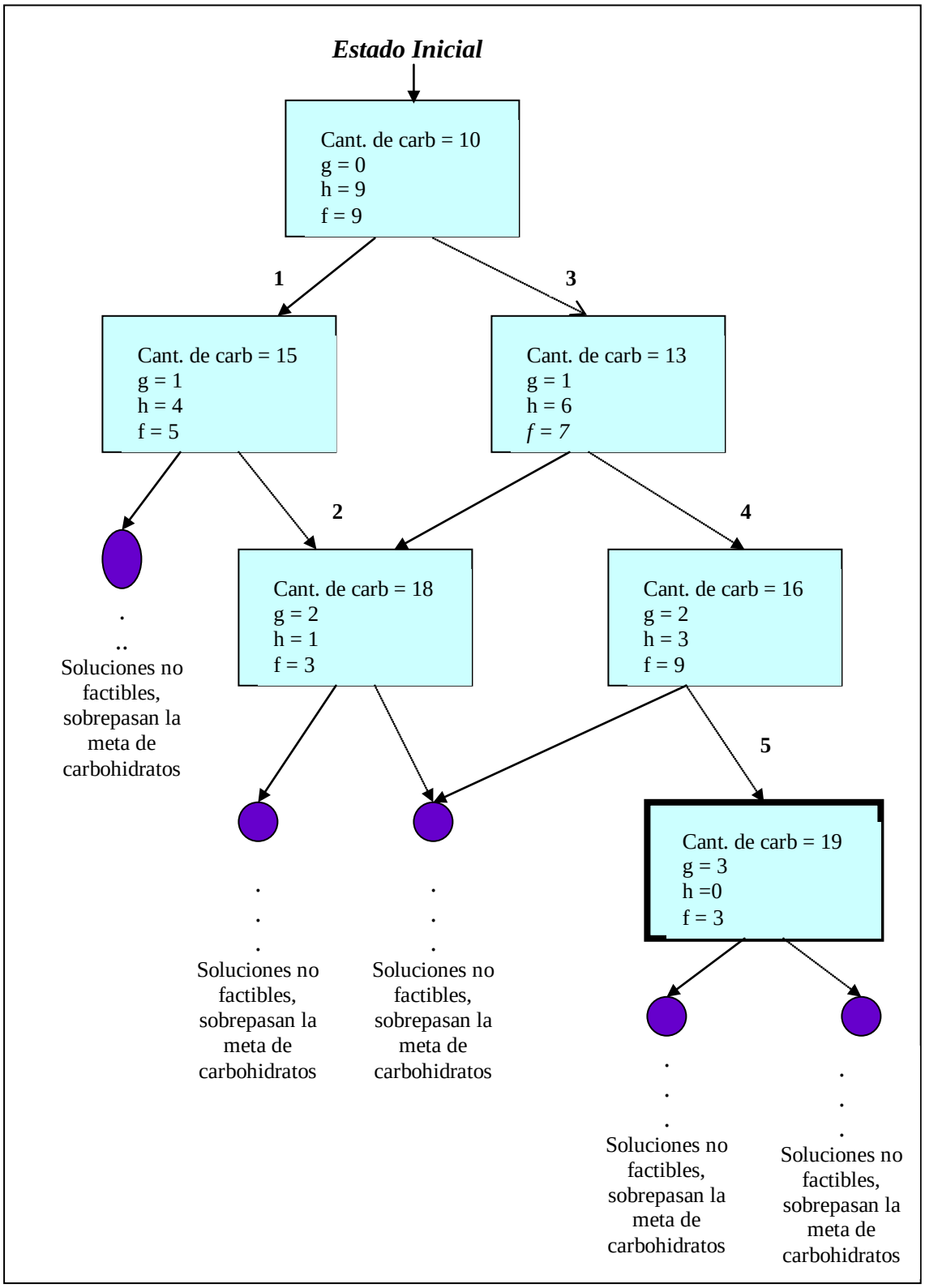



**Figura 14:** Árbol generado para las operaciones agregar pan y agregar pasta.

**Desarrollo de la herramienta implementando las técnicas de lógica difusa y búsquedas heurísticas.**

La fase de programación fue desarrollada con la herramienta visual orientada a objetos Delphi 6 de Borland (Charte, 2003), la cual permitió un manejo fácil y sencillo de las estructuras de datos definidas, de los procedimientos y de la base de datos de compuestos alimenticios. Se utilizaron conceptos de Ingeniería de Software (Pressman, 1993) tales como los diagramas de flujo de datos (DFD), los cuales permiten desarrollar al mismo tiempo los modelos del ámbito de información y del ámbito funcional. Por lo tanto, para guiar o facilitar la programación de la herramienta computacional para la combinación balanceada de nutrientes, se diseñó el DFD de nivel 0 (Anexo M), para indicar las entradas y salidas principales. Además cabe destacar que la programación de la herramienta fue basada en los procedimientos principales de las Tablas 23, 27 y 28. La definición de las estructuras de datos definidas y los procedimientos implementados se muestran en el Anexo R.

**Resultados de la evaluación de la herramienta en la fase diagnóstica.**

Luego de haber ejecutado los pasos anteriores, se evaluó la herramienta con el primer algoritmo heurístico de balanceo implementado, con la finalidad de analizar su eficiencia en el balanceo de compuestos alimenticios. Para lograr esto se utilizó un conjunto de 30 compuestos, que se obtuvieron de forma aleatoria en la interfaz de selección de sus componentes alimenticios.

La eficiencia se midió en base al número de compuestos alimenticios balanceados, al tiempo promedio empleado en cada balanceo y al tamaño promedio del espacio generado para alcanzar la solución.

Por lo tanto, la idea es comparar diferentes algoritmos heurísticos de balanceo para determinar cual es el más eficiente en cuanto a los tres parámetros nombrados anteriormente.

**Conclusiones y recomendaciones de la fase diagnóstica.**

El análisis de los resultados obtenidos en la evaluación de la herramienta con el primer algoritmo heurístico de balanceo permitió profundizar en el problema de investigación y concebir la propuesta de estudio, estableciéndose las conclusiones y recomendaciones pertinentes.

**Universo y Muestra.**

En este caso, el universo está formado por todos los tipos de componentes alimenticios presentes en la Tabla de Composición de Alimentos del INN (Instituto Nacional de Nutrición, Revisión 1999). Esta tabla contiene la información nutricional de 624 alimentos y la revisión de la misma fue coordinada por el INN en conjunto con el Instituto Nacional de Higiene "Rafael Rangel", el Instituto Venezolano de Investigaciones Científicas (IVIC), el Instituto de Medicina Experimental de la U.C.V y la Universidad Simón Bolívar.

De tal universo se tomó una muestra de 140 alimentos, la cual es representativa de las costumbres y hábitos nutricionales de la población venezolana. Los Anexos del B al L muestran la información nutricional en carbohidratos, lípidos y proteínas de los alimentos que constituyen la muestra, además, de las unidades de medida en las cuales son representados; y los valores mínimo y máximo reglamentarios de cada alimento que se deben consumir para tener una dieta bien balanceada. Estos valores fueron investigados en las Tablas de Proporcionamiento de Alimentos suministradas en la Escuela de Nutrición y Dietética de la Universidad del Zulia para la cátedra Alimentación Institucional I.

Tal muestra se llevó a una tabla en la Base de datos de manejo de compuestos alimenticios, la cual permitirá a un usuario formar un compuesto o comida, escogiendo componentes alimenticios dentro de un conjunto de categorías (carnes blancas, alimentos ricos en carbohidratos, pescados y mariscos, etc.) para satisfacer un requerimiento nutricional.

**Técnicas e Instrumentos de Recolección de Información.**

En lo que respecta a la investigación documental, esta se llevó a cabo a través de consultas en libros especializados referentes a temas de lógica difusa y búsquedas heurísticas, consultas en Internet, además de consultas a bibliografía en bibliotecas de universidades nacionales, tales como la UCLA, UNEXPO, Universidad Fermín Toro, Universidad Yacambú y la biblioteca Marcel Roche (IVIC). Para complementar la investigación documental se recibió asesoramiento de los siguientes especialistas: (a) Prof. Omaira Peña, en la realización de la metodología y otros aspectos del trabajo de grado; se desempeña actualmente como docente en la UCLA, (b) Lic. Belkis López de Lameda en tópicos relacionados con la investigación (Teoría de Conjuntos Difusos); se desempeña actualmente como docente en la UCLA, (c) Ing. Carlos Lameda (con maestría en Ciencias de la Computación mención Inteligencia Artificial) en tópicos relacionados con la investigación; se desempeña actualmente como docente e investigador en el área de lógica difusa en la UNEXPO. (d) Lic. en Nutrición y Dietética Mariana Tierno, en tópicos relacionados con la nutrición humana.

Por otro lado, en cuanto a la recolección de los datos requeridos para diseñar las entradas vagas o imprecisas, representar el conocimiento (diseño de la base de datos difusa) y evaluar la herramienta, éstos se obtuvieron por medio de libros de nutrición actualizados e información proporcionada por el INN referente a las necesidades de energía y de nutrientes para la población venezolana (Instituto Nacional de Nutrición, Revisión 2000), donde se establecen los rangos o normas para regir su consumo; además de las tablas de composición nutricional de los alimentos (Instituto Nacional de Nutrición, Revisión 1999). En cuanto al desarrollo la herramienta se utilizaron técnicas de diseño y programación actualizadas.

**Técnicas de análisis de los datos.**

La compatibilidad de las cantidades difusas totales (g) de carbohidratos, lípidos y proteínas en un compuesto alimenticio requerido por un usuario y sus metas calóricas

(cantidad recomendada en gramos de cada nutriente que se debe consumir en una comida según porcentaje de distribución) es evaluada por medio de índices de compatibilidad, que se calculan en término de medidas de posibilidad y necesidad. Estos índices permiten obtener información del grado de emparejamiento de los nutrientes que constituyen el compuesto y sus metas calóricas (meta en gramos) correspondientes; permitiendo con esto, obtener información detallada de los problemas de combinación o balanceo de nutrientes (si los hay) presentes en dicho requerimiento.

La comparación de la eficiencia de los algoritmos de balanceo implementados se midió en base al número de comidas balanceadas, al tiempo promedio de aplicación de las operaciones de balanceo y al número promedio de nodos generados.

**Resultados de la fase diagnóstica.**

En el primer algoritmo de balanceo implementado se utilizó el siguiente conjunto de operaciones sobre los componentes alimenticios:

1. Modificar su cantidad a un mínimo.
2. Modificar su cantidad a un promedio.
3. Modificar su cantidad a un máximo.
4. Eliminarlo del compuesto alimenticio.

Las cantidades que representan el mínimo, promedio y máximo son encontradas en la tabla de medidas por alimentos (Ver Tabla 16) de la base de datos de manejo de compuestos alimenticios.

Considerando que la meta principal es cambiar lo menos posible componentes alimenticios (minimizar el tiempo de búsqueda de soluciones), cada operación es pesada con el mismo costo, es decir, el cálculo de g fue hecho en base a un costo igual a 1 para todas estas operaciones (costo uniforme). La heurística utilizada en este algoritmo se describe a continuación:

$$h = \sum_{n \in nutrientesnoemparejados} w_n$$

En la ecuación anterior, $w_n$ pesa la importancia del nutriente $n$, es decir, a los tres nutrientes básicos considerados en el estudio se les asignó una importancia igual a 1 si no está emparejado con su meta calórica y 0 si está emparejado. Se aplicó este algoritmo a un conjunto de 30 compuestos alimenticios y se obtuvieron los resultados de la Tabla 30, los cuales fueron calculados con la información del Anexo N. Cabe destacar que el tiempo promedio de balanceo depende de la velocidad del procesador del equipo en que se pruebe el algoritmo, en este caso un Pentium III de 550 MHZ.

**Tabla 30.** Resultados aplicando el primer algoritmo de balanceo.

| ***Número de compuestos alimenticios balanceados.*** | ***Tiempo promedio de balanceo.*** | ***Número promedio de nodos generados.*** |
|---|---|---|
| 21 | 194 ms. | 259 |

**Conclusiones de la fase diagnóstica.**

En la aplicación del primer algoritmo de balanceo implementado se detectó que el problema principal fue el conjunto de operaciones aplicadas a los componentes alimenticios en las cuales se generó un espacio demasiado grande entre sus cantidades mínimo, promedio y máximo; es decir, utilizando este conjunto de operaciones no se evalúan otras posibilidades en las cuales se puedan conseguir mejores soluciones.

Por otra parte, el término heurístico caracteriza la extensión de desequilibrio nutricional más que el costo de la solución más cercana; en otras palabras, no estima el costo del número de operaciones que se requieren para nivelar un nutriente *n* con la finalidad de emparejarlo a su meta.

**Recomendaciones de la fase diagnóstica.**

Los resultados obtenidos permiten objetar que se deben definir operaciones sobre los componentes alimenticios las cuales evalúen sus cantidades a lo largo de sus rangos mínimos y máximos para encontrar soluciones o compuestos alimenticios que cumplan con determinados requerimientos nutricionales, es decir, que tengan un balance nutricional en sus componentes.

Se deben definir términos heurísticos que no incrementen el tamaño del espacio de estados (tamaño = número de nodos visitados x número de operaciones x número de componentes alimenticios) en la búsqueda de soluciones, en otras palabras, se deben desarrollar sólo compuestos alimenticios cuyos nutrientes emparejen más rápido sus metas calóricas. Con esto lo que se busca es incrementar la ganancia en consumo de memoria y tiempo de respuesta.

## Fase 2: Estudio de Factibilidad.

El estudio de la factibilidad es el estudio en el cual se analiza y evalúa la información técnica, económica y operativa correspondiente a la herramienta computacional propuesta, para establecer su viabilidad de llegar a desarrollarse. Esto, mediante la cuantificación de los recursos humanos, materiales y económicos que serán necesarios. Por lo tanto, se analizó la factibilidad técnica, operativa y económica del proyecto, las cuales se detallan a continuación.

**Factibilidad Técnica.**

En este punto se definen los recursos técnicos con que se cuenta y que pueden utilizarse para satisfacer los requerimientos de diseño y desarrollo de la herramienta computacional propuesta, tanto como de hardware, software y humanos. En otras palabras, en la factibilidad técnica se estudia la viabilidad de proyecto, en cuanto a su

funcionalidad, rendimiento y las restricciones que pueden afectar a la posibilidad de realización del mismo.

En este caso, se puede decir que el proyecto es factible técnicamente ya que existen los recursos humanos y la tecnología necesaria para llevar a cabo su diseño y desarrollo. Además se posee la habilidad suficiente para aplicar y utilizar dicha tecnología y así producir una herramienta computacional eficiente y confiable a un costo razonable, y la cual permita dar la flexibilidad para satisfacer necesidades futuras, tales como; acoplamiento de nuevos procedimientos, actualización, incremento en las operaciones, entre otras. Por otra parte, la tecnología seleccionada cumple con las garantía técnicas de calidad, exactitud, confiabilidad, facilidad de uso, seguridad y portabilidad.

También se cuenta con la bibliografía especializada, con herramientas de diseño y programación, así como del asesoramiento de especialistas en los tópicos de interés a la temática del proyecto.

**Factibilidad Operativa.**

El producto de esta investigación se considera factible operativamente, ya que se automatizarán procedimientos para la combinación balanceada de nutrientes, utilizando lógica difusa y búsquedas heurísticas, los cuales permitirán manejar y representar valores vagos o imprecisos, lográndose con esto, más precisión en los resultados. Convirtiéndose en un aporte para el área de nutrición, ya que la mayoría de los alimentos poseen vaguedad e imprecisión tanto en sus cantidades como en su composición nutricional, haciéndose necesario el manejo de cantidades vagas o imprecisas para calcular de una manera más adecuada el contenido nutricional de un compuesto alimenticio. Además, la interfaz de la herramienta computacional será amigable, es decir, de fácil uso y entendimiento para el usuario.

**Factibilidad Económica.**

Los recursos económicos para llevar a cabo este proyecto se encuentran dentro de los límites accesibles por parte del autor o persona encargada de llevarlo a cabo.

Por lo tanto, el proyecto se considera factible económicamente ya que la inversión económica requerida se considera aceptable, y se tienen los recursos financieros. Se realizó una estimación de gastos en base a la adquisición de la bibliografía especializada en ingeniería de software, adquisición de software y accesorios de computación, gastos de copias y fletes para recibir envío de información especializada y cursos de actualización en herramientas de programación. Estos gastos se resumen en la Tabla 31.

**Tabla 31.** Estimación de costos para la culminación del proyecto.

| ***Descripción de Costos.*** | |
|---|---|
| ***Costos de Desarrollo*** | ***Costo Estimado (Bs.)*** |
| Adquisición de bibliografía para diseño de software | 500.000,00 |
| Cursos de actualización en herramientas de programación. | 300.000,00 |
| Gastos de copias y fletes para recibir envío de información especializada para definir los requerimientos de diseño y desarrollo. | 200.000,00 |
| Adquisición de software | 100.000,00 |
| ***Costos de Hardware*** | ***Costo Estimado (Bs.)*** |
| Adquisición de hardware complementario y otros accesorios de computación. | 600.000.00 |
| ***TOTAL*** | 1.700.000,00 |

## Fase 3: Diseño de la Propuesta.

1. Se diseñó e implementó una herramienta computacional para obtener compuestos alimenticios con una combinación balanceada de nutrientes (utilizando lógica difusa y búsquedas heurísticas), los cuales estén adaptados a determinados requerimientos nutricionales; permitiendo además el manejo y representación de manera adecuada de cantidades vagas o imprecisas.
2. Se validó la herramienta en base a la evaluación de requerimientos nutricionales y del balanceo de sus componentes (de presentarse el caso), para obtener una combinación o compuesto alimenticio adaptado a esos requerimientos y que a su vez cumpla con normas de nutrición previamente establecidas, utilizando un algoritmo heurístico de balanceo eficiente.

## Fase 4: Ejecución y Evaluación de la Propuesta.

1. Se determinó la calidad de la herramienta computacional en base al cumplimiento de los objetivos para la cual fue implementada. En este caso de investigación los resultados de ejecución fueron los esperados.
2. Se establecieron las conclusiones y recomendaciones pertinentes.

# CAPÍTULO IV
# PROPUESTA DEL ESTUDIO

## Objetivo General.

La propuesta del estudio está fundamentada en diseñar e implementar una herramienta computacional para obtener compuestos alimenticios adaptados a los requerimientos nutricionales de los venezolanos, los cuales tengan una combinación balanceada de nutrientes, utilizando lógica difusa y búsquedas heurísticas.

## Objetivos Específicos.

1. Evaluar el contenido nutricional de un compuesto alimenticio requerido por un usuario utilizando la técnica de emparejamiento de patrones difusos ponderada, para determinar por medio de índices de posibilidad y necesidad, el nivel de compatibilidad entre las cantidades totales de sus nutrientes básicos (carbohidratos, lípidos y proteínas) y sus correspondientes metas calóricas.

2. Corregir el compuesto alimenticio (si los niveles de emparejamiento o compatibilidad indican desbalances nutricionales en sus componentes) utilizando el algoritmo heurístico de balanceo más eficiente.

## Descripción de la propuesta.

Luego de haber implementado la herramienta computacional cumpliendo los pasos de la fase diagnóstica, la propuesta de estudio consiste en mejorar la eficiencia del algoritmo heurístico de balanceo implementado en cuanto a las operaciones aplicadas sobre los componentes alimenticios y la definición de su término heurístico; dado que aplicando este algoritmo se detectó que el problema principal fue el conjunto de operaciones aplicadas a tales componentes, generándose un espacio demasiado grande entre sus cantidades mínimo, promedio y máximo. En otras palabras, estas operaciones aumentaron considerablemente el tamaño del espacio de estados en la búsqueda de soluciones. Además en este algoritmo se da una solución sólo si se generan 10000 estados o menos; esto con la finalidad de ahorrar memoria de CPU y tiempo de ejecución. Por otra parte, el término heurístico no estima el costo del número de operaciones que se requieren para nivelar un nutriente *n*, con la finalidad de emparejarlo con su meta calórica (g de nutrientes establecidos de acuerdo al requerimiento calórico y porcentaje de distribución). La idea entonces es implementar mejoras a este algoritmo e ir comparando las mismas para determinar cual de ellas logra balancear un compuesto alimenticio de la forma más eficiente. Tales mejoras están basadas en cambiar lo menos posible alimentos de los compuestos alimenticios (minimizar el número de operaciones de balanceo) y con esto disminuir el tamaño del espacio de búsqueda generado. Para lograr esto se procedió de la siguientes manera:

- Se definió un nuevo conjunto de operaciones sobre los componentes alimenticios las cuales se describirán a continuación: (a) Agregar una cantidad $a^f$ a la cantidad de alimento $f$ y (b) Restar o sustraer una cantidad $a^f$ a la cantidad de alimento *f*. Esta cantidad $a^f$ es la porción más pequeña de un alimento que puede ser razonablemente considerada en la práctica, por ejemplo, 1 rebanada de pan, 1 cucharadita de mermelada, etc. Por lo tanto este nuevo

conjunto de operaciones permite que se puedan aumentar o disminuir las cantidades de los alimentos incrementando o disminuyendo su porción más pequeña. En el algoritmo implementado en la fase diagnóstica, el cálculo de g fue en base a un costo igual a 1 (costo uniforme) para todas las operaciones. A partir de ahora, los costos de las operaciones sobre un alimento son: (a) La operación de aumentar la cantidad de un alimento con su cantidad incremento $a^f$ tiene un costo igual a 1, (b) Cuando la disminución de la cantidad del alimento causa eliminación el costo es igual a 3. (Esto para inducir una búsqueda la cual tienda a evitar estas eliminaciones) y (c) si la disminución del alimento no causa eliminación el costo es igual a 1.

Por lo tanto, el nuevo valor de *g* el cual denota el costo mínimo de transformación desde un compuesto alimenticio inicial a un estado *m* del mismo está expresado por la siguiente ecuación:

$$g^m = \sum_{f \varepsilon "alimentos"} \frac{\left| q_f^m - q_f^i \right|}{a^f} + \sum_{f \varepsilon "alimentos\ sin\ iciales" q_f^m = 0} Eliminar Costo$$

En esta ecuación, $q_f^m$ y $q_f^i$ son las cantidades de un alimento *f* en el compuesto alimenticio *m* (o estado actual) y el compuesto alimenticio inicial respectivamente; donde la cantidad $a^f$ , es el incremento de la cantidad del alimento *f*.

- Se implementó un segundo algoritmo heurístico de balanceo igualmente con la estructura de un algoritmo A (Rubio y otros, 1998) pero con el valor mejorado de *g* y su término heurístico basado en la necesidad de emparejamiento de un nutriente *n* con su meta calórica. Se analizará su comportamiento en la búsqueda de soluciones y los resultados obtenidos balanceando un conjunto de 30 compuestos alimenticios son mostrados en la Tabla 32.
- Se implementó un tercer algoritmo heurístico de balanceo con la estructura de un algoritmo A, con el valor mejorado de *g* y su término heurístico basado en la

distancia antes del emparejamiento de un nutriente *n* con su meta calórica. Se analizó su comportamiento en la búsqueda de soluciones y los resultados obtenidos balanceando un conjunto de 30 compuestos alimenticios son mostrados en la Tabla 33.

- Por último, se implementó un cuarto algoritmo heurístico de balanceo con la estructura de un Algoritmo A, el valor mejorado de *g* y su término heurístico basado en la distancia antes del emparejamiento mejorada del nutriente *n* con su meta calórica, por medio de la cual se induce una búsqueda donde el orden en que son aplicadas las operaciones sobre los alimentos es de poca importancia. Se analizó su comportamiento en la búsqueda de soluciones y los resultados obtenidos balanceando un conjunto de 30 compuestos alimenticios son mostrados en la Tabla 34.
- Luego se compararon los cuatro algoritmos implementados en cuanto al número de balanceos, tiempo promedio de balanceo y número promedio de nodos generados en balanceos exitosos, basándose en los resultados de las Tablas 32, 33 y 34, con la finalidad de determinar cual de ellos es el más eficiente.

# CAPÍTULO V
# EJECUCIÓN DE LA PROPUESTA

Se procedió a desarrollar los pasos establecidos en el Capítulo IV para ejecutar la propuesta, y con ello implementar diferentes algoritmos heurísticos de balanceo, para determinar cual de ellos es el más eficiente en la obtención de compuestos alimenticios con una combinación balanceada de nutrientes.

**Algoritmo de balanceo basado en la necesidad de emparejamiento.**

Este es una versión del algoritmo implementado en la fase diagnóstica, pero utilizando el valor mejorado de *g*. Bien es sabido que el término heurístico *h* es una estimación del costo para transformar un estado *m* a sus soluciones más cercanas, por lo tanto, el término heurístico utilizado está definido por la siguiente ecuación:

$$h = \frac{\sum_{n\varepsilon"nutrientes"} w_n.(1 - N_n)}{2}$$

donde $N_n$ es la necesidad de emparejamiento del nutriente *n* con su meta calórica (cantidad máxima en gramos que se debe consumir por cada nutriente según el requerimiento calórico y porcentaje de distribución) y $W_n$ es un valor que representa la importancia asignada a cada nutriente. En este caso se le asignó una importancia igual a 1 a todos los nutrientes considerados en el balance (carbohidratos, lípidos y proteínas).

El nuevo conjunto de operaciones definidas sobre los alimentos al igual que las definidas en la fase diagnóstica incrementan el tamaño del espacio de estados ya que: Espacio de estados generado = Número de estados visitados x Número de operaciones x Número de alimentos.

En el término heurístico del algoritmo implementado, el término 1- $N_n$ es casi siempre igual a 1 y disminuye solamente cuando el emparejamiento es casi perfecto. Se puede decir que un emparejamiento es casi perfecto, cuando la necesidad $N_n$ del nutriente *n* con respecto a su meta calórica está en el intervalo [0.9, 1], en otras palabras, cuando las coordenadas del número difuso que representa la cantidad total de un nutriente *n* están casi incluidas en el núcleo del número difuso que representa la cantidad meta para ese nutriente, en caso contrario el valor de la necesidad $N_n$ está por debajo de 0.9. Por lo tanto, en los casos cuando un compuesto alimenticio está muy desbalanceado y tiene al menos 5 alimentos se generan espacios de búsqueda muy grandes. Se aplicó este algoritmo a un conjunto de 30 compuestos alimenticios y se obtuvieron los resultados mostrados en la Tabla 32, los cuales fueron calculados con la información del Anexo O.

**Tabla 32.** Resultados aplicando el segundo algoritmo de balanceo.

| ***Número de compuestos alimenticios balanceados.*** | ***Tiempo promedio de balanceo.*** | ***Número promedio de nodos generados.*** |
|---|---|---|
| 24 | 3 seg. 86 ms. | 1360 |

**Algoritmo de balanceo basado en la distancia antes del emparejamiento.**

Este algoritmo está basado en la distancia antes del emparejamiento la cual puede definirse como la variación del nutriente *n* en un compuesto alimenticio *m* y cuya ecuación es:

$$\Delta_n(m, meta) = \frac{\phi_n^{meta} - \phi_n^{m}}{\alpha_n^{\max}}$$

donde $\phi_n^{meta}$ es la cantidad meta del nutriente *n*, $\phi_n^m$ la cantidad del nutriente *n* en el estado *m* del compuesto alimenticio y $\alpha_n^{\max}$ es el incremento máximo para el nutriente *n* en una operación, el cual se define por:

$$\alpha_n^{\max} = \max_{f \varepsilon "componentes"} (a^f \varphi_n^f)$$

con $\varphi_n^f$, la cual es la cantidad del nutriente *n* en un componente alimenticio *f* (1 g de) y $a^f$, es la cantidad incremento del componente alimenticio *f* (en gramos). Por lo tanto, $a^f \varphi_n^f$ es la cantidad de nutriente *n* que se agrega o elimina en una operación.

Por otra parte, $\Delta_n(m, meta)$ mide el costo mínimo en número de operaciones que se requieren para nivelar o emparejar el nutriente *n* con su meta calórica; en otras palabras, es el número de operaciones que se deben aplicar a un componente alimenticio para nivelar el nutriente *n*. De lo anterior se deduce que $\Delta_n(m, meta)$ es negativa si el nutriente está por encima de su meta calórica y es positiva si está por debajo. Entonces, para balancear un compuesto alimenticio *m* cada nutriente *n* tiene que estar nivelado, particularmente el que tenga la distancia máxima antes del emparejamiento (nutriente que tiene mayor desbalance por debajo o por encima de su meta). Por lo tanto, la solución más cercana está a mayor distancia, y se denota por:

$$\max_{n \varepsilon "nutrientes"} |\Delta_n(m, meta)|$$

De hecho, el número mínimo de operaciones "incremento" para emparejar el nutriente *n* más desbalanceado por debajo de su meta $(\Delta_n > 0)$ está dado por la siguiente ecuación:

$$h_{incremento}^m = \max(0, \max_{n \varepsilon "nutrientes"} (\Delta_n(m, meta)))$$

Al contrario de lo anterior, el número mínimo de operaciones "disminución" para emparejar el nutriente *n* más desbalanceado por encima de su meta $(\Delta_n < 0)$ está dado por la siguiente ecuación:

$$h^{m}_{dis\,\min\,ución} = -\min(0, \min_{n\varepsilon"nutrientes"}(\Delta_n(m, meta)))$$

En conclusión, el término heurístico utilizado en la implementación de este algoritmo, el cual denota la solución más lejos o que tiene mayor distancia, está representado por la siguiente ecuación:

$$h^{m} = h^{m}_{incremento} + h^{m}_{dis\,\min\,ución}$$

La aplicación de este algoritmo no ejecuta una evaluación optimista (no se aproxima a un algoritmo A*) de la distancia antes que emparejamiento debido a que el costo de la solución más cercana se estima en base al nutriente más desbalanceado por encima y por debajo de la meta calórica respectivamente, por lo tanto, al intentar balancear estos nutrientes con la mayor distancia antes del emparejamiento se producen desbalances en los otros componentes involucrados en el proceso; incrementándose el tamaño del espacio de búsqueda y el número de compuestos alimenticios inútilmente visitados. Se aplicó este algoritmo a un conjunto de 30 compuestos alimenticios y se obtuvieron los resultados mostrados en la Tabla 33, los cuales fueron calculados con la información del Anexo P.

**Tabla 33.** Resultados aplicando el tercer algoritmo de balanceo.

| ***Número de compuestos alimenticios balanceados.*** | ***Tiempo promedio de balanceo.*** | ***Número promedio de nodos generados.*** |
|---|---|---|
| 26 | 3 seg. 32 ms. | 1107 |

**Algoritmo de balanceo basado en la distancia antes del emparejamiento mejorada.**

Este algoritmo heurístico se diferencia del algoritmo anterior, debido a que el orden en que son aplicadas las operaciones (incremento/disminución) en los

componentes alimenticios es de poca importancia. Como es sabido, los términos $h^m_{incremento}$ y $h^m_{dis\,\min ución}$ indican el número mínimo de operaciones incremento/disminución a ser hechas independientemente. Entonces cuando se generan estados de un compuesto alimenticio, se definió un criterio para generar sólo aquellos compuestos los cuales permitan emparejar más rápido el nutriente *n* con la mayor distancia antes del emparejamiento. Por lo tanto, durante el cálculo de $h^m$ se definió un valor para indicar la operación (incremento/ disminución) a ser aplicada a los nutrientes, de la siguiente forma:

- Si $h^m_{incremento} > h^m_{dis\,\min ución}$ entonces aplicar operación incremento.
- Si $h^m_{incremento} < h^m_{dis\,\min ución}$ entonces aplicar operación disminución.
- Sin $h^m_{incremento} = h^m_{dis\,\min ución}$ entones aplicar ambas operaciones.

La aplicación de este algoritmo sí ejecuta una evaluación optimista (se aproxima a un algoritmo A*) de la distancia antes del emparejamiento debido a que hay menos comidas inútilmente visitadas, reduciéndose considerablemente el espacio generado de búsqueda (ganancia en consumo de memoria), lo cual conduce a una mayor eficiencia y rapidez en tiempo de ejecución. Se aplicó este algoritmo a un conjunto de 30 compuestos alimenticios y se obtuvieron los resultados mostrados en la Tabla 34, los cuales fueron calculados con la información del Anexo Q.

**Tabla 34.** Resultados aplicando el cuarto algoritmo de balanceo.

| ***Número de compuestos alimenticios balanceados.*** | ***Tiempo promedio de balanceo.*** | ***Número promedio de nodos generados.*** |
|---|---|---|
| 26 | 130 ms. | 228 |

**Estudio comparativo de los algoritmos heurísticos de balanceo implementados.**

1. El conjunto de operaciones aplicadas a los componentes alimenticios en el primer algoritmo heurístico de balanceo generó un espacio demasiado grande entre sus cantidades mínimo, promedio y máximo; es decir, no permiten evaluar otras posibilidades en las cuales se puedan conseguir mejores soluciones. Por otro lado se consiguen soluciones de forma rápida cuando los incrementos/disminución de alimentos caen en sus cantidades mínimo, promedio y máximo, y exista un emparejamiento entre éstas y sus metas calóricas; proponiendo una solución distinta a la planteada inicialmente por el usuario.

2. En el término heurístico del segundo algoritmo implementado, el término 1-$N_n$ es casi siempre igual a 1 y disminuye solamente cuando el emparejamiento es casi perfecto. Por lo tanto, en los casos cuando la cantidad total de un nutriente *n* en un compuesto alimenticio está muy desbalanceada por encima o por debajo de su meta, y el compuesto alimenticio tenga al menos 5 alimentos, se generan espacios de búsqueda muy grandes no consiguiendo en muchos casos la solución antes de que 10000 estados sean generados.

3. En el tercer algoritmo implementado el costo de la solución más cercana se estima en base a la cantidad de nutriente más desbalanceada por encima y por debajo de la meta calórica respectivamente. En este caso las operaciones de incremento/disminución son aplicadas a cada componente alimenticio, incrementándose el número de compuestos alimenticios inútilmente visitados, lo que origina un incremento considerable en el tamaño del espacio de búsqueda, no consiguiendo en muchos casos la solución antes de que 10000 estados sean generados.

4. El cuarto algoritmo implementado basado en la distancia antes del emparejamiento mejorada, además de reducir el número de operaciones incremento/disminución aplicadas a los componentes alimenticios, permite proponer de forma eficaz soluciones distintas a la planteada inicialmente por el usuario, cuando genera una comida balanceada.

# CAPÍTULO VI
# CONCLUSIONES Y RECOMENDACIONES

Se realizó una investigación para el diseño e implementación de una herramienta computacional para obtener compuestos alimenticios con una combinación balanceada de nutrientes utilizando lógica difusa y búsquedas heurísticas. Para ello se analizaron los requerimientos nutricionales de los venezolanos los cuales permitieron definir los criterios para la representación de los valores de entrada, del diseño de la base de datos de compuestos alimenticios y de la interacción herramienta-usuario. Se seleccionó la técnica de emparejamiento de patrones difusos a implementar y se analizaron algoritmos de búsqueda heurística para el balanceo de compuestos alimenticios. Se desarrolló la herramienta implementando estas técnicas de lógica difusa y búsquedas heurísticas; la cual se evaluó balanceando un conjunto de compuestos alimenticios, lo cual permitió analizar el comportamiento de los algoritmos heurísticos de balanceo implementados, comparando su eficiencia en la obtención de compuestos alimenticios con una combinación balanceada de nutrientes con los cuales se pretendió satisfacer ciertos requerimientos nutricionales.

De los resultados obtenidos en la ejecución de la propuesta, se concluye que los algoritmos de búsqueda heurística tipo A permiten resolver el problema de combinación o mezcla de alimentos, donde la definición de las operaciones aplicadas a tales alimentos y la definición del término heurístico para evaluar la distancia antes del emparejamiento fueron el punto central. No está demás acotar que pretender resolver este tipo de problema con métodos ciegos de búsqueda (búsquedas no informadas) tales como la búsqueda primero en anchura sería impráctico debido al crecimiento combinatorio del espacio de búsqueda.

El algoritmo heurístico de balanceo basado en la distancia antes del emparejamiento mejorada, además de reducir el número de operaciones incremento/disminución aplicadas a los componentes alimenticios (redujo considerablemente el tamaño del espacio de búsqueda), permitió proponer de forma eficaz soluciones distintas a la planteada inicialmente por el usuario, cuando se generaron compuestos alimenticios con una combinación balanceada de nutrientes; por lo tanto, este algoritmo realizó la mejor ejecución del procedimiento de balanceo.

Esta herramienta computacional es un aporte al área de nutrición y puede ser adaptada para:

- Manejar un número mayor de nutrientes (Vitaminas, minerales, etc.).
- Incluir enfermedades de usuarios que restrinjan el consumo de alimentos y sugieran dietas especiales.
- Abordar una mayor diversidad de usuarios (niños, embarazadas, deportistas), los cuales requieran mayores consumos calóricos que el normal.
- Crear una interfaz lingüística con el usuario, que por ejemplo emita frases tales como; “Tu comida es demasiado rica en grasas saturadas”, entre otras.

# REFERENCIAS BIBLIOGRÁFICAS


Botella Francisco 2001. Recopilación de Software Médicos. URL: http://medfam.org/softbio/softbio_.asp#nutricion (Consulta: Septiembre 15, 2004)

Buisson Jean-Christophe 1999. Approximate reasoning in computer-aided medical decision systems. The Handbooks of Fuzzy Sets. Norwell, MA: Kluwer, Vol. 7, ch. 11, pp. 337-361.

Buisson Jean-Christophe 2002. Nutri-Expert et Nutri-Advice, deux logiciels d'aide á la construction de repas equilibrés pour l'éducation nutritionnelle. Revues Information, Interaction, Intelligence, Vol. 1 (2), pp. 7-37.

Buisson Jean-Christophe and Garel Alexandre 2003. Balancing Meals Using Fuzzy Arithmetic and Heuristic Search Algorithms. IEEE Transactions On Fuzzy Systems. Vol. 11, Nº 1, pp. 68-78, February.

Charte Ojeda, Francisco 2003. Programación Delphi 7 y Kylix 3. Ediciones Anaya multimedia. Grupo Anaya, S.A.

Cayrol M., Farreny H. and Prade H. 1982. Fuzzy Pattern Matching. Kybernetes. Vol. 11, pp. 103-116.

Dubois, Didier and Prade Henry 1980. Fuzzy Sets and Systems. North-Holland Publishing Company. Vol. 3, pp. 37-48.

a. Dubois, Didier and Prade Henry 1985. A Review of Fuzzy Set Aggregation Connectives. Inform. Sci. Vol. 36 (1-2), pp. 85-121.

b. Dubois, Didier and Prade Henry 1985. Evidence Measures Based on Fuzzy Information. Automatic. Vol. 21, Nº 5, pp. 547-562.

Dubois D., Prade H. and Testemale C. 1988. Weighted Fuzzy Pattern Matching. Fuzzy Sets and Systems. North-Holland. Vol. 28, pp. 313-331.

Farreny H., Prade H. and Buisson C. 1985. The Development of a Medical Expert System and the Treatment of Imprecision in the framework of Possibility Theory. Information Sciences. Vol. 37, pp. 211-226.

Instituto Nacional de Nutrición, Revisión 1999. Tabla de composición de alimentos para uso práctico. Publicación nro. 52, Serie Cuadernos Azules. Ministerio de Sanidad y Asistencia Social. Caracas-Venezuela.

Instituto Nacional de Nutrición, Revisión 2000. Valores de referencia de energía y de nutrientes para la población venezolana. Publicación nro. 53, Serie Cuadernos Azules. Ministerio de Sanidad y Asistencia Social. Caracas-Venezuela.

Millán G., María del Pilar 1998. Menúes para todos los días. Productos Roche S.A. Edificio Roche, Avenida Diego Cisneros, Los Ruices, Caracas-Venezuela.

Nilsson J. Nils 2001. Inteligencia Artificial: Una Nueva Síntesis. McGraw-Hill/Interamericana de España, S.A.

Nilsson N., Hart P. and Raphael B. 1968. A formal Basis for the Heuristic Determination of Minimum Cost Path. IEEE Transactions of Systems Science and Cibernetics. Vol. SSC-4. Nº 2, pp. 100-107.

Pressman, Roger S. 1993. Ingeniería de Software: Un Enfoque Práctico. Tercera Edición. McGraw-Hill/Interamericana de España, S.A.

Rubio J., Muro P., y Bañares J. 1998. Búsqueda. Paper de apoyo para la cátedra de Inteligencia Artificial e Ingeniería del Conocimiento I. Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza, España. URL:http://www. lsi.upc.es/~bejar/iia/iabusq.pdf. (Consulta: Septiembre 08, 2004)

Taylor, Geoffrey 1981. Principios de la Nutrición Humana. Ediciones Omega, S.A. Barcelona, España.

Tsoukalas, Lefteri and Uhrig Robert 1997. Fuzzy and Neural Approaches in Engineering. A wiley-Interscience Publication. Cap. 4, pp. 77-102.

Turnin R., Beddok R., Clottes J., Abadie R., Martini P., Buisson C., Dupuy C., Bayard F. and Tauber P. 1992. Telematic Expert System Diabeto, a new tool for diet self monitoring for diabetic patients. Diabetes Care. Vol. 15, Nº 2.

Villazón Alberto y Arenas Gustavo 1993. Nutrición Enteral y Parenteral. Interamericana McGraw-Hill. Capítulo 2. pp. 8-13.

Yen John and Langari Reza 1999. Intelligence Control and Information. Prentice Hall. Preface by Lofti A. Zadeh.

Zadeh, Lotfi 1965. Fuzzy Sets. Information and Control. Vol. 8, pp. 338-353.
Zadeh, Lotfi 1978. Fuzzy Sets As A Basis For A Theory Of Possibility. North-Holland Publishing Company. Vol. 1, Nº 1 , pp. 3-28.

Zimmermann, H. J. 1996. Fuzzy Set Theory and Its Applications. Kluwer Academic Publishers. Cap. 5, 8.

# *Anexos*

# ANEXO A

## RESUMEN CURRICULAR DEL AUTOR

### *DATOS PERSONALES*

**Apellidos y Nombres:** Colón S. Luisa.

**Cédula de Identidad :** V-9903190

### *ESTUDIOS REALIZADOS*

**Superior**

- Universidad Centro Occidental "Lisandro Alvarado". Barquisimeto, Estado Lara.

***Título Obtenido: Ingeniero en Informática.***

### *Experiencia Docente*

- Docente de las Cátedras de Estructuras de Datos II, Laboratorio de Programación I y Programación I, dictadas en la carrera de Ingeniería en computación de la Universidad Fermín Toro. Barquisimeto, Edo. Lara.

- Docente de las Cátedras de Lógica Computacional y Computación, dictadas en la carrera de Ingeniería Electrónica en la Universidad Yacambú. Barquisimeto, Edo. Lara.

- Docente de la cátedra de Informática Básica en la Universidad Pedagógica Experimental Libertador desde Noviembre de 2001.

### *Estudios de Postgrado*

-Maestría en Ingeniería Industrial, con escolaridad culminada. Universidad Nacional Experimental Politécnica "Antonio José de Sucre". Barquisimeto, Edo. Lara. Inicio Septiembre de 1998. Culminación escolaridad julio 2001.

- Maestría en Ciencias de la Computación mención Inteligencia Artificial, con escolaridad culminada. Universidad Centroccidental "Lisandro Alvarado". Barquisimeto, Edo. Lara. Inicio Febrero 2001 culminación Febrero 2003.

### *Experiencia Laboral.*

- Actualmente me desempeño en el área de asesoría de sistemas informáticos.

# ANEXO B

## TABLA DE INFORMACIÓN NUTRICIONAL PARA HUEVOS, LÁCTEOS Y DERIVADOS

| ***Huevos, Lácteos y Derivados (En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g/cc)*** |
| Huevo Completo | 12.5 | 11.1 | 1.5 | 50 g (1 Ud), 75 g (1 y ½ Ud) y 100 g (2 Ud) |
| Leche líquida descremada | 3.6 | 0.9 | 5.0 | 50 cc (1/4 Taza) **,** 100 cc (1/2 Taza), 200 cc (1 Taza) |
| Leche líquida completa | 3.5 | 3.4 | 4.7 | 50 cc (1/4 Taza), 100 cc (1/2 Taza), 200 cc (1 Taza) |
| Queso duro de leche completa | 24.9 | 31.5 | 1.4 | 30 g (Trozo Peq.), 40 g (Trozo Med.), 60 g (Trozo Grande) |
| Queso duro de leche descremada | 38.2 | 5.7 | 4.9 | 30 g (Trozo Peq.), 40 g (Trozo Med.), 60 g (Trozo Grande) |
| Requesón | 13.4 | 10.8 | 2.6 | 25 g (Trozo Peq.), 35 g (Trozo Med.), 50 g (Trozo Grande) |
| Queso amarillo | 26.0 | 32.1 | 1.3 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40 g (2 Lonjas) |
| Queso blanco suave | 18.3 | 26.3 | 0.8 | 30 g (Trozo Peq.), 40 g (Trozo Med.), 60 g (Trozo Grande) |
| Yogurt de leche completa y frutas | 3.6 | 3.2 | 17.7 | 50 cc (1/4 Taza), 100 cc (1/2 Taza), 200 cc (1 Taza) |
| Yogurt de leche descremada | 4.6 | 0.6 | 6.9 | 50 cc (1/4 Taza), 100 cc (1/2 Taza), 200 cc (1 Taza) |
| Yogurt Light (firme) | 4.3 | 0.7 | 7.8 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO C

## TABLA DE INFORMACIÓN NUTRICIONAL PARA CARNES ROJAS Y VÍSCERAS

| ***Carnes Rojas y Vísceras (En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Carne Molida | 18,2 | 14,2 | 0 | 30 g (1/4 Taza), 50 g (1/2 Taza), 80 g (1 Taza) |
| Carne de Esmechar | 20,6 | 0,9 | 0 | 40 g (1/4 Taza), 60 g (1/2 Taza), 100 g (1 Taza) |
| Lagarto (sopa) | 20,5 | 0,2 | 0 | 40 g (Trozo Peq.), 60 g (Trozo Med.), 100 g (Trozo Grande) |
| Carne de Primera para bistec | 18,7 | 1,8 | 0 | 120 g(Ud. Peq.), 150 g(Ud. Med.), 180 g (Ud. Grande) |
| Carne de segunda para bistec | 21,7 | 2,7 | 0 | 120 g(Ud. Peq.), 150 g(Ud. Med.), 180 g (Ud. Grande) |
| Carne de cochino (Chuleta) | 18,3 | 19,6 | 0 | 100 g(Ud. Peq.), 120 g(Ud. Med.), 150 g (Ud. Grande) |
| Corazón | 16,8 | 4,9 | 0,5 | 80 g (Ud. Peq.), 120 g (Ud. Med.), 160g (Ud. Grande) |
| Hígado | 19,3 | 3,8 | 4,1 | 100 g(Ud. Peq.), 150 g(Ud. Med.), 200g (Ud. Grande) |
| Lengua | 18,4 | 12,3 | 2,1 | 40 g (1 Lonja), 80 g (2 Lonjas) , 120 g (3 Lonjas) |
| Panza | 15,0 | 3,8 | 2,4 | 40 g(1/4 Taza ), 80 g (1/2 Taza ), 120 g (1 Taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO D

## TABLA DE INFORMACIÓN NUTRICIONAL PARA CARNES BLANCAS

| ***Carnes Blancas (En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Gallina (muslo) | 18,5 | 18,4 | 0 | 100g (Ud. Peq.), 120 g(Ud. Med.), 150g (Ud. Grande) |
| Gallina (pechuga) | 18,5 | 18,4 | 0 | 80 g (1/4 Ud.), 160 g (1/2 Ud.) |
| Pavo (muslo) | 20 | 20,1 | 0 | 100g (Ud. Peq.), 120 g(Ud. Med.), 150g (Ud. Grande) |
| Pollo (muslo) | 18,2 | 3,8 | 0 | 80g (Ud. Peq.), 100 g(Ud. Med.), 120g (Ud. Grande) |
| Pollo (pechuga) | 23,1 | 1,2 | 0 | 60 g (1/4 Ud.), 120g(1/2 Ud.), 240 g (Ud. Grande) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO E

## TABLA DE INFORMACIÓN NUTRICIONAL PARA ALIMENTOS GRASOS Y EMBUTIDOS

| *Alimentos Grasos y Embutidos (En 100 g de parte comestible).* | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Aceite | 0 | 100 | 0 | 1 Cucharadita (5 g)<br>1 Cucharada (10 g) |
| Margarina | 0,6 | 81 | 0,4 | 1 Cucharadita (5 g)<br>1 Cucharada (10 g) |
| Mayonesa | 1,1 | 80 | 62 | 1 Cucharadita (5 g)<br>1 Cucharada (10 g) |
| Mantequilla | 0,6 | 81 | 0 | 1 Cucharadita (5 g)<br>1 Cucharada (10 g) |
| Crema de leche | 2 | 40,3 | 2,9 | 1 Cucharadita (5 g)<br>1 Cucharada (10 g) |
| Suero | 3,1 | 3,3 | 5,3 | 1 Cucharada (10 g)<br>2 Cucharadas (20 g) |
| Jamón de pavo | 15,2 | 0,7 | 2,2 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40 g (2 Lonjas) |
| Jamón de pollo | 14 | 0,8 | 2,0 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40 g (2 Lonjas) |
| Jamón de pierna | 13,2 | 3,9 | 1,6 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40(2 Lonjas) |
| Jamón de espalda | 13,1 | 6,1 | 2,1 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40 (2 Lonjas) |
| Salchicha de cerdo | 10,8 | 44,8 | 0,4 | 20 g (1 Ud.), 40 g (2 Ud.) |
| Tocineta | 9,1 | 65 | 1,6 | 10 g (1 Ud.), 20 g (2 Ud.) |
| Mortadela | 13 | 27,2 | 6,7 | 20 g (1 Lonja), 30 g (1 y ½ Lonja), 40 g (2 Lonjas) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO F

## TABLA DE INFORMACIÓN NUTRICIONAL PARA HARINAS Y CEREALES

| ***Harinas y Cereales (En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Pan de Sándwich | 9,4 | 3,0 | 49,6 | 1 Rebanada (25 g), 2 Rebanadas (50 g),4 Rebanadas (100 g) |
| Pan francés | 11,4 | 2,3 | 57,1 | 50 g(1Ud.), 75 g(1 y ½) , 100g(2 Ud.) |
| Pan integral | 9,0 | 2,6 | 45,4 | 1 Rebanada (25 g), 2 Rebanadas (50 g),4 Rebanadas (100 g) |
| Pastas enriquecidas | 15,0 | 1,9 | 69,3 | 60 g (1/2 Taza), 120 g (1 Taza), 180 g (1 y 1/2 Taza) |
| Arepa de harina de maíz | 3,8 | 0,6 | 36,9 | 60 g (Ud. Peq), 80 g (Ud. Med), 100 g (Ud. Grande) |
| Arroz blanco | 2,2 | 0,1 | 24,0 | 30 g (1/4 Taza), 60 g (1/2 Taza), 120 g (1 Taza) |
| Arroz integral | 2,4 | 0,3 | 20,8 | 30 g (1/4 Taza), 60 g (1/2 Taza), 120 g (1 Taza) |
| Avena en hojuelas | 9,6 | 7,3 | 61,0 | 1 Cucharada (10 g), 2 Cucharadas (20 g), 3 Cucharadas (30 g) |
| Cereal en hojuelas (Corn Flakes) | 6,7 | 0,4 | 60 | 15 g (1/4 Taza), 30 g ( 1/2 Taza), 60 g (1 Taza) |
| Cereal en hojuelas c/azúcar | 3,7 | 0,1 | 86,7 | 15 g (1/4 Taza), 30 g ( 1/2 Taza), 60 g (1 Taza) |
| Cereal con chocolate (tipo Musli) | 6,7 | 4,4 | 71,1 | 15 g (1/4 Taza), 30 g ( 1/2 Taza), 60 g (1 Taza) |
| Galletas de soda | 8,4 | 10 | 72 | 1 Paquete (30 g)<br>2 Paquetes (60 g) |
| Galletas dulces (tipo María) | 7,7 | 9,2 | 77,1 | 1 Paquete (30 g)<br>2 Paquetes (60 g) |
| Galletas (tipo Honey bran) | 8,0 | 13,5 | 66,8 | 1 Paquete (30 g)<br>2 Paquetes (60 g) |
| Galletas saladas( tipo Kraker bran) | 9,7 | 12,6 | 65 | 1 Paquete (30 g)<br>2 Paquetes (60 g) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO G

## TABLA DE INFORMACIÓN NUTRICIONAL PARA PESCADOS Y MARISCOS

| ***Pescados y Mariscos (En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Carite | 21,1 | 0,6 | 0 | 100 g (Ud. Peq.), 130 g (Ud. Med.), 180 g (Ud. Grande) |
| Sardina | 20,6 | 7,0 | 0 | 1 unidad (40 g)<br>2 unidades (80 g)<br>3 unidades (120 g) |
| Merluza | 20,5 | 1,0 | 0 | 100g (Ud. Peq.), 130 g (Ud. Med.), 180 g (Ud. Grande) |
| Atún (esmechado) | 26,1 | 0,7 | 0 | 25 g (1/4 Taza), 50 g(1/2 Taza), 100 g (1 Taza) |
| Calamares (rodajas) | 16,8 | 1,4 | 0 | 25 g (1/4 Taza), 50 g(1/2 Taza), 100 g (1 Taza) |
| Camarones | 19,7 | 1,3 | 0 | 25 g (1/4 Taza), 50 g(1/2 Taza), 100 g (1 Taza) |
| Pulpo picado | 14,2 | 0,6 | 0,4 | 25 g (1/4 Taza), 50 g(1/2 Taza), 100 g (1 Taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO H

## TABLA DE INFORMACIÓN NUTRICIONAL PARA VEGETALES Y HORTALIZAS

| **Vegetales y Hortalizas** ***( En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Acelga | 2 | 0,2 | 2,2 | 30 g (1/4 Taza), 45 g (1/2), 60 g (1) T |
| Aguacate | 1,3 | 14 | 5,2 | 30 g (Trozo Peq.), 40 g (Trozo Med.), 60 g (Trozo grande) |
| Auyama | 1,5 | 0,4 | 8,4 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Berenjena | 1 | 0,3 | 2,8 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Brócoli | 3,8 | 0,4 | 2,1 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Calabacín | 1,2 | 0,2 | 1,1 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Cebolla | 1,4 | 0,2 | 9,0 | 30 g (1/4 Taza), 50 g (1/2), 70 g (1) |
| Coliflor | 2,5 | 0,2 | 1,8 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Lechuga | 1,2 | 0,2 | 0,9 | 30 g(1/2 Taza), 50 g (1 Taza ), 70 g (1 y ½ Taza) |
| Pepino | 0,7 | 0,1 | 2,4 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Pimentón | 4,0 | 0,8 | 3,9 | 20 g (1/4 Taza), 40 g (1/2), 80 g (1) |
| Remolacha | 1,4 | 0,2 | 6,9 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Repollo | 1,4 | 0,2 | 4,3 | 30 g (1/2 Taza), 60 g (1 Taza), 90 g (1 y ½ Taza) |
| Tomate | 1,4 | 0,2 | 1,9 | 40 g (1/4 Taza), 80 g (1/2), 120 g (1) |
| Vainitas | 2,4 | 0,2 | 4,8 | 30 g(1/4 Taza), 60 g (1/2 Taza), 90 g (1 Taza) |
| Zanahoria | 1,0 | 0,2 | 6,5 | 40 g (1/4 Taza), 80 g (1/2 Taza), 120 g (1 Taza) |
| Apio | 0,6 | 0,6 | 20,3 | 50 g (1/4 Taza), 100 g (1/2 Taza), 150 g (1 Taza) |
| Ñame | 1,7 | 0,1 | 19,9 | 50 g(1/4 Taza ), 100 g (1/2 Taza), 150 g (1 Taza) |
| Papa | 2,0 | 0,1 | 14,8 | 80 g (Ud. Peq.), 100 g (Ud. Med.), 120 g (Ud. Grande) |
| Yuca | 0,8 | 0,2 | 30,9 | 80 g (Trozo Peq.), 100 g (Trozo Med.), 120 g (Trozo Grande) |
| Plátano verde | 1,4 | 1,3 | 34,9 | 40 g (1/4 Ud.), 80 g (1/2 Ud.), 160 g (1 Ud.) |
| Plátano maduro | 1,2 | 0,4 | 32,1 | 40 g (1/4 Ud.), 80 g (1/2 Ud.), 160 g (1 Ud.) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO I

## TABLA DE INFORMACIÓN NUTRICIONAL PARA FRUTAS

| ***Frutas ( En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Cambur manzano | 1,0 | 1,3 | 19,8 | 1 Ud.100 g , 2 Ud. 200 g. |
| Cambur Titiaro | 1,4 | 1,9 | 27,7 | 1 Ud. 30 g , 2 Ud. 60 g. |
| Coco | 1,9 | 11,9 | 0,6 | 30 g (1/4 Taza), 60 g (1/2), 100 g (1) |
| Durazno | 0,4 | 0,1 | 11,7 | 1 Ud. 30 g , 2 Ud. 60 g, 3 Ud. 90 g |
| Fresas | 0,8 | 0,5 | 7,9 | 30 g (1/4 Taza), 60 g (1/2), 120 g (1) |
| Guanábana | 1,0 | 0,4 | 14,9 | 30 g(1/4 Taza ), 60 g (1/2), 120 g (1) |
| Guayaba rosada | 1,0 | 0,4 | 5,4 | 1 Ud. 60 g, 2 Ud. 120 g. |
| Lechoza | 0,6 | 0,1 | 6,9 | 30 g (1/4 Taza), 60 g (1/2), 120 g (1) |
| Mandarina | 0,8 | 0,3 | 11,9 | 1 Ud. 60 g, 2 Ud.120 g. |
| Mango | 0,6 | 0,1 | 14,8 | 1 Ud. 100 g, 2 Ud. 200 g. |
| Manzana criolla | 0,4 | 0,2 | 9,7 | 1 Ud.100 g, 2 Ud. 200 g. |
| Manzana importada | 0,3 | 0,3 | 12,8 | 1 Ud. 120 g, 2 Ud. 240 g. |
| Melón | 0,6 | 0,2 | 4,3 | 30 g (1/4 Taza), 60 g (1/2), 120 g (1) |
| Mora | 1,2 | 0,6 | 13,9 | 30 g (1/4 Taza), 60 g (1/2), 120 g (1) |
| Naranja valencia | 0,7 | 0 | 8,8 | 1 Ud. 50 g, 2 Ud.100 g. |
| Níspero | 0,4 | 1,0 | 17,9 | 1 Ud. 60 g, 2 Ud.120 g. |
| Patilla | 0,5 | 0 | 4,8 | 30 g(1/4 Taza), 60 g (1/2), 120 g (1) |
| Parchita | 1,8 | 3,4 | 13,9 | 30 g(1/4 Taza), 60 g (1/2), 120 g (1) |
| Pera | 0,7 | 0,4 | 14,2 | 1 Ud. 80 g, 2 Ud.180 g. |
| Piña | 0,4 | 0,2 | 8,4 | 30 g (1/4 Taza), 60 g (1/2), 120 g (1) |
| Tamarindo | 2,4 | 0,3 | 73,7 | 30 g(1/4 Taza), 60 g (1/2 Taza) |
| Uva | 0,7 | 0,3 | 15,7 | 50 g (1/2 Taza), 100 g (1 Taza) |
| Ciruelas pasas | 2,3 | 0,5 | 65,9 | 30 g (1/4 Taza), 60 g (1/2 Taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO J

## TABLA DE INFORMACIÓN NUTRICIONAL PARA LEGUMINOSAS

| ***Leguminosas ( En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g)*** |
| Arvejas | 7,4 | 0,3 | 13,5 | 40 g (1/4 Taza ) 80 g (1/2 Taza), 160 g (1 Taza) |
| Caraotas blancas | 8,0 | 0,5 | 11,8 | 40 g(1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Caraotas negras | 8,5 | 0,3 | 11,3 | 40 g (1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Caraotas rojas | 8,6 | 0,5 | 13,0 | 40 g(1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Frijol | 8,2 | 0,4 | 14,0 | 40 g(1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Garbanzo | 9,5 | 1,8 | 14,7 | 40 g(1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Lentejas | 6,8 | 0,3 | 8,7 | 40 g (1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |
| Quinchoncho | 8,2 | 0,5 | 15,2 | 40 g(1/4 Taza), 80 g (1/2 Taza), 160 g (1 Taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO K

## TABLA DE INFORMACIÓN NUTRICIONAL PARA ALIMENTOS PREPARADOS

| *Alimentos Preparados ( En 100 g de parte comestible).* | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (g/cc)*** |
| Mermelada | 0,5 | 0,3 | 71,2 | Cucharadita (5 g)<br>Cucharada (10 g) |
| Pudín | 3.4 | 3,1 | 22,2 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Flan | 2,9 | 3,4 | 17,8 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Gelatina | 1,6 | 0 | 15,2 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Jalea de Mango | 0,5 | 1 | 43,7 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Papas fritas | 4,5 | 11,3 | 49,5 | 80 g (Ud. Peq.), 100 g (Ud. Med.), 120 g (Ud. Grande) |
| Tajadas | 1,4 | 9,2 | 34 | 40 g (1/4 Ud.), 80 g (1/2 Ud.) |
| Tostones | 1,9 | 10,1 | 38,3 | 40 g (1/4 Ud.), 80 g (1/2 Ud.) |
| Hamburguesa con queso | 12,3 | 11,8 | 23 | 1 Ud. 200 g |
| Hamburguesa doble carne doble queso | 16,3 | 16,1 | 18,7 | 1 Ud. 280 g |
| Salsa de tomate tipo ketchup | 2,0 | 0,4 | 20,6 | 1 Cucharada (10 g)<br>2 Cucharadas (20 g) |
| Helado de crema | 4,1 | 12,0 | 22,0 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Helado de leche | 5,0 | 4,0 | 25,4 | 45 cc (1/4 Taza), 90 cc (1/2 Taza), 180 cc (1 Taza) |
| Chocolate en barra | 3,8 | 16,8 | 76,1 | 1 Ud. 30 g. |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO L

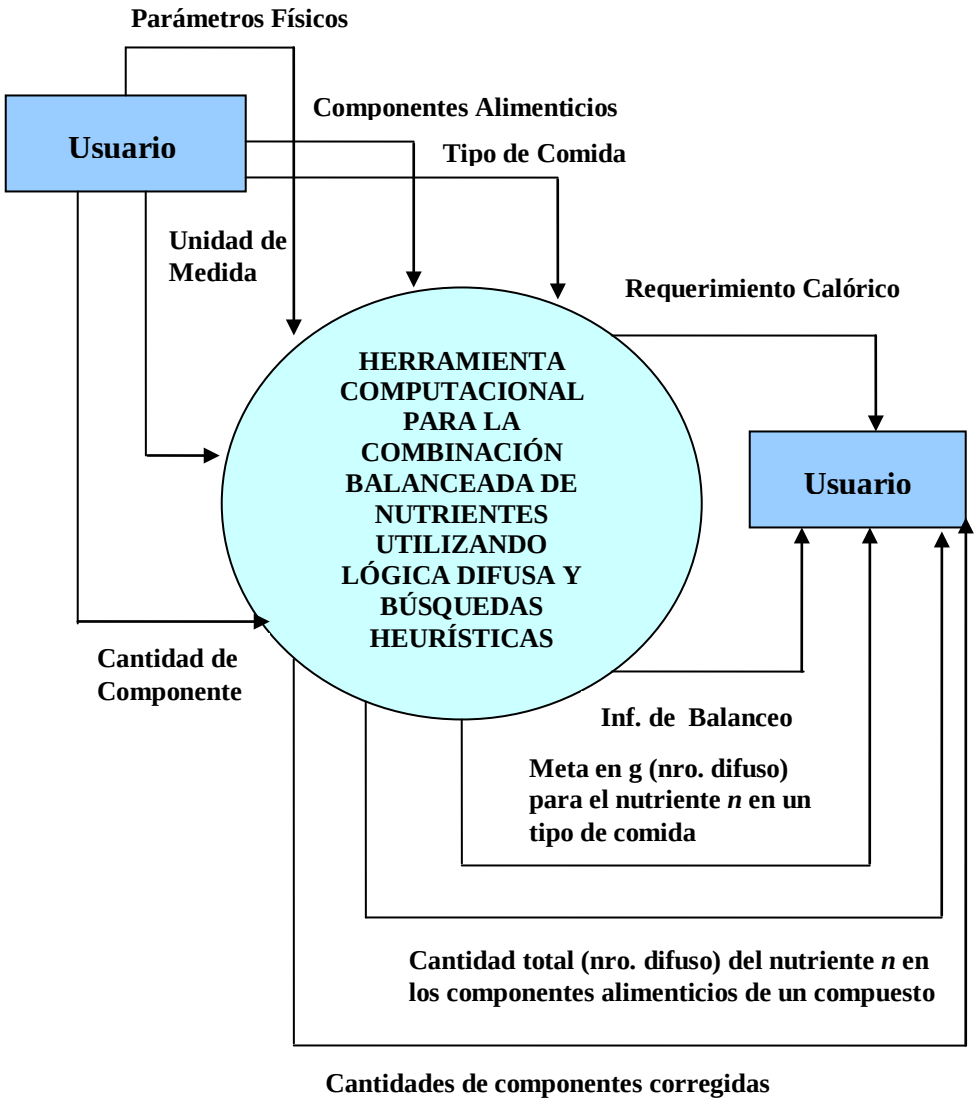
## TABLA DE INFORMACIÓN NUTRICIONAL PARA BEBIDAS ALCOHÓLICAS Y NO ALCOHÓLICAS.

| ***Bebidas Alcohólicas y No Alcohólicas ( En 100 g de parte comestible).*** | | | | |
|---|---|---|---|---|
| ***Alimento*** | ***Prot.*** | ***Grasas*** | ***Carb.*** | ***Unidad de Medida (cc)*** |
| Cerveza | 0,6 | 0,0 | 3,8 | 1 Botella (240 cc) |
| Ponche crema | 4,5 | 5,2 | 14,4 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Bebidas gaseosas | 0 | 0 | 12 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Café infusión s/azúcar | 0,3 | 0,1 | 0,8 | 25 cc (1/8 taza )<br>50 cc (1/4 taza )<br>100 cc (1/2 taza) |
| Jugo de caña | 0,2 | 0 | 15,5 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Agua de coco | 0,2 | 0,1 | 4,1 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Chicha de arroz | 0,6 | 0 | 23,8 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Malta | 0,1 | 0 | 13,7 | 1 Botella (240 cc) |
| Naranjada | 0.2 | 0 | 5,0 | 50 cc (1/4 Taza) ,100 cc (1/2 Taza), 200 cc (1 Taza) |
| Té infusión | 0,1 | 0 | 0,4 | 25 cc (1/8 taza )<br>50 cc (1/4 taza )<br>100 cc (1/2 taza) |

**Fuente:** Tabla de composición de alimentos del INN.

# ANEXO M

## DIAGRAMA CONTEXTUAL DE LA HERRAMIENTA COMPUTACIONAL



# ANEXO N

## TABLA DE INFORMACIÓN DE LOS RESULTADOS POR COMIDA GENERADOS POR EL PRIMER ALGORITMO HEURÍSTICO DE BALANCEO

| *Primer algoritmo heurístico de balanceo basado en el Desequilibrio Nutricional* | | | |
|---|---|---|---|
| ***Comida*** | ***Balanceada(Si/No)*** | ***Tiempo de balanceo (min:seg:ms)*** | ***Número de nodos generados*** |
| 1 | *Si* | *00:00:090* | *255* |
| 2 | *Si* | *00:00:030* | *92* |
| 3 | *No* | - | - |
| 4 | *Si* | *00:00:030* | *122* |
| 5 | *Si* | *00:00:010* | *30* |
| 6 | *Si* | *00:00:030* | *28* |
| 7 | *Si* | *00:00:591* | *864* |
| 8 | *Si* | *00:00:020* | *43* |
| 9 | *Si* | *00:01:993* | *2000* |
| 10 | *Si* | *00:00:020* | *24* |
| 11 | *No* | - | - |
| 12 | *No* | - | - |
| 13 | *No* | - | - |
| 14 | *Si* | *00:00:020* | *67* |
| 15 | *Si* | *00:00:020* | *28* |
| 16 | *Si* | *00:00:030* | *25* |
| 17 | *Si* | *00:00:040* | *115* |
| 18 | *Si* | *00:00:961* | *1392* |
| 19 | *Si* | *00:00:020* | *55* |
| 20 | *No* | - | - |
| 21 | *No* | - | - |
| 22 | *Si* | *00:00:030* | *21* |
| 23 | *Si* | *00:00:030* | *40* |
| 24 | *Si* | *00:00:040* | *72* |
| 25 | *No* | - | - |
| 26 | *No* | - | - |
| 27 | *Si* | *00:00:020* | *44* |
| 28 | *Si* | *00:00:020* | *52* |
| 29 | *No* | - | - |
| 30 | *Si* | *00:00:020* | *67* |

# ANEXO O

## TABLA DE INFORMACIÓN DE LOS RESULTADOS POR COMIDA GENERADOS POR EL ALGORITMO DE BALANCEO BASADO EN LA NECESIDAD DE EMPAREJAMIENTO

| ***Segundo algoritmo de balanceo basado en la Necesidad de Emparejamiento.*** | | | |
|---|---|---|---|
| ***Comida*** | ***Balanceada(Si/No)*** | ***Tiempo de balanceo (min:seg:ms)*** | ***Número de nodos generados*** |
| 1 | *Si* | *00:01:292* | *1597* |
| 2 | *Si* | *00:00:901* | *1170* |
| 3 | *No* | - | - |
| 4 | *Si* | *00:08:101* | *3899* |
| 5 | *Si* | *00:01:923* | *1675* |
| 6 | *Si* | *00:00:040* | *32* |
| 7 | *Si* | *00:31:114* | *4616* |
| 8 | *Si* | *00:00:051* | *175* |
| 9 | *No* | - | - |
| 10 | *Si* | *00:00:020* | *18* |
| 11 | *Si* | *00:00:050* | *117* |
| 12 | *Si* | *00:00:320* | *636* |
| 13 | *Si* | *00:00:020* | *32* |
| 14 | *Si* | *00:00:090* | *275* |
| 15 | *Si* | *00:00:030* | *38* |
| 16 | *Si* | *00:00:030* | *32* |
| 17 | *Si* | *00:25:077* | *5327* |
| 18 | *Si* | *00:03:040* | *2493* |
| 19 | *Si* | *00:00:110* | *326* |
| 20 | *Si* | *00:24:040* | *4170* |
| 21 | *Si* | *00:00:370* | *733* |
| 22 | *Si* | *00:00:030* | *26* |
| 23 | *Si* | *00:01:282* | *1579* |
| 24 | *No* | - | - |
| 25 | *No* | - | - |
| 26 | *Si* | *00:00:351* | *682* |
| 27 | *Si* | *00:00:040* | *113* |
| 28 | *Si* | *00:04:747* | *2875* |
| 29 | *No* | - | - |
| 30 | *No* | - | - |

# ANEXO P

## TABLA DE INFORMACIÓN DE LOS RESULTADOS POR COMIDA GENERADOS POR EL ALGORITMO DE BALANCEO BASADO EN LA DISTANCIA ANTES DEL EMPAREJAMIENTO

| *Tercer algoritmo de balanceo basado en la Distancia antes del Emparejamiento.* | | | |
|---|---|---|---|
| ***Comida*** | ***Balanceada(Si/No)*** | ***Tiempo de balanceo (min:seg:ms)*** | ***Número de nodos generados*** |
| 1 | *Si* | *00:00:120* | *345* |
| 2 | *Si* | *00:00:561* | *947* |
| 3 | *No* | - | - |
| 4 | *Si* | *00:02:213* | *2078* |
| 5 | *Si* | *00:01:052* | *1253* |
| 6 | *Si* | *00:00:020* | *19* |
| 7 | *Si* | *00:28:701* | *4510* |
| 8 | *Si* | *00:00:040* | *142* |
| 9 | *Si* | *00:00:310* | *616* |
| 10 | *Si* | *00:00:030* | *71* |
| 11 | *Si* | *00:00:040* | *87* |
| 12 | *Si* | *00:00:270* | *575* |
| 13 | *Si* | *00:00:030* | *28* |
| 14 | *Si* | *00:00:050* | *137* |
| 15 | *Si* | *00:00:030* | *38* |
| 16 | *Si* | *00:00:020* | *32* |
| 17 | *Si* | *00:19:448* | *4850* |
| 18 | *Si* | *00:00:090* | *57* |
| 19 | *Si* | *00:00:020* | *73* |
| 20 | *Si* | *00:18:146* | *3954* |
| 21 | *Si* | *00:00:241* | *550* |
| 22 | *Si* | *00:00:030* | *16* |
| 23 | *Si* | *00:01:322* | *1596* |
| 24 | *Si* | *00:10:235* | *4170* |
| 25 | *No* | - | - |
| 26 | *Si* | *00:00:050* | *113* |
| 27 | *Si* | *00:00:020* | *68* |
| 28 | *Si* | *00:03:385* | *2445* |
| 29 | *No* | - | - |
| 30 | *No* | - | - |

# ANEXO Q

## TABLA DE INFORMACIÓN DE LOS RESULTADOS POR COMIDA GENERADOS POR EL ALGORITMO DE BALANCEO BASADO EN LA DISTANCIA ANTES DEL EMPAREJAMIENTO MEJORADA

| *Cuarto algoritmo basado en la Distancia antes del Emparejamiento Mejorada.* | | | |
|---|---|---|---|
| ***Comida*** | ***Balanceada(Si/No)*** | ***Tiempo de balanceo (min:seg:ms)*** | ***Número de nodos generados*** |
| 1 | *Si* | *00:00:020* | *76* |
| 2 | *Si* | *00:00:171* | *362* |
| 3 | *No* | - | - |
| 4 | *Si* | *00:00:181* | *408* |
| 5 | *Si* | *00:00:010* | *15* |
| 6 | *Si* | *00:00:020* | *9* |
| 7 | *Si* | *00:00:131* | *294* |
| 8 | *Si* | *00:00:020* | *34* |
| 9 | *Si* | *00:00:040* | *89* |
| 10 | *Si* | *00:00:020* | *24* |
| 11 | *Si* | *00:00:020* | *22* |
| 12 | *Si* | *00:00:080* | *217* |
| 13 | *Si* | *00:00:020* | *13* |
| 14 | *Si* | *00:00:030* | *50* |
| 15 | *Si* | *00:00:020* | *18* |
| 16 | *Si* | *00:00:030* | *16* |
| 17 | *Si* | *00:00:130* | *329* |
| 18 | *Si* | *00:00:030* | *30* |
| 19 | *Si* | *00:00:010* | *26* |
| 20 | *Si* | *00:01:762* | *1447* |
| 21 | *Si* | *00:00:060* | *175* |
| 22 | *Si* | *00:00:020* | *8* |
| 23 | *Si* | *00:00:220* | *460* |
| 24 | *Si* | *00:00:301* | *525* |
| 25 | *No* | - | - |
| 26 | *Si* | *00:00:030* | *51* |
| 27 | *Si* | *00:00:010* | *24* |
| 28 | *Si* | *00:00:991* | *1204* |
| 29 | *No* | - | - |
| 30 | *No* | - | - |

# ANEXO R

## DEFINICIÓN DE LAS ESTRUCTURAS DE DATOS E IMPLEMENTACIÓN DE LOS PROCEDIMIENTOS MÁS IMPORTANTES

```
//Unidad que contiene la definición de las clases, estructuras de datos y los
//procedimientos más importantes para realizar el balanceo de compuestos
//alimenticios

unit uarbol;

interface

uses f_mensajeprogreso,Graphics,sysutils;

//Definición de las clases y estructuras de datos utilizadas para el
//balanceo de compuestos alimenticios
type

 TOpIneficientes   = (modminimo,modpromedio,modmaximo,eliminar);
 TOpEficientes     = (reducir,aumentar);
 THeuristica       = (ThNutricional,ThNecesidad,ThDistancia);
 TTipoAlgoritmo    = (TAEficiente,TAIneficiente,TAEficienteMejorado);
 TOpConLista       = (TOpCrear,TOpcopiar);
 ttiposcomidas     = (desayuno,medman,almuerzo,medtarde,cena);
 ttiponutriente    = (carbohidratos,lipidos,proteinas);

 tnrodifuso = record
    Cord_a1,Cord_b1,Cord_c1,Cord_d1:real;
 end;

 tmeta             = array[ttiponutriente] of real;
 tmetas            = array[ttiposcomidas]  of tmeta;
 tValorPosNec      = array[ttiponutriente] of real;
 TTotalesNutrientes    = array[ttiponutriente] of tnrodifuso;
 tmetasdif         = array[ttiposcomidas]  of TTotalesNutrientes;
 TDeltas           = array[ttiponutriente] of real;
 tPorcentajeComidas    = array[ttiposcomidas] of real;
 tcomidasselec     = set of ttiposcomidas;
```

```
tnalimentos         = array[ttiposcomidas] of byte;
TptrAlimento = ^talimento;

// Registro que contiene la información de un alimento seleccionado en
// la comida
talimento = record
            codalimento,
            codmedida      :integer;
            porcnucleo,
            porcdispersion,
            minimo,
            promedio,
            maximo,
            cantidad       :real;
            cantgramos,
            cantcarbohidratos,
            cantlipidos,
            cantproteinas   :tnrodifuso;
            proximo        :TptrAlimento;
end;

TAlimentos            = array[1..20] of talimento;
talimentosEnComidas     = array[ttiposcomidas] of talimentos;

// Clase que permite definir una la lista de alimentos, la cual está
// contenida en cada estado generado en los algoritmos de búsqueda heurística
TlistaAlimentos = class
 public
  Primero           :TptrAlimento;
  constructor create;
  procedure AgregarAlimento(alimento:talimento);
  procedure ModificarAlimento(codalimento: integer; nuevacantidad: real);
  function BuscarAlimento(codalimento: integer): TPtrAlimento;
  procedure EliminarAlimento(codalimento: integer);
  function maximoValorNutriente1(i: TTipoNutriente): real;
  function MaximoValorNutriente2(i: TTipoNutriente): real;
  destructor destroy;override;
  function AlgunoSobrepasaElMaximo: boolean;
end;

// Clase para manejar los estados que se van originando en los algoritmos de
// búsqueda heurística, es decir, un nodo del árbol que se va generando
```

```
TEstadoComida = class
  private
   HDisminucion,
   HAumento,
   F,H                :real;
   G                  :integer;
   HeuristicaUsada    :THeuristica;
   TotalesNutrientes  :TTotalesNutrientes;
   procedure copiarListaAlimentos(lista: TListaAlimentos);
   public
   Prox          :testadocomida;
   ListaAlimentos  :TlistaAlimentos;
   constructor create;
   procedure calcularF;
   procedure CalcularG(estadoAnt: TEstadoComida);
   procedure copiarEstado(estadoOrigen:TEstadoComida;
                          QueHacer:TOpConLista);
   procedure ActualizarTotales;
   destructor destroy;override;
  end;

  // Clase que maneja la lista de estados que se van generando cuando se le
  // aplica una operación a un estado inicial
  TListaEstadosGenerados = class
   private
   public
   //inicializa la lista en vacio
   Primero :TEstadoComida;
   constructor inicializar;
   function Estavacia:boolean;
   procedure Agregar(Estado:TEstadoComida);
   function menor2: testadocomida;
   procedure extraer2(estado: testadocomida);
   function ExisteEstado2(Estado: TEstadoComida): boolean;
   destructor destroy;override;
  end;

  // Registro y puntero que se utiliza para manejar los estados visitados
  // y que no han sido solución, es decir  que no lograron el balanceo
  TPtrEstadoVisitado = ^TEstadoVisitado;

  TEstadoVisitado = record
   totales :TTotalesNutrientes;
```

```
  prox    :TPtrEstadoVisitado;
end;

//Clase que maneja la lista de estados que han sido visitados
//Esta lista junto con la anterior conforman el árbol de búsqueda
TListaEstadosVisitados = class
   public
    /// inicializa la lista en vacío
    Primero :TPtrEstadoVisitado;
    constructor inicializar;
    procedure Agregar(Estado:TEstadoComida);
    function ExisteEstado(Estado: TEstadoComida): boolean;
    destructor destroy;override;
end;

//Clase que organiza todos los elementos y los métodos necesarios para
//lograr una solución a un problema de balanceo
Tarbol = class
private
  Heuristica            :THeuristica;
  MetaGrsNutrientes     :TTotalesNutrientes;
  EstadoAuxiliar,    // se utiliza temporalmente en uno de los métodos
  EstadoInicial,     // punto de partida para generar los siguientes estados
  EstadoSolucion        :TEstadoComida; //va a tomar valor cuando se
                                        //consiga solución
  ComidasGeneradas      :TListaEstadosGenerados;
  ComidasVisitadas      :TListaEstadosVisitados;
  horainicio: real;
  horafin: real;
  MaximoNodos: integer;
  respetarmaximo: boolean;
  mensajeprogreso           :Tfrmprogreso;
  function EstaBalanceada(comida: TEstadoComida): boolean;
  function ExtraerMejorComida: TEstadoComida;
  procedure GenerarComidasSucesoras(var comida: TEstadoComida);
  procedure GenerarComidasSucesorasEficiente(comida: TEstadoComida);
  procedure calcularH(var Estado:TEstadoComida);
  function EsFactible(Estado: TEstadoComida):boolean;
  procedure ActualizarEstado(var Estado: tEstadoComida;
                             EstadoAnt:tEstadoComida);
  procedure CrearEstadoInicial(Heuristica:THeuristica;
                               alimentos: talimentos; n:integer);
  procedure asignarHoraInicio;
```

```
    procedure AsignarHoraFin;
    procedure CalcularTiempoDuracion;
    function calcularDeltas1(i: ttiponutriente;Estado:TEstadoComida):real;
    procedure GenerarComidasSucesorasEficienteMejorado(comida:EstadoComida);
    function calcularDeltas2(i:ttiponutriente; estado: TEstadoComida):real;
    procedure AgregarcomidaGenerada;
    procedure completarbarra;
    procedure FinalizarBalanceo;
    procedure IniciarBalanceo;
  public
    NroNodosGenerados: integer;
    TiempoDuracion: string;
    constructor Create;
    destructor Destroy; override;
    function BalancearComida(TipoAlgoritmo:TTipoAlgoritmo;maximo:integer;
                             RespetarElMaximo:boolean):boolean;
    procedure inicializarArbol(HeuristicaUsada:THeuristica;
                               metas:TTotalesNutrientes;alimentos:TAlimentos;
                               n:integer);
    function DevolverComidaBalanceada:testadocomida;
  end;

var
  dispxNutriente       :array[ttiponutriente] of byte = (15,30,30);
  nombresnutrientes :array[ttiponutriente] of string = ('Carbohidratos',
                                                        'Lípidos',
                                                        'Proteínas');
  //Nombres de las comidas en string
  nombrescomidas:array[ttiposcomidas] of string[30]=('Desayuno',
                                                     'Merienda mañana',
                                                     'Almuerzo',
                                                     'Merienda tarde','Cena');

  //Color que va a tener cada comida en el gráfico de distribución del requerimiento
  //calórico
  colores      :array[ttiposcomidas] of tcolor = (clred,clwhite,clblue,
                                                  clyellow,clgreen);

implementation

uses operaciones;

//******** Métodos de la clase TListaAlimentos ************
```

```
constructor TListaAlimentos.create;
//Se utiliza para inicializar la lista de alimentos
begin
  Inherited Create;
  Primero:=nil;
end;

procedure TListaAlimentos.AgregarAlimento(alimento:talimento);
//Se utiliza para agregar un alimento a la lista de alimentos, la cual está
//contenida en un estado. Inserta los nodos al comienzo de la lista
var
   p:TptrAlimento;
begin
  new(p);
  with p^ do
  begin
    codalimento:=alimento.codalimento;
    codmedida:=alimento.codmedida;
    cantidad:=alimento.cantidad;
    cantgramos:=alimento.cantgramos;
    cantcarbohidratos:=alimento.cantcarbohidratos;
    cantlipidos:=alimento.cantlipidos;
    cantproteinas:=alimento.cantproteinas;
    minimo:=alimento.minimo;
    maximo:=alimento.maximo;
    porcnucleo:=alimento.porcnucleo;
    porcdispersion:=alimento.porcdispersion;
    promedio:=alimento.promedio;
  end;
  p^.proximo:=Primero;
  Primero:=p
end;

procedure ModificarCantidad(var valor:tnrodifuso;n,d:real;cantidad,
                    nuevacantidad:real);
//Se utiliza para determinar el correspondiente valor de la nueva cantidad
//del nutriente que se está modificando, en base de la cantidad que había
//anteriormente
var
  aux :real;
begin
  aux:=(CentroideDifuso(valor)/cantidad)*nuevacantidad;
  valor:=nitidotodifuso(aux,n,d);
```

```
end;


procedure TlistaAlimentos.ModificarAlimento(codalimento: integer;
                                    nuevacantidad: real);
//Se utiliza para modificar la cantidad que hay de un alimento en la lista de
//alimentos, la cual está contenida en un estado. Es utilizado para generar los
//nuevos estados. Modifica la cantidad del alimento y la cantidad de los
//nutrientes en función de la nueva cantidad
var
 p:TptrAlimento;
begin
  p:=BuscarAlimento(codalimento);
  if p <> nil
  then
  with p^ do
  begin
    ModificarCantidad(cantcarbohidratos,porcnucleo,porcdispersion,cantidad,
                nuevacantidad);
    ModificarCantidad(cantlipidos,porcnucleo,porcdispersion,cantidad,
                nuevacantidad);
    ModificarCantidad(cantproteinas,porcnucleo,porcdispersion,cantidad,
                nuevacantidad);
    cantidad:=nuevacantidad;
  end
end;

function TlistaAlimentos.BuscarAlimento(codalimento:integer):TPtrAlimento;
//Se utiliza para buscar un alimento dentro de la lista de alimentos,
//dado el código del mismo. Devuelve un puntero al alimento encontrado
var
 p:TPtrAlimento;
begin
  result:=nil;
  p:=Primero;
  while p <> nil do
  begin
    if p^.codalimento = codalimento
    then
    begin
      result:=p;
      Break
    end
```

```
   else p:=p^.proximo;
  end;
end;

//Procedimiento para eliminar un alimento de la lista
procedure TlistaAlimentos.EliminarAlimento(codalimento: integer);

var
  p,q:TptrAlimento;
begin
  p:=Primero;
  q:=Primero;
  while p <> nil do
  begin
    if p^.codalimento = codalimento then
    begin
      if q = p  // es el primero
      then Primero:=Primero.proximo
      else q^.proximo:=p^.proximo;
      Dispose(p);
      break
    end else
    begin
       q:=p;
       p:=p^.proximo;
    end
  end
end;

//Función para determinar el alimento que aporta la mayor cantidad de
//de un nutriente
function TlistaAlimentos.maximoValorNutriente1(i:TTipoNutriente):real;
var
  aux:real;
  p:TptrAlimento;
begin
  p:=Primero;
  result:=0;
  while p <> nil do
  begin
    case i of
    carbohidratos   :aux:=p^.minimo*CentroideDifuso(p^.cantcarbohidratos);
    lipidos         :aux:=p^.minimo*CentroideDifuso(p^.cantlipidos);
```

```
   proteinas      :aux:=p^.minimo*CentroideDifuso(p^.cantproteinas);
   end;
   if aux > result
   then result:=aux;
   p:=p^.proximo
  end;
end;

// ***********Métodos de la clase TEstadoComida ***********

constructor TEstadoComida.create;
//Es llamado cuando se crea un estado de la comida. Crea la lista de alimentos
begin
  Inherited Create;
  ListaAlimentos:=TlistaAlimentos.Create;
end;


procedure TEstadoComida.calcularF;
//Se utiliza para calcular F o función de evaluación de cada estado
begin
 F:=G + H
end;

procedure TEstadoComida.copiarListaAlimentos(lista: TListaAlimentos);
//Se utiliza para crear una nueva lista de alimentos basándose en una
//lista ya existente. Esto para que al manipular la original no afecte a ésta
var
  p:TptrAlimento;
begin
  ListaAlimentos.Primero:=nil;
  p:=lista.Primero;
  while p <> nil do
  begin
    ListaAlimentos.AgregarAlimento(p^);
    p:=p^.proximo
  end;
end;

procedure TEstadoComida.CalcularG(estadoAnt: TEstadoComida);
//Se utiliza para calcular la G necesaria para calcular la F.
//La G se calcula en función de la heurística usada
var
```

```
  p:TptrAlimento;
  cantidadini,
  aux:real;
begin
  case HeuristicaUsada of
   ThNutricional    :G:=estadoAnt.G + 1;
  else
    p:=ListaAlimentos.Primero;
    aux:=0;
    while p <> nil do
    begin
      cantidadini:=estadoAnt.ListaAlimentos.
                BuscarAlimento(p^.codalimento)^.cantidad;
      aux:=aux + (Abs(p^.cantidad - cantidadini)/p^.minimo);
      if p^.cantidad = 0 // si fue eliminado el alimento
      then aux:=aux + 3;
      p:=p^.proximo;
    end;
    G:=trunc(aux);
  end;
end;

procedure TEstadoComida.ActualizarTotales;
//Se utiliza para actualizar los totales de nutrientes que hay en una comida
//Esto es la sumatoria de los nutrientes de cada alimentos que haya en la
//lista de alimentos
var
  i:TtipoNutriente;
  p :TptrAlimento;
begin
  for i :=low(TtipoNutriente)  to high(TtipoNutriente) do
  begin
    TotalesNutrientes[i]:=inicializadifuso;
    p:=ListaAlimentos.Primero;
    while p<> nil do
    begin
      case i of
      carbohidratos:
      TotalesNutrientes[i]:=suma(TotalesNutrientes[i],
                          p^.cantcarbohidratos);
      lipidos:
      TotalesNutrientes[i]:=suma(TotalesNutrientes[i],
                          p^.cantlipidos);
```

```
      proteinas:
      TotalesNutrientes[i]:=suma(TotalesNutrientes[i],
                           p^.cantproteinas);
      end;
      p:=p^.proximo;
    end;
  end;
end;

procedure TEstadoComida.copiarEstado(estadoOrigen: TEstadoComida;
                              QueHacer:TOpConLista);
//Se utiliza cuando se desea asignar el valor de un estado a otro
//debido a que no se puede hacer una simple asignación
begin
  F            :=estadoOrigen.F;
  G            :=estadoOrigen.G;
  H            :=estadoOrigen.H;
  Prox           :=estadoOrigen.Prox;
  HeuristicaUsada :=estadoOrigen.HeuristicaUsada;
  TotalesNutrientes:=estadoOrigen.TotalesNutrientes;
  HDisminucion    :=estadoOrigen.HDisminucion;
  HAumento       :=estadoOrigen.HAumento;
  case QueHacer of
  TOpCrear  :copiarListaAlimentos(estadoOrigen.ListaAlimentos);
  TOpCopiar :ListaAlimentos:=estadoOrigen.ListaAlimentos;
  end;
end;


// ********* Métodos de la clase TListaEstadosGenerados *****************

function TListaEstadosGenerados.Estavacia:boolean;
// verifica si la lista esta vacia
begin
  Result:= Primero = nil
end;

procedure TListaEstadosGenerados.Agregar(Estado:TEstadoComida);
//Se utiliza para agregar un estado factible a la lista de estados
//generados. Debe crearse una instancia nueva del estado y debe tenerse
//una lista independiente. El estado se agrega al comienzo de la lista
var
  a:TEstadoComida;
```

```
begin
  a:=TEstadoComida.create;
  a.copiarEstado(estado,TopCrear);
  a.copiarListaAlimentos(estado.ListaAlimentos);
  if Primero = nil
  then a.Prox:=nil
  else a.Prox:=Primero;
  Primero:=a
end;

constructor TListaEstadosGenerados.inicializar;
//Inicializa la lista de estados generados en vacio
begin
  primero:=nil;
end;

function TListaEstadosGenerados.menor2: testadocomida;
//Se utiliza para determinar cual estado de la lista de estados generados posee
//el menor valor de F
var
  elmenor,
  p             :TEstadoComida;
  auxmenor      :real;
begin
  auxmenor:= Primero.F;
  p:=TEstadoComida.create;
  elmenor:=TEstadoComida.create;
  elmenor:=Primero;
  p:=Primero.Prox;
  while p <> nil do
  begin
    if p.F < auxmenor
    then
    begin
      elmenor:=p;
      auxmenor:=p.F
    end;
    p:=p.Prox;
  end;
  Result:=elmenor;
end;
```

```
procedure TListaEstadosGenerados.extraer2(estado:testadocomida);
//Se utiliza para extraer un nodo (estado o comida) de la lista de
//estados generados. Se extrae de la lista para que no vuelva a ser
//tomado en cuenta en la siguiente iteración del algoritmo
var
  q,p:TEstadoComida;
begin
  p:=TEstadoComida.create;
  q:=TEstadoComida.create;
  p:=Primero;
  q:=p;
  while p <> nil do
  begin
    if p = Estado
    then
    begin
      if p = q
      then primero:=p.Prox
      else q.Prox:=p.Prox;
      break
    end else
    begin
      q:=p;
      p:=p.Prox;
    end
  end;
end;

function TListaEstadosGenerados.ExisteEstado2(Estado:
TEstadoComida):boolean;
//Se utiliza para verificar si un estado ya ha sido generado y no ha
//sido visitado aún, para no volverlo a agregar a la lista de generados
//y así optimizar la creación de nodos
var
  p :TEstadoComida;
  aux :boolean;
  i :ttiponutriente;
begin
  result:=false;
  p:=TEstadoComida.create;
  p:=Primero;
  while p <> nil do
  begin
```

```
   //Se verifica el centroide de los tres nutrientes de cada
   //elemento en la lista, con los tres nutrientes del elemento buscado
   aux:=true;
   for i :=low(ttiponutriente)  to high(ttiponutriente) do
    begin
     if CentroideDifuso(p.TotalesNutrientes[i]) <>
       CentroideDifuso(Estado.TotalesNutrientes[i])
     then
     begin
       aux:=false;
       break
     end
   end;
   if aux = true
   then
   begin
     result:=true;
     break
   end;
   p:=p.Prox;
  end;
end;


// ************ Métodos de la clase TListaEstadosVisitados *****************

constructor TListaEstadosVisitados.inicializar;
//Inicializa la lista de estados visitados en vacío
begin
  Primero:=nil;
end;

procedure TListaEstadosVisitados.Agregar(Estado:TEstadoComida);
//Se utiliza para agregar un nodo a la lista de estados visitados
var
 aux :TPtrEstadoVisitado;
begin
 new(aux);
 with aux^ do
 begin
  totales:=Estado.TotalesNutrientes
 end;
 if Primero = nil
```

```
  then aux^.Prox:=nil
  else aux^.Prox:=Primero;
  Primero:=aux;
end;

function TListaEstadosVisitados.ExisteEstado(Estado:TEstadoComida): boolean;
//Función que sirve para buscar un estado dentro de la lista de estados.
var
  p:TPtrEstadoVisitado;
  i :ttiponutriente;
  aux:boolean;
begin
  result:=false;
  p:=Primero;
  while p <> nil do
  begin
    //Se verifica el centroide de los tres nutrientes de cada
    //elemento en la lista con los tres nutrientes del elemento buscado
    aux:=true;
    for i :=low(ttiponutriente)  to high(ttiponutriente) do
    begin
      if CentroideDifuso(p^.Totales[i]) <>
         CentroideDifuso(Estado.TotalesNutrientes[i])
      then
      begin
        aux:=false;
        break
      end
    end;
    if aux = true
    then
    begin
      result:=true;
      break
    end;
    p:=p^.Prox;
  end;
end;

// ************ Métodos de la clase TArbol ******************
constructor tarbol.create;
//Es llamado automáticamente cuando se instancia el árbol
begin
```

```
  Inherited Create;
  EstadoInicial:=TEstadoComida.Create;
  EstadoSolucion:=TEstadoComida.Create;
  EstadoAuxiliar:=TEstadoComida.create;

  ComidasGeneradas:=TListaEstadosGenerados.Create;
  ComidasVisitadas:=TListaEstadosVisitados.Create;

  ComidasGeneradas.inicializar;
  ComidasVisitadas.inicializar;
  mensajeprogreso:= Tfrmprogreso.Create(nil);
end;

destructor tarbol.destroy;
//Es llamado automáticamente cuando se libera el árbol al cerrarse el programa
begin
  EstadoAuxiliar.free;
  EstadoInicial.free;
  EstadoSolucion.free;
  ComidasGeneradas.free;
  ComidasVisitadas.free;
  inherited destroy;
end;

function tarbol.BalancearComida(TipoAlgoritmo:TTipoAlgoritmo;
                                maximo:integer;
                                RespetarElMaximo:boolean):boolean;
// Es el algoritmo principal. Genera los estados hijos de un estado inicial,
// determina cual de ellos es el mejor, y si este no es la solución
// genera mas nodos a partir de él. Culmina cuando no se puedan generar
//más nodos o cuando se encuentre la solución que balancea la comida
var
  Estadoactual:TEstadoComida;
begin
  MaximoNodos:=maximo;
  respetarmaximo:=respetarElmaximo;
  IniciarBalanceo;
  Estadoactual:=TEstadoComida.create;
  Estadoactual.copiarEstado(EstadoInicial,TOpCrear);
  result:=true;
  while not EstaBalanceada(Estadoactual) do
  begin
    ComidasVisitadas.Agregar(EstadoActual);
```

```
   case TipoAlgoritmo of
   TAIneficiente          :GenerarComidasSucesoras(EstadoActual);
   TAEficiente            :GenerarComidasSucesorasEficiente(EstadoActual);
    TAEficienteMejorado:
    GenerarComidasSucesorasEficienteMejorado(EstadoActual);
   end;
   if (not ComidasGeneradas.Estavacia) and
     (NroNodosGenerados < MaximoNodos)
   then EstadoActual:=ExtraerMejorComida
   else
   begin
    result:=false; // No se logró balancear la comida
    break
   end;
  end;
  //si se consiguió una solución se le asigna ésta (ComidaActual)
  //al campo Comida Solución del Árbol
  if result
  then EstadoSolucion:=Estadoactual;
  FinalizarBalanceo;
end;

procedure tarbol.InicializarArbol(HeuristicaUsada:THeuristica;
            metas:TTotalesNutrientes;alimentos:TAlimentos;n:integer);
//Se utiliza para inicializar el árbol, para empezar a buscar la solución
//generando nodos a partir del estado inicial (basándose en los alimentos)
//que fueron seleccionados por el usuario
begin
  TiempoDuracion:='';
  NroNodosGenerados:=0;
  MetaGrsNutrientes:= Metas;
  Heuristica := HeuristicaUsada;
  CrearEstadoInicial(Heuristica,alimentos,n);
  ComidasGeneradas.inicializar;
  ComidasVisitadas.inicializar;
end;

function tarbol.DevolverComidaBalanceada:testadocomida;
//Se utiliza cuando se desea saber cual es el estado (la comida) que
//generó la solución (en el caso de que ésta exista)
begin
  result:=EstadoSolucion;
end;
```

```
function Tarbol.EstaBalanceada(comida: TEstadoComida): boolean;
// Se utiliza para verificar si un estado (una comida)
// esta balanceada, y por consiguiente si es el estado solución.
// Se calcula verificando si los tres totales de los nutrientes
// de la comida están emparejados con la meta, la cual esta almacenada en
// en la clase tarbol
var
 i:ttiponutriente;
 a,b:real;
begin
 result:=true;
 for i :=low(ttiponutriente) to high(ttiponutriente) do
  begin
    if not Emparejado(MetaGrsNutrientes[i],
                comida.TotalesNutrientes[i],
                a,b)
    then
    begin
          result:=false;
          break
    end;
  end;
 if result and respetarmaximo
 then result:=not comida.ListaAlimentos.AlgunoSobrepasaElMaximo;
end;

procedure Tarbol.GenerarComidasSucesoras(var comida: TEstadoComida);
// Este algoritmo genera las comidas sucesoras utilizando las operaciones:
// -Modificar un alimento a un mínimo
// -Modificar a un promedio
// -Modificar a un máximo
// -Remover el alimento

var
 auxalimento    :talimento;
 p              :TptrAlimento;
 operacion      :TOpIneficientes;
begin
 p:=comida.ListaAlimentos.Primero;
 EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
 while p <> nil do
 begin
  if p^.cantidad <> 0 then
```

```
begin
  auxalimento:=p^;
  for operacion := low(TOpIneficientes) to high(TOpIneficientes) do
  begin
    case operacion of
    modminimo      : begin
                       EstadoAuxiliar.ListaAlimentos.
                       ModificarAlimento(p^.codalimento,p^.minimo);
                     end;
    modpromedio    : begin
                       EstadoAuxiliar.ListaAlimentos.
                       ModificarAlimento(p^.codalimento,p^.promedio);
                     end;
    modmaximo      : begin
                       EstadoAuxiliar.ListaAlimentos.
                       ModificarAlimento(p^.codalimento,p^.maximo);
                     end;
    eliminar       : begin
                       EstadoAuxiliar.ListaAlimentos.
                       ModificarAlimento(p^.codalimento,0);
                     end;
    end;

    //Actualizar el estado (calcular F y G )
    //verificar que tampoco esté en comidas generadas
    ActualizarEstado(EstadoAuxiliar,comida);
    if not ComidasVisitadas.ExisteEstado(EstadoAuxiliar) and
      not ComidasGeneradas.ExisteEstado2(EstadoAuxiliar)
    then
    begin
      inc(NroNodosGenerados);
      ComidasGeneradas.Agregar(EstadoAuxiliar);
      mensajeprogreso.progreso.StepBy(1);
      if NroNodosGenerados = maximoNodos
      then exit
    end
    else;
    //Hay que reestablecer el estado y el alimento para realizar la
    //siguiente operación
    EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
    p^:=auxalimento;
  end;
end;
```

```
   p:=p^.proximo;
  end;
end;

//Función para extraer el estado con el menor  valor de la función de evaluación
//de la lista de estados generados
function Tarbol.ExtraerMejorComida: TEstadoComida;
var
   aux:TEstadoComida;
begin
  aux:=TEstadoComida.create;
  aux:=ComidasGeneradas.menor2;
  ComidasGeneradas.extraer2(aux);
  Result:=aux
end;

procedure Tarbol.ActualizarEstado(var Estado:tEstadoComida;
                          EstadoAnt:tEstadoComida);
//Es utilizado para calcular F, G y H correspondientes a un estado
begin
  calcularH(Estado);
  Estado.CalcularG(EstadoAnt);
  Estado.calcularF;
end;

function Tarbol.EsFactible(Estado: TEstadoComida): boolean;
// Se utiliza para determinar si un estado que ha sido generado
// es factible para ser agregado en la lista de Estados Generados,
// para verificar luego si es solución o crear nuevos estados a partir de él
var
  aux,
  total :real;
  i :ttiponutriente;
  a,b:Real;
begin
  total:=0;
  result:=true;
  for i :=low(ttiponutriente)  to high(ttiponutriente) do
  begin
       aux:=CentroideDifuso(Estado.TotalesNutrientes[i]);
       if(CentroideDifuso(MetaGrsNutrientes[i]) < aux) and
         not Emparejado(MetaGrsNutrientes[i],Estado.TotalesNutrientes[i],a,b)
       then
```

```
      begin
          result:=false;
          break
      end;
      total:=total + aux;
  end;
  if total = 0
  then result:=false;
end;

procedure Tarbol.calcularH(var Estado:TEstadoComida);
//Se utiliza para calcular la H necesaria en la heurística, la cual permite
//calcular la F de un estado dado. Esta H se calcula de una forma u otra
//dependiendo de la heurística que se este utilizando
Var
  i :ttiponutriente;
  aux,min,max,
  a,b:real;
  Deltas  :TDeltas;
begin
 Estado.ActualizarTotales;
 case Heuristica of
 //Calcula la h del primer algoritmo de balanceo basado en el desequilibrio
 //nutricional
 ThNutricional    :
  begin
    Estado.H:=0;
    for i :=low(ttiponutriente) to high(ttiponutriente) do
    begin
          if not Emparejado(MetaGrsNutrientes[i],
                    Estado.TotalesNutrientes[i],
                    a,b )
          then Estado.H:=Estado.H + 1
          else ;
    end;
 end;
 //Calcula la h del segundo algoritmo de balanceo basado en la necesidad de
 //emparejamiento
 ThNecesidad      :
 begin
  aux:=0;
  for i :=low(ttiponutriente) to high(ttiponutriente) do
        aux:=aux + (1 - Calculo_Necesidad(MetaGrsNutrientes[i].Cord_a1,
```

```
                    MetaGrsNutrientes[i].Cord_b1,
                    MetaGrsNutrientes[i].Cord_c1,
                    MetaGrsNutrientes[i].Cord_d1,
                    Estado.TotalesNutrientes[i].Cord_a1,
                    Estado.TotalesNutrientes[i].Cord_b1,
                    Estado.TotalesNutrientes[i].Cord_c1,
                    Estado.TotalesNutrientes[i].Cord_d1,1));
        Estado.H:=aux / 2;
end ;

//Calcula la h del tercer algoritmo de balanceo basado en la distancia
//antes del emparejamiento
ThDistancia     :
begin
 min:=1000;
 max:=0;
 for i :=low(ttiponutriente)  to high(ttiponutriente) do
 begin
  Deltas[i]:=calcularDeltas1(i,Estado);
  if deltas[i] <= 0
  then
        if deltas[i] <= min
        then min:=deltas[i]
        else
  else
        if deltas[i] >= 0
        then
              if deltas[i] >= max
              then max:=deltas[i]
              else
 end;
 if min = 1000
 then min:=0
 else min:=-1*min;
 Estado.H:= min + max;
 Estado.HDisminucion:=min;
 Estado.HAumento:=max;
end;
ThDistanciaMejor :
begin
  min:=1000;
  max:=0;
  for i :=low(ttiponutriente)  to high(ttiponutriente) do
```

```
   begin
     Deltas[i]:=calcularDeltas2(i,Estado);
     if deltas[i] < 0
     then
       if deltas[i] < min
       then min:=deltas[i]
       else
     else
       if deltas[i] > 0
       then
         if deltas[i] > max
         then max:=deltas[i]
         else
   end;
   if min = 1000
   then min:=0
   else min:=-1*min;
   Estado.H:= min + max;
   Estado.HDisminucion:=min;
   Estado.HAumento:=max;
 end;
 end;
end;

procedure Tarbol.GenerarComidasSucesorasEficiente(comida: TEstadoComida);
// Este algoritmo genera las comidas sucesoras, se diferencia del otro
// porque este es mas eficiente. Utiliza las siguientes operaciones:
// -Agregar una cantidad Af al alimento
// -Quitar una cantidad Af del alimento

var
  auxalimento     :talimento;
  p               :TptrAlimento;
  operación       :TOpEficientes;
begin
  p:=comida.ListaAlimentos.Primero;
  EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
  while p <> nil do
  begin
    if p^.cantidad <> 0 then
    begin
      auxalimento:=p^;
      for operacion := low(TOpEficientes) to high(TOpEficientes) do
```

```
    begin
     case operacion of
     reducir      :
      if auxalimento.cantidad > 0 then
       begin
        EstadoAuxiliar.ListaAlimentos.
        ModificarAlimento(p^.codalimento,p^.cantidad - p^.minimo);
       end;
     aumentar     :
     if not respetarmaximo or (auxalimento.cantidad < auxalimento.maximo)
     then
       begin
        EstadoAuxiliar.ListaAlimentos.
        ModificarAlimento(p^.codalimento,p^.cantidad + p^.minimo);
       end;
     end;
     // actualizar el estado (calcular F y G )
     // verificar que tampoco este en comidas generadas
     ActualizarEstado(EstadoAuxiliar,EstadoInicial);
     if not ComidasVisitadas.ExisteEstado(EstadoAuxiliar) and
       not ComidasGeneradas.ExisteEstado2(EstadoAuxiliar)
     then
     begin
      inc(NroNodosGenerados);
      ComidasGeneradas.Agregar(EstadoAuxiliar);
      mensajeprogreso.progreso.StepBy(1);
      if NroNodosGenerados = maximoNodos
      then exit
     end
     else ;
     // Hay que reestablecer el estado y el alimento para realizar la
     // siguiente operación
     EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
     p^:=auxalimento;
    end;
   end;
   p:=p^.proximo;
  end;
end;

procedure Tarbol.CrearEstadoInicial(Heuristica: THeuristica;
                          alimentos: talimentos; n: integer);
// Se utiliza para crear el estado inicial del árbol, el cual es el punto
```

```
// de partida del algoritmo
var
 i:byte;
begin
  with EstadoInicial do
  begin
    HeuristicaUsada:=Heuristica;
    ListaAlimentos.Primero:=nil;
    for i:=1 to n do
          ListaAlimentos.AgregarAlimento(alimentos[i]);
    ActualizarTotales;
    G:=0;
    H:=0;
    F:=0;
  end;
end;

procedure Tarbol.asignarHoraInicio;
begin
   horainicio:=time;
end;

procedure Tarbol.AsignarHoraFin;
begin
  horafin:= Time
end;

procedure Tarbol.CalcularTiempoDuracion;
begin
  TiempoDuracion:=Formatdatetime('nn:ss:zzz',(horafin - horainicio)) +
              'min:seg:milseg';
end;

//Función para calcular la variación de un nutriente n
function Tarbol.calcularDeltas1(i: ttiponutriente;estado:TEstadoComida): real;
var
  aux:real;
begin
    aux:=Estado.ListaAlimentos.maximoValorNutriente1(i);
    if aux = 0  // para evitar división entre 0
    then result:=0
    else result:=(CentroideDifuso(MetaGrsNutrientes[i]) -
              CentroideDifuso(estado.TotalesNutrientes[i])) / aux;
```

```
end;

//Procedimiento que implementa el cuarto algoritmo basado en la distancia
//antes del emparejamiento mejorada
procedure Tarbol.GenerarComidasSucesorasEficienteMejorado(comida:
                                        TEstadoComida);
var
 aux           :TOpEficientes;
 auxalimento    :talimento;
 ambas         :boolean;
 p             :TptrAlimento;
 operacion      :TOpEficientes;
begin
 calcularH(comida);
 p:=comida.ListaAlimentos.Primero;
 EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
 ambas:=false;
 if (formatfloat('0.0',EstadoAuxiliar.HDisminucion) =
    formatfloat('0.0',EstadoAuxiliar.HAumento))
 then ambas:=true
 else
    if(EstadoAuxiliar.HDisminucion > EstadoAuxiliar.HAumento)
    then aux:=reducir
    else aux:=aumentar;
 while p <> nil do
 begin
  if p^.cantidad <> 0 then
  begin
    auxalimento:=p^;
    for operacion := low(TOpEficientes) to high(TOpEficientes) do
    begin
      if ambas then aux:=operacion;
      case aux of
      reducir:
        if auxalimento.cantidad > 0
        then begin
             EstadoAuxiliar.ListaAlimentos.
             ModificarAlimento(p^.codalimento,p^.cantidad - p^.minimo);
            end;
      aumentar:
        if  not respetarmaximo or (auxalimento.cantidad < auxalimento.maximo)
        then
```

```
          begin
              EstadoAuxiliar.ListaAlimentos.
              ModificarAlimento(p^.codalimento,p^.cantidad + p^.minimo);
          end;
        end;
        // Actualizar el estado (calcular F y G )
        // verificar que tampoco este en comidas generadas
        ActualizarEstado(EstadoAuxiliar,EstadoInicial);
        if not ComidasVisitadas.ExisteEstado(EstadoAuxiliar) and
          not ComidasGeneradas.ExisteEstado2(EstadoAuxiliar)
        then
        begin
          inc(NroNodosGenerados);
          ComidasGeneradas.Agregar(EstadoAuxiliar);
          mensajeprogreso.progreso.StepBy(1);
          if NroNodosGenerados = maximoNodos
          then exit
        end
        else ;
        // Hay que reestablecer el estado y el alimento para realizar la
        // siguiente operación
        EstadoAuxiliar.copiarEstado(comida,TOpcopiar);
        p^:=auxalimento;
        if not ambas then break  //para que se haga sólo una operación
      end;
    end;
    p:=p^.proximo;
  end;
end;

//Función que permite calcular la variación de un nutriente n con respecto a la
//meta calórica
function Tarbol.calcularDeltas2(i:ttiponutriente;estado:TEstadoComida): real;
var
 aux1,aux2 :real;
begin
  aux2:=estado.ListaAlimentos.MaximoValorNutriente2(i);
  if aux2 = 0   // para evitar división entre 0
  then result:=0
  else
  begin
    aux1:=(MetaGrsNutrientes[i].Cord_a1 - MetaGrsNutrientes[i].Cord_c1) -
        (estado.TotalesNutrientes[i].Cord_a1 -
```

```
        estado.TotalesNutrientes[i].Cord_c1);
   if aux1 > 0
   then result:=aux1
   else
   begin
    aux1:=(estado.TotalesNutrientes[i].Cord_b1 -
         estado.TotalesNutrientes[i].Cord_d1) -
        (MetaGrsNutrientes[i].Cord_b1 - MetaGrsNutrientes[i].Cord_d1);
    if aux1 > 0
    then result:=aux1
    else result:=0
   end
  end
end;

//Procedimientos para liberar espacio de memoria
destructor TEstadoComida.destroy;
begin
      ListaAlimentos.Destroy;
      inherited destroy;
end;

destructor TlistaAlimentos.destroy;
var
      p,q:TptrAlimento;
begin

 p:=Primero;
 while p<> nil do
 begin
  q:=p;
  p:=p^.proximo;
  dispose(q);
 end;
 inherited destroy;
end;

destructor TListaEstadosGenerados.destroy;
var
      p,q:TEstadoComida;
begin
 p:=Primero;
 while p<> nil do
```

```
 begin
   q:=p;
   p:=p.Prox;
   q.destroy;
 end;
 inherited destroy;
end;

destructor TListaEstadosVisitados.destroy;
var
 p,q:TPtrEstadoVisitado;
begin
 p:=Primero;
 while p<> nil do
 begin
   q:=p;
   p:=p^.prox;
   dispose(q);
 end;
 inherited destroy;
end;

//Procedimiento para determinar cuando un alimento sobrepasa el máximo
//permitido
function TlistaAlimentos.AlgunoSobrepasaElMaximo: boolean;
var
 p:TptrAlimento;
begin
 p:=Primero;
 result:=false;
 while p <> nil do
 begin
   if p^.cantidad > p^.maximo
   then
   begin
         Result:=true;
         break
   end else
   p:=p^.proximo;
 end;

end;
```

```
procedure Tarbol.completarbarra;
var
      i:integer;
begin
 i := mensajeprogreso.progreso.Position;
 while i <= mensajeprogreso.progreso.max do
 begin
  mensajeprogreso.progreso.StepIt;
  inc(i)
 end
end;

procedure Tarbol.FinalizarBalanceo;
begin
 AsignarHoraFin;
 CalcularTiempoDuracion;
 completarbarra;
 mensajeprogreso.close
end;

procedure Tarbol.IniciarBalanceo;
begin
 asignarHoraInicio;
 mensajeprogreso.progreso.StepBy(1);
 mensajeprogreso.progreso.Max:=MaximoNodos;
 mensajeprogreso.show;
 mensajeprogreso.Refresh;
end;
end.


//Unidad que contiene los procedimientos para calcular la posibilidad y
//la necesidad de emparejamiento
unit operaciones;

interface

uses uarbol,sysutils,controls;

function inicializadifuso: tnrodifuso;
function suma(RPatron,Rdato:tnrodifuso):tnrodifuso;
function Emparejado(Meta,valor:tnrodifuso;var posibilidad,
                    necesidad:real):boolean;
```

```
function EstanBalanceadosNutrientes(posibilidades,
                                    necesidades:tValorPosNec):boolean;
function nitidotodifuso(Rgrm,RPorcN,RPorcD: real):tnrodifuso;
function CentroideDifuso(nrodifuso: tnrodifuso): real;
function Calculo_Necesidad(a1,b1,c1,d1,a2,b2,c2,d2,Importancia:Real):real;
function Calculo_Posibilidad(a1,b1,c1,d1,a2,b2,c2,d2,importancia:Real):real;
function ValorDifusoPromedio(nro:tnrodifuso):real ;
function THeuristicaToString(Heuristica:THeuristica):string ;
function TAlgoritmoToString(algoritmo:TTipoAlgoritmo):string ;
function difusotostring(nrodifuso: tnrodifuso): string;
function stringtodifuso(nrodifuso:string ): tnrodifuso;
function Multiplicar(RDifuso:tNroDifuso; RCant:Real):tnrodifuso;
function fractostr(numero: real): string;
function Calcularedad(fechanac: tdate): integer;


implementation

//Función para calcular la fracción de una medida
function fractostr(numero: real): string;
var
   nveces:byte;
   fracstr:string;
begin
   if frac(numero)=0
   then result:=inttostr(trunc(numero))
   else
   begin
     nveces:=trunc(frac(numero) / 0.125);
     case nveces of
     1: fracstr:='1/8';
     2: fracstr:='1/4';
     3: fracstr:='3/8';
     4: fracstr:='1/2';
     5: fracstr:='5/8';
     6: fracstr:='3/4';
     7: fracstr:='7/8';
     end;
     if int( numero) = 0
     then result:=fracstr
     else result:=inttostr(trunc(numero))+' y '+fracstr;
   end
end;
```

```
//Función para calcular la edad de un usuario
function Calcularedad(fechanac: tdate): integer;
begin
   result:=trunc((trunc(date) - trunc(fechanac))/365)
end;

//Función para multiplicar un número difuso por un valor entero
function Multiplicar(RDifuso:tNroDifuso; RCant:Real):tnrodifuso;
Begin
With result Do {Coordenadas de la meta calórica}
Begin
 Cord_a1:=RDifuso.Cord_a1*Rcant;
 Cord_b1:=RDifuso.Cord_b1*Rcant;
 Cord_c1:=RDifuso.Cord_c1*Rcant;
 Cord_d1:=RDifuso.Cord_d1*Rcant;
End;
End;

//Función para convertir una cadena que representa un número difuso
function stringtodifuso(nrodifuso:string ): tnrodifuso;
var
   central,varianza1,
   varianza2,extremo :real;
   posi :byte;
begin
   posi:=pos('±',nrodifuso);
   if posi = 0
   then
   begin
        central:=strtofloat(nrodifuso);
        with result do
        begin
             Cord_a1:=central ;
             Cord_b1:=central ;
             Cord_c1:=0;
             Cord_d1:=0;
        end;
   end
   else
   begin
        central:=strtofloat( copy(nrodifuso,1,posi - 2) );
        delete(nrodifuso,1,posi);
        posi:=pos('±',nrodifuso);
```

```
        varianza1:=strtofloat( copy(nrodifuso,1,posi - 2));
        delete(nrodifuso,1,posi);
        varianza2:= strtofloat( nrodifuso);
        with result do
        begin
              Cord_a1:=central - varianza1;
              Cord_b1:=central + varianza1;
              Cord_c1:=varianza2;
              Cord_d1:=varianza2;
        end;
   end
end;

//Función para convertir un número difuso a string
function difusotostring(nrodifuso: tnrodifuso): string;
var
   central,varianza1,
   varianza2,extremo :real;

begin
   central:=(nrodifuso.Cord_a1 + nrodifuso.Cord_b1) / 2;
   result:= formatfloat('0.0',central);
   result:= formatfloat('0.0',central);
   if  (nrodifuso.Cord_c1 <> 0) and (nrodifuso.Cord_d1 <> 0) then
   begin
      varianza1:= central - nrodifuso.Cord_a1;
      varianza2:=  nrodifuso.Cord_c1;
      result:=result + ' ± ' + formatfloat('0.0',varianza1) +
            ' ± ' + formatfloat('0.0',varianza2 );
   end
end;

//Función para evaluar el tipo de heurística
function THeuristicaToString(Heuristica:THeuristica):string ;
begin
   case Heuristica  of
   ThNutricional  :result:='Nutricional' ;
   ThNecesidad    :result:='Necesidad' ;
   ThDistancia    :result:='Distancia';
   end;
end;

//Función para evaluar el tipo de algoritmo
```

```
function TAlgoritmoToString(algoritmo:TTipoAlgoritmo):string ;
begin
   case algoritmo of
   TAEficiente     :result:='Eficiente' ;
   TAIneficiente   :result:='Ineficiente' ;
   TAEficienteMejorado :result:='Eficiente mejorado';
   end;
end;

//Función para convertir un número nítido a difuso
function nitidotodifuso(Rgrm,RPorcN,RPorcD: real):tnrodifuso;
Var
Nucleo, Dispersion:Real;
Begin
Nucleo:=Rgrm*(RPorcN/100);
Dispersion:=Rgrm*(RPorcD/100);
With result Do
 begin
   Cord_a1:=Rgrm-Nucleo;
   Cord_b1:=Rgrm+Nucleo;
   Cord_c1:=Dispersion;
   Cord_d1:=Dispersion;
 end;
end;

//Función para sumar las coordenadas de dos números difusos,
//los cuales representan la cantidad total del nutriente n en una comida
//y su meta calórica respectivamente
function suma(RPatron,Rdato:tnrodifuso):tnrodifuso;
Var
a1,b1,c1,d1,a2,b2,c2,d2,SumCorda,SumCordb,SumCordc,SumCordd:Real;
begin
With RPatron Do {Coordenadas de la meta calórica}
begin
 a1:=Cord_a1;
 b1:=Cord_b1;
 c1:=Cord_c1;
 d1:=Cord_d1;
end;
With Rdato Do
begin
 a2:=Cord_a1;
 b2:=Cord_b1;
```

```
 c2:=Cord_c1;
 d2:=Cord_d1;
end;
SumCorda:=a1+a2;
SumCordb:=b1+b2;
SumCordc:=c1+c2;
SumCordd:=d1+d2;
With result Do
begin
 Cord_a1:=SumCorda;
 Cord_b1:=SumCordb;
 Cord_c1:=SumCordc;
 Cord_d1:=SumCordd;
end;
end;

//Inicializa las coordenadas de un número difuso
function inicializadifuso: tnrodifuso;
begin
   with result do
   begin
         Cord_a1:=0;
         Cord_b1:=0;
         Cord_c1:=0;
         Cord_d1:=0;
   end
end;

//Función para calcular la necesidad de emparejamiento entre dos números
//difusos, los cuales representan la cantidad total del nutriente en una comida
//y su meta calórica respectivamente
function Calculo_Necesidad(a1,b1,c1,d1,a2,b2,c2,d2,Importancia:Real):real;
Var
  Necesidad,Import,NA,NB:Real;
Begin
NA:=0.0;
NB:=0.0;
Import:=1-Importancia;
{Parte donde se determina si la necesidad es igual a 1}
if ((a1<=a2-c2) and (b1>=b2+d2)) or ((a1=a2) and (b1=b2) and (c1=0) and
   (c2=0) and (d1=0) and (d2=0))
then Necesidad:=1
else{Parte donde se determina si la necesidad es igual a 0}
```

```
if(a2<a1-c1) or (b2>b1+d1) or (((c1<>0) or (c2<>0)) and (a1<a2)) or
  (((d1<>0) or (d2<>0)) and (b1<b2))
then if (c1<>0) and (d1<>0) and (c2=0) and (d2=0) and (a2<b1) and (b1<b2)
    then  Necesidad:=(b1-b2+d1)/(d1+d2) {En este caso NA=1}
    else if (a1<a2) and (a2<b1) and ((b1<b2) or (b2<b1))
        then Begin {Cálculo de la necesidad cuando NA y NB son
                    diferentes de cero}
             If (c1<>0) and (c2<>0) and (d1<>0) and (d2<>0)
             Then Begin
                  NA:=(a2-a1+c1)/(c1+c2);
                  NB:=(b1-b2+d1)/(d1+d2);
                 End
             Else Necesidad:=0;
             if NA<NB
             then Necesidad:=NA
             else Necesidad:=NB;
            End
         else if ((b1=b2) and (a1<a2)) or (b1<b2)
             then if (d1=0) and (d2=0)
                  then Necesidad:=0
                  else Necesidad:=(b1-b2+d1)/(d1+d2)
             else if (a1=a2) and (b2<b1)
                  then Necesidad:=(a2-a1+c1)/(c1+c2)
                  else Necesidad:=0
else {Parte donde se calcula la necesidad cuando 0<NA<1 y 0<NB<1}
     Begin
      {Condiciones que se chequean para calcular NA}
      If (c1=0) and (c2=0)
      then if a1<=a2
           then NA:=1
           else NA:=0
      else if (a1<a2) and ((c2=0) and (d2=0))
           then NA:=1 {Se considera en la condición cuando la
                       necesidad es 0}
           else NA:=(a2-a1+c1)/(c1+c2);
      {Condiciones que se chequean para calcular NB}
      if (d1=0) and (d2=0)
       then if b1>=b2
            then NB:=1
            else NB:=0 {Se considera en la condición cuando la
                        necesidad es 0}
       else if ((a2<a1) and ((c2=0) and (d2=0))) or ((a1=a2) and
             (c2<>0) and (d2<>0))
```

```
            then NB:=1
            else NB:=(b1-b2+d1)/(d1+d2);
         if NA<NB
         then Necesidad:=NA
         else Necesidad:=NB;
        End;
if Import>Necesidad
then result:=Import
else result:=Necesidad;
End;

//Función para calcular la posibilidad de emparejamiento entre dos números
//difusos, los cuales representan la cantidad total del nutriente n en una comida
//y su meta calórica respectivamente
function Calculo_Posibilidad(a1,b1,c1,d1,a2,b2,c2,d2,importancia:Real):real;
Var
Posibilidad,Import:Real;

Begin
Import:=1-Importancia;
Posibilidad:=0.0;
If ((b1+d1<a2-c2) or (b2+d2<a1-c1))
Then Posibilidad:=0
Else If ((a2<=b1) and (b1<=b2)) or ((a1<=b2) and (b2<=b1))
   Then Posibilidad:=1
   Else
      If (b1<a2) and (a1-c1<a2-c2) or (b1<a2)
       Then Posibilidad:=1-((a2-b1)/(c2+d1))
       Else
        If ((b2<a1) and (a2-c2<a1-c1)) or (b2<a1)
         Then Posibilidad:=1-((a1-b2)/(c1+d2))
         Else;
If Import>Posibilidad
Then result:=Import
Else result:=Posibilidad;
End;

//Función para determinar si un nutriente está emparejado con su meta calórica
function Emparejado(Meta,valor:tnrodifuso;var posibilidad,
              necesidad:real):boolean;
begin
   posibilidad:=Calculo_Posibilidad(Meta.Cord_a1,Meta.Cord_b1,Meta.Cord_c1,
                        Meta.Cord_d1,valor.Cord_a1,valor.Cord_b1,
```

```
                        valor.Cord_c1,valor.Cord_d1,1);

   necesidad :=Calculo_Necesidad(Meta.Cord_a1,Meta.Cord_b1,Meta.Cord_c1,
                        Meta.Cord_d1,valor.Cord_a1,valor.Cord_b1,
                        valor.Cord_c1,valor.Cord_d1,1);
   if (posibilidad=1) and (necesidad>=0.7)
   then  result:=true
   else  result:=false;
end;

//Determina si un nutriente está balanceado es decir, si su posibilidad es
//igual a 1 y su necesidad es mayor o igual a 0.7
function EstanBalanceadosNutrientes(posibilidades,
                        necesidades:tValorPosNec):boolean;
var
   i:ttiponutriente;
   menorpos,
   menornec :real;
begin
   menorpos:=2;
   menornec:=2;
   for i :=low(ttiponutriente)  to high(ttiponutriente)  do
   begin
        if posibilidades[i] < menorpos
        then menorpos:=posibilidades[i];
        if necesidades[i] < menornec
        then menornec:=necesidades[i];
   end;
   if (menorpos=1) and (menornec>=0.7)
   then  result:=true
   else  result:=false;
end;

//Calcula el centroide o punto central de un número difuso
function CentroideDifuso(nrodifuso: tnrodifuso): real;
begin
   result:=(nrodifuso.Cord_a1 + nrodifuso.Cord_b1) / 2;
end;

end.
```