UNIVERSIDAD NACIONAL

"SANTIAGO ANTÚNEZ DE MAYOLO"

ESCUELA DE POSTGRADO

"OPTIMIZACION DIFUSA EN LA FERTILIZACIÓN AGRICOLA"

Tesis para optar el grado de Doctor

en Ciencia e Ingeniería de la Computación

ESMELIN NIQUIN ALAYO

Asesor: Dr. EDMUNDO RUBEN VERGARA MORENO

Huaraz – Ancash - Perú

2011

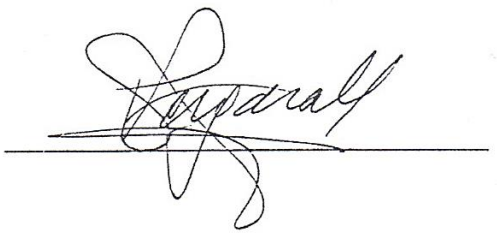
N° Registro: TD001

MIEMBROS DEL JURADO

Doctor José Del Carmen Ramírez Maldonado Presidente

Doctor Jesús Edilberto Espinola Gonzales Secretario

Doctor Edmundo Rubén Vergara Moreno Vocal



ASESOR

Doctor **Edmundo Rubén Vergara Moreno**

AGRADECIMIENTO

- En primer lugar a Dios, por tenerme con salud y brindarme excelentes oportunidades en mi vida.
- A mis hijos Renzo y André por permitir tomar parte del tiempo que debería dedicárselos como padre a ellos, así como les agradezco por las muestras de ilimitado afecto y amor hacia mi persona.
- A mi sobrino Marck Arturo a quien le deseo toda clase de éxitos y ventura en su vida futura, el mismo que me apoyó como desarrollador del programa computacional.
- Un reconocimiento especial a la señorita Cecilia Blanco, invalorable amiga quien con sus palabras de aliento contribuyó a la realización de este sueño tan anhelado.
- A mi asesor, Dr. Edmundo Vergara Moreno, excelente amigo quien con sus acertadas orientaciones y apoyo permanente hicieron posible la realización de la presente tesis.
- Al Dr. Jesús Espinola Gonzáles, Dr. Ángel Cobo Ortega y a todos los profesores del doctorado, quienes nos brindaron su apoyo y conocimientos para lograr culminar con éxito este reto personal y profesional.
- Al personal administrativo de la Escuela de Postgrado de la UNASAM por su dinamismo e identificación con la institución, así como por la celeridad en los trámites administrativos establecidos.

A mis hijos: Renzo y André, mis más preciados tesoros, para quienes espero ser un ejemplo y un buen guía en este largo camino que tienen a futuro y logren perseguir, alcanzar y realizar todos sus sueños e ilusiones y ser personas de bien.

A mi madre Flor, ejemplo de vida, amor y lucha continua, a mi padre Segundo, a mis hermanos, hermanas, a toda mi familia, amigos y amigas que de una u otra manera me brindaron su apoyo constante e incondicional para conseguir el objetivo trazado.

INDICE

Pág.

“ OPTIMIZACIÓN DIFUSA EN LA FERTILIZACIÓN AGRÍCOLA”

Esmelin Niquín Alayo, email: esmelinn@yahoo.es, esmelin.niquin@gmail.com
Profesor Principal del departamento de Matemáticas de la Facultad de Ciencias
Escuela de Postgrado de la Universidad Nacional “Santiago Antúnez de Mayolo”
Doctorado en Ciencia e Ingeniería de la Computación.

RESÚMEN

La aplicación de los conceptos teóricos de la lógica difusa ha permitido en este trabajo la modelización matemática sobre fertilización de terrenos agrícolas, considerando los costos difusos y con la ayuda de los lenguajes de programación y herramientas informáticas se ha logrado diseñar un software denominado FERTIDIF que coadyuvará a la toma de una decisión adecuada sobre el uso de fertilizantes y nutrientes que se requiere para un determinado cultivo.

Se ha resuelto el problema de fertilización difuso (PFD), mediante la adaptación de metodologías de solución propuestos por Lai-Hwang y Leberling, métodos que transforman un problema de programación difuso con costos difusos en un problema de optimización multiobjetivo, el mismo que mediante la ayuda del software FERTDIF es fácilmente resuelto, proporcionando a los especialistas en fertilización agrícola tomar decisiones sobre cantidades de fertilizantes a usar así como determinar el costo mínimo del proceso de fertilización de un cultivo determinado.

Palabras clave: Optimización Lineal Difusa, fertilización agrícola, programación lineal multiobjetivo.

“FUZZY PROGRAMMING IN THE FERTILIZATION AGRICULTURAL”

Esmelin Niquín Alayo, email: esmelinn@yahoo.es, esmelin.niquin@gmail.com
Profesor Principal del departamento de Matemáticas de la Facultad de Ciencias
Escuela de Postgrado de la Universidad Nacional “Santiago Antúnez de Mayolo”
Doctorado en Ciencia e Ingeniería de la Computación.

ABSTRAC

The application of theoretical of fuzzy logic in this paper has allowed mathematical modelling on fertilization if agricultural land, considering fuzzy cost and with the help of programming languages and tools has been designing software called FERTIDIF this will help to making a decision about the use of fertilizers and nutrients required for a particular crop.

We have solved the problem of fuzzy fertilization (PFD), by adapting methods of solution proposed by Lai-Hwang and Leberling, methods that transform a fuzzy problemwith fuzzy costs in a multi objective optimizacion problem, the same as using the FERTDIF software help is easily solved by providing agricultural fertilization specialist make decisions about amounts of fertilizer to use determination well as the minimum cost the process of fertilization of a given crop.

Key words: Linear Diffuse optimization, agricultural fertilization, linear programming I multitarget.

I INTRODUCCIÓN

En toda sociedad el bienestar social de las personas que lo conforman, es el fin supremo. Una de los componentes de este bienestar es la buena salud y esto se consigue con una adecuada alimentación. Por su parte se puede conseguir una adecuada alimentación con productos agrícolas y ganaderos de buena calidad. En síntesis la buena alimentación de las personas del mundo depende directamente de la calidad y cantidad de los productos que se consumen.

Con el incremento acelerado de la población a nivel mundial y la carencia de tierras vírgenes que quedan por incorporar a la agricultura, el problema de la alimentación del hombre tiene que ser resuelto mejorando el rendimiento de la productividad, rendimiento basado al uso de semillas seleccionadas, de abonos adecuados y el aumento de los regadíos.

En el proceso de la fertilización agrícola en el Perú y en el mundo dicho proceso se realiza solamente utilizando el sentido común, comprando fertilizantes preparados como Urea, Fosfatos entre otras y en base a la experiencia se procede a la fertilización del terreno considerando el tipo de cultivo. Sin embargo esto no es lo más adecuado y se podría estar haciendo una inversión que no justifique la productividad del terreno, teniendo en cuenta que los costos de los fertilizantes son cambiantes y la disponibilidad de los recursos también está sujeto a la disponibilidad los mismos en el mercado y aun mas que se debe pensar en obtener la máxima productividad como la mínima inversión, esto nos induce a pensar que nos encontramos ante un problema de Programación Lineal Difusa con funciones a optimizar multiobjetivo.

Es en este contexto que el presente trabajo pretende dotar a los encargados de la producción agrícola de una gama de posibles alternativas factibles con la finalidad de abonar los terrenos agrícolas con los fertilizantes adecuados, considerando que los costos no deben ser demasiados altos y con lo cual se pretende que como consecuencia se obtengan productos de óptima calidad, adicionalmente se diseñara un programa que permitirá hacer los cálculos necesarios a fin de optimizar tiempo en cuanto se refiere a la solución de un modelo matemático de optimización (Programación Lineal Difusa).

La estructura sobre el cual se desarrolla el presente trabajo se inicia con una revisión bibliográfica sobre fertilización de cultivos los Conjuntos Difusos: definición, propiedades, operaciones, y algunos otros conocimientos sobre esta teoría difusa, posteriormente se usaran modelos de Programación Matemática (Modelos de Dieta) clásicos, los mismos que mediante el uso de conceptos teóricos sobre Conjuntos Difusos y Programación Difusa en nuestro caso será transformado en un problemas de fertilización difusa considerando que los costos de los fertilizantes son cambiantes en el tiempo y nos interesa determinar las cantidades optimas de fertilizantes a usar en un proceso de fertilización así como es menester tener una idea del costo del proceso de fertilización de un determinado cultivo, aquí también se hara una descripción de las metodologías de solución a un problema de programación multiobjetivo, propuestas por Lai-Hwang y Leberling.

La parte fundamental del trabajo que corresponde a los resultadoses esta enfocado a realizar la adaptación de las metodologías de solución planteadas por Lai y Hwang así como por Leberling a nuestro Problema de Fertilización Difusa (PFD)

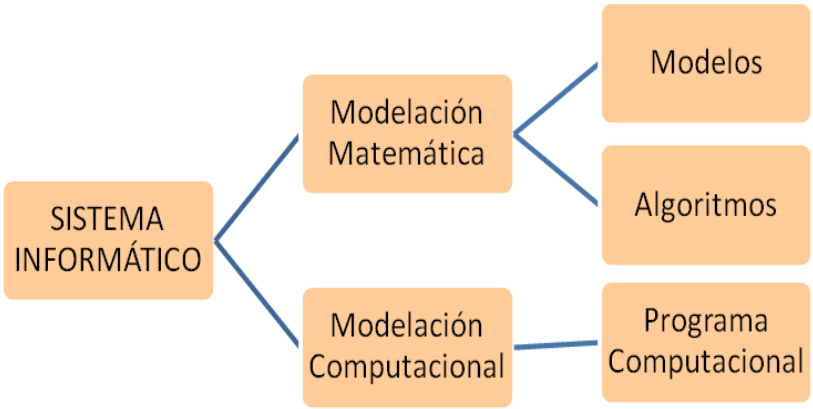
para un determinado cultivo, en determinados terrenos agrícolas. Toda esta teoría se ejemplicara con un ejemplo práctico de aplicación el mismo que haciendo uso del software interactivo FERTDIF diseñado para la presente investigación, resolverá el problema planteado de una manera rápida y con soluciones confiables.

1.1 OBJETIVOS

1.1.1.- Objetivo General.- Definir un modelo matemático de optimización difusa que permita determinar de manera óptima tipos y cantidad de fertilizantes de tal manera que su costo sea mínimo y su rendimiento sea máximo.

1.1.2.- Objetivos Específicos

- Acopio de información de fertilizantes
- Creación del modelo de optimización difusa
- Crear el programa computacional que implemente el modelo y su algoritmo de solución.



- Validar el sistema informático mediante comparación con algunos sistemas informáticos existentes en el mercado y que resuelven problemas de optimización clásicos.

1.1.3.- Hipótesis

El uso de un modelo matemático de optimización Difusa mejora la productividad Agrícola minimizando los costos de fertilización.

1.1.4.- Variables

Variable Independiente

Información de fertilizantes

Variable Dependiente

Modelo de Optimización Difuso para fertilización

II. MARCO TEÓRICO

2.1 Antecedentes

En nuestro país actualmente no se encuentra información disponible, sobre la existencia de estudios y software, que permita realizar una planificación óptima para realizar la fertilización de cultivos y terrenos agrícolas.

El problema de la selección de abonos adecuados [38], para mejorar el rendimiento de la tierra y al mismo tiempo sin perjudicar el ambiente en general y manteniendo los componentes del fruto aptos para el consumo humano, no es muy fácil porque depende de las componentes químicas del suelo [40], del agua con la que se riega y del tipo de plantas a cultivar [39].

La selección de abonos adecuados también debe hacerse de tal manera que el costo sea mínimo, en tal sentido éste constituye una variante del *problema de la dieta* (PD) [5], pues se trata de seleccionar la dieta (combinación de abonos) de menor costo pero satisfaciendo los requerimientos nutricionales de las plantas que se cultivan en las condiciones de la tierra y de regadío existentes.

Por otro lado dentro de este contexto los datos con las que se trabajan no son exactos sino aproximaciones a la realidad. Para estos casos la teoría de conjuntos difusos proporciona una mejor herramienta para su representación y la programación lineal difusa proporciona las técnicas de solución de problemas de programación con datos imprecisos. El problema de la dieta en el que algunos datos son imprecisos se denomina el *problema de la dieta difusa* (PDD) [5],

La solución del problema de planificación de fertilización en terrenos agrícolas implica la preparación por así decirlo de una fertilización (abonamiento) que satisfaga los requerimientos nutricionales para un adecuado desarrollo de las plantas con el objetivo de tener un mejor producto o mayor productividad pero con el menor costo posible.

Este problema corresponde a un problema clásico de dietas, dentro de un contexto socio económico no estable, referente específicamente a la variación en los precios de los fertilizantes en tiempos prolongados.

En tal sentido, el modelo correspondiente es el del Problema de Dietas, con la modificación de que los coeficientes de la función objetivos son costos imprecisos representados mediante números difusos [4, 19]. Modelo que será resuelto inicialmente mediante los métodos de la programación lineal difusa y posteriormente estos métodos de solución serán implementados en un programa

computacional usando un lenguaje de programación para crear el interfaz con el cual interactuara el usuario de este software.

El problema de la dieta fue planteado por Jerome Cornfield en 1941 y resuelto por primera vez mediante la programación lineal por Stigler como consta en [12, 14], sin embargo existe poca bibliografía sobre su enfoque difuso, entre ellos se tiene [5,6], pese a su gran importancia dentro de las ciencias económicas como por ejemplo en el análisis insumo-producto, teoría del consumo, etc.

El problema de la Fertilización (PF) tiene como meta determinar la combinación más económica de productos fertilizantes de tal manera que satisfaga las necesidades nutricionales mínimas o máximas requeridas. Para expresar simbólicamente, consideremos el conjunto de los fertilizantes $A = \{A_1, ..., A_n\}$ el conjunto de los nutrientes $N = \{N_1, N_2, ..., N_m\}$ y los siguientes símbolos:

c_i = el costo del fertilizante A_j ; j = 1, ….,n

x_j = la cantidad del fertilizante A_j que se debe incluir en el proceso de fertilización; j = 1, ….,n

a_{ij} = la cantidad del nutriente N_i contenido en el fertilizante A_j, $i = 1, ..., m$, $j = 1,...,n$

p_i = cantidad mínima requerida del nutriente N_i, $i = 1,...,m$.

P_i = cantidad máxima requerida del nutriente N_i, $i = 1,...,m$.

m_i = cantidad mínima requerida del fertilizante A_j, $j = 1,...,n$.

M_i = cantidad máxima requerida del fertilizante A_j, $j = 1,...,n$.

Luego, el modelo matemático del PD formulado como un problema de programación lineal es la siguiente:

$$\text{Min } \Sigma_{j=1,\ldots,n}\, c_j x_j$$

Sujeto a:

$$p_i \le \Sigma_{j=1,\ldots,n}\, a_{ij} x_j \le P_i, \quad i = 1, \ldots, m \qquad (1)$$

$$m_j \le x_j \le M_j, \qquad j = 1, \ldots, n$$

Las *m* primeras restricciones indican que la cantidad total de nutrientes en el proceso de fertilización no debe ser inferior ni superior a las cantidades mínimas y máximas permitidas, mientras que las otras *n* restricciones acotan la cantidad de cada fertilizante por las cantidades mínimas y máximas permitidas.

En el modelo (1) (versión clásica) se considera que los parámetros son completamente conocidos, sin embargo en la realidad mayormente solo se conocen vagamente, digamos aproximadamente. Por ejemplo se suele tener la información de que el fertilizante A_i cuesta "alrededor de n unidades monetarias" esta información pueden interpretarse de muchas maneras, o que existe mucha probabilidad de que dicho alimento cueste exactamente n unidades monetarias, o que el precio posiblemente es n – h ó n+ h unidades monetarias. Esto puede ocurrir con respecto a los requerimientos mínimos y máximos de los fertilizantes y nutrientes, o respecto a la cantidad de nutrientes que contiene cada fertilizante o todos los casos. El problema de fertilización con estas características lo denominaremos el problema de fertilización difuso (PFD).

La representación de los precios de los productos que no son estables en periodos de tiempo prolongado, es mediante los números difusos [1, 4, 18, 20, 32], que en su forma más simple se representa utilizando una terna de números, denominado

números difusos triangulares. Cada costo difuso ($\tilde{c}_i$) (número difuso triangular que es definido como un conjunto difuso, convexo normal y acotado en la recta real) está descrito por una función de pertenencia seccionalmente lineal como se muestra en la figura 1.

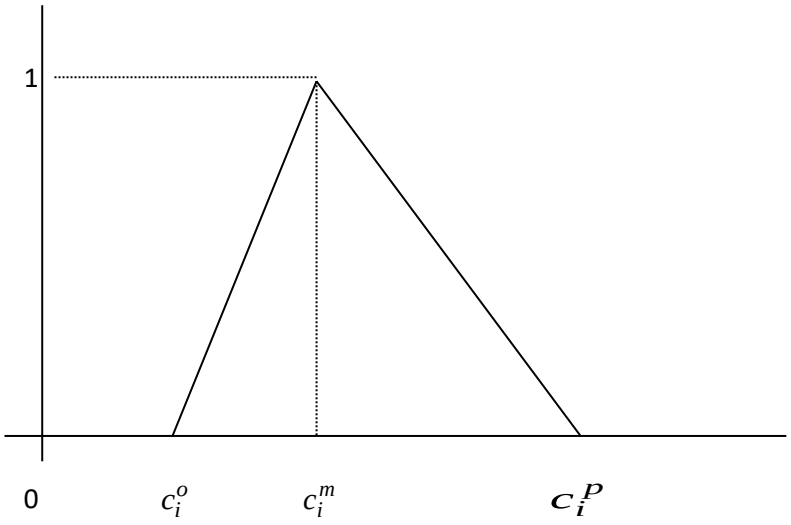


Figura 1: Número difuso triangular $\tilde{c}_i$.

Luego el modelo del problema de fertilización difusa (PFD), con costos y restricciones difusas, se enuncia de la siguiente forma:

$$\text{Min} \sum_{i=1}^{n} \tilde{c}_i x_i$$

s.a.

$$p_i \leq_f \Sigma_{j=1,\ldots,n}\, a_{ij} x_j \leq_f \mathrm{P}_i, \quad i = 1, \ldots, m \qquad (2)$$

$$m_j \leq x_j \leq \mathrm{M}_j, \qquad j = 1, \ldots, n$$

El símbolo "$\leq_f$" representa la imprecisión de la cantidad de los recursos disponibles (que también se utiliza el símbolo $\leq$.) y significa que el decisor

admite el incumplimiento de las restricciones (entendida como ≤) naturalmente con un margen de tolerancia apropiado

Debo indicar que el autor de este trabajo cuenta con la experiencia necesaria para resolver con éxito el reto planteado ya que anteriormente desarrollo su tesis de maestría en lo que corresponde a fertilización fuzzy considerando que las restricciones eras inciertas, en el presente trabajo se complica el problema pues se asume que los costos de los recursos (fertilizantes) son inciertos en esta realidad cambiante y como consecuencia de ello se tendrá que hacer un trabajo que desembocará en un problema de programación multiobjetivo [2, 11,35].

2.2. Bases teóricas

2.2.1 FERTILIZACIÓN AGRÍCOLA

2.2.1.1 Fertilidad del suelo

Los conceptos teóricos han sido extraídos de los libros [39,41,42]. Uno de los problemas que aquejan la productividad en el sector agrario está relacionado a la disponibilidad de suelos capaces de soportar diferentes cultivos con un alto índice de productividad.

La capacidad del suelo varía de acuerdo con las diferencias en la formación del suelo y por consiguiente podemos afirmar que existen diferentes tipos o clases de suelos, debemos tener en mente considerar que los suelos no es materia inerte, sino dinámica y está en constante transformación, las cuales pueden ser físicas, químicas y biológicas, estos cambios ocurren fundamentalmente en la capa superficial (25 cm. – 30 cm.).

Las transformaciones provocan un cambio gradual en las propiedades de los suelos, lo cual se refleja en reacciones que pueden derivar en ser adecuados o

inadecuados para la formación de las raíces y por consiguiente en una adecuada capacidad nutricional de la planta a fin de obtener adecuado crecimiento y productividad óptima.

Las transformaciones fundamentalmente se dan a través de:

- Fenómenos climáticos como temperatura, humedad y viento.
- Prácticas agrícolas como fertilización, encalado y aradura de los terrenos de cultivo.

Las actividades más importantes para un crecimiento adecuado de las plantas son:

- Proporcionar una estructura de suelo, con una adecuada proporción balanceada de materiales sólidos, agua y oxigeno.
- Conservar una buena estructura del suelo.
- Suministrar nutrientes oportunos y adecuados.
- Evitar reacciones extremas del suelo agrícola.

La carencia de una o más de estas actividades puede traer como consecuencia el empobrecimiento de los suelos y por consiguiente estos se convertirán en terrenos eriazos inadecuados para la producción óptima de cultivos.

Así por ejemplo, las cantidades de nutrientes disponibles para la planta son de vital importancia en el desarrollo y producción de la misma. Estas cantidades pueden determinarse a través de análisis del suelo y tejidos de la planta, estos análisis serán la base para las recomendaciones en cuanto se refiere a la aplicación de fertilizantes, lo cual traerá como consecuencia compensar las deficiencias de nutrientes del suelo.

Para aplicar la fertilización a un nivel adecuado de disponibilidad, es importante seleccionar los tipos de fertilizantes en cantidades adecuadas y adoptar los procedimientos de aplicación recomendados para evitar pérdidas de nutrientes.

A través de una fertilización regular, es posible la obtención de residuos vegetales, como raíces y paja lo cual contribuye a suministrar en forma indirecta ingentes cantidades de materia orgánica. Estos residuos orgánicos y los residuos directos de los fertilizantes mejoran la estructura del suelo y con una adecuada rotación y fertilización de cultivos, puede establecerse y conservarse la fertilidad del suelo.

En este trabajo se abordará fundamentalmente la parte de fertilización del suelo, para lo cual se hará uso de un modelo matemático el cual será resuelto haciendo uso de la programación lineal difusa.

Debo indicar que en el presente trabajo se darán las recomendaciones sobre fertilización de cultivos, los cuales serán fertilizados en las cantidades y/o proporciones que se obtengan de la resolución del modelo matemático.

2.2.1.2 Suelos agrícolas

Las plantas crecen en la capa superficial de la tierra, el cual permite un medio adecuado para las raíces.

A diferencia de las rocas, el suelo es la superficie suelta de la tierra, el suelo es un cuerpo naturalmente desarrollado en que tiene lugar procesos físicos, químicos y biológicos.

Desde el punto de vista de la agricultura**, el suelo es el medio donde crecen las plantas.** El suelo es también el almacén de donde las plantas toman las sustancias nutritivas, agua y aire para crecer y desarrollar.

Los suelos se han formado a partir de materias madres (rocas) por la influencia de los procesos de desintegración y traslocación durante largos periodos.

Las actividades físicas y químicas desintegran las rocas y los fragmentos de estas, transformándolas en minerales gruesos y limos que vienen a ser lo que posteriormente se denomina suelo.

La desintegración física del suelo es causada por las siguientes acciones:

- Cambios diarios estacionales de temperatura.
- Hielos y deshielos.
- Erosión.
- Prácticas agrícolas.
- Acción de plantas y animales.

La descomposición química del suelo es el resultado de las siguientes acciones:

- Disolución de las materias solubles.
- Reacciones de las partículas sólidas con la solución del suelo.
- Reacciones de los constituyentes con el aire.
- Reacciones de los constituyentes con las raíces.

Los factores climáticos en la formación del suelo son:

- Precipitación, las altas precipitaciones originan las corrientes de agua, estas arrastran las partículas medias y finas de la superficie y las depositan en terrazas o en terrenos bajos.
- Acción del viento, éste arrastra las partículas minúsculas del suelo, según la dirección que lleva, abandonándolas cuando reduce suficientemente su velocidad.
- Temperaturas, como resultado de las bajas temperaturas se forman las masas de hielo, estas transportan parte del material madre encerrado en ellas, dejándolo en el lugar donde el hielo se derrite. Los terremotos y los deslizamientos de tierra también influyen localmente en la formación de los suelos.

La formación de los suelos ha pasado por diferentes periodos de desarrollo, estos han transformado los materiales primarios en suelos a través de las siguientes etapas:

- Materias madres
- Principios de desintegración de las materias madres en la parte superior del suelo.
- Materias madres parcialmente desintegrados
- Suelo actual.

Las componentes del suelo determinan sus propiedades o características, así tenemos:

- El volumen del suelo ideal consta de 50% de materias sólidas (45% de materias no orgánicas y 5% de materias orgánicas). Además consta de 25% de agua y 25% de aire.
- Los cuatro componentes se encuentran subdivididos y mezclados de tal manera que el agua y el aire llenan los poros que quedan entre las partículas sólidas.
- El contenido del agua puede variar considerablemente, de allí que la lluvia y la irrigación del terreno son factores importantes.

El agua del suelo desempeña las siguientes funciones:

- Satisface los requerimientos de humedad de la planta.
- Sirve como vehículo de transporte de los elementos nutritivos.
- Disuelve los nutrientes, formando soluciones que son absorbidas por las raíces.
- Controla el volumen de aire en el suelo.
- Controla la fluctuación de la temperatura en el suelo.

El aire consiste en una mezcla de gases, estos gases llenan parte de los poros, parte de los gases pueden ser absorbidos por las raíces y los microorganismos o pueden ser disueltas en la solución del suelo.

La textura del suelo se refiere a la composición del mismo en grupos de partículas de diferentes tamaños las que se identifican como se muestra en la Tabla 1.

Tabla 1: Textura del suelo

Descripción del tamaño	Nombre común	Identificación
Muy grueso	Piedras y grava	A simple vista
Grueso	Arena	A simple vista
Fino	Limo	Con microscopio
Muy fino	Arcilla	Con microscopio

Fuente: Facultad de Ciencias Agrarias - UNASAM

La textura del suelo, en relación con sus propiedades agrícolas tiene la importancia siguiente:

Suelos arenosos.- Retienen poca humedad y tienden a secarse, tienen poca capacidad para retener los nutrientes. Poseen por naturaleza bajo índice de fertilidad, alta porosidad y rápida percolación (no retiene agua). Es necesario aplicar frecuentemente materiales orgánicos y nutrientes inorgánicos, son fácilmente trabajables.

- **Suelos francos y francos limosos.-** Poseen buena penetración y retiene bien el agua y los nutrientes, su fertilidad natural va de media a alta, se pierde poco agua y nutrientes por lixiviación, los mejores suelos están ubicados en este rango.
- **Suelos franco arcillosos y arcillosos.-** Tienen poca penetración de agua , retienen grandes cantidades de humedad, parte de la cual no esta disponible para la planta, la perdida de nutrientes por percolación en estos suelos es muy reducida, carece de porosidad y contiene poco aire, sus

problemas más saltantes son el apelmazamiento, la formación de costras, el drenaje y la labranza. Para prevenir el apelmazamiento y la formación de terrones grandes, se aplica cal y materia orgánica.

Tabla 2: Relación de color y drenaje de los suelos

Color del suelo	Drenaje
Rojo	Excelente
Rojo café o café	Bueno
Amarillo brillante	Medio
Amarillo pálido	Moderado
Gris	Malo

Fuente: Facultad de Ciencias Agrarias - UNASAM

El color del suelo es el resultado de las cantidades de materia orgánica y de algunos minerales específicos. El color no siempre es un indicador de fertilidad, pero existe una relación entre el color del suelo y el drenaje, tal como se muestra en la Tabla 2.

2.2.1.3 Productividad del suelo

La productividad del suelo es su capacidad para producir cultivos. Para que el suelo sea productivo es necesario que sea fértil. Sin embargo un suelo fértil no es necesariamente productivo. Por ejemplo existen suelos fértiles en zonas áridas que no pueden producir sin riegos.

El suelo es fértil cuando contiene y suministra a las raíces cantidades adecuadas de nutrientes, agua y aire para que el cultivo produzca bien. Tiene una estructura y profundidad adecuadas para proporcionar un ambiente favorable al desarrollo de las plantas.

Un buen suelo mantendrá sus condiciones favorables durante un largo periodo, inclusive ante influencias adversas climáticas y de vegetación.

El agricultor tiene la posibilidad de mantener y mejorar la fertilidad natural del suelo a través de la aplicación de medidas tales como:

- Análisis de suelos para determinar el suministro de nutrientes.
- Preparación adecuada del suelo.
- Aplicación de fertilizantes, cal de acuerdo con los análisis y necesidades de los cultivos.
- Restauración continúa del contenido de residuos vegetales y animales.
- Adecuada rotación de cultivos.

Materia orgánica (MO).- La materia orgánica esta formada por materiales frescos, plantas parcial y completamente descompuestas, humus. El humus es el producto final de la descomposición de la materia orgánica en el suelo.

Un suelo rico en materia orgánica y buena estructura permite que las raíces penetren mejor.

La materia orgánica actúa como granulador en las partículas minerales. La materia orgánica y los entes microbianos forman migajones, los mismos que crean una estructura desmenuzable lo cual es una característica de los suelos productivos.

La materia orgánica suministra energía a los microorganismos del suelo, sin estos no habría actividad biológica ni descomposición de la materia orgánica y habría limitación en la formación de nódulos en las raíces de las leguminosas.

La materia orgánica fundamentalmente proporciona nutrientes tales como nitrógeno, fósforo y azufre.

2.2.1.4 Elementos nutritivos de las plantas

Los vegetales absorben nutrientes del suelo y del aire atmosférico a través de su raíz, hojas y tallos verdes, también pueden asimilarlos cuando se mojan las partes aéreas con soluciones acuosas nutritivas.

Cierto número de nutrientes son utilizados en cantidades importantes, mientras otros lo son en pequeñas dosis.

Dentro de los nutrientes de importancia tenemos:

- Carbono (C)
- Hidrogeno (H)
- Oxigeno (O)
- Nitrógeno (N)
- Fósforo (P)
- Potasio (K)
- Azufre (S)
- Calcio (Ca)
- Magnesio (Mg)
- Hierro (Fe)

- Cobre (Cu)

En un grupo secundario tenemos:

- Zinc (Zn)
- Manganeso (Mn)
- Boro (Bo)
- Cloro (Cl)
- Molibdemo (Mo).

Se debe tener presente que los vegetales no absorben directamente las moléculas de los nutrientes tales como se indican, sino que estas moléculas al ponerse en contacto con el suelo e interrelacionarse con las demás moléculas sufren transformaciones lo cual da como resultado sustancias que si son absorbidas por los vegetales en forma iónica; con excepción del carbono, hidrógeno y oxigeno.

Carbono.- Es el elemento químico de mayor importancia para los vegetales, porque forma parte de todos los tejidos de la planta, este elemento lo fijan las plantas absorbiendo por las hojas y partes verdes el anhídrido carbónico del aire, en virtud de un proceso fisiológico denominado fotosíntesis o función clorofílica, donde existe una segunda fase en la cual la planta expulsa oxigeno.

Para que la fotosíntesis se de, se necesita del concurso de la luz solar, cuya energía radiactiva contribuye a la formación de la materia viva de la planta, que produce el crecimiento de los tejidos.

A partir de la asimilación del carbono, se forman en las hojas los principios inmediatos que han de integrar los tejidos vegetales: azucares, grasas, proteínas, vitaminas y hormonas, la fotosíntesis también depende de que exista fósforo y

potasio en las hojas, estos elementos son imprescindibles para el crecimiento vegetal, motivo por el cual se fertiliza la tierra con abonos fosfopotásicos.

Oxigeno.- El oxigeno que penetra en la planta procede del aire atmosférico en virtud de la función respiratoria que realizan los vegetales, lo cual consiste en la absorción del oxigeno y desprendimiento de anhídrido carbónico lo cual se da a través de las raíces, tallos, hojas. flores y frutos.

La respiración es fundamental para las plantas, cuando esta función no se realiza se tiene como efecto que el proceso de nutrición y asimilación clorofílica aceleran la asfixia respiratoria de la planta con la consiguiente muerte de la misma.

De una manera especial, las raíces absorben el oxígeno del aire interpuesto en el suelo y compuestos químicos oxidantes, como los nitratos, cuyo papel es decisivo en la respiración radicícola. Dicho oxígeno se integra en la savia que oxigena los tejidos del crecimiento de la planta.

La asfixia radicícola es un accidente grave porque produce la podredumbre de las raíces, que una vez iniciada no tienen curación posible.

En los suelos permeables con espacio poroso y óptima circulación de aire, las raíces respiran bien, lo cual activa la absorción nutritiva de las mismas. Lo contrario sucede en terrenos arcillosos, deduciéndose que las fórmulas de abonado deben incrementarse en los suelos filtrantes a fin de optimizar al máximo la potencia asimiladora que desarrollan las raíces en tales terrenos.

Hidrógeno.- El hidrógeno no penetra en la planta en forma gaseosa sino como componente del agua lo cual es absorbido por las raíces y sirve al mismo tiempo como vehículo de transporte de los nutrientes de las plantas.

Nitrógeno.- Este elemento nutritivo lo absorben las raíces de los siguientes compuestos en solución del suelo:

- Del anión nítrico (nitratos) de rápida asimilación
- Del catión amonio (sales amónicas) cuya intensidad de absorción varía con las distintas especies vegetales.
- De principios orgánicos nitrogenados, por las plantas provistas de micorrizas (simbiosis de hongos con raíces) que se forman en las tierras ácidas.

Los abonos nitrogenados son rápidamente absorbidos por la planta y pasan a ser un elemento constitutivo de la clorofila, debemos indicar que el nitrógeno es uno de los elementos principales de la nutrición vegetal, lo cual obliga a tener que fertilizar los suelos con abonos nitrogenados.

Fósforo.- El fósforo es absorbido de las raíces de los aniones fosfóricos de la solución del suelo. Durante la fotosíntesis se forma un fosfoglicerido, lo que exige la presencia de fósforo para la nutrición clorofílica, el dulzor de las frutas depende de la riqueza del suelo en fosfatos y de la permeabilidad del terreno, lo cual aumenta la respiración radicícola y por consiguiente la absorción nutritiva.

Para obtener de los cultivos cosechas importantes, es necesario aumentar la riqueza del suelo en fósforo, mediante la aplicación de abonos fosfatados.

Potasio.- El potasio es absorbido por las raíces vegetales en forma de catión, de los compuestos potásicos de la solución del suelo después de ser ionizados.

Por su carácter radiactivo, desprende electrones cuando recibe la luz solar, los cuales son imprescindibles para la fotosíntesis, por ello la escasez de potasio detiene la asimilación del carbono.

Como los suelos no contienen en general dosis suficientes de potasio, es necesario incorporar al suelo fertilizante de esta clase.

Fertilidad del suelo: Es una cualidad del suelo resultante de la interacción entre las características físicas, químicas y biológicas del mismo y que consiste en la capacidad de poder suministrar condiciones necesarias para el crecimiento y desarrollo de las plantas.

Se debe indicar que en base a la experiencia y estudios de gente comprometida en la docencia y agricultura (curso AS-F02-Facultad de Ciencias Agrarias-UNASAM) indican lo siguiente:

Un suelo exhibe una **fertilidad física** ideal cuando:

- Es de textura media y con suficiente materia orgánica para favorecer la circulación del aire y el almacenamiento de agua.
- Posee una buena estructuración (migajosa) y por tanto es poroso.
- Tiene una buena profundidad efectiva y es adecuadamente permeable.

Un suelo exhibe una **fertilidad química** ideal cuando:

- Posee PH ligeramente ácido a neutro (6 – 7).

- Posee un alto CIC, alto porcentaje de saturación de bases y adecuado balance catiónico.
- Posee P y K disponibles, lo suficientemente altos.
- La CE es al menos menor a 2 mS/cm.

Un suelo exhibe una **fertilidad biológica** ideal cuando:

- Posee un alto porcentaje de materia orgánica.
- Posee adecuado drenaje.
- No se hace abuso de agroquímicos.
- Se aplica rotación de cultivos planificados.

2.2.1.5. Teoria difusa (FUZZY)

El término difuso (“fuzzy”) fue propuesto por Zadeh en 1962, quien en 1965 [30], formalmente publicó el famoso tratado sobre “Fuzzy Sets” (conjuntos difusos), la teoría de los conjuntos difusos (fuzzy) es desarrollada como una extensión de la teoría de conjuntos. Haciendo uso de la teoría de conjuntos difusos se desarrollas modelos más robustos y flexibles para resolver problemas complejos del mundo real y sistemas que involucran algunos aspectos inherentes al ser humano, contribuye a la toma de decisiones no solo al considerar la existencia de alternativas bajo restricciones dadas (optimizar un sistema dado), sino que también desarrolla nuevas alternativas (diseñando un sistema). La teoría de conjuntos difusos es también aplicada en muchos campos, tales como la investigación de operaciones, gestión en ciencias, teoría de control, inteligencia artificial, sistemas expertos, comportamiento de aspectos humanos, etc.

Frecuentemente en la vida real hay hechos inciertos o conceptos que albergan valores de verdad que se encuentran entre la pertenencia (1) y la no pertenencia (0). En estos casos la lógica matemática convencional no responde con eficacia.

Esto da lugar a un nuevo tipo de lógica, llamada lógica difusa, la cual puede determinar valores intermedios fraccionales entre 0 y 1 de la lógica binaria. En este acápite haremos una breve descripción de los conjuntos y números difusos, así como de ciertas propiedades y operaciones con ellos.

2.2.1.5.1 Subconjuntos Difusos

En la teoría clásica, un conjunto es definido también por una función característica. Supongamos que A es un subconjunto de S definido como un mapeo de los elementos de S a los elementos del conjunto {0,1}, representados como un conjunto de pares ordenados con exactamente un par ordenado presente por cada elemento de S, la segunda componente del par ordenado es un elemento del conjunto {0,1}, donde 0 representa la no pertenencia y el 1 la pertenencia.

En la siguiente oración: x está en A, se evalúa su valor de verdad encontrando el par ordenado cuyo primer elemento es x, si el segundo elemento del par ordenado es 1 entonces la oración es verdadera, en cambio si es 0 entonces es falsa. Por lo tanto el elemento x pertenece o no pertenece al conjunto A.

Si el rango del mapeo anterior se considera, al intervalo [0,1] en lugar del conjunto {0,1}, en donde el valor 0 denotaría la no pertenencia completa y por el contrario el valor 1 denotaría la pertenencia completa y cualquier otro valor intermedio determinaría grados intermedios de pertenencia. Esto es lo que se denomina grado de pertenencia difuso.

Debemos indicar que los grados difusos no son equivalentes a los porcentajes usados en probabilidades. Pues las medidas de probabilidad establecen si algo pasara o no, en cambio lo difuso establece el grado mediante el cual algo ocurrirá o no ante la existencia de una condición.

A continuación se definirá lo que es un conjunto difuso:

Sea X un conjunto cuyos elementos notaremos por *x*, y sea A un subconjunto de *X*. La pertenencia de un elemento *x* de *X* al conjunto A viene dada por la función característica

$$\mu_A(x) = \begin{cases} 1 & \text{si y sólo si } x \in A \\ 0 & \text{si y sólo si } x \notin A \end{cases}$$

{1,0} es el llamado conjunto valoración (rango).

Si el conjunto valoración es el intervalo real [0,1], A se denomina un conjunto difuso y μ_A mide el grado de pertenencia del elemento *x* al conjunto A.

Un conjunto difuso A se denota por el conjunto de pares $A = \{(x, \mu_A(x)), x \in X\}$

Definición de Conjuntos Difusos.- Sea *X* un conjunto universal clásico, A es llamado un conjunto difuso en *X*, si definimos una función $\mu_A : X \to [0,1]$ denominada función de pertenencia, tal que a cada elemento *x* de *X* le asocia un número $\mu_A(x)$ entre 0 y 1, denominado el grado de pertenencia de *x* en A. Equivalentemente diremos que un conjunto difuso *A* se define como la función de pertenencia que enlaza o empareja los elementos de un dominio o universo *X* con elementos del conjunto [0,1].

Así de esta manera cuando más cerca este $\mu_A(x)$ del valor 1 mayor será la pertenencia del elemento *x* al conjunto *A*, los valores de pertenencia varían entre 0 (la no pertenencia total) y 1 (pertenencia total).

Representación.- un conjunto difuso *A* puede representarse como un conjunto de pares de valores, cada elemento $x \in X$ con su grado de pertenencia $\mu_A(x)$, esto es $A = \{(x, \mu_A(x)), x \in X\}$.

Otra representación es dada en forma de suma de pares esto es:

$A = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i}$, en este caso los $\mu_A(x_i) = 0$ no se consideran. Además la suma no se considera como la operación algebraica.

Cardinalidad.- La cardinalidad de un conjunto difuso A se define como:

$$Card(A) = \sum_{x \in X} \mu_A(x)$$

Función de pertenencia.- Un conjunto difuso también puede representarse geométricamente como una función, especialmente cuando el universo es continuo.

Abscisa (eje X): universo *X*.

Observación.- En una función de pertenencia $A: X \to [0,1]$, la representación de la función A debe ser simple, debido a que simplifican muchos cálculos y no se pierde exactitud, debido precisamente a que se está definiendo un concepto difuso, así pues se puede tener formas trapezoidales, triangulares, parabólicas, acampanadas, etc.

Soporte de un conjunto Difuso.- Es dado por elementos de X que pertenecen a A con grado de pertenencia mayor que cero esto es Sop(A) = {x$\in X$ / $\mu_A(x)$) > 0}

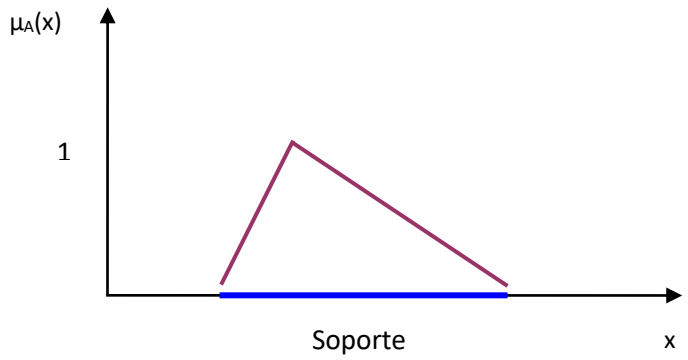


Figura 2: Soporte de un conjunto difuso

Conjunto α nivel (α - corte).- El conjunto α-corte de un conjunto difuso A es un subconjunto clásico de X que está dado por:

$$A_\alpha = \{x / \mu_A(x) \geq \alpha, x \in X\}$$

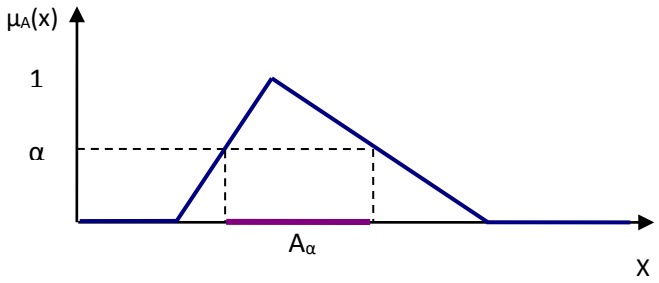


Figura 3: α-corte de un conjunto difuso

Conjunto Difuso Convexo.- Un conjunto difuso A es convexo si y sólo si para todo par de puntos x_1 y x_2 en X, la función de pertenencia de A satisface

$$\mu_A[\delta x_1 + (1-\delta)x_2] \geq mín[\mu_A(x_1), \mu_A(x_2)] \text{ donde } \delta \in [0,1].$$

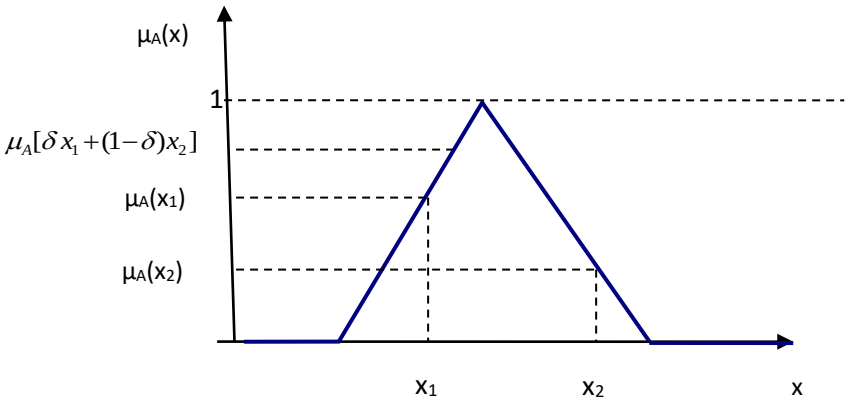


Figura 4: Conjunto difuso convexo

En otras palabras $\mu_A[\delta x_1 + (1-\delta)x_2] \geq mín[\mu_A(x_1), \mu_A(x_2)]$ donde $\delta \in [0,1]$, significa que un conjunto es convexo si para cualquier punto entre x_1 y x_2, este debe tener un grado mayor o igual que el mínimo de x_1 y x_2.

Normalidad.- Un conjunto difuso A es normal si y sólo si el supremo de $\mu_A(x)$ en X es 1, es decir $\sup_X \mu_A(x) = 1$.

Número Difuso.- Un conjunto difuso convexo y normalizado definido en $\Re$ cuya función de pertenencia es seccionalmente continua es llamado número difuso.

2.2.1.5.2 Operaciones Básicas con Conjuntos Difusos:

Resumiremos algunas operaciones básicas con conjuntos difusos:

Inclusión.- Sean A y B dos subconjuntos difusos en X, entonces A está incluido en B si y sólo si $\mu_A(x) \leq \mu_B(x), \quad \forall x \in X$. Denotaremos la inclusión por $A \subset B$.

Inclusión Difusa.- Si el universo es finito, se puede determinar el grado en el cual un conjunto difuso A, está incluido en otro conjunto difuso B, de la forma siguiente:

$$S(A,B) = \frac{1}{Card(A)}\{Card(A) - \sum_{x \in X} máx(0, \mu_A(x) - \mu_B(x))\}$$

Igualdad.- Los conjuntos difusos A y B son llamados iguales si y sólo si $\mu_A(x) = \mu_B(x)$, $\forall x \in X$. Denotaremos la igualdad por $A = B$.

Complementación.- Los conjuntos difusos A y B en X son complementarios si y sólo si $\mu_A(x) = 1 - \mu_B(x), \quad \forall x \in X$. Denotaremos el complemento de A por $B = A^c$ ó, $A = B^c$ donde A^c y B^c son complementos de A y B respectivamente.

Intersección .- La intersección de dos conjuntos difusos A y B en X está definida por:

$$\mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\}, \quad \forall x \in X.$$

Unión.- La unión de dos conjuntos difusos A y B en X está definida por: $\mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\} \quad , \forall x \in X$.

Diferencia.- La diferencia $A - B$ de los conjuntos difusos A y B en X es caracterizada por $\mu_{A \cap B^C}(x) = min(\mu_A(x), \mu_{B^C}(x))$; donde B^c es el complemento de B, con $\mu_{B^c}(x) = 1 - \mu_B(x)$.

2.2.1.5.3 Métodologia de Aproximación de Zimmermann

Zimmermann [6,17], considera la función objetivo como una restricción difusa teniendo como recurso a b_0 y su correspondiente tolerancia t_0, este b_0 viene a ser una meta a lograr de la función objetivo. Teniendo en cuenta estos datos, Zimmermann transforma el modelo (2) en el siguiente modelo:

$$
\begin{aligned}
&\textit{hallar } \mathrm{x} \\
&\textit{tal que}: \\
&\qquad cx \geq_f b_0 \\
&\qquad (Ax)_i \leq_f b_i \quad \forall\, i \\
&\qquad x \geq 0
\end{aligned}
$$

donde las restricciones difusas están definidas respectivamente por las funciones de pertenencia que se expresan a continuación:

$$
\mu_0(x) = \begin{cases} 1 & ; \quad \mathtt{si} \quad cx > b_0 \\ 1 - \dfrac{(b_0 - cx)}{t_0} & ; \quad \mathtt{si}\ b_0 - t_0 \leq cx \leq b_0 \\ 0 & ; \quad \mathtt{si} \quad cx < b_0 - t_0 \end{cases}
$$

$$
\mu_i(x) = \begin{cases} 1\ , & ; \quad \mathtt{si} \quad (Ax)_i < b_i \\ 1 - \dfrac{[(Ax)_i - b_i]}{t_i} & ; \quad \mathtt{si}\ b_i \leq (Ax)_i \leq b_i + t_i \\ 0\ , & ; \quad \mathtt{si} \quad (Ax)_i > b + t_i \qquad \forall i \end{cases}
$$

y sus correspondientes gráficas se aprecian en las figuras (5) y (6) que se muestran a continuación:

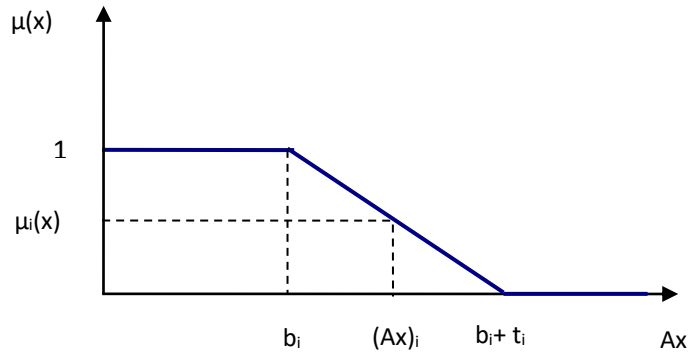


Figura 5: Función de pertenencia de las restricciones difusas

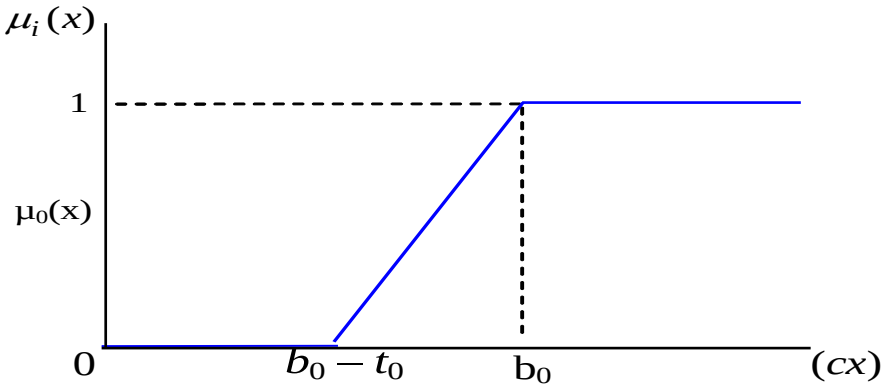


Figura 6: Función de pertenencia de la meta difusa.

Luego usando el operador ***max–min*** de Bellman y Zadeh, tenemos que la solución óptima se puede obtener maximizando μ_D, es decir, resolviendo

$$\text{Max}\{\min[\,\mu_0(x), \mu_1(x), ..., \mu_m(x)\,]\}$$

donde μ_D es la función de pertenencia del espacio decisión D, dado por:

$$\mu_D = min\,(\,\mu_0, \mu_1, ..., \mu_m).$$

Si $\mu_D = \alpha$ el modelo es equivalente a

$$
\begin{aligned}
&max\ \alpha \\
&sujeto\ a: \\
&\quad \mu_0(x) = 1 - (b_0 - cx)/t_0 \geq \alpha \\
&\quad \mu_i(x) = 1 - [(Ax)_i - b_i]/t_i \geq \alpha \qquad i = 1,...,m \\
&\qquad \forall i, \alpha \in [0,1]
\end{aligned}
$$

o lo que es lo mismo

$$
\begin{aligned}
&max\ \alpha \\
&sujeto\ a: \\
&\quad cx \geq b_0 - (1-\alpha)\, t_0 \\
&\quad (Ax)_i \leq b_i + (1-\alpha) t_i \qquad \forall i \\
&\quad x \geq 0, \alpha \in [0,1]
\end{aligned}
$$

Obviamente, el modelo se puede resolver mediante técnicas de la programación lineal clásica.

2.3 Definición de términos

A. Fuzzy

El término Fuzzy significa difuso, borroso, impreciso, que no está perfectamente definido o completamente limitado.

B. Variables Lingüísticas

Son variables cuyos valores son descritos cualitativamente y cuantitativamente por un conjunto difuso.

C. Conjuntos difusos

Son conjuntos con fronteras uniformes o suaves.

D. Restricciones Difusas

Son aquellas restricciones en las cuales los recursos no son conocidos con precisión.

E. Costos Difusos

Son aquellos cuyos costos o utilidades se conocen vagamente (con imprecisión, no con exactitud). Por tanto se representan mediante

números difusos

F. Programación difusa

En general un problema de programación lineal difusa se puede expresar de la siguiente manera:

$$\max \sum_{j=1}^{n} c_j x_i$$

$$\textit{sujeto a}:$$

$$\sum_{j=1}^{n} A_{ij} x_j \leq B_i \quad (i \in N_m)$$

$$x_j \geq 0 \quad (j \in N_n)$$

Donde A_{ij} , B_i y c_j son variables difusas. Es decir que estas variables manejan cierta holgura permitiendo así modelar casos donde las variables se encuentran dentro de un rango dado y con un acercamiento a la realidad.

G. Decisión Difusa

Una aplicación importante de la teoría de conjuntos difusos es en la teoría de toma de decisiones, debido a que en ella las informaciones siempre son imprecisas. En un proceso de decisión el principal ingrediente incluye un conjunto de alternativas, un conjunto de restricciones para la elección entre las alternativas diferentes, y una ejecución (función objetivo). En un ambiente difuso (con informaciones imprecisas), según Bellman y Zadeh, la función objetivo se puede representar en términos de metas imprecisas, por lo tanto las restricciones como la meta constituyen conjuntos difusos y la decisión difusa es el conjunto difuso de alternativas resultantes de la intersección de tales conjuntos difusos.

Así, Bellman y Zadeh propusieron que una decisión difusa puede ser definida como el conjunto difuso de alternativas resultantes de la intersección de la meta y

las restricciones difusas. Esto es: $D = G \cap C$ es el conjunto difuso resultante de la intersección de G y C y tiene como función de pertenencia $\mu_D = \mu_G \cap \mu_C$ (figura 7).

Una mejor decisión (maximizante), entonces, puede ser definida como sigue:

$$\mu_D(x^M) = \begin{cases} \max \mu_D(x), & x \in X \\ 0 \quad , \text{ en otra parte.} \end{cases}$$

Donde μ_D es la función de pertenencia del espacio de decisión D.

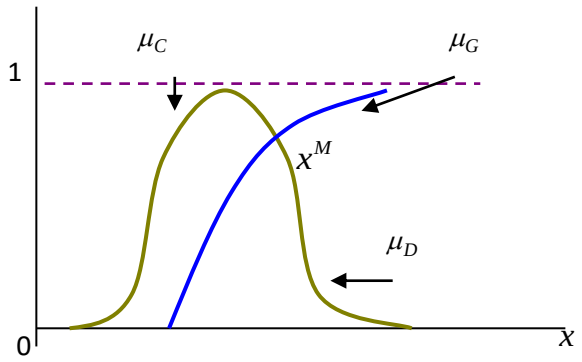


Figura 9: Intersección de la meta G y restricción C

En general cuando se utiliza este tipo de decisión difusa, diremos que se está utilizando el operador ***max-min***.

Donde el ***min*** representa la intersección de las funciones de pertenencia de las restricciones y meta, mientras que **max** significa que se debe elegir la mejor decisión, es decir aquella alternativa con el mayor grado de pertenencia (aceptación).

III. MATERIALES Y METODOS

En el presente trabajo se utilizara la información relativa a:

- La composición química de los fertilizantes.
- Las propiedades físicas, químicas y biológicas de los suelos (análisis de suelos).
- Las fórmulas de abonamiento.

- Los conceptos y métodos de solución de problemas de Programación Lineal clásica.
- La teoría de los conjuntos difusos.
- Modelo del problema de dietas.
- Métodos de la programación lineal difusa

Se ha requerido de los siguientes medios y materiales:

Equipos.

- Una Laptp personal marca Lenovo
- Una impresora láser HP.

Softwares:

- Windows XP Profesional.
- LINGO, software especial para resolver problemas de programación lineal clásica.
- WINQSB, software especial para resolver problemas de programación lineal clásica.
- FertiDif. Sistema informatico diseñado por el tesista.

3.1. Tipo y diseño de la investigación.

3.1.1. Tipo de investigación

- El presente trabajo es de Tipo Explicativo con Metodología Cualitativa – Cuantitativa.

3.1. 2. Diseño de la investigación.

Se recurrirá a comparar los resultados obtenidos con la aplicación de la Lógica Difusa y las diferentes Metodologías de solución planteadas las cuales se implementaran en un sistema computacional y de esa manera facilitar los cálculos. Posteriormente se hará dos cultivos de periodo de producción corto, uno fertilizado de manera tradicional y el otro con los resultados obtenidos en el estudio del presente proyecto.

3.2. Población y muestra

- La población de estudio en la investigación es el universo de fertilizantes.
- La Muestra es similar a la población.

3.3. Validación de la investigación

Se hará mediante la comparación de los resultados obtenidos con el software diseñado y los resultados obtenidos con los programas LINGO y WINQSB ello enfocado en los costos obtenidos y en la determinación de la cantidad de fertilizantes a usar en un proceso de fertilización de un determinado cultivo.

3.4. Método

En el presente trabajo partiremos de un modelo con coeficientes de la función objetivo difusos los mismos que son caracterizados porque son aquellos modelos cuyo costo o utilidades se conocen vagamente (son imprecisos o no son exactos). Por consiguiente se pueden representar mediante números difusos, y el modelo que describe este comportamiento es dado por:

$$\text{Max } Z = \tilde{C}\, x$$

Sujeto a:

$$Ax \leq b \qquad (3)$$

$$x \geq 0$$

Obviamente Z es también un número difuso, pero x puede ser un vector de números difusos o no difusos y cada costo difuso se describe mediante su correspondiente función de pertenencia $\mu(x)$.

3.4.1. Modelos de Fertilización Agrícola Difusa

La condición de vaguedad de cada parámetro en (3) puede generar un modelo difuso correspondiente, que en cualquier caso estarán comprendidos dentro de los cinco modelos, de acuerdo a la clasificación hecha por Lai-Hwang [17] y Verdegay [27], siguientes:

a) Problemas con restricciones difusas, que se derivan de las situaciones en las que no se conoce con precisión la cantidad de los recursos con la que se dispone, simbólicamente se expresa

$$\text{Min } z = cx$$

Sujeto a:

$$Ax \leq_f b \qquad (4)$$

$$x \geq 0$$

donde el símbolo "$\leq_f$" representa la imprecisión de la cantidad de los recursos disponibles (que también se utiliza el símbolo $\leq$.) y significa que el decisor admite el incumplimiento de las restricciones (entendida como $\leq$) naturalmente con un margen de tolerancia apropiado.

b) Problemas con metas difusas, corresponde a aquellos problemas de programación lineal en la que el decisor aspira el "costo total sea sustancialmente menor que *p*", se representa por:

$$Min\ \tilde{z} = cx$$

Sujeto a:

$$Ax \leq b \qquad (5)$$

$$x \geq 0$$

donde $\tilde{z}$ representa un número difuso definido del siguiente modo:

$$u(z) = \begin{cases} 1 & z < p \\ f(z) & p \leq z \leq p + t \\ 0 & z > p + t \end{cases}$$

Con f una función monótona no creciente y $t > 0$ un margen de tolerancia.

c) Problemas con la función objetivo difuso, que incluye aquellos en las que los costos son conocidos vagamente, y se representa por:

$$Min\ z = \tilde{c}x$$

Sujeto a:

$$Ax \leq b \qquad (6)$$

$$x \geq 0$$

donde $\tilde{c} = (\tilde{c}_1, \cdots, \tilde{c}_n)$ es un vector de números difusos, con cada componente difuso definido mediante una función de pertenencia u_j: $\mathbf{R} \rightarrow [0,1]$, para cada $j = 1,...,n$.

d) Problemas con coeficientes tecnológicos y recursos difusos, son aquellos caracterizados porque los coeficientes tecnológicos y los recursos son conocidos vagamente, cuya formulación es la siguiente:

$$Min \ \ z = cx$$

Sujeto a:

$$\sum_{j=1}^{n} \tilde{a}_{ij} \leq^{f} \tilde{b}_i , \ \ i = 1,...,m \qquad (7)$$

$$x \geq 0$$

donde cada $\tilde{a}_{ij}$ y $\tilde{b}_i$ es un número difuso definido mediante su correspondiente función de pertenencia, y el símbolo $\leq^{f}$ representa la relación de orden entre dos números difusos definido de acuerdo a la naturaleza del problema en concreto, debido a que existen diferentes formas de ordenar los números difusos [26, 27].

e) Problemas completamente difusos, comprende los casos más imprecisos, es decir cuando tanto los costos, coeficientes tecnológicos y los recursos con imprecisos, y se representa por:

$$Min \ \ z = \tilde{c} \, x$$

Sujeto a:

$$\sum_{j=1}^{n} \tilde{a}_{ij} \leq^{f} \tilde{b}_i , \ \ i = 1,...,m \qquad (8)$$

$$x \geq^{f} 0$$

donde los símbolos son interpretados como en (6) y (7) de acuerdo al caso.

La programación lineal difusa que estudia los modelos (4)-(7), que tienen como coeficientes a los números difusos, son denominados la *programación posibilística* . [3,4,16,17,21,22].

Los modelos difusos correspondientes de los PDD son inmediatos, y particularmente para el caso general es de la forma:

$$\text{Min} \sum_{j=1}^{n} \tilde{c}_j x_j$$

Sujeto a:

$$\tilde{p}_i \leq^f \sum_{j=1}^{n} \tilde{a}_{ij} x_j \leq^f \tilde{P}_i, \quad i = 1,...,m \tag{9}$$

$$\tilde{m}_j \leq^f x_j \leq^f \tilde{M}_j, \quad j = 1, ..., n$$

donde cada uno de los parámetros difusos y los símbolos son como en (8).

El propósito fundamental del presente trabajo es resolver el modelo (6) haciendo uso de conceptos de optimización y paralelamente se diseñara un programa computacional que nos permita resolver (6 con rapidez y con soluciones óptimales.

El modelo de fertilización agrícola difuso ser resuelto mediante una adaptación adecuada de los métodos aproximados de Lai_Hwang y Leberling. Los mismos que luego serán implementados en un programa computacional que permitirá obtener resultados en forma rápida. A continuación daremos la descripción de las metodologías antes mencionadas.

3.4.2 Métodos de solución

Existen diferentes métodos de solución dependiendo del modelo a resolver. La diferencia entre ellos se dan o bien en la forma de representación de los parámetros difusos o bien en el procedimiento mismo. En el presente caso trataremos dos métodos de solución que por su formulación y sencillez permitirán resolver el problema planteado (3). Sin embargo debo dejar indicado que si la información de dicho modelo es imprecisa es de imaginar que la solución (es) obtenida (as) es también imprecisa, obteniéndose en consecuencia no soluciones en el sentido tradicional, sino soluciones difusas, los cuales se usarán dependiendo de la experiencia y conocimiento sobre el tema del decisor o encargando de tomar decisiones.

Se dijo, que en la formulación del modelo dado en (3), los coeficientes difusos hacen que la función objetivo también sea difusa. La representación, esto es, la función de pertenencia de la función objetivo depende de la representación de los coeficientes, que por cierto tienen múltiples formas de representación, y de la forma como se relacionan éstas. La diferencia entre los métodos de solución de modelos de la forma (3) propuestos radican en dos características básicas, que son: la forma cómo se representan los coeficientes difusos y la forma cómo se relacionan para representar a la función objetivo difusa.

Cabe indicar, que de aquí en adelante, cuando se trabaje con los coeficientes que son números difusos, se presentan algunas diferencias debido a que los números difusos no se conciben de una manera única. Esta situación conduce a la existencia de métodos generales que engloban todas las concepciones, y otros métodos particulares en los que se utilizan números difusos con características particulares.

A continuación desarrollaremos dos enfoques para resolver el problema planteado (3), uno propuesto por Lai y Hwang y el otro propuesto por Leberling.

3.4.2.1 Aproximación de Lai y Hwang:

Aquí se considera el modelo (3) con costos representados por números difusos en su forma triangular, es decir, cada costo tiene la forma de $\tilde{C}_i = (C_i^m,\ C_i^p,\ C_i^o)$, para todo i, con función de pertenencia denominada distribución de posibilidad triangular tal como se muestra de la Figura 7, dondeC_i^m, es el valor más posible (posibilidad = 1), C_i^pel valor más pesimista, y el C_i^o el valor más optimista. De este modo la función de pertenencia para el número $\tilde{C}_i$ es π_i (distribución posibilidad), que puede ser expresada como el grado de ocurrencia de un evento, por esta razón (3) se denomina también programación posibilística.

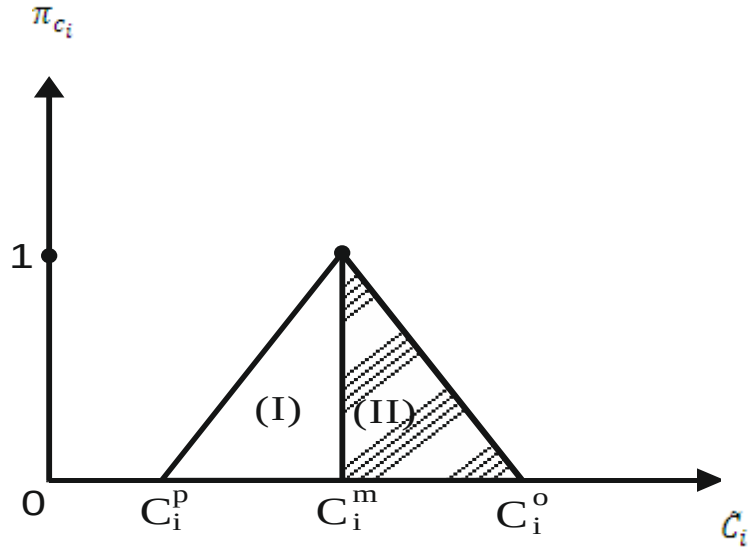


Figura 7: La distribución posibilística de $\tilde{C}_i$

C_i^m = Valor más posible (posibilidad = 1)

C_i^p = valor más pesimista

C_i^o = valor más optimista

π_i = grado de ocurrencia de un evento

Para resolver el problema (3), Lai y Hwang [19,20], utilizan la filosofía del empresario (la de enrumbar sus actividades a fin de aumentar su posibilidad de ganancia), proponen transformar en un problema de programación lineal multiobjetivo con tres objetivos a solucionar, estos objetivos son:

(a) Minimizar la posibilidad de obtener menor ganancia.

(b) Maximizar la posibilidad de más alta ganancia.

(c) Maximizar la posibilidad de obtener mayor ganancia.

Si se tiene en cuenta que las variables x_i representan números reales, la función objetivo de (3) es equivalente a:

$$\max_{x \in F} \sum_{i=1}^{n} \left(C_i^m x_i,\ C_i^p x_i, C_i^o x_i \right)$$

ó

$$\max_{x \in F} \left((C^m)^t x,\ (C^p)^t x, (C^o)^t x, \right) \qquad (10)$$

Donde: $C^m = (C_1^m, C_2^m, \ldots C_n^m)$,

$C^p = (C_1^p, C_2^p, \ldots C_n^p)$,

$C^o = (C_1^o, C_2^o, \ldots C_n^o)$

F = *representa el conjunto de restricciones para* x

La función objetivo corresponde a una función objetivo difusa o imprecisa con la distribución de posibilidad triangular.

Este objetivo difuso está completamente definido geométricamente por tres puntos o vértices $((C^m)^t x, 1), (C^p)^t x, 0) y\ ((C^o)^t x, 0)$. Así, maximizamos la

función objetivo, desplazando estos tres puntos críticos hacia la derecha. Afortunadamente, las coordenadas verticales de los puntos críticos son fijados solo en 1 ó en 0, por lo que serán consideradas sólo las tres coordenadas horizontales. Por tanto el problema a resolver es:

$$\max_{x \in F} ((C^m)^t x,\ (C^p)^t x, (C^o)^t x,)$$

Donde $((C^m)^t x,\ (C^p)^t x, (C^o)^t x,)$ es el vector de las tres funciones objetivos, $(C^m)^t x,\ (C^p)^t x\ \ y\ (C^o)^t x$ se escriben en ese orden para mantener la forma triangular (normal y convexa) de la distribución posibilidad.

En lugar de maximizar estos tres objetivos simultáneamente vamos a maximizar, $(C^m)^t x$, minimizar $[(C^m)^t x\ -\ (C^p)^t x]$ y maximizar$[(C^o)^t x -\ (C^m)^t x]$, donde las últimas dos funciones objetivos son medidas relativas desde$(C^m)^t x$ (ver Figura 8).

Los tres nuevos (funciones) objetivos nos garantizan los argumentos previos de desplazamiento de la función de posibilidad triangular hacia el lado derecho

$$\pi_{c_i}$$

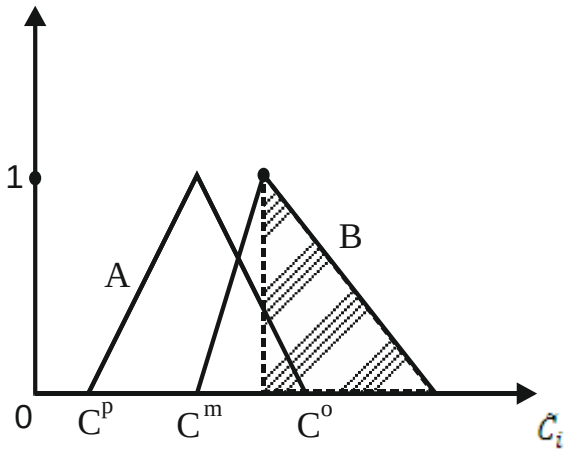


Figura 8: Estrategia para resolver max$\tilde{C}^t x$

Esto nos lleva al siguiente problema auxiliar de la ecuación (10).

$$\min Z_1 = (C^m - C^p)^t x$$

$$\min Z_2 = (C^m)^t x \qquad (11)$$

$$\min Z_3 = (C^o - C^m)^t x$$

Sujeto a: x$\in F$

El problema clásico de programación lineal multiobjetivo que corresponde a la ecuación (11) es equivalente a maximizar el valor más posible (el grado de posibilidad = 1). Al mismo tiempo hemos minimizado el lado inferior de la distribución posibilidad. Esto significa minimizar la región (I) de la Figura (7), lo cual constituye "el riesgo de obtener la menor ganancia"; y también hemos maximizado la región (II) de la distribución de posibilidad, la cual es equivalente a "la posibilidad de obtener la mayor ganancia".

De lo que se observa en la Figura (8), es evidente que preferiríamos la distribución posibilidad B que la de A. Para solucionar (11) haremos uso del método de programación lineal difusa de Zimmermann [33].

El primer paso será considerar que cada función objetivo es como una meta difusa. Luego se debe definir su función de pertenencia correspondiente. Con tal propósito calculamos, lo que se denomina Solución Ideal Positiva (PIS) y la Solución Ideal Negativa (NIS) de las tres funciones objetivos y estas son:

$$Z_1^{PIS} = \min_{x \in F} (C^m - C^p)^t x, \qquad Z_1^{NIS} = \max_{x \in F} (C^m - C^p)^t x \tag{12}$$

$$Z_2^{PIS} = \max_{x \in F} (C^m)^t x, \qquad Z_2^{NIS} = \min_{x \in F} (C^m)^t x \tag{13}$$

$$Z_3^{PIS} = \max_{x \in F} (C^o - C^m)^t x, \qquad Z_3^{NIS} = \min_{x \in F} (C^o - C^m)^t x \tag{14}$$

Luego las funciones de pertenencia lineales para estas funciones objetivos pueden ser definidas como:

$$\mu_{Z_1} = \begin{cases} 1 & Si\ Z_1 < Z_1^{PIS} \\ \frac{Z_1^{NIS} - Z_1}{Z_1^{NIS} - Z_1^{PIS}} & Si\ Z_1^{PIS} \leq Z_1 \leq Z_1^{NIS} \\ 0 & SiZ_1 > Z_1^{NIS} \end{cases} \tag{15}$$

Cuya gráfica se muestra en la Figura (9)

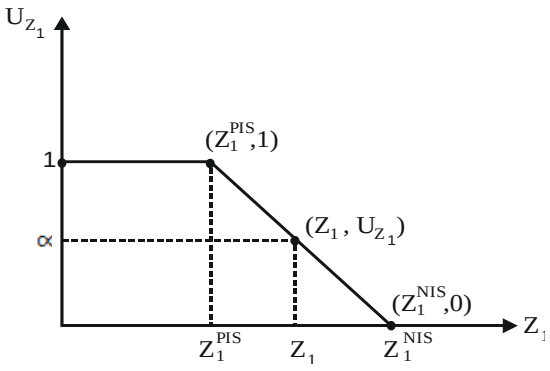


Figura 9: Función de pertenencia para Z_1.

De donde se obtiene que:

$$\mu_{Z_1} - 0 = m(Z_1 - Z_1^{NIS})$$

$$\mu_{Z_1} = \frac{1-0}{Z_1^{PIS} - Z_1^{NIS}}(Z_1 - Z_1^{NIS})$$

$$\mu_{Z_1} = \frac{Z_1 - Z_1^{NIS}}{Z_1^{PIS} - Z_1^{NIS}}$$

$$\mu_{Z_1} = \frac{Z_1^{NIS} - Z_1}{Z_1^{NIS} - Z_1^{PIS}}$$

Con lo cual se tiene (9)

En forma similar de la Figura (10) obtenemos la función de pertenencia para Z_2

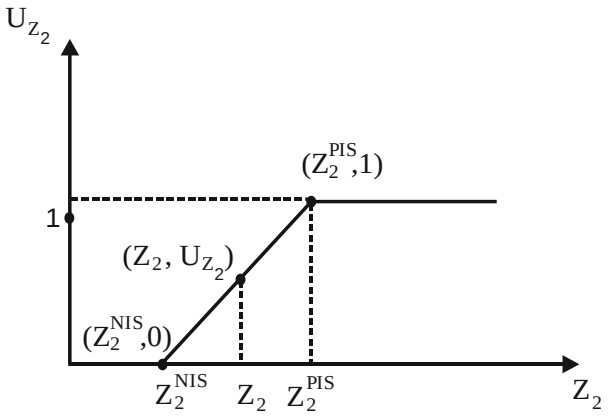


Figura 10: Función de pertenencia para Z_2

del gráfico obtenemos:

$$\mu_{Z_2} - 0 = \frac{1-0}{Z_2^{PIS} - Z_2^{NIS}}(Z_2 - Z_2^{NIS})$$

$$\mu_{Z_2} = \frac{Z_2 - Z_2^{NIS}}{Z_2^{PIS} - Z_2^{NIS}}$$

$$\mu_{Z_2} = \begin{cases} 1 & Si\ Z_2 < Z_2^{PIS} \\ \frac{Z_2 - Z_2^{NIS}}{Z_2^{PIS} - Z_2^{NIS}} & Si\ Z_2^{NIS} \leq Z_2 \leq Z_2^{PIS} \\ 0 & SiZ_2 > Z_2^{NIS} \end{cases} \quad (16)$$

Ahora construimos la función de pertenencia para Z_3 a partir del gráfico de la Figura (11)

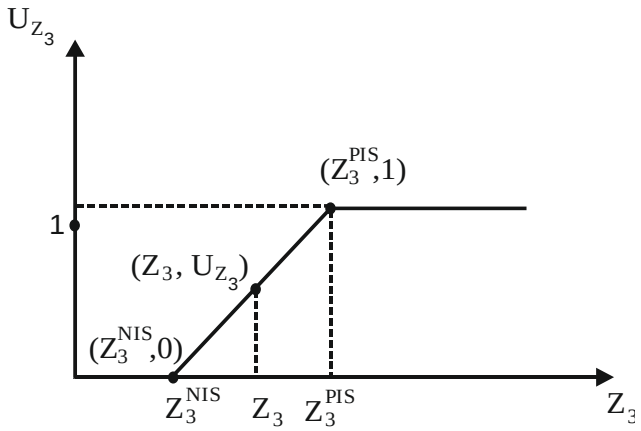


Figura 11: Función de pertenencia para Z_3

$$\mu_{Z_3} - 0 = \frac{1-0}{Z_3^{PIS} - Z_3^{NIS}}(Z_3 - Z_3^{NIS})$$

$$\mu_{Z_3} = \frac{Z_3 - Z_3^{NIS}}{Z_3^{PIS} - Z_3^{NIS}}$$

Luego tendremos:

$$\mu_{Z_3} = \begin{cases} 0 & Si\ Z_3 < Z_3^{NIS} \\ \frac{Z_3 - Z_3^{NIS}}{Z_3^{PIS} - Z_3^{NIS}} & Si\ Z_3^{NIS} \leq Z_3 \leq Z_3^{PIS} \\ 1 & SiZ_3 > Z_3^{PIS} \end{cases} \quad (17)$$

Finalmente solucionamos el problema de programación mediante el método de Zimmermann, el cual es equivalente a un modelo de programación con un solo objetivo, el mismo que tiene la siguiente forma:

$$\max \propto$$

Sujeto a: $\mu_{Z_j} \geq \propto$ j = 1,2,3 (18)

$$X \in F$$

o lo que es lo mismo

$$\max \propto$$

Sujeto: $\mu_{Z_1} \geq \propto$

$\mu_{Z_2} \geq \propto$ (19)

$\mu_{Z_3} \geq \propto$

$X \in F$ con $\propto \in [0,1]$

Donde la solución óptima de (19) nos provee una solución satisfactoria bajo la estrategia previa de minimizar el riesgo de ganar menos (obtener menor ganancia) y maximizar el valor posible y posibilidad de obtener la ganancia más alta.

3.4.2.2 Método de metas difusas en problemas de optimización lineal multiobjetivo

Muchos problemas reales conducen a modelos de optimización lineal multiobjetivo de la forma:

Optimizar Z(X)

Sujeto a:

$AX \leq b$

$X \geq 0$

donde $Z(X) = (Z_1(X), Z_2(X),\ldots Z_K(X))$, con cada $Z_K(X)$ una función lineal y optimizar puede ser maximizar o minimizar para cada K = 1, … n. Estos problemas se denominan problemas de optimización lineal multiobjetivo (POLMO).

Los primeros intentos de formulación y solución de estos problemas se debe a Pareto (1896), quien para resolverlo propuso transformar a un problema de un solo objetivo para aplicar técnicas conocidas.

En este tipo de problemas no es posible definir la idea de solución óptima, solo podemos definir soluciones no dominadas actualmente conocidas como soluciones eficientes, soluciones admisibles o soluciones Pareto óptimas. El concepto de "Pareto Optimalidad" fue introducido por Koopmans en 1951. Un enfoque más general como parte de la programación matemática fue iniciada por Kuhh y Tucker en 1951.

Después de un periodo poco fructífero en Polmo, debido principalmente a la aparición del método Simplex (1949) de la programación lineal.

En 1961 Charnes y Cooper fueron los reiniciadores de un periodo fructífero entre los cuales destaca Zadeh, Ktinger, Yum Zeteny. Buena parte de métodos importantes en POLMO lo encontramos en [58].

Actualmente se cuenta con diferentes métodos de solución de POLMO, tales como: métodos de maximización vectorial [11], programación de metas [11], técnicas iterativas [60], método de ponderación [4], método de puntos interiores [1, 2, 50, 51] y métodos difusos [42].

Los métodos difusos combinan la programación de metas y la programación lineal difusa [41], para resolver los problemas de POLMO.

En el presente trabajo haré uso de la metodología de Leberting [25], para dar solución planteado sobre fertilización agrícola.

3.4.2.3 Aproximación de Leberling:

Consideremos el modelo de todas las funciones objetivo, de la forma siguiente:

Maximizar Z(X)

Sujeto a: $AX \leq b$

$$X \geq 0 \qquad \text{……(20)}$$

Sea $F = \{X \in \mathrm{IR}^n / AX \leq b,\ X \geq 0\}$, el conjunto factible del problema (20).

El primer problema que aparece en estos problemas es la definición de una solución óptima. Como ya se comentó al inicio de este método, no se puede definir una solución óptima para un POLMO, pues una solución óptima para una función objetivo no es necesariamente óptima para los otros objetivos. Puesto que es muy difícil en la práctica que exista una solución factible X_0 que sea simultáneamente óptima para Z_1, Z_2,… Z_m, en tal sentido debemos de sentirnos satisfechos con un concepto de óptimo mucho más débil. Entre es el concepto de óptimo de Pareto o solución eficiente, que de una forma un tanto imprecisa se podría enunciar así:

"El punto factible X_0 es "óptimo" del problema (20) si no existe X factible que mejore todos los objetivos Z_1, Z_2,… Z_m.

Definición (solución eficiente u óptimo de Pareto). Dado el POLMO (20), diremos que $X_0 \in F$ es una solución eficiente si $\nexists X_1 \in F + q\ Z_i(X_1) \geq Z_i(X_0)$ y $Z_i(X_1) = Z_i(X_0)$ para algún $i \in \{1,2,\ldots,m\}$.

El conjunto de todas las soluciones eficientes se denota por E.

DEFINICION: (Soluciones de compromiso)

Es una solución eficiente elegido como una solución óptima mediante algún criterio.

Uno de los criterios usados es una función P: $\mathrm{IR}^n > F \rightarrow \mathrm{IR}$ denominada función de preferencia tal que:

$$X^* \in E \quad y \quad \bigwedge_{X \in F} P(X) \leq P(X^*)$$

Donde $\wedge$ es un operador.

La técnica difusa que trataremos aquí permite obtener una solución de compromiso sin la necesidad de encontrar todas las soluciones eficientes y sin hacer uso de una función de preferencia.

Consideremos en IR^n el orden usual.

$$(\mu_1, \mu_2, \dots, \mu_m) \leq (v_1, v_2, \dots, v_m) \ si \ \mu_i \leq v_i, \ \forall i = \overline{1,m}$$

Definición: El punto $X_0 \in F$ es una solución del problema (14) si no existe $X_1 \in F$ tal que $Z(X_1) \geq Z(X_0)$ y $F(X_1) \neq Z(X_0)$.

La definición dada indica que ningún punto $X_1 \in F$ mejora simultáneamente todos los objetivos. No obstante, y al no ser total el orden natural de IR^m, el que X_0 sea solución máxima no implica que:

$$Z(X_0) \geq Z(X)$$

Y solo se verifica la desigualdad anterior cuando Z(X) y $Z(X_0)$ son comparables (dos elementos de IR^m son comparables si uno de ellos es menor o igual al otro).

Otro aspecto que debemos notar es lo siguiente: Si en el problema (14) se tuviese una sola función objetivo (m=1) y si X_0, X_0^1 fueran soluciones del mismo, se tendría que cumplir que $Z(X_0) = Z(X_0^1)$. Sin embargo esto no es cierto en general cuando m > 1, ya que $Z(X_0) \ y \ Z(X_0^1)$ no tienen necesariamente que ser comparables. El hecho de que el valor óptimo que toma la función objetivo no es único en general es el que motiva la definición siguiente.

Definición: Se llama línea de eficiencia del problema (14) al conjunto.

$$\{Z(X): X \text{ es solución de (14)}\} \subset IR^m$$

Obviamente en el caso de la programación escalar (m=1) la línea de eficiencia de un problema se reduce a un punto (si el problema tiene solución) o simplemente es vacía (si no hay solución). Esto no es cierto si m es mayor que 1.

3.4.2.4 Método de programación de metas

Dado un modelo POLMO de la forma:

Optimizar $Z = Cx$

Sujeto a: $AX \leq b$

$X \geq 0$ ……(21)

Donde $Z = (Z_1, Z_2, \ldots Z_k)^t$ es el vector de los objetivos. C es una kxn – matriz de constantes, x es un n-vector columna de variables de decirión. A es una mxn matriz de constantes, b es un m-vector columna de constantes y optimizar significa maximizar para algunos objetivos y minimizar para otros.

Es evidente que los objetivos se pueden reagrupar de forma tal que (21) asuma la forma:

$\max Z_1 = C_1X$

$\min Z_2 = C_2X$ ……..(22)

Sujeto a: $AX \leq b$

$X \geq 0$

Donde $Z_1 = (Z_1, Z_2, \ldots Z_{k1})^t$, $Z_2 = (Z_{K1+1}, Z_{k+2}, \ldots K_k)^t$

Son vectores objetivos, C_1 es una k_1x n matriz y C_2 es una $(k–k_1)$x n matriz de constantes, los otros elementos son los mismos del modelo (21).

El método de metas consiste de transformar las funciones objetivos, excepto una, en restricciones fijando una ruta para cada una de ellas. Las metas deberán ser dadas por el decisor, por ejemplo: “el ingreso no debe ser inferior a P_1 para las

funciones de maximización y en la forma" el costo total no debe ser superior a P_2. Así el modelo (22) se transforma, sin pérdida de generalidad en:

$$maxZ_1 = C_1^1 X$$

Sujeto a: $C_1X \geq P_1$

$C_2X \leq P_2$

$AX \leq b$ …….(23)

$X \geq 0$

Donde C_1^1 es la primera fila de la matriz C_1 y C_1^1 es la matriz C_1 con la primera fila excluida.

El modelo (23) es un problema que puede resolverse con cualquier método de la programación lineal. Sin embargo debemos señalar algunos inconvenientes de este método entre los cuales podemos resaltar:

1.- Pese a la experiencia que pueda tener el decimal para determinar las metas, no se puede garantizar que la solución obtenida sea necesariamente una solución eficiente.

2.- La elección de una función objetivo en el modelo (23) es subjetiva, y evidentemente proporcionará una solución óptima con respecto al objetivo elegido y no se garantiza que sea una solución eficiente.

3.- Al proporcionar las metas se considera como restricciones que no se encuentran interrelacionadas con los demás objetivos con lo cual se desvirtuar del contexto real del problema original planteado en el que los objetivos están relacionados entre si. Aquí es donde para optar por mejores formulaciones haremos uso del método difuso.

En la formulación difusa además de introducir metas en el mismo sentido que en la metodología de programación con metas se debe introducir las tolerancias y la función de pertenencia [10, 19, 59]. La diferencia de los diversos métodos difusos existentes en la literatura radica en la forma de determinación de las metas y tolerancias así como en la definición de las funciones de pertenencia.

En el presente trabajo haremos uso de funciones de pertenencia seccionalmente lineales o denominadas simplemente “lineales”

IV. RESULTADOS

El presente trabajo responderá a la interacción de varios factores, los mismos que se denominaran el diagnostico sobre el proceso de fertilización, consideramos que los factores mas importantes al parecer de los especialistas en esta área del campo agrario son:

a) Factor suelo: PH, materia orgánica, N, P, K, disponibles, textura y condiciones físicas

b) Factor clima: precitación y temperatura, por tanto si el cultivo es bajo riego y/o secano. Así como los riegos de exposición a sequías, granizadas y heladas.

c) Factor cultivo: potencialidad de producción, las extracciones de nutrientes, variedades (nativa, mejorada), y las expectativas del agricultor.

d) Factor Tecnología y manejo: uso de semilla mejorada, pesticidas, herbicidas, maquinaria, etc.

e) Factor económico: entendiendo que la meta del productor no es solo el máximo rendimiento, sino la mínima inversión y la obtención de la máxima rentabilidad.

Por otro lado cabe hacerse las siguientes preguntas:

¿Cuánto? → Dosis o fórmula de fertilización

¿Qué? → Tipo de fertilizante a aplicar

¿Cómo? → Modalidad de aplicación

¿Cuándo? → Oportunidad y momento de su aplicación

El presente trabajo tiene como finalidad dar respuesta a los dos primeras interrogantes, considerando que las dos últimas son decididas por el gerente y/o el técnico en agricultura y la decisión no es sujeto de mucho análisis y en algunos casos es mucho más de carácter personal.

En esta parte se hará una adaptación de los diferentes métodos estudiados en la capitulo anterior a fin de resolver nuestro problema de fertilización agrícola.

4.1 Adaptación de los modelos al caso de minimización

4.1.1 Adaptación de la aproximación de Lai - Hwang

Para el caso de un problema de programación lineal de minimización (minimizar el costo total) tendremos el siguiente problema equivalente a partir de la Figura 8

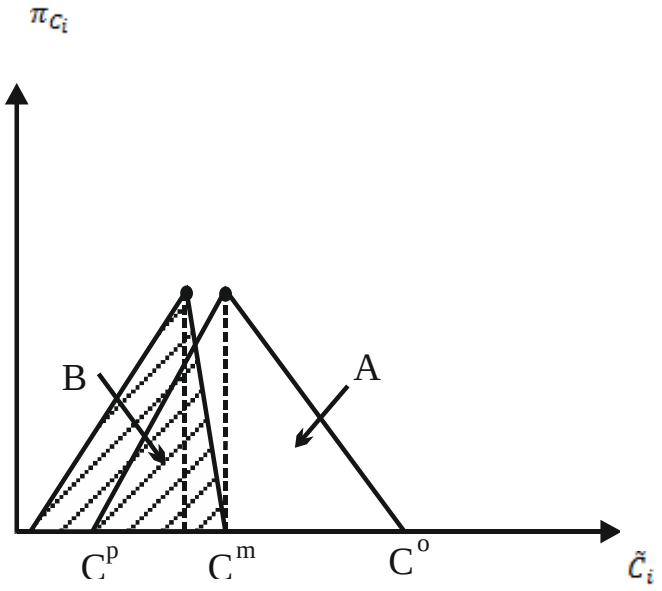


Figura 12: Estrategia para resolver min $\tilde{C}X$.

donde debemos maximizar la región B, minimizar la región A y minimizar el valor más posible, así obtendremos el siguiente problema.

$$\begin{cases} \max Z_1^l = (C^m - C^p)^t X \\ \qquad X \in F \\ \min Z_2^l = (C^o - C^m)^t X \\ \qquad X \in F \\ \min Z_3^l = (C^m)^t X \\ \qquad X \in F \end{cases} \tag{24}$$

Problema que puede ser resuelto usando el procedimiento de Zimmermann. Para ello es necesario construir las funciones de pertenencia para $\mu_{Z_1^l}$, $\mu_{Z_2^l}$, $\mu_{Z_3^l}$, de la manera siguiente:

Para Z_1^l

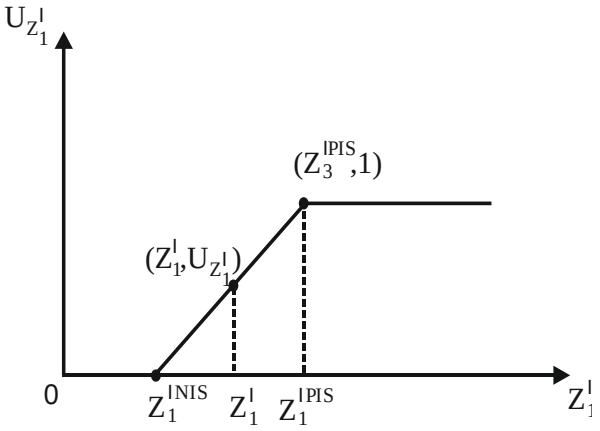


Figura 13: Función de pertenencia para Z_1^l

$$\mu_{Z_1^l} - 0 = \frac{1-0}{Z_1^{lPIS} - Z_1^{lNIS}} \left(Z_1^l - Z_1^{lNIS}\right)$$

$$\mu_{Z_1^l} = \frac{Z_1^l - Z_1^{lNIS}}{Z_1^{lPIS} - Z_1^{lNIS}}$$

Luego tendremos la siguiente función de pertenencia:

$$\mu_{Z_1^l} = \begin{cases} 0 \quad , & \text{Si } Z_1^l < Z_1^{lNIS} \\ \frac{Z_1^l - Z_1^{lNIS}}{Z_1^{lPIS} - Z_1^{lNIS}}, & Si\ Z_1^{lNIS} \leq Z_1^l \leq Z_1^{lPIS} \\ 1 \quad , & Si\ Z_1^l > Z_1^{lPIS} \end{cases} \tag{25}$$

Realizamos lo mismo para Z_j^l, esto es para j = 2,3

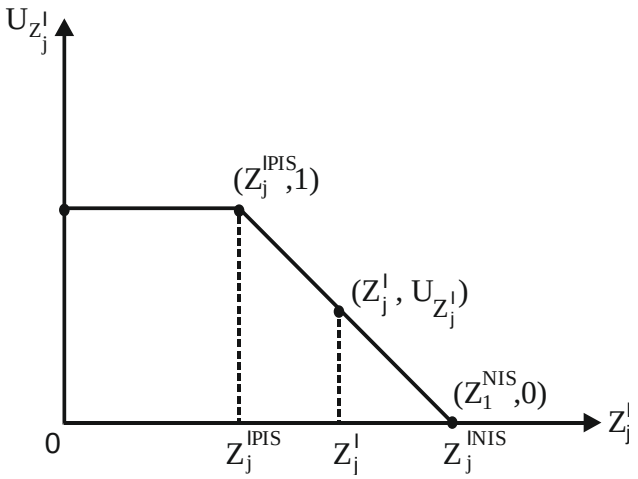


Figura 14: Función de pertenencia para Z_j^l; j = 2,3

Del gráfico se obtiene que:

$$\mu_{Z_j^l} = \begin{cases} 1 \quad , & \text{Si } Z_j^l < Z_j^{lPIS} \\ \frac{Z_j^l - Z_j^{lNIS}}{Z_j^{lPIS} - Z_j^{lNIS}}, & Si\ Z_j^{lPIS} \leq Z_j^l \leq Z_j^{lNIS} \\ 0 \quad , & Si\ Z_j^l > Z_j^{lNIS} \end{cases} \tag{26}$$

Luego tendremos que resolver el problema siguiente:

$max \propto$

Sujeto a : $\mu_{Z_1^l} \geq \propto$

$\mu_{Z_2^l} \geq \propto$ (27)

$\mu_{Z_3^l} \geq \propto$

$X \in F$ con $\propto \in [0,1]$

Necesitamos determinar $\mu_{Z_j^l} \geq \propto$; j = 1,2,3

Esto es:

Para $\mu_{Z_1^l} \geq \propto$, considerando $Z_j^l = \left(C_i^m - C_i^p\right)^t X$

Tendremos de $\frac{Z_1^l - Z_1^{lNIS}}{Z_1^{lPIS} - Z_1^{lNIS}} \geq \propto$

$$\frac{\left(C_i^m - C_i^p\right)^t X - Z_1^{lNIS}}{Z_1^{lPIS} - Z_1^{lNIS}} \geq \propto \Leftrightarrow \left(C_i^m - C_i^p\right)^t X - Z_1^{lNIS} \geq \propto \left(Z_1^{lPIS} - Z_1^{lNIS}\right)$$

$$\Leftrightarrow \left(C_i^m - C_i^p\right)^t X - \propto \left(Z_1^{lPIS} - Z_1^{lNIS}\right) \geq Z_1^{lNIS} \qquad (28)$$

Similarmente realizamos el cálculo para $\mu_{Z_2^l}$ y $\mu_{Z_3^l}$

Para $\mu_{Z_2^l} \geq \propto$ considerando $Z_2^l = (C_i^m)^t X$ en

$\frac{Z_2^l - Z_2^{lNIS}}{Z_2^{lPIS} - Z_j^{lNIS}} \geq \propto$ tendremos que: $\frac{(C_i^m)^t X - Z_2^{lNIS}}{Z_2^{lPIS} - Z_2^{lNIS}} \geq \propto$

$$\Leftrightarrow (C_i^m)^t X - Z_2^{lNIS} \geq \propto \left(Z_2^{lPIS} - Z_2^{lNIS}\right)$$

$$\Leftrightarrow (C_i^m)^t X - \propto \left(Z_2^{lPIS} - Z_2^{lNIS}\right) \geq Z_2^{lNIS} \qquad (29)$$

En $\mu_{Z_3^l} \geq \propto$ considerando $Z_3^l = (C_i^o - C_i^m)^t X$ en

$\frac{Z_3^l - Z_3^{lNIS}}{Z_3^{lPIS} - Z_3^{lNIS}} \geq \propto \quad \Leftrightarrow \quad \frac{(C_i^o - C_i^m)^t X - Z_3^{lNIS}}{Z_3^{lPIS} - Z_3^{lNIS}} \geq \propto$

$$\Leftrightarrow (C_i^o - C_i^m)^t X - Z_3^{lNIS} \geq \propto \left(Z_3^{lPIS} - Z_3^{lNIS}\right)$$

$$\Leftrightarrow (C_i^o - C_i^m)^t X - \propto \left(Z_3^{lPIS} - Z_3^{lNIS}\right) \geq Z_3^{lNIS} \qquad (30)$$

Usando (28), (29) y (30) tendremos el siguiente problema a resolver:

$max \propto$

Sujeto a:

$$\left(C_i^m - C_i^p\right)^t X - \propto \left(Z_1^{lPIS} - Z_1^{lNIS}\right) \geq Z_1^{lNIS}$$

$$(C_i^m)^t X - \propto \left(Z_2^{lPIS} - Z_2^{lNIS}\right) \geq Z_2^{lNIS} \qquad (31)$$

$$(C_i^o - C_i^m)^t X - \propto \left(Z_3^{lPIS} - Z_3^{lNIS}\right) \geq Z_3^{lNIS}$$

$$X \in F$$

$$\propto \in [0,1]$$

Previamente debemos determinar Z_j^{lPIS} y Z_j^{lNIS} con j = 1,2,3, donde ellos se obtienen al resolver en cada caso los siguientes problemas:

$$\begin{cases} Z_1^{lPIS} = maxZ_1^l = \left(C_i^m - C_i^p\right)^t X \\ \qquad\qquad Si\ X \in F \\ Z_1^{lNIS} = minZ_1^l = \left(C_i^m - C_i^p\right)^t X \\ \qquad\qquad Si\ X \in F \end{cases} \qquad (32)$$

$$\begin{cases} Z_2^{lPIS} = minZ_2^l = \left(C_i^o - C_i^m\right)^t X \\ \qquad\qquad Si\ X \in F \\ Z_2^{lNIS} = maxZ_2^l = \left(C_i^o - C_i^m\right)^t X \\ \qquad\qquad Si\ X \in F \end{cases} \qquad (33)$$

$$\begin{cases} Z_3^{lPIS} = minZ_3^l = \left(C_i^m\right)^t X \\ \qquad\qquad Si\ X \in F \\ Z_3^{lNIS} = maxZ_3^l = \left(C_i^m\right)^t X \\ \qquad\qquad Si\ X \in F \end{cases} \qquad (34)$$

Finalmente con estos datos resolvemos el problema (31), el mismo que será la solución para el caso en que el problema sea de minimización.

4.1.2 Adaptación de la Aproximación de Leberling:

Leberling [25]; propone que (20) sea transformado a un problema con metas difusas, en donde, tanto las metas así como las tolerancias sean calculadas a partir del modelo original y que las correspondientes funciones de pertenencia sean

lineales. Entonces las metas difusas están caracterizadas por sus respectivas funciones de pertenencia dadas de la manera siguiente:

$$\mu_j(\mathrm{X}) = \begin{cases} 0 \quad , & \text{para } Z_j(X) \leq Z_j^m \\ \dfrac{Z_j(X) - Z_j^m}{d_j^0}, & \text{para } Z_j^m \leq Z_j(X) \leq Z_j^0 \\ 1 \quad , & \text{para } Z_j(X) \geq Z_j^0 \end{cases} \tag{35}$$

Y su gráfica correspondiente se muestra en la Figura (15).

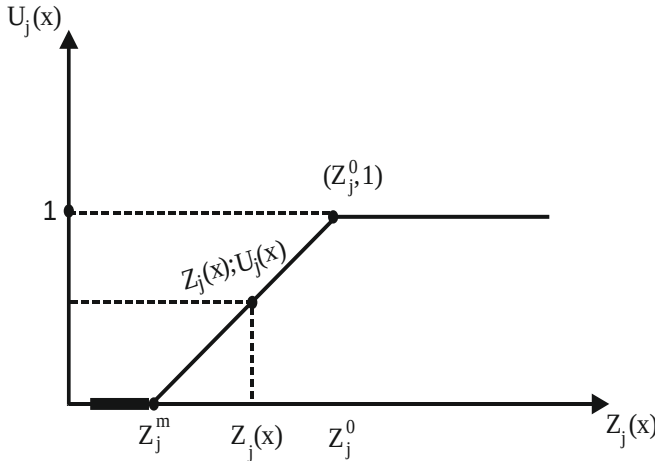


Figura 15: Función de pertenencia de las metas difusas (max)

Donde:

$$\mu_j(\mathrm{X}) = \frac{1-0}{z_j^0 - z_j^{\mathrm{m}}}\left(Z_j(X) - Z_j^{\mathrm{m}}\right)$$

$\mu_j(\mathrm{X}) = \frac{z_j(X) - z_j^{\mathrm{m}}}{z_j^0 - z_j^{\mathrm{m}}}$ donde si denotamos

$d_j^0 = Z_j^0 - Z_j^{\mathrm{m}}$ tendremos

$\mu_j(\mathrm{X}) = \frac{z_j(X) - z_j^{\mathrm{m}}}{d_j^0}$, donde definimos

Para j = 1, ……. K

$$Z_j^m = min\left(Z_1(X_1^0), \dots Z_j(X_{j+1}^0), \dots Z_j(X_k^0)\right)$$

$$Z_j^o = Z_j(X_j^0)$$

$$d_j^o = Z_j^0 - Z_j^m > 0$$

Donde X_j^0 viene a ser la solución óptima de la j-ésima función objetivo.

Finalmente el problema (23) usando el método de aproximación de Zimmermann es resuelto.

OBSERVACION:

Si el problema (20) corresponde a un problema de minimización, en ese caso las metas difusas estarán caracterizados por:

$$\mu_j(\mathrm{X}) = \begin{cases} 0 \quad , & si\ Z_j^{\mathrm{l}}(X) \geq Z_j^{\mathrm{l}m} \\ \frac{Z_j^{\mathrm{l}m} - Z_j^{\mathrm{l}}(\mathrm{X})}{Z_j^{\mathrm{l}o} - Z_j^{\mathrm{l}m}}, & si\ Z_j^{\mathrm{l}0} \leq Z_j^{\mathrm{l}}(X) \leq Z_j^{\mathrm{l}m} \\ 1 \quad , & si\ Z_j^{\mathrm{l}}(X) \leq Z_j^{\mathrm{l}0} \end{cases} \qquad \text{.....(36)}$$

Cuya gráfica se muestra en la Figura 16.

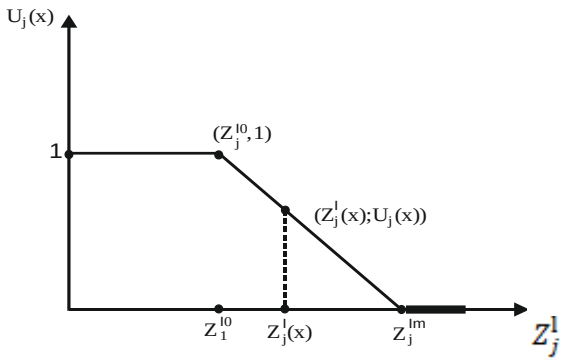


Figura 16: Función de pertenencia de las metas difusas (min)

De donde se tiene que:

$$\mu_j(\mathrm{X}) - 0 = \frac{1-0}{Z_j^{\mathrm{l}o} - Z_j^{\mathrm{l}m}}\left(Z_j^{\mathrm{l}}(X) - Z_j^{\mathrm{l}m}\right)$$

$$\mu_j(\mathrm{X}) = \frac{z_j^{\mathrm{l}}(X) - z_j^{\mathrm{lm}}}{z_j^{\mathrm{lo}} - z_j^{\mathrm{lm}}}$$

Y se define para j = 1,….,K

$$Z_j^{\mathrm{lm}} = max\left(Z_j^{\mathrm{l}}(X_1^0), \ldots Z_j^{\mathrm{l}}(X_{j-1}^0), Z_j^{\mathrm{l}}(X_{j+1}^0), \ldots Z_j^{\mathrm{l}}(X_K^0)\right)$$

$$Z_j^{\mathrm{l0}} = Z_j^{\mathrm{l}}(X_j^0)$$

$$d_j^{\mathrm{l0}} = Z_j^{\mathrm{l0}} - Z_j^{\mathrm{lm}}$$

Con lo cual (30) queda expresado como:

$$\mu_j(\mathrm{X}) = \begin{cases} 0 \quad , & si\, Z_j^{\mathrm{l}}(X) \geq Z_j^{\mathrm{lm}} \\ \dfrac{Z_j^{\mathrm{lm}} - Z_j^{\mathrm{l}}(\mathrm{X})}{Z_j^{\mathrm{l0}} - Z_j^{\mathrm{lm}}}, & si\ Z_j^{\mathrm{l0}} \leq Z_j^{\mathrm{l}}(X) \leq Z_j^{\mathrm{lm}} \\ 1 \quad , & si\, Z_j^{\mathrm{l}}(X) \leq Z_j^{\mathrm{lm}} \end{cases}$$

Y finalmente usando el método de aproximación de Zimmermann el problema de minimización es resuelto.

Como en nuestro caso el problema de fertilización agrícola se ajusta al modelo de minimización, ilustraremos el método planteado a un cultivo de patatas, sin usar un espectro amplio de fertilizantes para hacer más entendible la parte teórica, obviamente con el programa computacional que se construirá en el presente trabajo se pueden incrementar el número de fertilizantes, pues los cálculos se hará de manera rápida en el ordenador.

4.1.3 Ejemplo: Se dispone de una hectárea de terreno agrícola en la costa del Perú, en el cual se desea sembrar patatas, los resultados del análisis de suelos del terreno se muestra en la Tabla 3, si se debe aplicar fertilizantes nitrogenados (N), fosfóricos (P) y potásicos (k) para complementar los nutrientes que aporta el terreno, ¿cuál será la combinación de fertilizantes a utilizar y el costo mínimo del

proceso de fertilización? Si además se dispone en el mercado de los fertilizantes que se muestran en la Tabla 4.

Como datos adicionales en el proceso de fertilización se dispondrá de una tecnología media y se espera que la producción de patatas sea de 25 TM/Ha, la extracción del suelo de N, P_2O_5 y K_2O en Kg/Ha es de 120 – 60 – 200; los coeficientes del uso de nutrientes del suelo (E1) es de 0.20 – 0.20 – 0.40 y los coeficientes del uso del nutriente de fertilizante (F3) son 0.70 – 0.25 – 0.80

Tabla 3: Analisis de Suelos

Nº Lote	Suelo	PH	% M.O	%N	P Disponible ppm	K Disponible ppm
1	Limoso	6.8	1.40	0.07	28	98

Fuente: Facultad de Ciencias Agrarias .

Tabla 4: Fertilizantes disponibles en el mercado

Fertilizante	% N	% P	% K	Costo por Kg. en N.S.
Super Guano	20	20	15	(1, 1.2, 1.5)
Guano de la Isla	10	10	2	(0.8, 1, 1.2)
Fosfato de Amónico	16	48	0	(1, 1.15, 1.30)

Fuente: Propia

Solución del problema:

Se hará uso de la fórmula general de abonamiento siguiente:

$$Q_{(\ )} = \frac{E - SF1 - MF2}{F3} \qquad (37)$$

Con ella determinaremos la cantidad de nutrientes (N, P_2O_5 y K_2O) que se requiere en cada caso, quedando estos determinados por:

$$Q_{(N)} = \frac{E - SF1 - MF2}{F3} \quad (38)$$

$$Q_{(P_2O_5)} = \frac{E - SF1 - MF2}{F3} \quad (39)$$

$$Q_{(K_2O)} = \frac{E - SF1 - MF2}{F3} \quad (40)$$

Donde:

E = Extracción del fertilizante del suelo

S = Aporte del suelo

F1 = Coeficiente del uso del nutriente del suelo.

F2 = Coeficiente de uso de la materia orgánica (M.O) aplicada.

F3 = Coeficiente del uso del nutriente del fertilizante.

Previamente determinaremos el aporte de nutrientes del suelo, ello en base al análisis de suelos y resultados obtenidos.

1.- Para Nitrógeno

Cálculo del %N – Total = (% M.O)/20

= (1.40)/20

= 0.07%N – Total

% = ppm/10000

0.07 = ppm/10000 entonces ppm N = 700

ppm (2) = kg/Ha, (donde el valor 2 es el factor para suelo teórico)

700 (2) = [1400 kg/Ha] por año

Cálculo en base al coeficiente de mineralización:

Costa 2 – 3 % se considera $\bar{x}$ = 2.5%

Sierra 1 – 3 % se considera $\bar{x}$ = 2%

En nuestro caso es para la costa: al 2.5%

1400 (0.0025) = 35 kg/Ha por año

Luego entonces el aporte del suelo por campaña de seis meses, será la mitad para el medio año, esto es 17.5 kg/Ha

2.- Para Fósforo

P = 28 ppm (alto)

ppm (2) = kg/Ha entonces 28(2) = 56 kg/Ha

$P \xrightarrow[\leftarrow]{2.29} P_2O_5$

Entonces (56 kg/Ha) (2.29) = 128 kg P_2O_5/Ha, será el aporte del suelo

3.- Para Potasio

K = 98 ppm (bajo)

ppm (2) = kg/Ha entonces 98(2) = 196 kg/Ha

$P \xrightarrow[\leftarrow]{1.2} K_2O$

Entonces (196 kg/Ha) (1.2) = 235 kg K_2O/Ha, será el aporte del suelo

Haciendo los reemplazos de los datos dados y encontrados más aun considerando F2 = 0 (porque no se ha añadido materia orgánica), en las fórmulas de abonamiento en cada caso se obtendrá:

$$Q_{(N)} = \frac{120 - 17.5\ x\ 0.20 - 1.40\ x\ 0}{0.70} = 166.6 \approx 167\ \text{kg/Ha}$$

$$Q_{(P_2O_5)} = \frac{60 - 128\ x\ 0.20 - 1.40\ x\ 0}{0.25} = 137.28 \approx 137\ \text{kg/Ha}$$

$$Q_{(K_2O)} = \frac{200 - 235\ x\ 0.40 - 1.4\ x\ 0}{0.80} = 132.5 \approx 133\ \text{kg/Ha}$$

Según expertos en agricultura los resultados a considerar se muestran en la tabla 5

Tabla 5: Resultados

	M kg/Ha	P_2O_5 Kg/Ha	K_2O Kg/Ha
Fórmula Calculada	167	137	133
Fórmula Recomendada	170	140	135

Fuente: Facultad de Ciencias Agrarias

Con las fórmulas recomendados formularemos el modelo, entendiendo que en este caso los costos son variables (difusos) se tiene los valores.

C_i^m = Valor mas posible (posibilidad = 1)

C_i^o = Valor más optimista

C_i^p = Valor más pesimista

π_i = grado de ocurrencia de un evento

De la Figura 9 se tiene que para el caso de minimización:

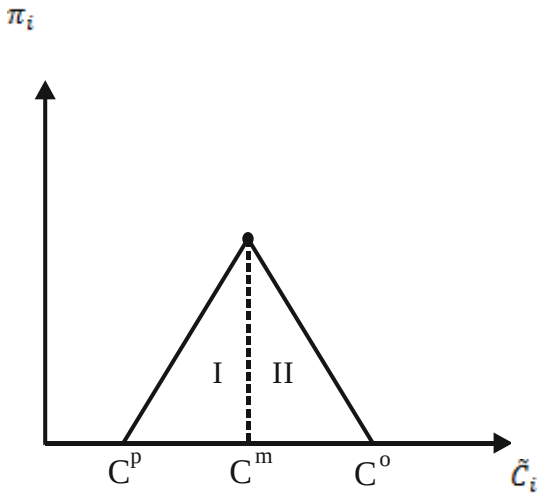


Figura 17: Distribución posibilística de $\tilde{C}_i$

I = Región optimista debe maximizarse

II = Región pesimista debe minimizarse.

Esto es:

$$max \propto \left(C^m - C^p\right)^t X$$

$$X \in F$$

$$min\left(C^o - C^m\right)^t X \qquad (41)$$

$$X \in F$$

$$min\left(C^m\right)^t X$$

$$X \in F$$

Con los datos del problema planteamos el modelo siguiente:

$$\min W = \sum_{i=1}^{3} \left(C_i^m X_i, C_i^p X_i, C_i^o X_i\right)$$

Sujeto a:

$$X \in F$$

O $\min W = ((C^m)^t X, (C^p)^t X, (C^o)^t X)$

Sujeto a:

$$X \in F$$

Asumiendo además una cantidad mínima y máxima requerida para el caso de nitrógeno (N), Fósforo (P) y Potasio (K) en el proceso de fertilización y considerando los costos difusos, es decir $\tilde{C}_1, \tilde{C}_2, \tilde{C}_3$ donde:

$\tilde{C}_1$= (1, 1.2, 1.5)

$\tilde{C}_2$= (0.9, 1, 1.2)

$\tilde{C}_3$= (1, 1.15, 1.30)

Tendremos el siguiente problema de fertilización con costos difusos

$$\min W = \tilde{C}_1 X_1 + \tilde{C}_2 X_2 + \tilde{C}_3 X_3$$

Sujeto a:

$150 \leq 0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$ ……..(N)

$140 \leq 0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$ ……..(P) ……(42)

$115 \leq 0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$ ……(K)

$X_1, X_2, X_3 \geq 0$

Donde [150,190], [140,160], [115,155] son las cantidades mínimas y máximas requeridas de Nitrógeno, Fósforo y Potasio requeridas para este cultivo de patatas.

Solución I.- Usando el método de aproximación de Lai-Hwang planteamos el problema (42) en la forma siguiente:

$$maxZ_1^1 = \left(C_i^m - C_i^p\right)^t X$$

Sujeto a:

$$X \in F$$

$$minZ_2^1 = (C_i^o - C_i^m)^t X \qquad \text{……(43)}$$

Sujeto a:

$$X \in F$$

$$minZ_3^1 = (C_i^m)^t X$$

Sujeto a:

$$X \in F$$

O equivalentemente:

$$maxZ_1^1 = \left(C_i^m - C_i^p\right)^t X$$

$$minZ_2^1 = (C_i^o - C_i^m)^t X \qquad \text{……(44)}$$

$$minZ_3^1 = (C_i^m)^t X$$

Sujeto a:

$$X \in F$$

Donde se debe considerar $X \in F$ como el conjunto de restricciones para (N), (P) y (K) obtenidas en (36)

$$C_1^m = 1.2 \qquad C_1^o = 1.5 \qquad C_1^p = 1$$

$$C_2^m = 1 \qquad C_2^o = 1.2 \qquad C_2^p = 0.9$$

$$C_3^m = 1.15 \qquad C_3^o = 1.3 \qquad C_3^p = 1$$

Para luego determinar:

a) $maxZ_1^l = (C_1^m - C_1^p)X_1 + (C_2^m - C_2^p)X_2 + (C_3^m - C_3^p)X_3$

$= (C_1^m - C_1^p)X_1 + (C_2^m - C_2^p)X_2 + (C_3^m - C_3^p)X_3$

⇨ $maxZ_1^l = 0.2X_1 + 0.1X_2 + 0.15X_3$ ……(45)

b) $minZ_2^l = (C_i^o - C_i^m)^t X$

$= (C_1^o - C_1^m)X_1 + (C_2^o - C_2^m)X_2 + (C_3^o - C_3^m)X_3$

$= (1.5 - 1.2)X_1 + (1.2 - 1)X_2 + (1.3 - 1.15)X_3$

⇨ $minZ_2^l = 0.3X_1 + 0.2X_2 + 0.15X_3$ ……(46)

c) $minZ_3^l = (C_i^m)^t X$

$= C_1^m X_1 + C_2^m X_2 + C_3^m X_3$

⇨ $minZ_3^l = 1.2X_1 + 1X_2 + 1.15X_3$ ……(47)

Usando (45), (46) y (47) tendremos el problema multiobjetivo planteado de la forma siguiente:

$maxZ_1^l = 0.2X_1 + 0.1X_2 + 0.15X_3$

$minZ_2^l = 0.3X_1 + 0.2X_2 + 0.15X_3$

$minZ_3^l = 1.2X_1 + 1X_2 + 1.15X_3$ ……(48)

Sujeto a:

$$150 \leq 0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$120 \leq 0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$115 \leq 0.15X_1 + 0.02X_2 + 0.00X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

Calculo de Z_j^{lPIS} y Z_j^{lNIS} con j = 1,2,3

$$Z_1^{lPIS} = maxZ_1^l = 0.2X_1 + 0.1X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

$$Z_1^{lNIS} = minZ_1^l = 0.2X_1 + 0.1X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

$$Z_2^{lPIS} = minZ_2^l = 0.3X_1 + 0.2X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

$$Z_2^{lNIS} = maxZ_2^l = 0.3X_1 + 0.2X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

Finalmente haremos el cálculo para Z_3^{lPIS} y Z_3^{lNIS}

$$Z_3^{lPIS} = minZ_3^l = 1.2X_1 + 1X_2 + 1.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

$Z_3^{lNIS} = max Z_3^l = 1.2X_1 + 1X_2 + 1.15X_3$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.X_3 \leq 155$$

$$X_1, X_2, X_3$$

Al realizar los cálculos se tiene:

$Z_1^{lPIS} = 160$ $\quad$ $Z_1^{lNIS} = 153.33$

$Z_2^{lPIS} = 230$ $\quad$ $Z_2^{lNIS} = 244.54$

$Z_3^{lPIS} = 920$ $\quad$ $Z_3^{lNIS} = 996.30$

Usando los datos de Z_j^{lPIS} y Z_j^{lNIS}, conseguimos las funciones de pertenencia $\mu_{Z_j^l}$ para cada función objetivo, esto es:

a) Para la función Z_1^l objetivo consentimos la función de pertenencia $\mu_{Z_1^l}$ haciendo uso del gráfico que se muestra en la Figura 18.

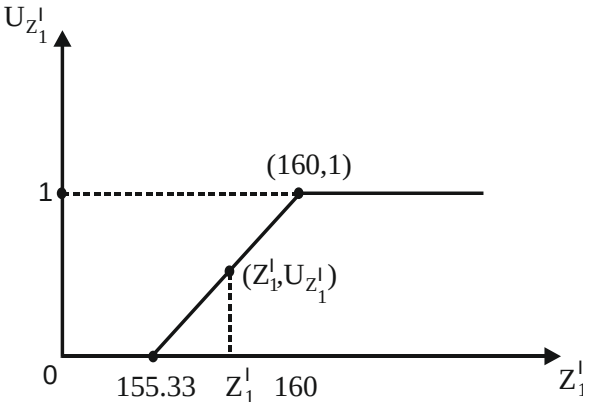


Figura 18: Función de pertenencia para Z_1^l de donde:

$$\mu_{Z_1^l} = \begin{cases} 0 \quad , & \text{Si } Z_1^l < 153.33 \\ \dfrac{Z_1^l - 153.33}{160 - 153.33}, & Si\ 153.33 \leq Z_1^l \leq 160 \\ 1 \quad , & Si\ Z_1^l > 160 \end{cases}$$

O lo que es lo mismo

$$\mu_{Z_1^l} = \begin{cases} 0 \quad , & \text{Si } Z_1^l < 153.33 \\ \dfrac{Z_1^l - 153.33}{6.67}, & Si\ 153.33 \leq Z_1^l \leq 160 \\ 1 \quad , & Si\ Z_1^l > 160 \end{cases}$$

Por otro lado:

$$\mu_{Z_2^l} = \begin{cases} 1 \quad , & \text{Si } Z_2^l < Z_2^{lPIS} \\ \dfrac{Z_2^l - Z_2^{lNIS}}{Z_2^{lPIS} - Z_2^{lNIS}}, & Si\ Z_2^{lPIS} \leq Z_2^l \leq Z_2^{lNIS} \\ 0 \quad , & Si\ Z_2^l > Z_2^{lNIS} \end{cases}$$

$$\mu_{Z_2^1} = \begin{cases} 1 \quad , & \text{Si } Z_2^1 < 230 \\ \dfrac{Z_2^1 - 244.54}{230 - 244.54}, & Si\ 230 \leq Z_2^1 \leq 244.54 \\ 0 \quad , & Si\ Z_2^1 > 244.54 \end{cases}$$

Lo que es lo mismo:

$$\mu_{Z_2^1} = \begin{cases} 1 \quad , & \text{Si } Z_2^1 < 230 \\ \dfrac{Z_2^1 - 244.54}{-14.54}, & Si\ 230 \leq Z_2^1 \leq 244.54 \\ 0 \quad , & Si\ Z_2^1 > 244.54 \end{cases}$$

ó

$$\mu_{Z_2^1} = \begin{cases} 1 \quad , & \text{Si } Z_2^1 < 244.54 \\ \dfrac{244.54 - Z_2^1}{14.54}, & Si\ 230 \leq Z_2^1 \leq 244.54 \\ 0 \quad , & Si\ Z_2^1 > 244.54 \end{cases}$$

Cuya gráfica de la función de pertenencia para Z_2^1 se muestra en la Figura 19.

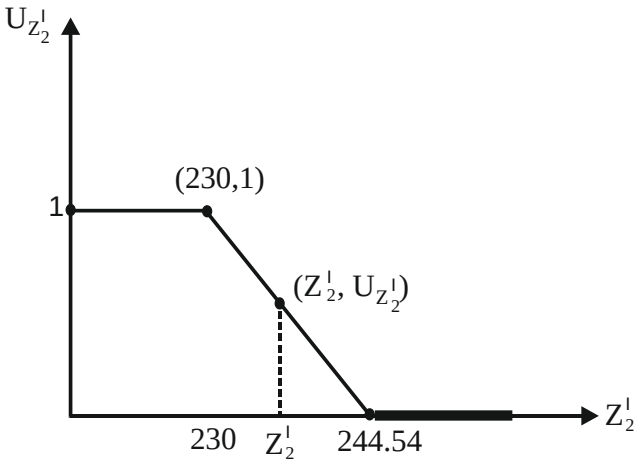


Figura 19: Función de pertenencia para Z_2^1

finalmente calculamos $\mu_{Z_3^1}$, esto es:

$$\mu_{Z_3^l} = \begin{cases} 1 \quad , & \text{Si } Z_3^l < Z_3^{lPIS} \\ \dfrac{Z_3^{lNIS} - Z_3^l}{Z_3^{lNIS} - Z_3^{lPIS}}, & Si\ Z_3^{lPIS} \leq Z_3^l \leq Z_3^{lNIS} \\ 0 \quad , & Si\ Z_3^l > Z_3^{lNIS} \end{cases}$$

$$\mu_{Z_3^l} = \begin{cases} 1 \quad , & \text{Si } Z_3^l < 920.0 \\ \dfrac{996.36 - Z_3^l}{996.36 - 920}, & Si\ 920 \leq Z_3^l \leq 996.36 \\ 0 \quad , & Si\ Z_3^l > 996.36 \end{cases}$$

$$\mu_{Z_3^l} = \begin{cases} 1 \quad , & \text{Si } Z_3^l < 920.0 \\ \frac{996.36 - Z_3^l}{76.36}, & Si\ 920 \leq Z_3^l \leq 996.36 \\ 0 \quad , & Si\ Z_3^l > 996.36 \end{cases}$$

Cuya gráfica se visualiza en la Figura 20.

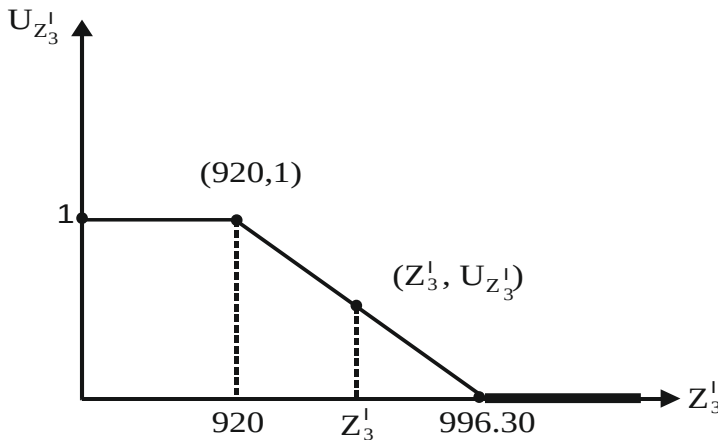


Figura 20: Función de pertenencia para Z_3^l

Con las funciones $\mu_{Z_j^l}$, j = 1,2,3 definidas aplicamos el método de Zimmermann para obtener la solución final del problema multiobjetivo siguiente:

$max \propto$

Si : $\mu_{Z_1^l} \geq \propto$

$$\mu_{Z_2^1} \geq \propto \qquad \text{..……...(49)}$$

$$\mu_{Z_3^1} \geq \propto$$

Sujeto a:

$$X \in F$$

Previamente realizamos los cálculos de $\mu_{Z_1^1} \geq \propto$, $\mu_{Z_2^1} \geq \propto$, y $\mu_{Z_3^1} \geq \propto$ esto es:

CASO I:

Si $\mu_{Z_1^1} \geq \propto \Leftrightarrow \frac{Z_1 - 153.3}{6.67} \geq \propto$

$$\Leftrightarrow Z_1 - 153.33 \geq 6.67 \propto$$

$$\Leftrightarrow Z_1 - 6.67 \propto > 153.33$$

Pero $Z_1 = 0.2X_1 + 0.1X_2 + 0.15X_3$ entonces sustituyendo en la inecuación anterior tendremos:

$$0.2X_1 + 0.1X_2 + 0.15X_3 - 6.67\propto \geq 153.33 \qquad \text{……(50)}$$

CASO II:

Si $\mu_{Z_2^1} \geq \propto \Leftrightarrow \frac{244.54 - Z_2^1}{14.54} \geq \propto$

$$244.54 - Z_2^1 \geq 14.54 \propto$$

$$Z_2^1 + 14.54 \propto \leq 244.54$$

Pero $Z_2^1 = 0.3X_1 + 0.2X_2 + 0.15X_3$

Luego tendremos:

$$0.3X_1 + 0.2X_2 + 0.15X_3 + 14.54\propto \leq 244.54 \qquad \text{……(51)}$$

CASO III:

Si $\mu_{Z_3^1} \geq \propto \Leftrightarrow \frac{996.36 - Z_3^1}{76.36} \geq \propto$

$$996.36 - Z_3^l \geq 76.36 \propto$$

$$Z_3^l + 76.36 \propto \leq 996.36$$

Pero $Z_3^l = 1.2X_1 + 1.X_2 + 1.15X_3$

$1.2X_1 + 1X_2 + 1.15X_3 + 76.36 \propto \leq 996.36$ ……(52)

Usando (50), (51) y (52) debemos resolver el problema siguiente:

$max \propto$

Sujeto a:

$$0.2X_1 + 0.1X_2 + 0.15X_3 - 6.67 \propto \geq 153.33$$

$$0.3X_1 + 0.2X_2 + 0.15X_3 + 14.54 \propto \leq 244.54$$

$$1.2X_1 + 1X_2 + 1.15X_3 + 76.36 \propto \leq 996.36$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$ ……(53)

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

Con $\propto \in [0,1]$, $X_1, X_2, X_3 \geq 0$

Los resultados que se obtienen al resolver (47) son:

$X_1 = 781.38$

$X_2 = 0$

$X_3 = 7.75$

$\propto = 0.61$

Notar que $\propto = 0.61$es un grado de adaptación que esta sobre la media.

Usando X_1, X_2, X_3 calculamos Z_1^l, Z_2^l, Z_3^l y $\mu_{Z_1^l}, \mu_{Z_2^l}, \mu_{Z_s^l}$ de la forma siguiente:

$$Z_1^l = (0.2)(781.38) + (0.10)(0) + (0.15)(7.75) = 157.86$$

$$\mu_{Z_1^l} = \frac{157.86 - 153.33}{6.67} = 0.67$$

Resultados que se muestran en la Figura (21)

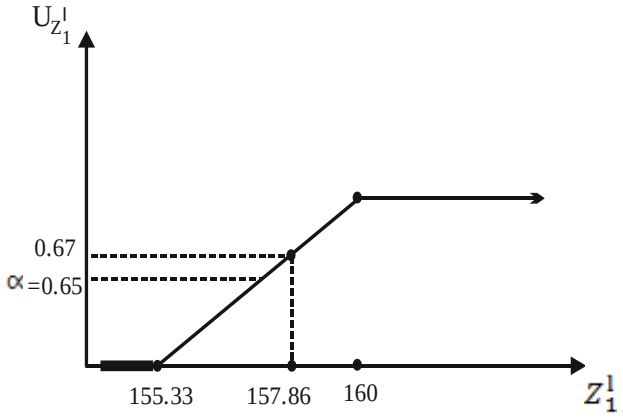


Figura 21: Grado de aceptación para función objetivo Z_1^l

Similarmente tendremos que:

$$Z_2^l = (0.3)(781.38) + (0.2)(0) + (0.15)(7.75) = 235.57$$

$$\mu_{Z_2^l} = \frac{244.54 - 235.57}{14.54} = 0.616$$

$$Z_3^l = (0.2)(781.38) + 1(0) + (1.15)(7.75) = 946.58$$

$$\mu_{Z_s^l} = \frac{996.36 - 946.58}{76.36} = 0.65$$

Notar que los valores $\mu_{Z_j^l}$ cumplen con la condición de ser mayores o iguales que $\propto$.

Finalmente debemos interpretar la solución del problema

$$\min W = X_1\tilde{C}_1 + X_2\tilde{C}_2 + X_3\tilde{C}_3$$

Sujeto a:

$$X \in F$$

Ó

$$\min W = X_1\tilde{C}_1 + X_2C_2 + X_3C_3$$

Sujeto a:

$$150 \leq 0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$140 \leq 0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$115 \leq 0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

Donde X_1, X_2, X_3 ya fueron determinados faltando determinar el número difuso que corresponde a la función objetivo en el entendido de que si los costos unitarios son difusos entonces el costo total es también un número difuso, para ello usamos el producto de un escalar por un número difuso, esto es:

$$\min W = X_1\tilde{C}_1 + X_2\tilde{C}_2 + X_3\tilde{C}_3$$

$$= 781.38\ (1, 1.2, 1.5) + 0\ (0.9, 1, 1.2) + 7.75\ (1, 1.15, 1.30)$$

mín W = (789.13, 953.11, 1182.77)

Cuya representación difusa es aproximadamente

$$\pi_{W_i}$$

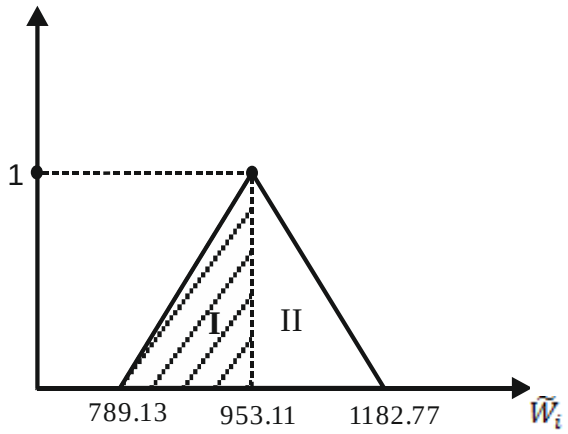


Figura 22: Distribución posibilística de $\tilde{W}_i$ maximizando la región I y se ha minimizado la región, lo cual era el objetivo del problema.

Solución II.- Usando el método de aproximación de Leberling planteamos el problema (36) en la forma siguiente:

$$\min W = \tilde{C}_1 X_1 + \tilde{C}_2 X_2 + \tilde{C}_3 X_3$$

Sujeto a:

$150 \leq 0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$ …….. (N)

$140 \leq 0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$ …….. (P) …… (54)

$115 \leq 0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$ ……... (K)

$X_1, X_2, X_3 \geq 0$

El mismo que usando la aproximación de Lai y Hwang se transforma en el siguiente problema:

$$max Z_1^{10} = \left(C_i^m - C_i^p\right)^t X$$

Sujeto a:

$$X \in F$$

$$minZ_2^{lo} = (C_i^o - C_i^m)^t X$$

Sujeto a:

$$X \in F$$

$$minZ_3^{lo} = (C_i^m)^t X$$

Sujeto a:

$$X \in F$$

O

$$maxZ_1^{lo} = \left(C_i^m - C_i^p\right)^t X$$

$$minZ_2^{lo} = (C_i^o - C_i^m)^t X$$

$$minZ_3^{lo} = (C_i^m)^t X$$

Sujeto a:

$$X \in F$$

En el caso de nuestro ejemplo debemos resolver:

$$maxZ_1^{lo} = 0.2X_1 + 0.1X_2 + 0.15X_3 \quad \text{..... (I)}$$

$$minZ_2^{lo} = 0.3X_1 + 0.2X_2 + 0.15X_3 \quad \text{..... (II)}$$

$$minZ_3^{lo} = 1.2X_1 + 1.X_2 + 1.15X_3 \quad \text{..... (III)}$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160 \quad \text{..... (IV)}$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

A partir de este modelo expresado de (I), (II), (III) y (IV) Leberting aplica su metodología de solución, esto es determinando las metas y posteriormente las funciones de pertenencia.

Notar que el problema planteado corresponde a un problema multiobjetivo. Con funciones objetivos $Z_1^{10}, Z_2^{10}, Z_3^{10}$, las mimas que constituyen las metas a lograr.

Determinamos $Z_1^{10}, Z_2^{10}, Z_3^{10}$, para ello resolvemos cada caso:

CASO A:

$$max Z_1^{10} = 0.2X_1 + 0.1X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.03 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

Cuyos resultados una vez resuelto el problema es:

$X_1^0 = (800, 0, 0)$ con $X_1 = 800$, $X_2 = X_3 = 0$

$Z_1^{10} = 160$

CASO B:

$$minZ_2^{10} = 0.3X_1 + 0.2X_2 + 0.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.03 \leq 155$$

$$X_1, X_2, X_3 \geq 0$$

Cuya solución es:

$X_2^0 = (766.66, 0, 0)$ con $X_1 = 766.66$, $X_2 = X_3 = 0$

$Z_2^{10} = 230$

CASO C:

$$minZ_3^{10} = 1.2X_1 + 1.X_2 + 1.15X_3$$

Sujeto a:

$$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$$

$$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$$

$$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$$

$$0.15X_1 + 0.02X_2 + 0.0X_3 \geq 115$$

$$0.15X_1 + 0.02X_2 + 0.03 \leq 155$$

Con solución:

$X_3^0 = (766.66, 0, 0)$; donde $X_1 = 766.66$, $X_3 = X_2 = 0$

$Z_3^{l0} = 920.00$

De la solución de los Casos A, B y C se obtiene:

$Z_1^{l0} = 160$

$Z_2^{l0} = 230$

$Z_3^{l0} = 920$

Resultados que constituyen las metas a lograr.

Ahora determinamos las tolerancias de cada función objetivo $\left(d_j^{lo} = Z_j^{l0} - Z_j^{lm}\right)$.

Para ello determinamos los Z_j^{lm}.

Si j = 1 => $Z_1^{lm} = min\left(Z_1^l(X_2^0),\ Z_1^l(X_3^0)\right)$

Donde:

a) $Z_1^l(X_2^0) = (0.2)\ (766.66) + (0.1)\ (0) + (0.15)\ (0)$

$Z_1^l(X_2^0) = 153.33$

b) $Z_1^l(X_3^0) = (0.2)\ (766.66) + (0.1)\ (0) + (0.15)\ (0)$

$Z_1^l(X_3^0) = 153.33$

De a y b $Z_1^{lm} = min\ (153.33, 153.33) = 153.33$

Si j = 2 => $Z_2^{lm} = max\left(Z_2^l(X_1^0),\ Z_2^l(X_3^0)\right)$

Donde:

c) $Z_2^l(X_1^0) = (0.3)X_1 + 0.2\ X_2 + 0.15X_3$

$= (0.3)\ (800) + (0.2)\ (0) + (0.15)\ (0)$

$Z_2^l = 240$

d) $Z_2^l(X_3^0) = (0.3)\ (766.66) + (0.2)\ (0) + (0.15)\ (0)$

$Z_2^1(X_3^0) = 229.99$

De c y d tendremos $Z_2^{1m} = max$ (240, 229.99) = 240

Si j = 3 => $Z_3^{1m} = max\left(Z_3^1(X_1^0), Z_3^1(X_2^0)\right)$

e) $Z_3^1(X_1^0) = 0.2X_1 + 1.X_2 + 1.15X_3$

$= (1.2)\ (800) + 1\ (0) + (0.15)\ (0)$

$Z_3^1(X_1^0) = 960$

f) $Z_3^1(X_2^0) = (1.2)X_1 + 1.X_2 + (1.15)X_3$

$= (1.2)\ (766.66) + 1\ (0) + (0.15)\ (0)$

$Z_3^1(X_2^0) = 919.99$

De e y f tendremos $Z_3^{1m} = max$ (960.00, 919.99) = 960.00

⇨ $Z_3^{1m} = 960.00$

Como siguiente paso definimos la función de pertenencia para Z_1^1 (Figura 23).

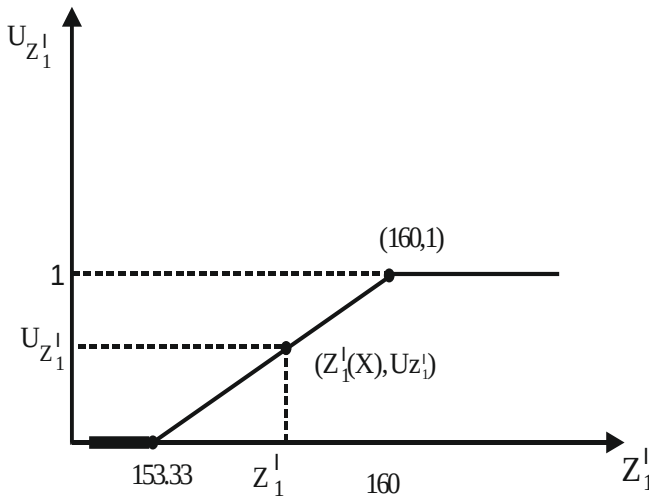


Figura 23: Función de pertenencia para $Z_1^1(X)$.

$$\mu_{Z_1^l} = \begin{cases} 0 \quad , & si\ Z_1^l \leq 153.33 \\ \dfrac{Z_1^l(X) - 153.33}{6.67}, & si\ \ 153.33 \leq Z_1^l \leq 160 \\ 1 \quad , & si\ Z_1^l \geq 160 \end{cases}$$

Hacemos lo mismo para Z_2^l (Figura 24) y Z_3^l (Figura 25)

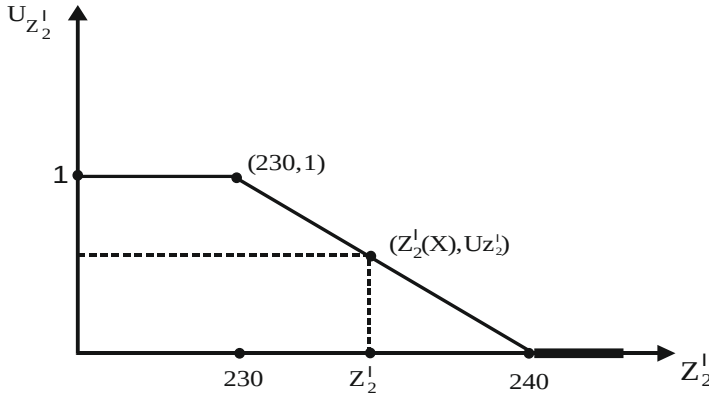


Figura 24: Función de pertenencia para Z_2^l donde:

$$\mu_{Z_2^l} = \frac{1}{-10}\left(Z_2^l - 240\right) = \frac{240 - Z_2^l}{10}$$

Luego:

$$\mu_{Z_2^l} = \begin{cases} 1 \quad , & si\ Z_2^l \leq 230 \\ \dfrac{240 - Z_2^l}{10}, & si\ \ 230 \leq Z_2^l \leq 240 \\ 0 \quad , & si\ Z_2^l \geq 240 \end{cases}$$

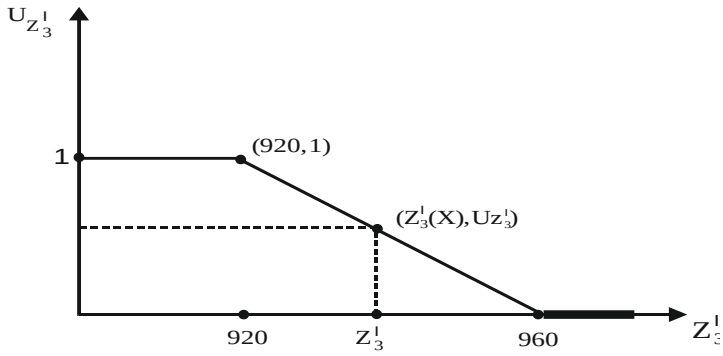


Figura 25: Función de pertenencia para Z_3^l del gráfico se tiene que:

$$\mu_{Z_s^1} = \begin{cases} 1 \quad , & si\ Z_3^1 \leq 920 \\ \dfrac{960 - Z_3^1}{40}, & si\ 920 \leq Z_3^1 \leq 960 \\ 0 \quad , & si\ Z_3^1 \geq 960 \end{cases}$$

Con estos resultados determinamos

$\mu_{Z_3^1} \geq \propto$ j = 1,2,3

Es decir debemos determinar

$\mu_{Z_1^1} \geq \propto$

$\mu_{Z_2^1} \geq \propto$

$\mu_{Z_3^1} \geq \propto$

CASO I:

Si $\mu_{Z_1^1} = \frac{Z_1^1 - 153.33}{6.67} \geq \propto \Rightarrow Z_1^1 - 6.67 \propto \geq 153.33$

Pero $Z_1^1(X) = 0.2X_1 + 0.1X_2 + 0.15X_3$

Entonces tendremos:

$0.2X_1 + 0.1X_2 + 0.15X_3 - 6.67 \propto \geq 153.33$ ….(*1)

CASO II:

Si $\mu_{Z_2^1} = \frac{240 - Z_2^1}{10} \geq \propto \Rightarrow Z_2^1(X) + 10 \propto \leq 240$

Pero $Z_2^1(X) = 0.3X_1 + 0.2X_2 + 0.15X_3$

Entonces se tiene:

$0.3X_1 + 0.2X_2 + 0.15X_3 + 10 \propto \leq 240$ ….(*2)

CASO III:

Si $\mu_{Z_3^1} = \frac{960 - Z_3^1}{40} \geq \propto \Rightarrow Z_3^1 + 40 \propto \leq 960$

Donde $Z_3^1(X) = 1.2X_1 + 1.X_2 + 1.15X_3$

Entonces se tiene:

$1.2X_1 + 1.X_2 + 1.15X_3 + 40 \propto \leq 960$ ….(*3)

Haciendo uso de (*1), (*2), (*3) se tiene el siguiente modelo a resolver:

$max \propto$

Sujeto a:

$0.2X_1 + 0.1X_2 + 0.15X_3 - 6.67 \propto \geq 153.33$

$0.3X_1 + 0.2X_2 + 0.15X_3 + 10 \propto \leq 240$

$1.2X_1 + 1X_2 + 1.15X_3 + 40 \propto \leq 960$

$0.20X_1 + 0.10X_2 + 0.16X_3 \geq 150$

$0.20X_1 + 0.10X_2 + 0.16X_3 \leq 190$

$0.20X_1 + 0.10X_2 + 0.48X_3 \geq 120$

$0.20X_1 + 0.10X_2 + 0.48X_3 \leq 160$

$0.15X_1 + 0.02X_2 + 0X_3 \geq 115$

$0.15X_1 + 0.02X_2 + 0X_3 \leq 155$

Con $\propto \in [0,1]$, $X_1, X_2, X_3 \geq 0$

Al resolver el sistema se tiene:

$X_1 = 783.32$

$X_2 = X_3 = 0$

$\propto = 0.5$

Finalmente determinamos la meta difusa.

$$\tilde{W} = X_1\tilde{C}_1 + X_2\tilde{C}_2 + X_3\tilde{C}_3$$

$\tilde{W}$ = 783.32 (1, 1.2, 1.5) + 0 (0.9, 1, 1.2) + 0 (1, 1.15, 1.3)

$\tilde{W}$ = (783.32, 939.98, 1174.98)

De lo resuelto, se recomienda comprar un solo fertilizante (hiperguano) al precio más barato lo cual se refleja en el costo mínimo de 783.32 N/S.

Tabla 6: Resultados

RESULTADOS COMPARATIVOS		
WINQSB	LAI: X_1 = 781.38 X_2 = 0 X_3 = 7.75 LEBERLING: X_1 = 783.32 X_2 = X_3 = 0	α = 0.61 W=(789.13, 953.11, 1182.77) α = 0.5 $\tilde{W}$ =(783.32,939.98, 1174.98)
LAI-HWANG (FERTIDIF)	X= (766.984,0,9.28992)	α =0.616279 Minimo : = (776.274,931.064,1162.55)
LEBERLING (FERTIDIF)	X= (769.307,0,0,)	α =0.5 Minimo : = (769.307,923.168,1153.96)

Fuente: Propia

4.2 Descripción del Software FERTIDIF

4.2.1 Alcanze del Software

El presente software ha sido diseñado pensando en la utilidad y velocidad de calculo en cuanto se refiere a la planificación de un proceso de fertilización de

un determinado cultivo, entendiendo que en estos tiempos tan cambiantes los costos de los fertilizantes son variables dinámicas que podrían conllevar a cometer errores en los montos estimados para efectuar el proceso, este hecho es mitigado con la aplicación de dichos cálculos usando el presente software, el mismo que es de fácil manipulación por técnicos o especialistas sobre fertilización. Entendiéndose que el presente programa computacional puede ser mejorado de ser necesario con el aporte de expertos en el campo agrario.

4.2.2 Planteamiento del problema

La necesidad de determinar la mejor solución en un problema que ha sido correctamente planteado, entre las soluciones posibles o disponibles, es que conlleva al estudio de teorías y a proponer métodos adecuados en el campo científico donde surge el problema en cuestión.

En el presente trabajo se usará modelos de Programación Matemática, como punto de inicio en el modelamiento del problema de fertilización agrícola, para posteriormente mediante el uso de conceptos teóricos de la teoría de conjuntos y programación difusa abordar de una manera eficiente y elegante la transformación y solución de dichos problemas**.**

En problemas de esta naturaleza, se refiere a la aplicación de la matemática en problemas de la vida real, en la mayoría de las veces es necesario dejar de lado el rigor matemático y proponer soluciones prácticas, ya que con el rigor matemático son pocos o muy pocos los que están capacitados para comprender los fundamentos teóricos del tema a desarrollar, en cambio si la aplicación es práctica y sencilla se tendrá un mayor número de interesados en el tema, el propósito no

es presentar ampulosamente una relación de teoremas, corolarios, lemas y sus demostraciones formales, pues el objetivo central del trabajo es presentar una aplicación práctica de la matemática en la vida real.

Para abordar este tema se tuvo que realizar investigaciones en el campo del análisis de suelos y fertilización de suelos agrícolas, gracias a ello se logró determinar las cantidades requeridas de nutrientes para fertilizar suelos agrícolas, cuando se tiene el propósito de sembrar un determinado cultivo.

4.2.3 Justificacion

El razonamiento del cual partimos fue determinar las cantidades óptimas de fertilizantes así como el costo que ello implicaba, de esta manera se definió a los fertilizantes como variables, las cuales debíamos determinar, los costos de los fertilizantes usualmente se consideran estables en el mercado, por otro lado se considera que la cantidad mínima o máxima de fertilizantes y nutrientes son cantidades aproximadas y no hay una cantidad definida, variando esta de un lugar a otro, con estas informaciones básicas y teniendo otros parámetros considerados en el análisis de suelos como el porcentaje de materia orgánica que aporta el terreno, grado de acidez del terreno (pH), porcentajes de nitrógeno, ppm de fósforo y potasio, se formuló un modelo matemático de programación lineal difusa, el mismo que una vez solucionado nos permitirá elegir la mejor opción para fertilizar un terreno agrícola.

4.2.4 Objetivos

4.2.4.1 Objetivo general

Desarrollar un software de acuerdo a mi tesis de Doctorado.

4.2.4.2 Objetivos especificos

- Dar una solución difuza al costo del fertilizante que se comprara, incluyendo cuanto se debe de usar.
- El usuario podrá modificar la base de datos de fertilizantes.
- Mejorar la fertilización por medio de este software.
- Totalmente interactivo con el usuario.
- Multiplataforma (Linux, Mac Os X, Windows)
- Tenga una buena performarce en tiempo para sus calculos.
- Que sea totalmente mantenible

4.2.5 Especificacion de requisitos

4.2.5.1 El software se desarrollara en C++ con las librerias de Qt4.

4.2.5.2 Para la creación y manipulación de la base de datos se usara SQLite

4.2.8.3 Se implementara 2 metodos que resolverán este problema tal como se especifica en la tesis original.

4.2.6 Requisitos de informacion

4.2.6.1 Información de fertilizantes con sus datos respectivos y costos difuzos.

4.2.6.2 Información de cultivos.

4.2.7 Restricciones del software

4.2.7.1 Unicamente es difuzo on respecto al costo.

4.2.7.2 Solo esta desarrollado para los tres componentes básicos de los cultivos . Estos elementos son P, K , N.

4.2.7.3 No es dependiente de un servidor de base de datos.

4.2.7.4 Su base de datos no es distribuida.

4.2.8 Requisitos funcionales

4.2.8.1 El sistema será modelado en clases

4.2.8.2 Modificación de base de datos de fertilizantes y cultivos.

4.2.8.3 Interacción con Openoffice.

4.2.8.4 Interacción con la impresora.

4.2.9 Diagramas de casos de uso

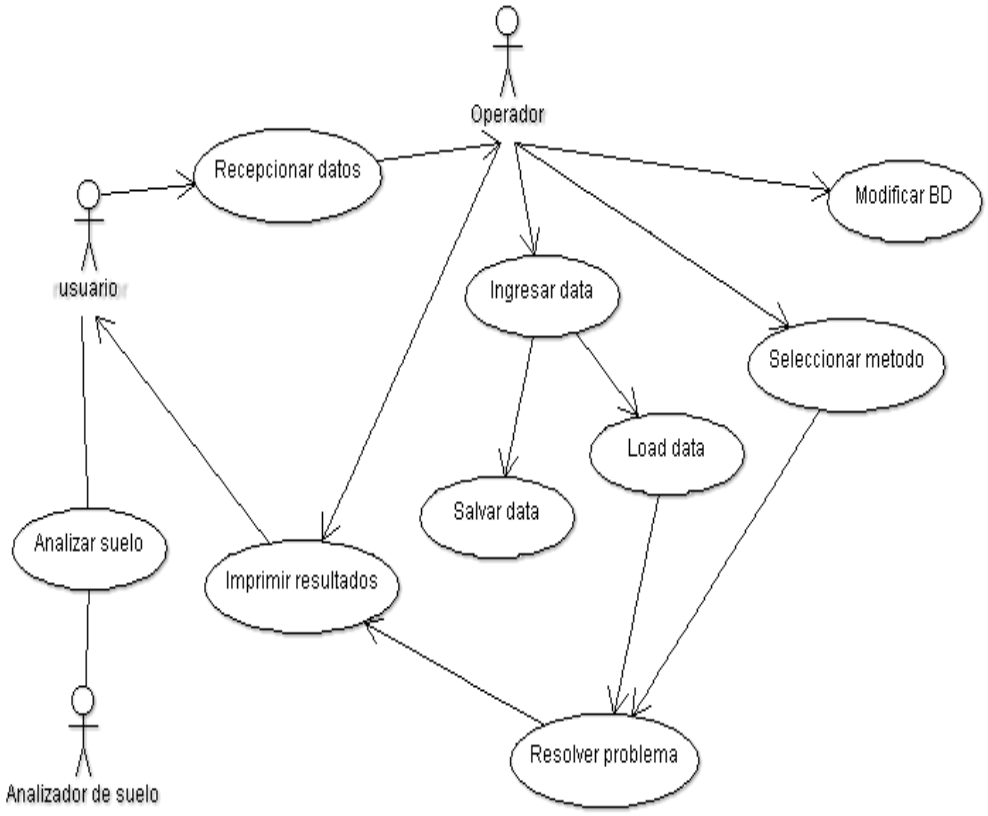


Figura 27: Casos de uso para el programa FERTIDIF

4.2.10 Definicion de actores

Usuario : Agricultor, ing agronomo u alguien relacionado a cultivos.

Operador: Encargado de recibir lista de resultados del analisis de suelo , lista de fertilizantes y emitir los resultados.

Analizador_suelo: Persona encargada del analisis de suelos.

4.2.11 Documentacion de los casos de uso

Imprimir resultados: imprime datos de fertilizantes y los resultados

Seleccionar método: selecciona el método que desarrollada el problema.

Analizar suelo: Análisis del suelo

Modificar BD: modificación de la base de datos de fertilizantes.

Ingresar Data: Ingresa la data desde teclado o archivo .fd hacia GUI después al problema para su desarrollo.

Load Data: Transforma los datos de fertilizantes, análisis de suelo al problema que resolveremos en su forma matricial

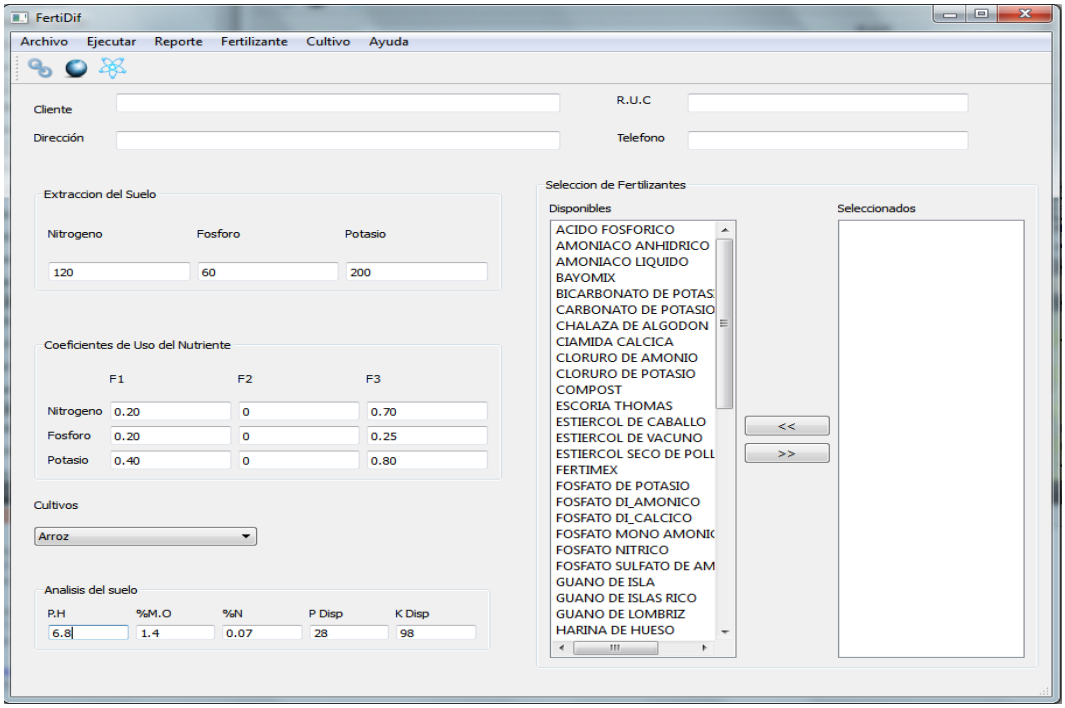
Salvar Data: Almacena la data ingresada y seleccionada del GUI a un archivo .fd

4.2.12 Requisitos no funcionales

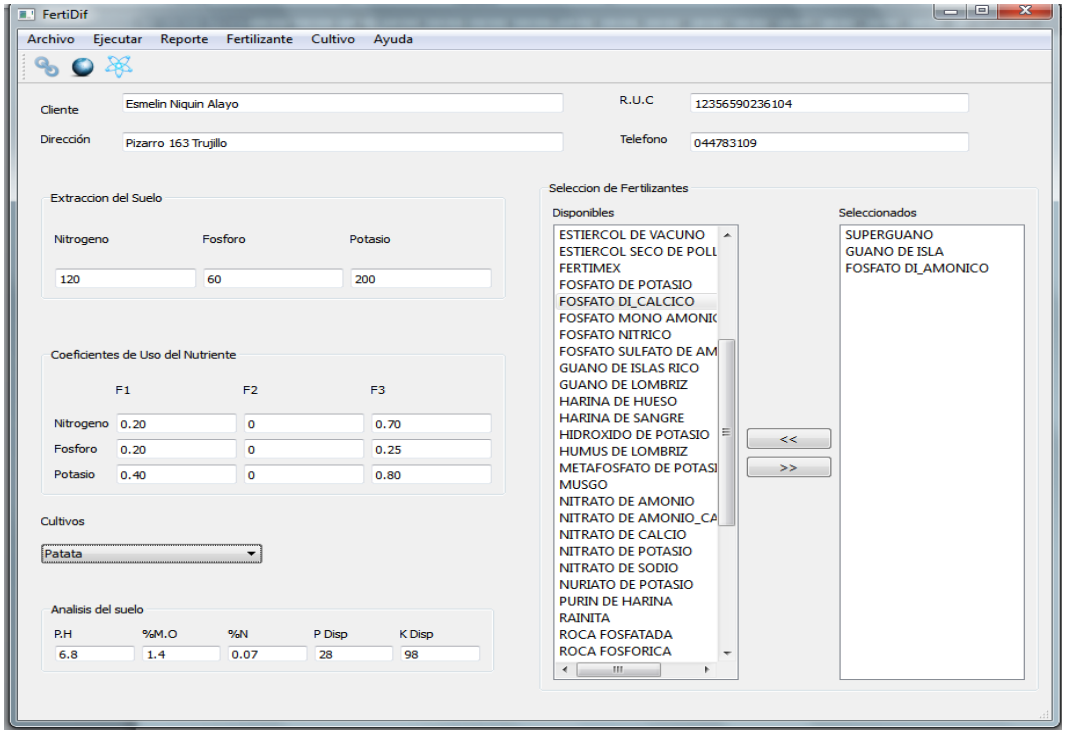
- Buena performance en tiempo.
- Entendible a cualquier usuario.
- Contara con un manual de ayuda.
- Se podra ejecutar en cualquier plataforma.
- Las licencias son por año.
- Soporte por correo o en forma presencial.
- Facil modificalidad del producto, si se desea tener mas funciones.
- Alta perfomance del lenguaje de programación.

4.2.13 **Ventanas y formularios del programa**

Ventana principal



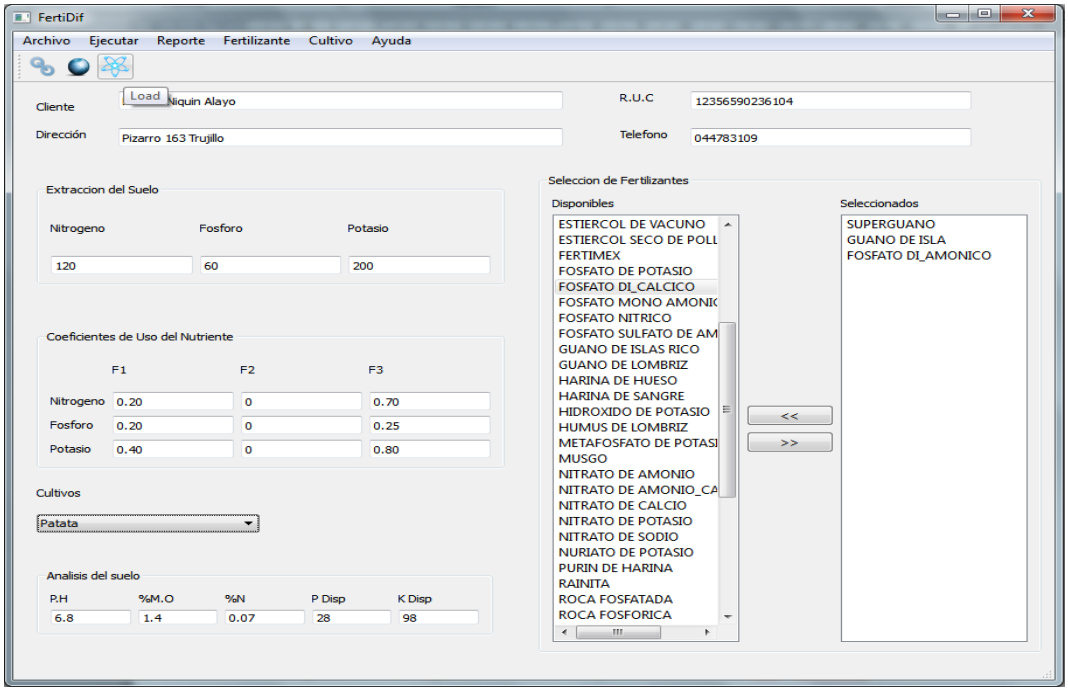
Es la ventana de presentación del programa, aquí el usuario ingresa los datos relacionados al suelo, selecciona el cultivo y los fertilizantes a usar en la fertilización así como también se ingresan los datos del cliente.



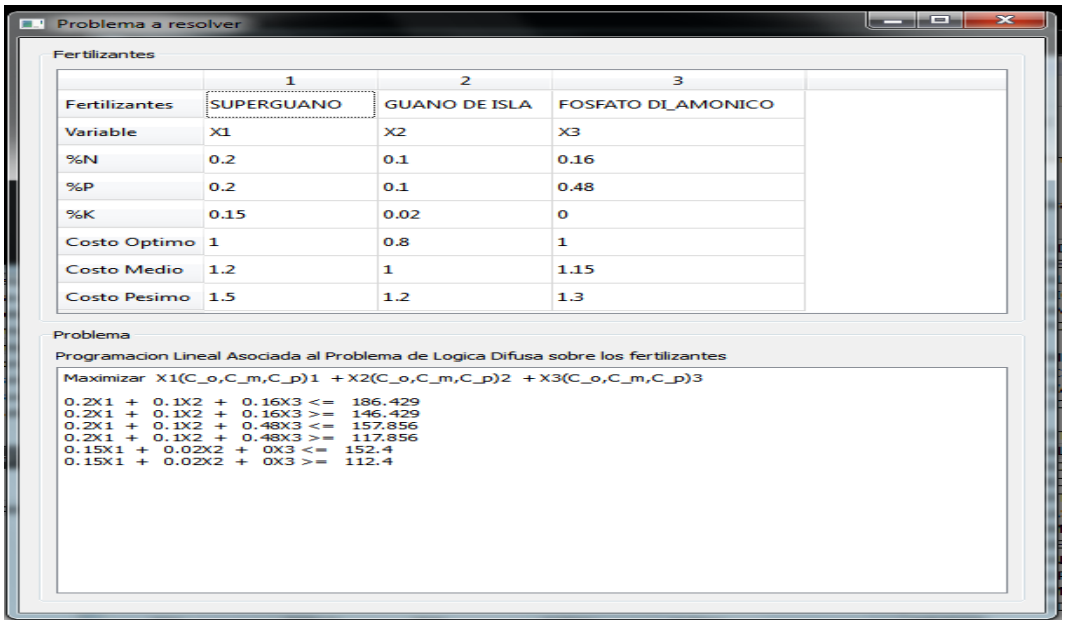
4.2.13.1 Ventana de cargar datos

Después de ingresar procedos a cargar los datos que son necesarios para la solución del problema. El cargar datos es el mismo para los métodos de Lai o Leberling

Para realizar esta carga, hacemos click en el icono de figura de átomo.



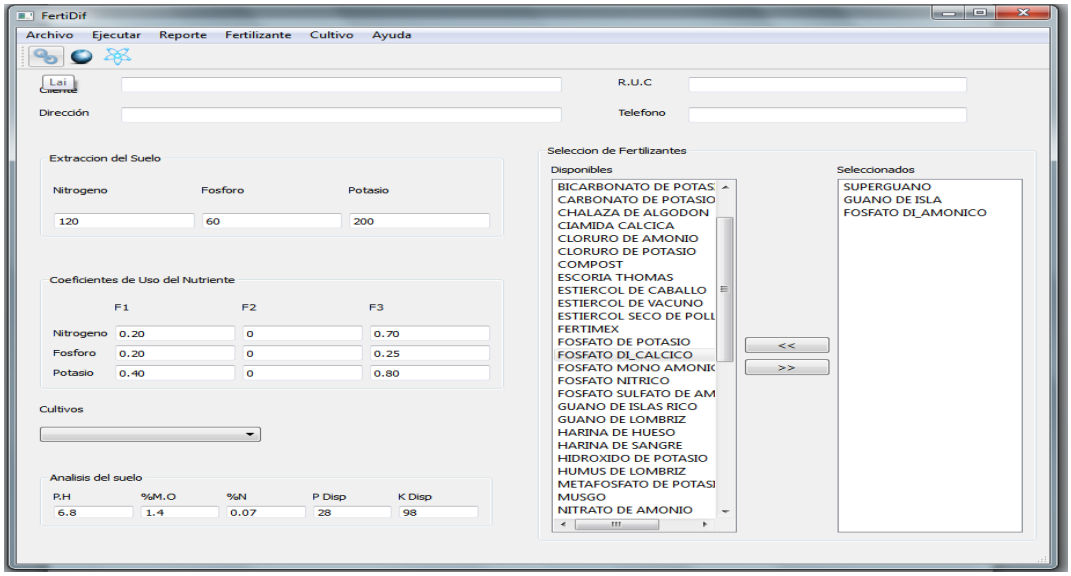
Y obtenemos la siguiente ventana.



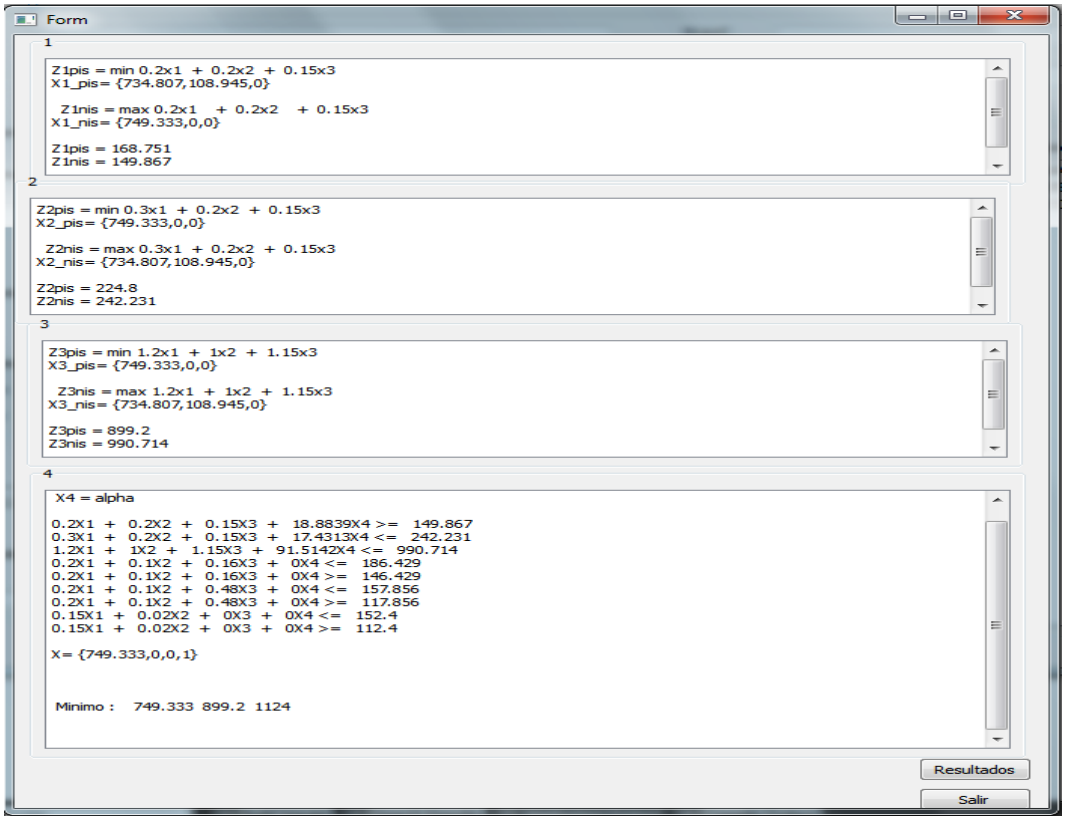
4.2.13.2 Ventanas de resultados

4.2.13.2.1 Aproximación de Lai-Hwang

Para resolver nuestro problema planteado con el método de Lai y Hwang, hacemos click en el icono de cadenas.

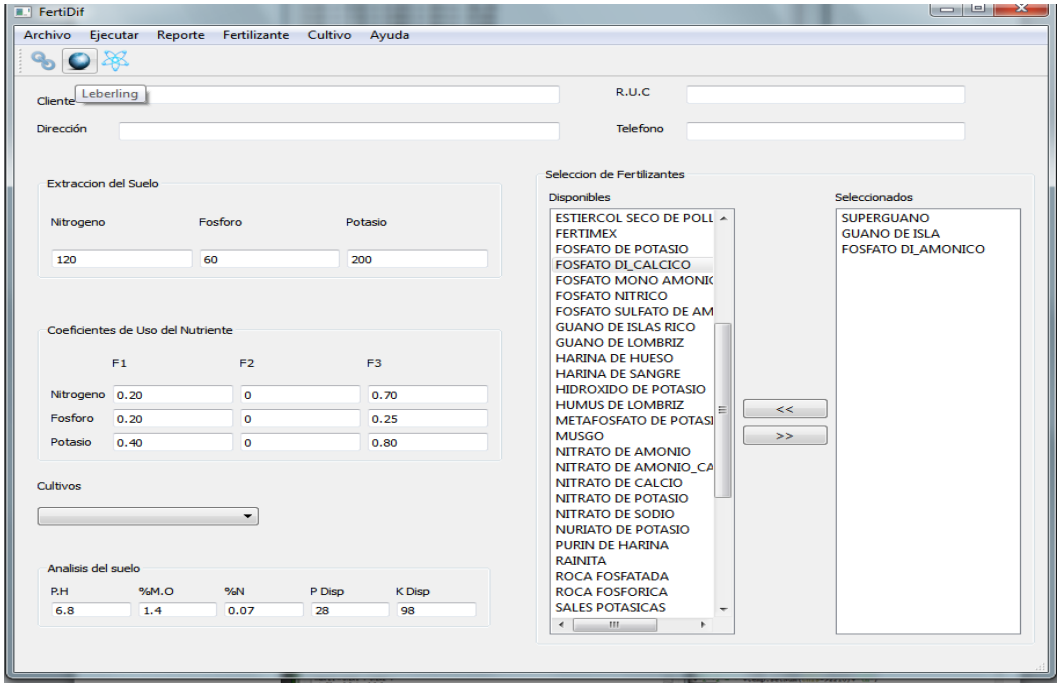


Obtenemos la siguiente ventana de resultados

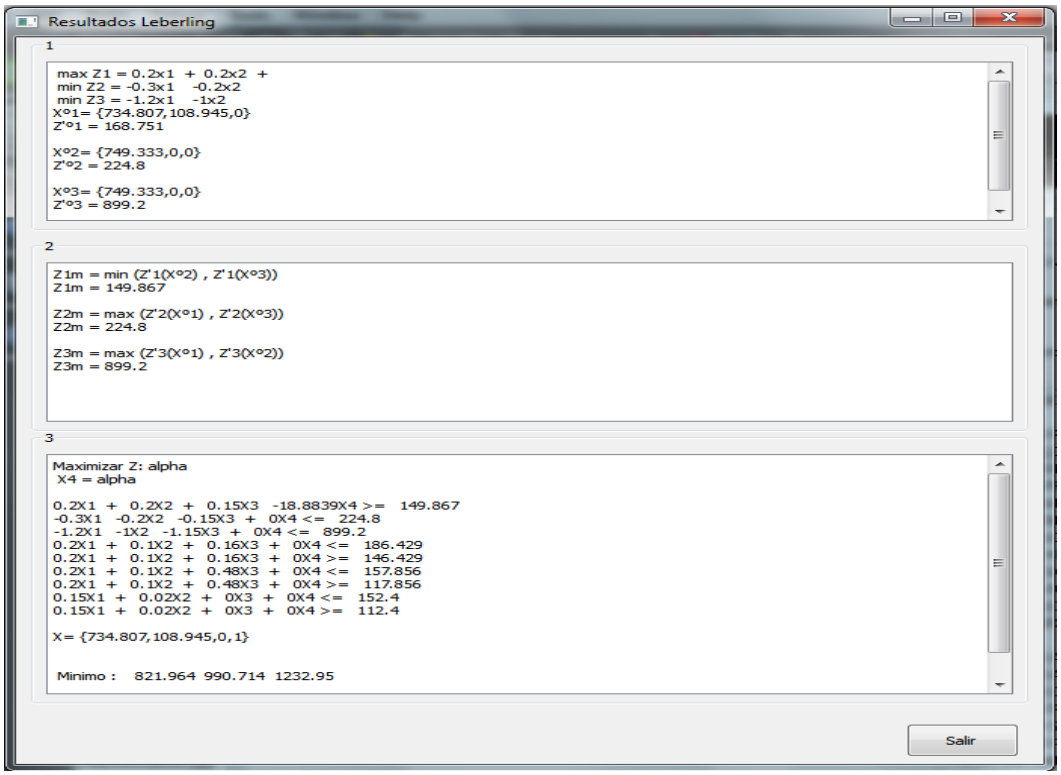


4.2.13.3.2 Leberling

Para resolver nuestro problema planteado con el método de Lai , hacemos click en el icono de mundo



Y el resultado obtenido se muestra en otra ventana.



4.2.14 Menus

4.2.14.1 Menu Archivo

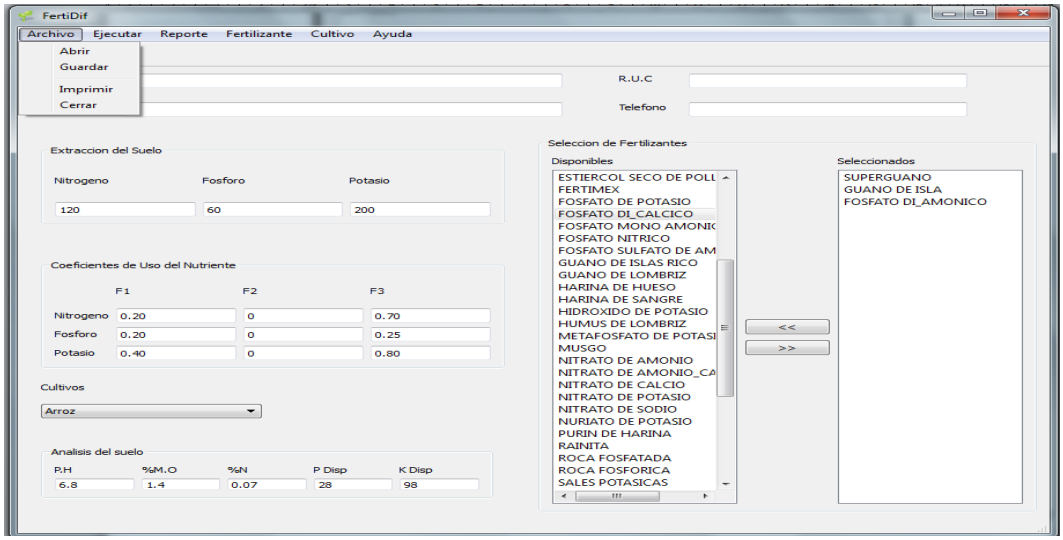
En este menú tenemos las siguientes opciones:

-Abrir

-Guardar

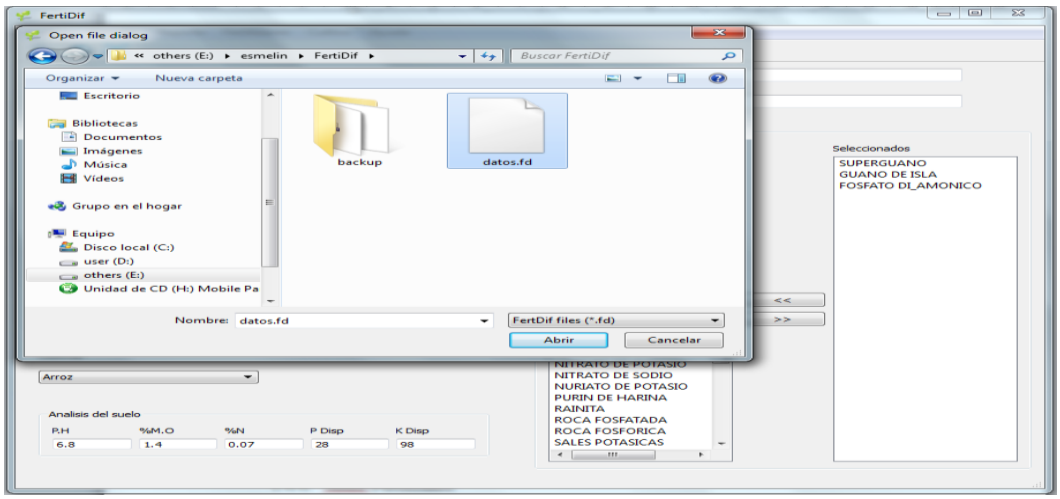
-Imprimir

-Cerrar



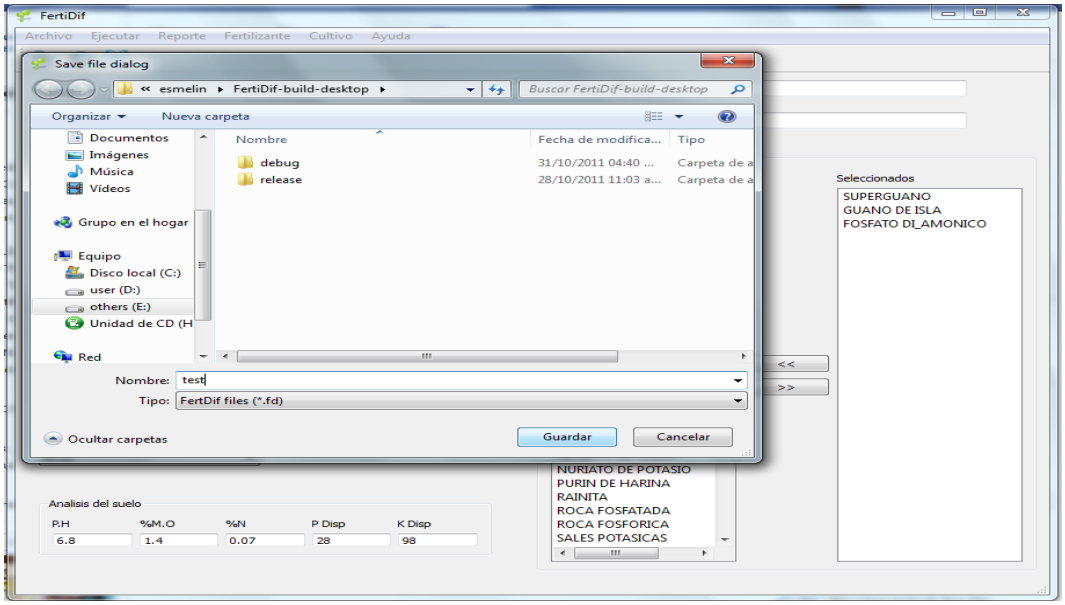
Abrir

Esta opción carga datos desde un archivo (.fd) que almacena información de una operación realizada anteriormente.



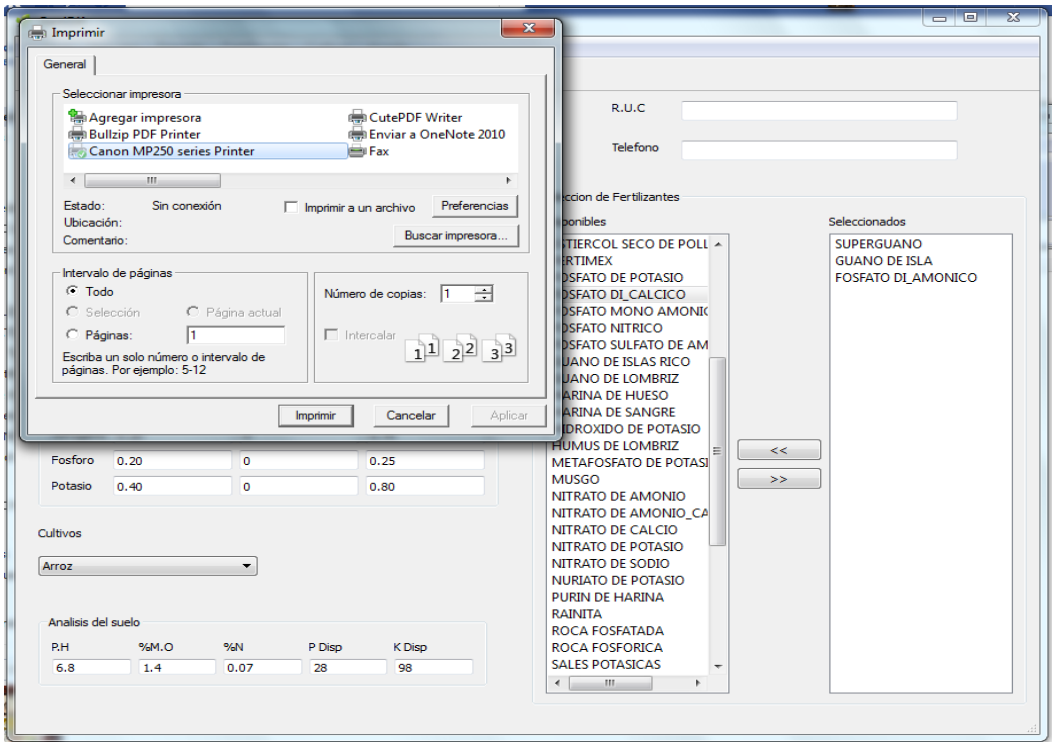
Guardar

También podemos almacenar los datos del problema que tratamos . La extensión del archivo es “.fd” .



- **Imprimir**

Nos permite imprimir los datos y resultados obtenidos del programa.

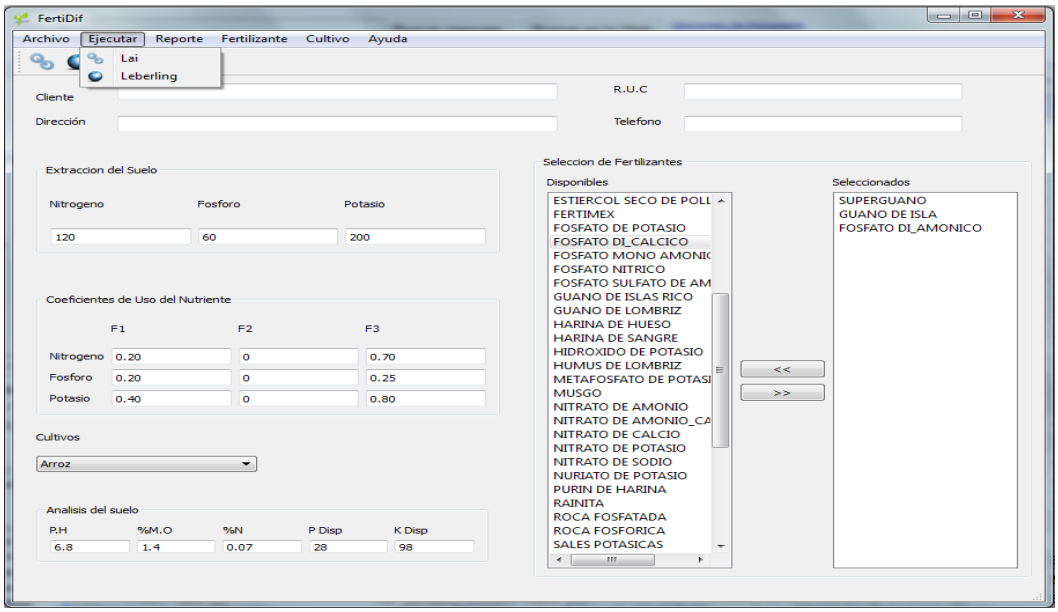


- **Cerrar**

Cierra todas las ventanas del programa.

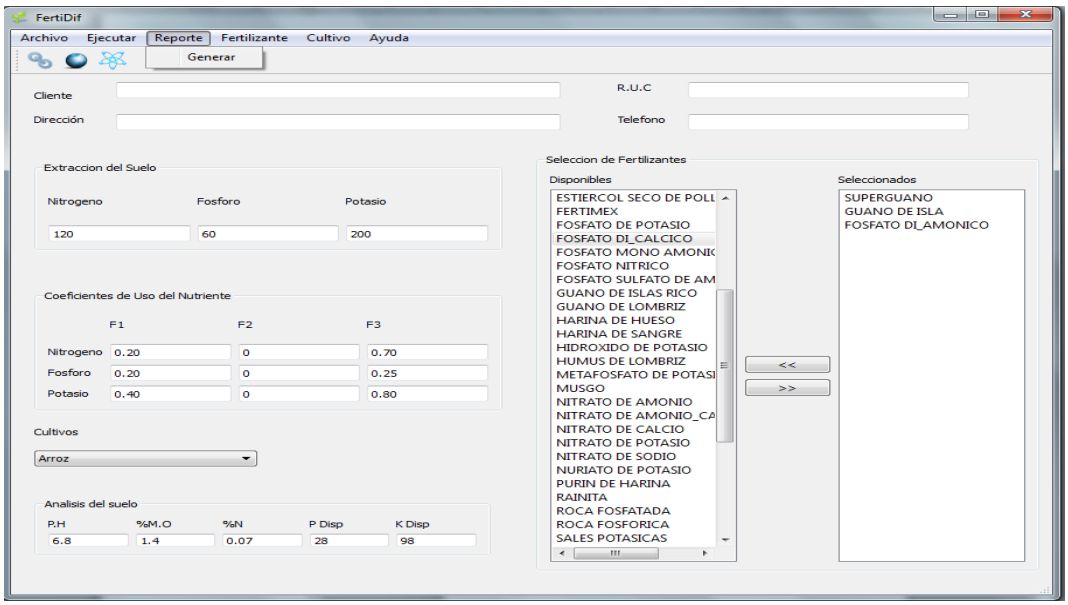
4.2.14.2 Menu Ejecutar

Cuenta los métodos de Lai y Leberling para la resolución del problema.



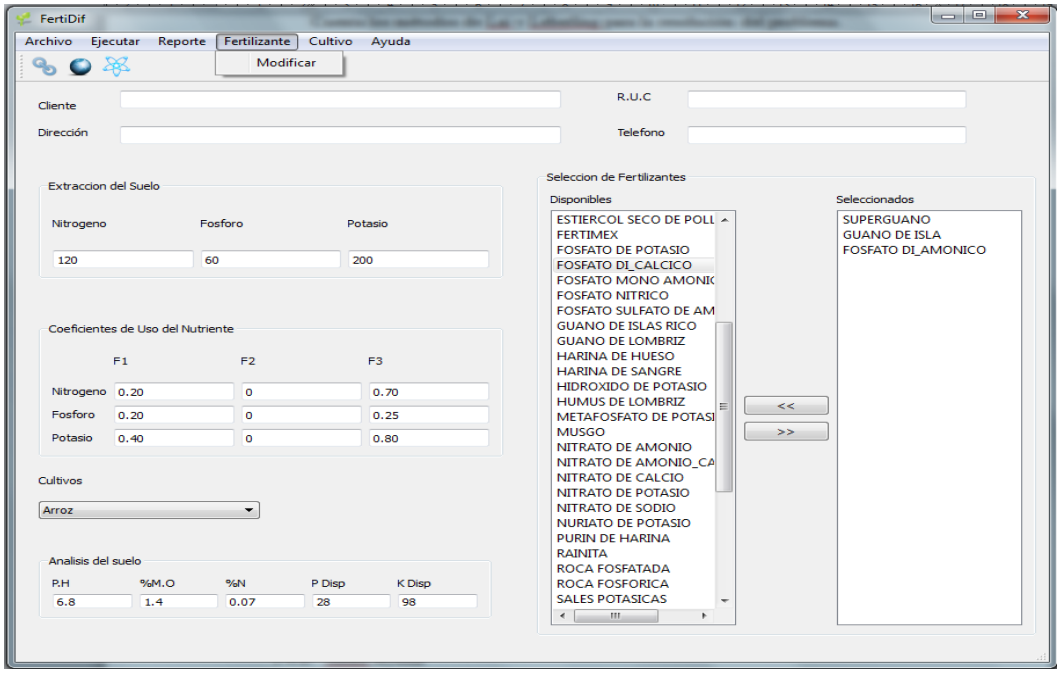
4.2.14.3 Menu Reportar

Genera un reporte con los datos ingresados y resultados obtenidos.

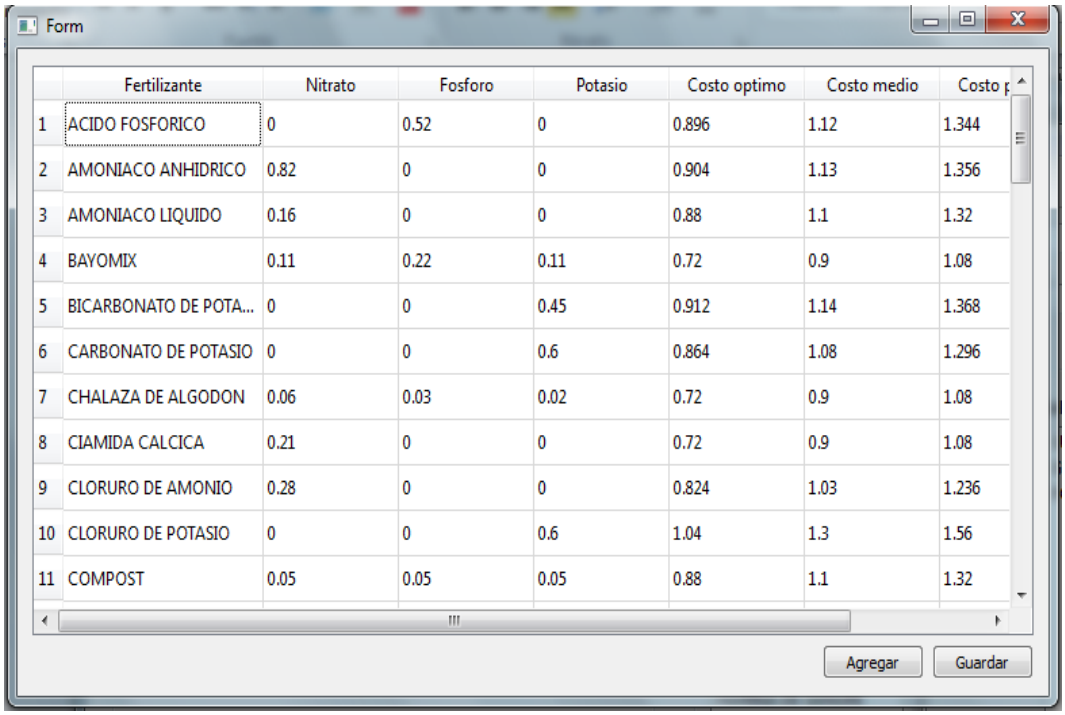


4.2.14.4 Menu Fertilizante

Esta opción permite modificar la base de datos de fertilizantes de nuestro programa así como también agregar nuevos fertilizantes.

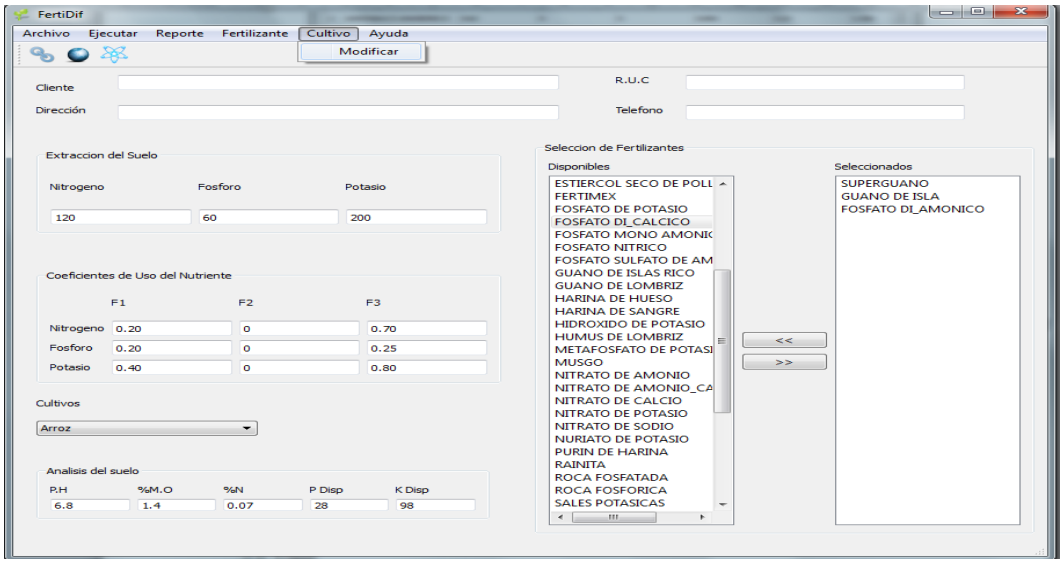


4.2.14.5 Modificar



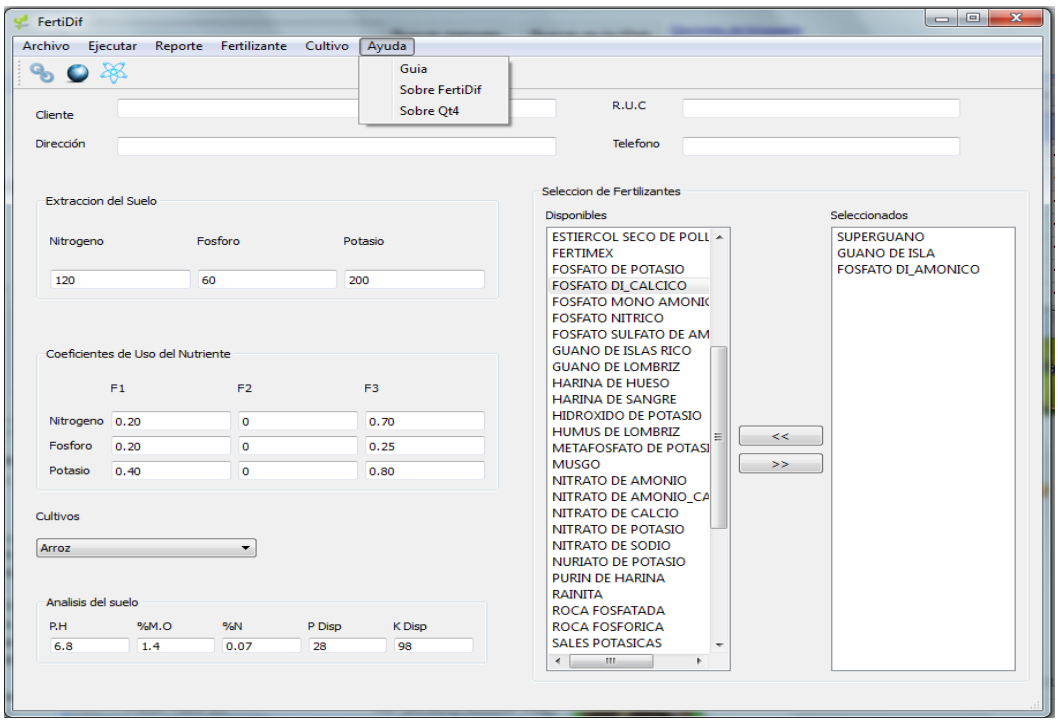
4.2.14.6 Menu Cultivo

Puedes modificar la base de datos de cultivos de nuestro programa así como también agregar nuevos cultivos.



4.2.14.7 Menu Ayuda

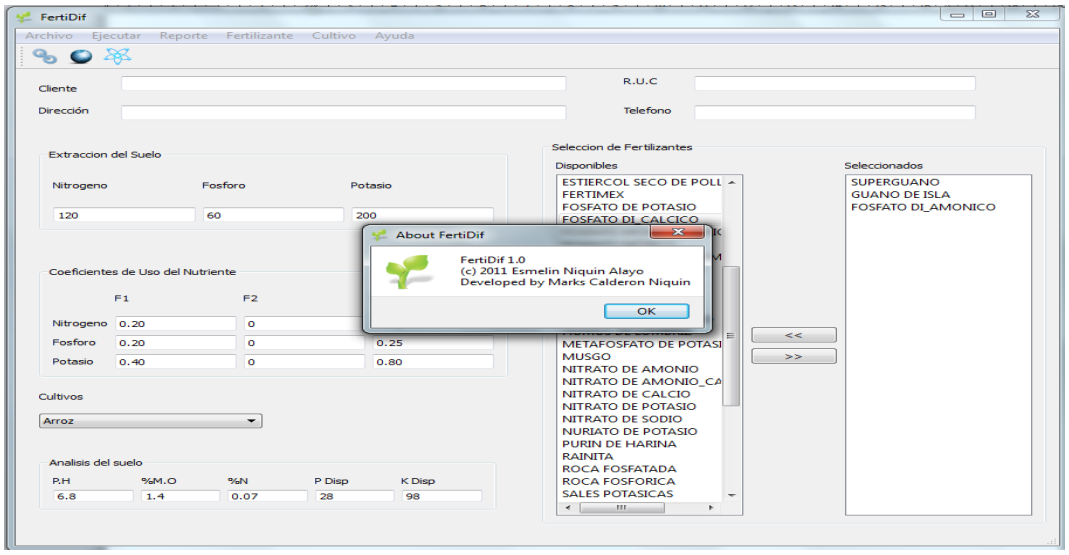
Cuenta con las opciones de Guia, Sobre FertiDif y Qt4



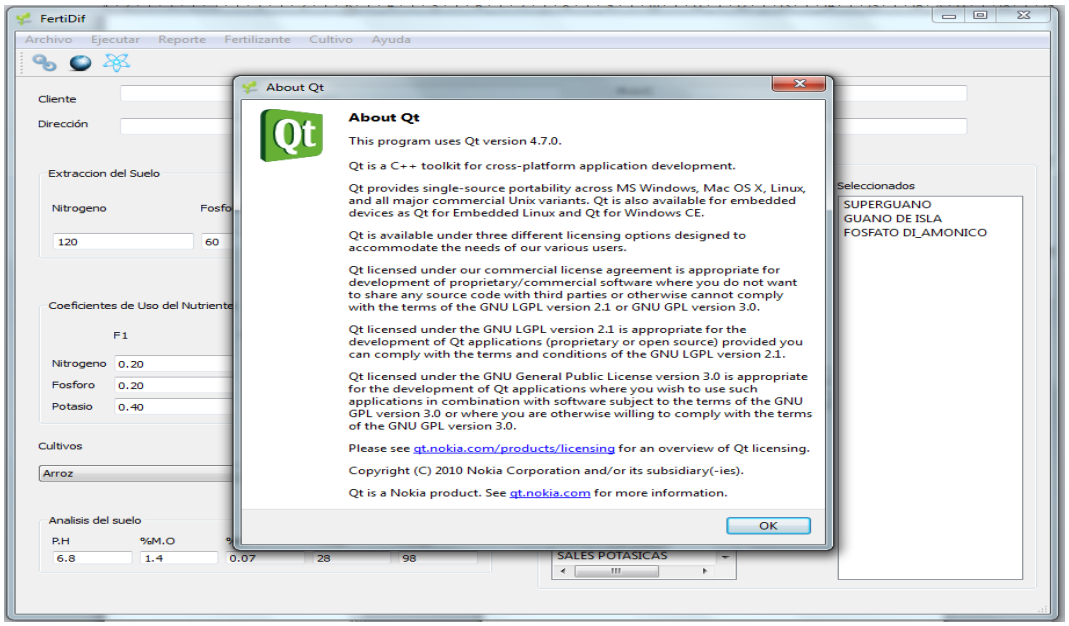
4.2.14.8 Guia

Por desarrollar en un futuro próximo, conjuntamente con un equipo de especialistas del agro sobre procesos de fertilización agrícola.

4.2.14.9 Sobre FertiDif



4.2.14.10 Sobre Qt4



V. DISCUSIÓN

Desarrollado el proyecto planteado y haciendo uso de las metodologías adaptadas de Lai-Hwang y Leberling a nuestro problema de fertilización de cultivos agrícolas se ha podido constatar que la teoría matemática es de suma importancia en la modelización matemática del problema real y en la obtención de la solución analítica del problema, es decir es posible diseñar un nexo que relacione los datos reales y la matemática aplicada, de allí que es posible la formulación de un modelo matemático que describe el problema de fertilización en este caso se trata de un problema de fertilización difuso (PFD) pues se considera que los costos de los fertilizantes (abonos) son variables.

En el caso de Lai-Hwang, los costos difusos hacen que el problema clásico de dieta se torne en un problema difuso, aquí se consideran que los porcentajes de los nutrientes que componen los mismos son cantidades fijas y el problema fundamental radica que la función objetivo con costos difusos se considere difusa con el consiguiente problema de obtener la solución, Lai-Hwang, proponen la metodología de solución para el caso de que la función obletivo sea de maximización y mediante ciertas manipulaciones y asumiendo ciertos principios del mundo empresarial transforman el problema con una función objetivo en un problema multiobjetivo y luego usando la metodología de Zimmermann que considera la función objetivo como difusa y haciendo uso de un determinado tipo de funciones de pertenencia del tipo lineal se logra resolver el problema planteado. Cabe indicar que se hicieron las adecuaciones pertinentes de esta metologia para resolver el problema en el caso que la función objetivo

corresponde a un caso de minimización, podría pensarse como el caso del problema dual al resuelto por Lai-Hwang.

En el caso de la metodología de Leberlin se usa la programación por metas y partiendo del modelo adaptado de Lai-Hwang (programa multiobjetivo) y definiendo las funciones de pertenencia se logra obtener la solución al problema real planteado.

En menestar afirmar que con ambas metodologías de solución al final se reducían a un problema clásico se programación y con la implementación del método Simplex y su aplicación se obtuvieron resultados con muy poca diferencia, motivo por lo cual pueden darse como validos los resultados obtenidos en ambos casos. Indicando que a mi modesto enetender la metodología de Lai-hwang es mucho mas simple de entender y ejecutar en la solución de problemas concretos planteados.En ambos casos las soluciones no son únicas.

En lo referente al software FERTIDIF, es menestar indicar que es de fácil manipulación o interactuar entre el usuario y el programa, donde todos los datos son ingresados por pantalla, iniciando el proceso previamente con un análisis de suelos para tener los parámetros que se requiere para usar las formulas de fertilización ya definidas y el software una vez ingresados toda la información requerida formula el modelo de optimización multiobjetivo y nos provee la opción de elegir el método de solución, bien sea mediante Lai-Hwang ó Leberling en ambos casos el tiempo de calculo es muy rápido y los resultados confiables.

VI. CONCLUSIONES

1. De la bibliografía especializada que se ha revisado se puede concluir concluye que la teoría difusa es una buena teoría y útil herramienta que permite representar mediante modelos matemáticos situaciones imprecisas de la vida cotidiana. En el presente proyecto los números difusos nos ha permitido representar la información de costos variados en el mercado. Con esta ayuda se ha podido representar la problemática de la planificación en fertilzación de cultivos agricolas mediante el modelo de la dieta difusa.

2. La solución que se obtiene luego de resolver el problema es dado por un número difuso, pero que el software reporta con un determinado grado de aceptación. Por otro lado en el mismo reporte se proporciona el menor costo, el costo intermedio y el mayor costo, dependiendo que el costo de adquisición sea el menor, intermedio o mayor costo, la decisión queda a criterio del especialista.

3. De los resultados obtenidos se ha podido inferir que ambos métodos de solución proporcionan resultados con muy pocas diferencias, lo cual nos garantiza la consistencia de los métodos y algoritmos para la solución al problema planteado.

4. Usanndo el modelo difuso de la dieta y los correspondientes métodos de la Programación Lineal Difusa se ha desarrollado FERTIDIF, un software en su versión primera, software que servirá de herramienta de apoyo en la planificación de fertilización de terrenos y cultivos agrícolas.

5. FERTIDIF permitirá determinar la cantidad de cada tipo de insumo que se debe incluir en un proceso de fertilización agrícola de manera tal que satisfaga los requerimientos nutricionales de un determinado cultivo y con la ventaja de generar un menor costo.

VII RECOMENDACIONES

1. En este análisis de mínimo costo hay variables que no se consideran, por lo complicado que resulta modelar su efecto individual en el contexto de la planificación agrícola, como por ejemplo: grado de solubilidad, salinización producto de la fertilización o enmienda, granulometría del producto. Es importante que esas variables que parecen secundarias en primera instancia, las considere el técnico al momento de tomar una decisión, ya que en situaciones particulares pueden influir en los resultados esperados.

2. Se recomienda que las dosis sean fijadas luego de un análisis real de la disponibilidad de nutrientes en el suelo, por eso es fundamental que se realice el análisis de suelos y si fuera necesario también un análisis foliar.

3. Se recomienda que cuando se presente un problema de programación lineal clásico y se observa que hay datos no tan precisos o se tiene ideas vagas de ellos se formule el problema como un problema de programación difusa.

4. Se recomienda que al abordar la solución de un problema de programación difusa se usen funciones de pertenencia lineales, pues son fácilmente manipulables y las soluciones son confiables.

5. Se recomienda a los técnicos en fertilización agrícola hagan uso de este programa de ayuda en sus labores.

6. Finalmente se recomienda a quienes trabajen en esta área de la optimización incorporen en sus equipos de investigación a especialistas del área en la cual se investiga y se desarrollara el modelo matemático.

VIII BIBLIOGRAFIA

[1] Bellman RE, Zadeh LA. Decision-making in a fuzzy environment Management Science 1970.

[2] Bitran GR. Linear multiple objective problems with interval coefficients. Management Science 1980.

[3] Buckley JJ. Possibilistic linear programming with triangular fuzzy numbers. Fuzzy Sets and Systems.

[4] Buckley JJ. Solving possibilistic linear programming problems, Fuzzy Sets and Systems 1989.

[5] Cadenas JM, Pelta DA, Pelta HR, Verdegay JL. Fuzzy Diet Problems in Argentinian Farms: a Case Study. European Journal of Operations Research 2004.

[6] Cadenas JM, Verdegay JL. PROBO: an interactive system in fuzzy linear programming. Fuzzy Sets and Systems 1995.

[7] Coenen F, Bench-Capon T. Maintenance of Knowledge-Based Systems: Theory, Techniques and Tools. Cornwall (Gran Bretaña): Academic Press, 1993.

[8] Chanas S. The use of parametric programming in FLP. Fuzzy Set and Systems 1983.

[9] Dantzig GB. Linear Programming and Extensions. Princeton-NJ: Princeton University Press, 1963.

[10] Delgado M, Herrera F, Verdegay JL, Vila MA. Post-optimality analysis on the membership functions of a fuzzy linear programming problem. Fuzzy Sets and Systems 1993.

[11] Delgado M, Verdegay JL, Vila MA. Relating different approaches to solve linear programming problems with imprecise costs. Fuzzy Sets and Systems 1990.

[12] Dorfman R, Samuelson PA, Solow RM. Programación Lineal y Análisis Económico. Madrid: Aguilar ediciones, segunda edición, 1964.

[13] Gass S. Programación Lineal: Métodos y aplicaciones. Mexico: Ed. Continental, 1985.

[14] Gurewich O, Gurewich N. VISUAL C++ 6 EN 21 DIAS. Mexico: Editorial Prentice may, 2003.

[15] Inuiguchi M, Ichihashi H. Relative modalities and their use in possibilistic linear programming. Fuzzy Sets and Systems 1990.

[16] Inuiguchi M, Sakawa M. Possible and necesary optimality tests in possibilistic linear programming problems. Fuzzy Sets and Systems 1994.

[17] Lai YJ, Hwang CL. Fuzzy Mathematical Programming, methods and applications. Lecture notes in economics and mathematical systems 394. Berlin: Springer-Verlag, 1993.

[18] Lai YJ, Hwang CL. Interactive fuzzy linear programming. Fuzzy Sets and Systems 1992.

[19] Lai YJ, Hwang CL. A new approach to some possibilistic linear programming problems. Fuzzy Sets and Systems 1992.

[20] Lai YJ, Hwang CL. Possibilistic linear programming for managing interest rate risk. Fuzzy Sets and Systems 1993.

[21] Luhandjula MK. On possibilistic linear programming. Fuzzy Sets and Systems 1986.

[22] Pressman RS. Ingenieria del software. Un enfoque practico. Mexico: Editorial McGrawHill, 5ta Edición, 2002.

[23] Rommelfanger H, Slowinski R. Fuzzy linear programming with single or multiple objetive functions. En Slowinski R. eds. Decision Analysis, Operations Research and Statistics of Handbooks of Fuzzy Sets. Boston: Luwer Academic Publishers, 1998.

[24] Rommelfanger H, Hanuscheck R, Wolf J. Linear programming with fuzzy objectives. Fuzzy Sets and Systems 1989-

[25] Tanaka, H., H. Ichihashi y K. Asai; A formulation of fuzzy linear programing problems basad on comparison of fuzzy numbers. In Kacprzyk 1984.

[26] Tanaka, H., T. Okuda y K. Asai, On fuzzy mathematical programming, Journal of Cybernetics 3 1974.

[27] Verdegay, J. L., A dual approach to solve the fuzzy linear programming problem, Fuzzy Sets and Systems 14,1984.

[28] Verdegay, J.L. Fuzzy mathematical programming, Approximate Reasoning in Decision Analysis - Gupta, M.M. y E. Sanchez (eds) (North_holland, Amsterdam, 1992.

[29] Verdegay, J. L.; Fuzzy optimization: Models, methods and Perspectives; 6thIFSA-95 World Congress. Sou Paulo - Brazil, 1995.

[30] Zadeh, L.A.: Fuzy sets, Information and Control 8, 1965 .

[31] Zhao, R., R. Govind y G. Fan, The complete decision set of the generalized symetyrical fuzzy linear programming problem, Fuzzy Sets and Systems 51, 1992

[32] Zimmermann, H. J., Description and optimization of fuzzy system, International Journal of general System 2,1976 .

[33] Zimmermann, H. J., Fuzzy programming and linear programming with several objective functions, Fuzzy Sets and Systems 1, 1978.

[34] Belton V. and Stewart T., Multiple Criteria Decision Analysis, Kluwer Academic Publishers, 2001.

[35] Coello C., Lamont G., Van Veldhuizen D., Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, 2002.

[36] Dasgupta P., Chakrabarti P., Desarkar S., Multiobjective Heuristic Search: An Introduction to Intelligent Search Methods for Multicriteria Optimization, Morgan Kaufmann Publishers, 2004.

[37] Deb K., Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, New York, 2001.

[38] A. Gros, Abonos: guía práctica de la fertilización, Ed. Mundi-prensa, Madrid, 1981.

[39] A. Casas C. y E. Casas B., El análisis de suelo-agua-planta y su aplicación en la nutrición de cultivos hortícolas en la zona peninsular, Caja Rural de Almería, 1999.

[40] R. E. White, Principles and practice of soil science, the soil as a natural resource, Oxford-Blackell Science, Londres, 2000.

[41] J. García F. y R. García Del Caz: Edafología y Fertilización Agrícola, Ed. Aedos, Barcelona, 1982.

[42] M. F. Orozco L; Suelos y Fertilización, Ed. Trillas, 2000.

[43] B. Aghezzaf; An interactives interior point algorithm for multiobjetive linear programming, Letters , 2001

[44] A. Arbel y S. Oren; theory and methodology: using approximate gradients in developing an interactive interior primal-dual multiobjetive linear programming algorithm; European Journal of Operational Research, 1996.

[45] A. Balbas; Programación Matemática, AC, Madrid, 1987.

[46] A. Charnes; Manegement Models And Industrial Applications of Linear Programming, New York, 1961.

[47] E. Hannan; Linear programming with multiple fuzzy goals, 1981

[48] H. Leberling; On finding compromise solutions in multicriterial problems using the fuzzy min operator, 1981.

[49] H.M. Ramadan; The relationship between goal programming and fuzzy programming, 1989.

[50] H. Rommelfanger; Fuzzy linear programming with single or multiple objective functions; Boston, 1992.

[51] T.B. Trafalis; An interior point multiobjetive programming approach for production planning with uncertain information, 1999.

[52] T.B. Trafalis; An interactive analytic center trade-off cutting plane algorithm for multiobjetive linear programming; 1999.

[53] M. Zeleny; Linear Multiobjetive Programming, Springer, New York, 1974.

[54] Buckley JJ. Possibilistic linear programming with triangular fuzzy numbers. Fuzzy Sets and Systems 1988; 26: 135-138.

[55] Buckley JJ. Solving possibilistic linear programming problems, Fuzzy Sets and Systems 1989; 31: 329-341.

[56] E.Niquín; Programación lineal Difusa y su Aplicación en la Toma de Decisiones en la Planificación de fertilización de Tierras de cultivo, tesis de maestro (2006).

ANEXO 1

Código fuente del Sistema Informático

1. Clase Simplex :

Header

```
1. #ifndef SIMPLEX_H
2. #define SIMPLEX_H
3. #include <vector>
4. using std::vector;
5. class simplex
6. {
7.     public:
8.         simplex();
9.         virtual ~simplex();
10.            //inicializa variables
11.
     void set_data(vector<vector<double> > M, vector<
   double> f, vector<double> B,
12.                            vector<char> sig );
13.            //funcion para resolver simplex
14.            bool solved_simplex();
15.            //calcculamos el minimo
16.            double minimum_simplex();
17.             //mostramos datos en consola
18.            void show_Data();
19.            //retorna el valor de las variables
   XD
20.            vector<double> retornarX();
21.            //funcion que evalua el objetivo
22.            double eval_obj();
23.            //limpia las matrices
24.            void limpiar();
25.         private:
26.            vector<double> Objetivo; //vector
   objetivo
27.
   vector<vector<double> > rest; //matriz a calcular
28.            vector<double> B; //vector B
29.            vector<int> slack;
30.            /*vol2*/
31.            vector<vector<double> > mat; //matriz
32.            vector<int> posFas; // posiciones
   para 2 fases
33.            /*limits double*/
34.             double min;
```

```
35.              double max;
36.
37.
38.             /*matriz ampliado*/
39.             vector<vector<double> > M;
40.             double resultado;
41.
42.
    int select_column(size_t); //selecciona columna
43.             int select_row(int); //selecciona
  fila
44.             /*calcule new table*/
45.
    void calcule_new_table(int row, int column);
46.             /*reemplace in object function*/
47.             void elimina_artificial();
48.     };
49.     #endif // SIMPLEX_H
```

Code:

```
1. #include "simplex.h"
2. #include <vector>
3. #include <cmath>
4. #include <cstdio>
5. #include <cstdlib>
6. #include <limits>
7. #include <iostream>
8. #include <functional>
9. #include <algorithm>
10.
11.     using namespace std;
12.     simplex::simplex()
13.     {
14.         //constructor
15.         min = numeric_limits<double>::min();
16.         max = numeric_limits<double>::max();
17.     }
18.     simplex::~simplex()
19.     {
20.         //destructor
21.         Objetivo.clear();
22.         rest.clear();
23.         B.clear();
24.          M.clear();
25.     }
```

```
26.     void simplex::limpiar()
27.     {
28.         Objetivo.clear();
29.         rest.clear();
30.         B.clear();
31.         slack.clear();
32.          mat.clear();
33.         posFas.clear();
34.         M.clear();
35.     }
36.     void simplex::calcule_new_table(int row, int
  column)
37.     {
38.         double temp;
39.         double error= 1.0e-10;
40.         int n ;
41.         temp = mat[row][column];
42.         n = mat[row].size()-1;
43.         /*dividimos para que nos quede 1 el
  elemento (row, column)*/
44.         for(int i = 0; i<n ; i++)
45.         {
46.             if(mat[row][i] != 0 )
47.             {
48.                 mat[row][i] /=temp;
49.             }
50.         }
51.         //n now change
52.         n = mat.size();
53.         /* ahora eliminamos  los demas elementos
  en esa columna*/
54.         for(int j =0; j<n; j++)
55.         {
56.             if(j != row)
57.             {
58.                 temp = mat[j][column];
59.                 if(temp!=0)
60.                 {
61.
    for(size_t k=0; k<mat[j].size()-1; k++)
62.                     {
63.                                 mat[j][k]-
  = (temp*mat[row][k] );
64.
   if(fabs(mat[j][k])>0 && fabs(mat[j][k]) <= error)
65.                                  mat[j][k] = 0.0;
```

```
66.                     }
67.                 }
68.
69.             }
70.         }
71.     }
72.     bool simplex::solved_simplex()
73.     {
74.         vector<double>::iterator it;
75.         vector<double> ult; //fila de
  coeficientes
76.         int last;
77.         int index; //columna pivote
78.         int row; //fila pivote
79.         bool flag = true;
80.         last = mat.size();
81.         ult= mat[last-1];
82.        //Fase 1 : prepropesamios la data
83.
    while((it=find_if(ult.begin(), ult.end(),bind2nd(
  greater<double>(),0)))!=ult.end()) /* si hay
  entradas negativas*/
84.         {
85.             index=this->select_column(last-1);
86.             if( index == -1) {
87.                 break;
88.             }
89.             if((row=select_row(index))== -1){
90.                 flag = false;
91.                 break;
92.             }
93.             calcule_new_table(row, index);
94.             ult= mat[last-1];
95.             slack[row] = index;    }
96.        //eleiminados elementos artificiales
97.         this->elimina_artificial();
98.         /*fase 2*/
99.         ult= mat[last-2];
100.        flag = true;
101.
    while((it=find_if(ult.begin(), ult.end(),bind2nd(
  greater<double>(),0)))!=ult.end()) /* si hay
  entradas negativas*/
102.        {
103.            index=this->select_column(last-2);
104.            if( index == -1) {
```

```
105.                break;
106.            }
107.            if((row=select_row(index))== -1){
108.                flag = false;
109.                break;
110.            }
111.
112.            calcule_new_table(row, index);
113.             slack[row] = index;
114.            ult= mat[last-2];
115.        }
116.        return  flag;
117.    }
118.    double simplex::minimum_simplex()
119.    {
120.        /*
121.          cambiamos el signo al vector Objetivo
122.        */
123.        for(size_t i=0; i<Objetivo.size(); i++)
124.          Objetivo[i] *=-1;
125.        /*
126.       cambiamos el resto    */
127.        for(size_t j=0; j<rest.size(); j++)
128.        {
129.
    for(size_t k=0; k<rest[j].size(); k++)
130.              rest[j][k] *= -1;
131.        }
132.        return solved_simplex();
133.    }
134.    double simplex::eval_obj()
135.    {
136.        double result = 0;
137.        vector<double> xs;
138.        size_t n;
139.        n= Objetivo.size();
140.        xs= this->retornarX();
141.        for(size_t j=0; j<xs.size(); j++)
142.        {        result += Objetivo[j]* xs[j] ;
143.        }
144.        return result;
145.    }
146.    void simplex::set_data(vector<vector<double>
  > M,vector<double> F,vector<double> B,vector<char>
   sig)
147.    {
```

```
148.            mat.push_back(F); //agregamos
  objetivo
149.            Objetivo=F;
150.            size_t n=sig.size();
151.            double uno;
152.               size_t k=0;
153.            vector<double> zeros;
154.
155.            vector<int> column; //columna donde
  se inserto
156.            for(size_t i=0; i<n; i++)
157.            {
158.                for(size_t j=0; j<i; j++)
159.                   zeros.push_back(0);
160.                 if(sig[i]=='='||sig[i]=='<')
161.                {
162.                    uno=1.0;
163.               }
164.                else if(sig[i]=='>') //amplia si
  x1+x2....+xn > bj
165.                {
166.                   uno = -1.0;
167.                   posFas.push_back(i+1);
168.                }
169.                 zeros.push_back(uno);
170.
  M[i].insert(M[i].end(), zeros.begin(), zeros.end()
  );
171.
    if(sig[i]=='>') column.push_back(M[i].size());
172.                if(k<M[i].size())
173.                    k = M[i].size();
174.                mat.push_back(M[i]);
175.                zeros.clear();
176.            }
177.             /*C- Z*/
178.            mat.push_back(F);
179.            /*fase 1*/
180.            double sum =0;
181.            vector<double> temp;
182.            for(size_t i=0; i<F.size(); i++)
183.            {
184.                sum=0;
185.
    for(size_t j=0; j<posFas.size(); j++)
186.                {
```

```
187.                     sum+= mat[posFas[j]][i];
188.                 }
189.                 temp.push_back(sum);
190.             }
191.             size_t x;
192.                 for(size_t y=F.size(); y<k ; y++)
193.                 {
194.                     temp.push_back(0);
195.                 }
196.               for(x=0; x<column.size();x++)
197.              {
198.                  cout<<column[x]<<endl;
199.                  temp[column[x]-1]=-1.0;
200.              }
201.              mat.push_back(temp);
202.              /*rellenar matriz con ceros
   faltantes*/
203.             for(size_t i=0; i< mat.size(); i++)
204.             {
205.
     for(size_t j=mat[i].size(); j<k; j++)
206.                 {
207.                     mat[i].push_back(0);
208.                 }
209.             }
210.             /*rellenar variables artificiales*/
211.             x=0;
212.             n = posFas.size();
213.
    for(unsigned int i=0; i< mat.size(); i++)
214.             {
215.                 vector<double> cero(n,0);
216.                 if(x<posFas.size())
217.                 {
218.                    if(posFas[x]==(int)i)
219.                     {
220.                            cero[x]=1.0;
221.                            x++;
222.                     }
223.
224.                 }
225.
   mat[i].insert(mat[i].end(),cero.begin(),cero.end()
   );
226.                 if(i==0)
227.                 {
```

```
228.
  mat[i].push_back(0); mat[i].push_back(0); //`rimer
  a fila 2 ultimos elementos
229.                }
230.                else{
231.                    if((i-1)<B.size())
232.                    {
233.                        mat[i].push_back(B[i-
  1]); //el elemtno b
234.
  mat[i].push_back(max);   //un gran M para comenzar
  :D
235.                    }
236.                    else{
237.
  mat[i].push_back(0); mat[i].push_back(0);  // los
  2 ultimos casos 000
238.                    }
239.                }
240.            }
241.        /*agregamos los slacks*/
242.        slack.clear();
243.        for(size_t u=0; u<mat.size(); u++)
244.         slack.push_back(-1);
245.    }
246.    void simplex::show_Data()
247.    {
248.    cout.precision(6);
249.        for(size_t k=0; k<slack.size();k++)
250.          cout<<slack[k]<<"  ";
251.          cout<<endl<<endl;
252.        for(size_t i=0; i<mat.size(); i++)
253.        {
254.            for(size_t j=0; j<mat[i].size(); j++)
255.            {
256.                cout<<mat[i][j]<<" ";
257.            }
258.            cout<<endl;
259.        }
260.    }
261.     int simplex::select_column(size_t last)
262.    {
263.        int indx=-1;
264.        size_t ult=last;
265.        double temp =-1;
266.        for(size_t j=0; j<mat[ult].size()-2; j++)
```

```
267.        {
268.
     if(mat[ult][j]>0 && temp<mat[ult][j]) //buscamos
  al mayor
269.            {
270.                temp = mat[ult][j];
271.                indx = j;
272.            }
273.        }
274.        return indx;
275.    }
276.    int simplex::select_row(int p)
277.    {
278.        double ratio = max;
279.        int ind = -1;
280.        int tam= mat[0].size();
281.        //seleccionamos el que tiene mayor ratio
282.        for(size_t i=1; i<mat.size()-2; i++)
283.        {
284.            if(mat[i][p]==0)
285.               mat[i][tam-1] = max;
286.            else if(mat[i][p]<0)
287.                mat[i][tam-1] = min;
288.            else {
289.                mat[i][tam-1]= mat[i][tam-
  2]/mat[i][p];
290.
291.                if(mat[i][tam-1]<ratio)
292.                {
293.                    ratio = mat[i][tam-1];
294.                    ind = i;
295.                }
296.            }
297.        }
298.        return ind;
299.    }
300.    void simplex::elimina_artificial()
301.    {
302.        size_t n = posFas.size();
303.        for(size_t i=0; i<mat.size(); i++)
304.        {
  mat[i].erase(mat[i].begin()+(mat[i].size()-n-
  2),mat[i].end()-2);
305.        }
306.    }
307.    vector<double> simplex::retornarX()
```

```
308.    {
309.        vector<double> A(Objetivo.size(),0);
310.        unsigned int n = slack.size();
311.        for(unsigned int i=0; i< n; i++)
312.        {
313.
     if(slack[i]>=0 && slack[i]<= (int)Objetivo.size(
   ))
314.            {
   A[slack[i]] = *(mat[i].end()-2); //extramos x
   desde la matriz final
315.             }
316.        }
317.        return A;
318.    }
```

2. Clase Lai:

Header

```
1. #ifndef LAI_H
2. #define LAI_H
3. #include <vector>
4. #include "datos.h"
5. #include "database.h"
6. #include "simplex.h"
7. using std::vector;
8. class Lai
9. {
10.     public:
11.         Lai();
12.         double min_z1_nis(vector<double> );
13.         double max_z1_pis(vector<double> );
14.         double max_z2_nis(vector<double> );
15.         double min_z2_pis(vector<double> );
16.         double max_z3_nis(vector<double> );
17.         double min_z3_pis(vector<double> );
18.
     void set_constrains(vector<vector<double> >,vect
   or<double>,vector<char> );
19.
   vector<double> calcular(vector<double> z1_min,vect
   or<double> z1_max,
```

```
20.
  vector<double> z2_min,vector<double> z2_max,
21.
  vector<double> z3_min, vector<double> z3_max);
22.
   vector<double> min_resul(vector<ferti> sel, vecto
  r<double> X);
23.       double z1_pis,z1_nis;
24.       double z2_pis,z2_nis;
25.       double z3_pis,z3_nis;
26.       vector<double> X_z1_pis,X_z1_nis;
27.       vector<double> X_z2_pis,X_z2_nis;
28.       vector<double> X_z3_pis,X_z3_nis;
29.       vector<vector<double> > R;
30.       vector<double> B;
31.       vector<char> signos;
32.    private:
33.
34.
35.
36.
    void ampliar(double z1_pis,double z1_nis, double
   z2_pis,double z2_nis,
37.
   double z3_nis,double z3_pis,vector<double> z1,vec
  tor<double> z2,vector<double> z3);
38.        simplex X;
39.    };
40.    #endif // LAI_H
```

Code

```
1. #include "lai.h"
2. #include <iostream>
3. using namespace std;
4. Lai::Lai()
5. {
6. }
7.
8. void Lai::set_constrains(vector<vector<double> > M
  , vector<double> T,vector<char> sig)
9. {
10.        R=M;
11.        B= T;
12.        signos=sig;
13.    }
```

```
14.
15.     double Lai::max_z1_pis(vector<double> F)
16.     {
17.         X.limpiar();
18.         X.set_data(R,F,B,signos);
19.         X.solved_simplex();
20.         X_z1_pis = X.retornarX();
21.        return X.eval_obj();
22.     }
23.
24.     double Lai::min_z1_nis(vector<double> F)
25.     {
26.         X.limpiar();
27.         X.set_data(R,F,B,signos);
28.         X.solved_simplex();
29.         X_z1_nis = X.retornarX();
30.         return (X.eval_obj()==0)?0:-X.eval_obj();
31.     }
32.     double Lai::max_z2_nis(vector<double> F)
33.     {
34.         X.limpiar();
35.         X.set_data(R,F,B,signos);
36.         X.solved_simplex();
37.         X_z2_nis = X.retornarX();
38.     return X.eval_obj();
39.     }
40.     double Lai::min_z2_pis(vector<double> F)
41.     {
42.         X.limpiar();
43.         X.set_data(R,F,B,signos);
44.         X.solved_simplex();
45.         X_z2_pis = X.retornarX();
46.         return (X.eval_obj()==0)?0:-X.eval_obj();
47.     }
48.     double Lai::max_z3_nis(vector<double> F)
49.     {
50.         X.limpiar();
51.         X.set_data(R,F,B,signos);
52.         X.solved_simplex();
53.         X_z3_nis = X.retornarX();
54.        return X.eval_obj();
55.     }
56.     double Lai::min_z3_pis(vector<double> F)
57.     {
58.         X.limpiar();
59.         X.set_data(R,F,B,signos);
```

```
60.         X.solved_simplex();
61.         X_z3_pis = X.retornarX();
62.      return (X.eval_obj()==0)?0:-X.eval_obj();
63.     }
64.
65.     void Lai::ampliar(double z1_pis,double z1_nis
  , double z2_pis,
66.
     double z2_nis,double z3_nis,double z3_pis,
67.
  vector<double> z1,vector<double> z2,vector<double>
    z3)
68.     {
69.         z1.push_back((z1_nis-z1_pis));
70.         z2.push_back((z2_nis-z2_pis));
71.         z3.push_back((z3_nis-z3_pis));
72.
  R.insert(R.begin(),z3);  B.insert(B.begin(),z3_nis
  ); signos.insert(signos.begin(),'<');
73.
  R.insert(R.begin(),z2); B.insert(B.begin(),z2_nis)
  ; signos.insert(signos.begin(),'<');
74.
  R.insert(R.begin(),z1); B.insert(B.begin(),z1_nis)
  ; signos.insert(signos.begin(),'>');
75.         for(size_t i=3; i<R.size();i++)
76.             R[i].push_back(0);
77.     }
78.
79.     vector<double> Lai::min_resul(vector<ferti> s
  el, vector<double> X)
80.      {
81.          vector<double> L(3,0);
82.          for(size_t i=0; i<sel.size();i++)
83.          {
84.              L[0]+= sel[i].cost_optimo*X[i];
85.              L[1]+= sel[i].cost_medio*X[i];
86.              L[2]+= sel[i].cost_pesimo*X[i];
87.          }
88.          return L;
89.      }
90.
91.     vector<double> Lai::calcular(vector<double> z
  1_min,vector<double> z1_max,
92.
  vector<double> z2_min,vector<double> z2_max,
```

```
93.
  vector<double> z3_min, vector<double> z3_max)
94.     {
95.          z1_pis = this->max_z1_pis(z1_max);
96.         z1_nis = this->min_z1_nis(z1_min);
97.         z2_pis = this->min_z2_pis(z2_min);
98.         z2_nis = this->max_z2_nis(z2_max);
99.         z3_pis = this->min_z3_pis(z3_min);
100.        z3_nis = this->max_z3_nis(z3_max);
101.         cout<<endl<<endl<<"z  pis y nis \n\n";
102.
103.        cout<<"z1_nis: "<<z1_nis<<"   z1_pis:
  "<<z1_pis<<endl;
104.        cout<<"z2_nis: "<<z2_nis<<"   z2_pis:
  "<<z2_pis<<endl;
105.            cout<<"z3_nis: "<<z3_nis<<"   z3_pis:
  "<<z3_pis<<endl;
106.          /*ampliamos la matriz ahora operamos
  con simplex*/
107.         this-
  >ampliar(z1_pis,z1_nis,z2_pis,z2_nis,z3_nis,z3_pis
  ,z1_max,z2_max,z3_max);
108.          X.limpiar();
109.         vector<double> cero(z3_min.size(),0);
110.         cero.push_back(1);
111.         X.set_data(R,cero,B,signos);
112.         X.show_Data();
113.         if(!X.solved_simplex())
114.             return vector<double> ();
115.        X.show_Data();
116.         vector<double> valor_X=X.retornarX();
117.         size_t n=valor_X.size(),i;
118.         double z1,z2,z3;
119.         double uz1,uz2,uz3;
120.         //calculamos z1
121.         z1=0;
122.         for(i=0;i<z1_max.size();i++)
123.             z1+= (z1_max[i]*valor_X[i]);
124.         uz1= (z1-z1_nis)/(z1_pis-z1_nis);
125.         //calculamos z2
126.         z2=0;
127.         for(i=0;i<z2_max.size();i++)
128.             z2+= (z2_max[i]*valor_X[i]);
129.         uz2= (z2-z2_nis)/(z2_pis-z2_nis);
130.         //calculamos z3
131.         z3=0;
```

```
132.         for(i=0;i<z3_max.size();i++)
133.             z3+= (z3_max[i]*valor_X[i]);
134.         uz3= (z3-z3_nis)/(z3_pis-z3_nis);
135.       for( i=0; i< valor_X.size() ; i++)
136.           cout<<valor_X[i]<<"  ";
137.         cout<<endl;
138.         cout<<"z1:  "<<z1<<"  z2: "<<z2<<"  z3:
   "<<z3<<endl;
139.         cout<<"uz1:  "<<uz1<<"  uz2: "<<uz2<<"
    uz3: "<<uz3<<endl;
140.      return valor_X;
141.     }
```

3. Clase Leberling

Header

```
1. #ifndef LEBERLING_H
2. #define LEBERLING_H
3. #include <vector>
4. #include "simplex.h"
5. #include "database.h"
6. using std::vector;
7. class Leberling
8. {
9. public:
10.         Leberling();
11.
     void set_constrains(vector<vector<double> >,vect
   or<double>,vector<char> );
12.         //void calcular(vector<double>
   z1,vector<double> z2,vector<double> z3);
13.         /*calculamos el max de z1*/
14.
     double calc_z1m(vector<double> x2,vector<double>
    x3, vector<double> z1);
15.         /*calculamos el min de z2*/
16.
     double calc_z2m(vector<double> x1,vector<double>
    x3, vector<double> z2);
17.         /*calculamos el min de z3*/
18.
     double calc_z3m(vector<double> x1,vector<double>
    x2, vector<double> z3);
19.         //evaluacion del resultado
```

```
20.
    vector<double> min_resul(vector<ferti> sel, vecto
   r<double> X);
21.          /*realizamos el calculo general una vez
   obtenis max z1 , min z2 y z3*/
22.
    vector<double> calcular( vector<double> z1_max, v
   ector<double> z2_min,  vector<double> z3_min);
23.          //matriz para calculo de simplex
24.          vector<vector<double> > R;
25.          vector<double> B;  //valores B de todas
   las ecuaciones
26.          vector<char> signos; //vector de
   simbolos <, > , = de cada ecuacion
27.          vector<double> x10,x20,x30; //variables
   para la consturccion del modelo
28.          double z110,z210,z310;
29.          double z1m,z2m,z3m;
30.     private:
31.         //funcion para ampliar matriz
32.
     void ampliar(double z10,double z1m, double z20,d
   ouble z2m,
33.
    double z30,double z3m,vector<double> z1,vector<do
   uble> z2,vector<double> z3);
34.
35.         //simplex que desarrollará el modelo
36.         simplex X;
37.     };
38.     #endif // LEBERLING_H
```

Code:

```
1. #include "leberling.h"
2. #include <vector>
3. #include <iostream>
4. using namespace std;
5. Leberling::Leberling()
6. {
7. }
8. void Leberling::set_constrains(vector<vector<doubl
   e> > M, vector<double> T,vector<char> sig)
9. {
10.         R.clear();
11.         R=M;
```

```
12.         B.clear();
13.         B= T;
14.         signos.clear();
15.         signos=sig;
16.     }
17.
18.     double Leberling::calc_z1m(vector<double> x2,
    vector<double> x3, vector<double> z1)
19.     {
20.         double suma=0;
21.         double suma1=0;
22.         size_t i;
23.         for( i=0;i<z1.size(); i++)
24.             suma+=(z1[i]*x2[i]);
25.         for( i=0;i<z1.size(); i++)
26.             suma1+=(z1[i]*x3[i]);
27.         return (suma<suma1)?suma:suma1;
28.     }
29.     double Leberling::calc_z2m(vector<double> x1,
    vector<double> x3, vector<double> z2)
30.     {
31.         double suma=0;
32.         double suma1=0;
33.         size_t i;
34.         for( i=0;i<z2.size(); i++)
35.             suma+=(z2[i]*x1[i]);
36.
37.         for( i=0;i<z2.size(); i++)
38.             suma1+=(z2[i]*x3[i]);
39.         return (suma<suma1)?suma1:suma;
40.     }
41.     double Leberling::calc_z3m(vector<double> x1,
    vector<double> x2, vector<double> z3)
42.     {
43.         double suma=0;
44.         double suma1=0;
45.         size_t i;
46.         for( i=0;i<z3.size(); i++)
47.             suma+=(z3[i]*x1[i]);
48.         for( i=0;i<z3.size(); i++)
49.             suma1+=(z3[i]*x2[i]);
50.         return (suma<suma1)?suma1:suma;
51.     }
52.
53.     void Leberling::ampliar(double z10,double z1m
   , double z20,double z2m,
```

```
54.
     double z30,double z3m,vector<double> z1,vector<d
   ouble> z2,vector<double> z3)
55.     {
56.         z1.push_back(-(z10-z1m));
57.         z2.push_back((z2m-z20));
58.         z3.push_back((z3m-z30));
59.
   R.insert(R.begin(),z3);  B.insert(B.begin(),z3m);
   signos.insert(signos.begin(),'<');
60.
   R.insert(R.begin(),z2); B.insert(B.begin(),z2m); s
   ignos.insert(signos.begin(),'<');
61.
   R.insert(R.begin(),z1); B.insert(B.begin(),z1m); s
   ignos.insert(signos.begin(),'>');
62.         for(size_t i=3; i<R.size();i++)
63.             R[i].push_back(0);
64.     }
65.
66.     vector<double> Leberling::min_resul(vector<fe
   rti> sel, vector<double> X)
67.     {
68.         vector<double> L(3,0);
69.         for(size_t i=0; i<sel.size();i++)
70.         {
71.             L[0]+= sel[i].cost_optimo*X[i];
72.             L[1]+= sel[i].cost_medio*X[i];
73.             L[2]+= sel[i].cost_pesimo*X[i];
74.         }
75.         return L;
76.     }
77.     vector<double> Leberling::calcular( vector<do
   uble> z1_max, vector<double> z2_min,  vector<doubl
   e> z3_min)
78.     {    /*max z1'*/
79.        X.limpiar();
80.        X.set_data(R,z1_max,B,signos);
81.        if(!X.solved_simplex())
82.            return vector<double> ();
83.        x10=X.retornarX();
84.        z110=X.eval_obj();
85.        cout<<"x10:  es\n";
86.         for(unsigned int i=0; i<x10.size();i++)
87.             cout<<x10[i]<<"  ";
88.         cout<<" evaluado : "<<z110<<endl;
```

```
89.       /*min z2*/
90.       X.limpiar();
91.       X.set_data(R,z2_min,B,signos);
92.       if(!X.solved_simplex())
93.           return vector<double> ();
94.       x20=X.retornarX();
95.       z210= (X.eval_obj()==0)?0:-X.eval_obj();
96.       cout<<"x20:  es\n";
97.        for(unsigned int i=0; i<x20.size();i++)
98.            cout<<x20[i]<<"  ";
99.        cout<<" evaluado : "<<z210<<endl;
100.      /*min z3*/
101.      X.limpiar();
102.      X.set_data(R,z3_min,B,signos);
103.      if(!X.solved_simplex())
104.          return vector<double> ();
105.      x30=X.retornarX();
106.      z310=(X.eval_obj()==0)?0:-X.eval_obj();
107.      cout<<"x30:  es\n";
108.       for(unsigned int i=0; i<x30.size();i++)
109.           cout<<x30[i]<<"  ";
110.       cout<<" evaluado : "<<z310<<endl;
111.
112.       //multiplicamos z2_min x -1
113.       for(size_t i =0; i<z2_min.size();i++)
114.           z2_min[i]*=-1;
115.       //multiplicamos z3_min x -1
116.       for(size_t i =0; i<z3_min.size();i++)
117.           z3_min[i]*=-1;
118.
119.      /*calculamos las tolerancias*/
120.      z1m = calc_z1m(x20,x30,z1_max);
121.      z2m = calc_z2m(x10,x30,z2_min);      //
  z2m = (z2m==0)?0:-z2m;
122.      z3m = calc_z3m(x10,x20,z3_min);  //  z3m =
  (z3m==0)?0:-z3m;
123.       cout<<"tolerancias : z1m: "<<z1m<<"  z2m:
  "<<z2m<<"  z3m: "<<z3m<<endl;
124.    this-
  >ampliar(z110,z1m,z210,z2m,z310,z3m,z1_max,z2_min,
  z3_min);
125.       /*ahora calculamos*/
126.      vector<double> cero(z3_min.size(),0);
127.      cero.push_back(1);
128.      X.limpiar();
129.      X.set_data(R,cero,B,signos);
```

```
130.       X.show_Data();
131.       if(!X.solved_simplex())
132.           return vector<double> ();
133.       vector<double> valor_X=X.retornarX();
134.       for(size_t i=0; i< valor_X[i]; i++)
135.            cout<<valor_X[i]<<"  ";
136.         cout<<endl;
137.       return valor_X;
138.    }
```

4. Clase Database

Heaader

```
1. #ifndef DATABASE_H
2. #define DATABASE_H
3. #include <string>
4. #include <vector>
5. #include <QSqlDatabase>
6. #include <QSqlError>
7. #include <QSqlQuery>
8. using std::string;
9. using std::vector;
10.     /*
11.       estructura de dato (dato) que contiene
12.     informacion de un fertilizante
13.
14.     */
15.
16.     typedef struct
17.     {
18.         string nombre;
19.         double nitrato;
20.         double fosforo;
21.         double potasio;
22.         double cost_optimo;
23.         double cost_medio;
24.         double cost_pesimo;
25.     } ferti;
26.     typedef struct
27.     {
28.         string nombre;
29.     } cultivo;
30.     class Database
31.     {
32.     public:
```

```
33.         Database();
34.         ~Database();
35.         bool load_fertilizante(string
  name); //carga una base de datos de fertilizante
36.         bool load_cultivo(string name); //carga
  una base de datos de cultivos
37.
  vector<string> get_fert(QString name_table);  //li
  sta los nombre de fertilizantes de la tabla :
  name_table
38.         ferti
  consult_fert(QString name_table,QString key); //co
  nsulta en fertilizantes por clave en el tabla:
  name_table
39.
40.
  vector<string> get_cultivo(QString name_table); //
  obtiene el valor de un cultivo
41.         cultivo
  consult_cultivo(QString name_table,QString key); /
  /consulta en cultivo por clave en el tabla:
  name_table
42.     private :
43.             QSqlDatabase db_fert; //BD_FERTI
44.             QSqlDatabase db_cultivo; // BD
  cultivo
45.     };
46.     #endif // DATABASE_H
```

Code:

```
1. #include "database.h"
2. #include <QSqlDatabase>
3. #include <QSqlError>
4. #include <QSqlQuery>
5. #include <QMessageBox>
6. #include <QtGui>
7. #include <vector>
8. #include <string>
9. using std::string;
10.     using std::vector;
11.     using namespace std;
12.     Database::Database()
13.     {
```

```
14.     }
15.     Database::~Database()
16.     {
17.       db_fert.close();
18.       db_cultivo.close();
19.     }
20.     bool Database::load_fertilizante(string name)
21.     {
22.
   db_fert = QSqlDatabase::addDatabase("QSQLITE"); //
   seleccionamos el tipo de BD SQLite
23.
   db_fert.setDatabaseName(QString::fromStdString(nam
   e)); // agregamos el nombre de la BD
24.             if (!db_fert.open()) {
25.                // muestra el mensaje si no puede
   conectarse con la BD
26.
27.
     QMessageBox::critical(0, qApp->tr("Cannot open
   database"),
28.                           qApp->tr("Unable to
   establish a database connection.\n"
29.                                    "This example
   needs SQLite support. Please read "
30.                                    "the Qt SQL
   driver documentation for information how "
31.                                    "to build
   it.\n\n"
32.                                    "Click Cancel
   to exit."), QMessageBox::Cancel);
33.                   return false;
34.                   }
35.         return true;}
36.     bool Database::load_cultivo(string name)
37.     {
38.
   db_cultivo = QSqlDatabase::database(); //seleccion
   amos el tipo de BD SQLite
39.
   db_cultivo.setDatabaseName(QString::fromStdString(
   name)); // agregamos el nombre de la BD
40.             if (!db_cultivo.open()) {
41.                 //muestra el mensaje si no puede
   conectarse con la BD
```

```
42.
    QMessageBox::critical(0, qApp->tr("Cannot open
  database"),
43.                          qApp->tr("Unable to
  establish a database connection.\n"
44.                                   "This example
  needs SQLite support. Please read "
45.                                   "the Qt SQL
  driver documentation for information how "
46.                                   "to build
  it.\n\n"
47.                                   "Click Cancel
  to exit."), QMessageBox::Cancel);
48.
49.                   return false;
50.                   }
51.         return true;
52.     }
53.     vector<string> Database::get_fert(QString nam
  e_table)
54.     {
55.         vector<string> L; //lista de tipo de
  fertilizantes
56.         string temp;
57.         QSqlQuery query; //clase para realizar
  las consultas en SQL
58.         query.exec("SELECT *FROM "+name_table);
59.         while(query.next())
60.         {
61.
  temp = query.value(0).toString().toStdString();
62.             L.push_back(temp);
63.         }
64.         return L;
65.     }
66.     vector<string> Database::get_cultivo(QString
  name_table)
67.     {
68.         vector<string> L; //lista de tipo de
  cultivos
69.         string temp;
70.         QSqlQuery query; //clase para realizar
  las consultas en sQL
71.         query.exec("SELECT *FROM "+name_table);
72.
73.         while(query.next())
```

```
74.         {
75.
  temp = query.value(0).toString().toStdString();
76.             L.push_back(temp);
77.         }
78.         return L;
79.     }
80.     ferti
  Database::consult_fert(QString name_table,QString
  key)
81.     {
82.         ferti F;
83.         QSqlQuery query; //clase para realizar
  las consultas en SQL
84.          query.exec("SELECT *FROM "+name_table+"
  WHERE nombre='"+key+"' ");
85.         while(query.next())
86.         {
87.
  F.nombre  = query.value(0).toString().toStdString(
  );
88.
  F.nitrato = query.value(1).toDouble();
89.
  F.fosforo = query.value(2).toDouble();
90.
  F.potasio = query.value(3).toDouble();
91.
  F.cost_optimo= query.value(4).toDouble();
92.
  F.cost_medio= query.value(5).toDouble();
93.
  F.cost_pesimo= query.value(6).toDouble();
94.             break;
95.         }
96.         return F;
97.     }
98.     cultivo
  Database::consult_cultivo(QString name_table,QStri
  ng key)
99.     {
100.        cultivo S;
101.        QSqlQuery query; //clase para realizar
  las consultas en sQL
102.        query.exec("SELECT *FROM "+name_table+"
  WHERE nombre='"+key+"' ");
```

```
103.        while(query.next())
104.        {
105.
  S.nombre = query.value(0).toString().toStdString()
  ;         break;
106.        }
107.        return S;
108.    }
```

5. Clase Suelo

Header

```
1. #ifndef SUELO_H
2. #define SUELO_H
3. class suelo
4. {
5. public:
6.     double ext_suelo[3]; // o: nitrogeno 1:
  fosforo 2: potasio
7.     double coef_us_nitrogeno[3]; //0: f1 1: f2
   2:f3
8.     double coef_us_fosforo[3]; //0: f1 1: f2  2:f3
9.     double coef_us_potasio[3]; //0: f1 1: f2
   2:f3
10.         double anali_suelo[5]; // 0:ph 1:mo 2: n
  3: pDisp 4: kdisp
11.         double QN,QP,QK;
12.         double aportNu[3]; //0: nit 1: fos  2:
  pot
13.         suelo();
14.         ~suelo();
15.         void set_ext_suelo(double []); //ingreso
  de coeficientes del suelo
16.
     void set_coef_us_nit(double []);//coeficiente de
  nit. para cada fertilizante
17.
     void set_coef_us_fos(double []);//coeficiente de
  fosf. para cada fertilizante
18.
     void set_coef_us_pot(double []);//coeficiente de
  potasio para cada fertilizante
```

```
19.
    void set_anali_suelo(double []);//coeficientes
  del analisis de suelo
20.         void calc_aport(); //calcula el aporte de
  nutrientes
21.         void calc_abona(); //calcule la cantidad
  de nutrientes
22.     private:
23.         double factor;
24.         double costa; //factor de la costa
25.         double sierra; //factor de la sierra
26.     };
27.     #endif // SUELO_H
```

Code:

```
1. #include "suelo.h"
2. #include <iostream>
3. using namespace std;
4. suelo::suelo()
5. {
6.  QN=QP=QK=0;
7.  factor=2;
8.  costa=2.5;
9.  sierra=2;
10.     }
11.     suelo::~suelo()
12.     {}
13.     void suelo::set_ext_suelo(double A[3])
14.     {    // o: nitrogeno 1: fosforo 2: potasio
15.       ext_suelo[0]= A[0];
16.       ext_suelo[1]= A[1];
17.       ext_suelo[2]= A[2];
18.     }
19.     void suelo::set_coef_us_nit(double A[3])
20.     {    //0: f1 1: f2  2:f3
21.         coef_us_nitrogeno[0]= A[0];
22.         coef_us_nitrogeno[1]= A[1];
23.         coef_us_nitrogeno[2]= A[2];
24.     }
25.     void suelo::set_coef_us_fos(double A[3])
26.     {
27.          //0: f1 1: f2  2:f3
28.         coef_us_fosforo[0]= A[0];
```

```
29.         coef_us_fosforo[1]= A[1];
30.         coef_us_fosforo[2]= A[2];
31.     }
32.
33.     void suelo::set_coef_us_pot(double A[3])
34.     {
35.          //0: f1 1: f2  2:f3
36.         coef_us_potasio[0]= A[0];
37.         coef_us_potasio[1]= A[1];
38.         coef_us_potasio[2]= A[2];
39.     }
40.     void suelo::set_anali_suelo(double A[5])
41.     {
42.          // 0:ph 1:mo 2: n 3: potDisp 4: kdisp
43.
44.         anali_suelo[0] = A[0];
45.         anali_suelo[1] = A[1];
46.         anali_suelo[2] = A[2];
47.         anali_suelo[3] = A[3];
48.         anali_suelo[4] = A[4];
49.     }
50.     void suelo::calc_aport()
51.     {
52.          // o: nitrogeno 1: fosforo 2: potasio
53.
   aportNu[0] = (1000*anali_suelo[1]*costa)/200;
54.         aportNu[1] = anali_suelo[3]*2*2.28;
55.         aportNu[2] = anali_suelo[4]*2.4;
56.     }
57.     void suelo::calc_abona()
58.     {    /*
59.          ext_suelo[0] is extracion de N
60.          aportNu[0] aporte del nitrogeno
61.          coef_us_nitrogeno[0] coef para f1
62.          coef_us_nitrogeno[1] coef para f2
63.          coef_us_nitrogeno[2] coef para f3
64.          anali_suelo[1] es mo
65.         */
66.         QN = (ext_suelo[0]-
   aportNu[0]*coef_us_nitrogeno[0] -
    anali_suelo[1]* coef_us_nitrogeno[1])/ coef_us_ni
   trogeno[2];
67.         /*
68.          ext_suelo[1] is extracion de P
69.          aportNu[1] aporte del potasio
70.          coef_us_potasio[0] coef para f1
```

```
71.          coef_us_potasio[1] coef para f2
72.          coef_us_potasio[2] coef para f3
73.          anali_suelo[1] es mo
74.         */
75.         QP = (ext_suelo[1]-
  aportNu[1]*coef_us_fosforo[0] -
   anali_suelo[1]* coef_us_fosforo[1])/ coef_us_fosf
  oro[2];
76.         /*
77.          ext_suelo[2] is extracion de P
78.          aportNu[2] aporte del potasio
79.          coef_us_fosforo[0] coef para f1
80.          coef_us_fosforo[1] coef para f2
81.          coef_us_fosforo[2] coef para f3
82.          anali_suelo[1] es mo
83.
84.         */
85.         QK = (ext_suelo[2]-
  aportNu[2]*coef_us_potasio[0] -
   anali_suelo[1]* coef_us_potasio[1])/coef_us_potas
  io[2];
86.     }
```

6. Clase datos

Header

```
1. #ifndef DATOS_H
2. #define DATOS_H
3. #include "database.h"
4. #include "suelo.h"
5. class datos
6. {
7. public:
8.     suelo tierra;
9.     datos();
10.         datos(vector<ferti> ); //constructor con
  lista de datos
11.         ~datos();
12.         vector<ferti> List; //lista de objetos
  fertilizantes
13.         vector<vector<double> > M; //matriz
14.         vector<double> B; //B de ecuaciones
15.         vector<char> signos; //signos de cada
  ecuacion <, > , =
```

```
16.
  vector<vector<double> > gen_max(); //maximo
17.
  vector<vector<double> > get_restricciones(){return
   M;} //restricciones
18.
  vector<double> get_B(){return B;} //obtenemos B
19.
  vector<char> get_signos(){return signos;} //obtene
  mos signos
20.         void clear(); //limpiamos la data
21.         void set_suelo(suelo s){tierra=s;} //set
  para nuestro tipo de suelo con su respectivamos
  mediciones
22.         void set_fert(vector<ferti> ); //set de
  lista de fertilizantes seleccionados
23.         void genera(); //crea la matriz de
  restricciones
24.         vector<double> min_z1(); //min z1:
  cost_medio-cost_pesimo
25.         vector<double> max_z1(); //max z1 :
  cost_pesimo- cost_medio
26.         vector<double> max_z2(); // max z2 :
  .cost_medio
27.         vector<double> min_z2(); //min z2 : 0 o -
  cost_medio
28.         vector<double> max_z3(); //max z3:
  cost_medio-cost_optimo
29.         vector<double> min_z3(); //min z3:
  cost_optimo- cost_medio
30.      private:
31.         void hacer_min(double& r);
32.      };
33.    #endif // DATOS_H
```

Code:

```
1. #include "datos.h"
2. #include <algorithm>
3. #include <iostream>
4. using namespace std;
5. datos::datos(vector<ferti> L)
6. {
7.   set_fert(L);
8. }
9. datos::datos()
```

```
10.     {
11.     }
12.     datos::~datos(){
13.     }
14.     void datos::set_fert(vector<ferti> L)
15.     {
16.         List.clear();
17.        List = L ;
18.     }
19.     void datos::hacer_min(double &r)
20.     {
21.         r*=(-1);
22.     }
23.     struct myclass {
24.       void operator() (double& i) {i=-1*i;}
25.     } myobject;
26.     void datos::genera()
27.     {
28.         vector<double> temp;
29.         vector<double> temp1;
30.         vector<double> temp2;
31.           M.clear();
32.           signos.clear();
33.           B.clear();
34.         for(size_t i=0; i<List.size(); i++)
35.         {
36.
  temp.push_back(List[i].nitrato); //fila Nitrato
37.
  temp1.push_back(List[i].fosforo); //fila fosforo
38.
  temp2.push_back(List[i].potasio); //fila potasio
39.         }
40.     /*maximo para N*/
41.         B.push_back(tierra.QN+20);
42.         M.push_back(temp);
43.         /*Minimo para N */
44.         B.push_back(tierra.QN-20);
45.          M.push_back(temp);
46.         /*maximo para P*/
47.         B.push_back(tierra.QP+20);
48.         M.push_back(temp1);
49.         /*minimo para P*/
50.           B.push_back(tierra.QP-20);
51.          M.push_back(temp1);
52.           /*Maximo para K */
```

```
53.         B.push_back(tierra.QK+20);
54.         M.push_back(temp2);
55.         /*minimo para K*/
56.         B.push_back(tierra.QK-20);
57.          M.push_back(temp2);
58.
    signos.push_back('<'); signos.push_back('>');
59.
    signos.push_back('<'); signos.push_back('>');
60.
    signos.push_back('<'); signos.push_back('>');
61.     }
62.
63.     vector<double> datos::min_z1()
64.     {
65.         vector<double> coef;
66.         for(size_t i=0; i<List.size(); i++)
67.         {
68.             coef.push_back(List[i].cost_medio-
   List[i].cost_pesimo);
69.         }
70.         return coef;
71.     }
72.     vector<double> datos::max_z1()
73.     {
74.         vector<double> coef;
75.         for(size_t i=0; i<List.size(); i++)
76.         {
77.             coef.push_back(List[i].cost_pesimo-
   List[i].cost_medio);
78.         }
79.         return coef;
80.     }
81.     vector<double> datos::max_z2()
82.     {
83.         vector<double> coef;
84.         for(size_t i=0; i<List.size(); i++)
85.         {
86.             coef.push_back(List[i].cost_medio);
87.         }
88.         return coef;
89.     }
90.     vector<double> datos::min_z2()
91.     {
92.         vector<double> coef;
93.         for(size_t i=0; i<List.size(); i++)
```

```
94.         {
95.
    coef.push_back(List[i].cost_medio==0?0:-
    List[i].cost_medio);
96.         }
97.         return coef;
98.     }
99.     vector<double> datos::max_z3()
100.    {
101.        vector<double> coef;
102.        for(size_t i=0; i<List.size(); i++)
103.        {
104.            coef.push_back( List[i].cost_medio-
    List[i].cost_optimo );
105.        }
106.        return coef;
107.    }
108.    vector<double> datos::min_z3()
109.    {
110.        vector<double> coef;
111.        for(size_t i=0; i<List.size(); i++)
112.        {
113.            coef.push_back(  List[i].cost_optimo-
     List[i].cost_medio );
114.        }
115.        return coef;
116.    }
```

7. Clase equations:

Header

```
1. #ifndef EQUATIONS_H
2. #define EQUATIONS_H
3. #include <QWidget>
4. #include <vector>
5. #include <QString>
6. using std::vector;
7. namespace Ui {
8.     class equations;}
9. class equations : public QWidget
10.     {
11.         Q_OBJECT
12.     public:
13.         explicit equations(QWidget *parent = 0);
14.         ~equations();
```

```
15.         /*variables para metodo de lai*/
16.         vector<double> min_z1;  double z1_pis;
17.         vector<double> max_z1;  double z1_nis;
18.         vector<double> min_z2;  double z2_pis;
19.         vector<double> max_z2;  double z2_nis;
20.         vector<double> min_z3;  double z3_nis;
21.         vector<double> max_z3;  double z3_pis;
22.         /*variables para metodo de alpha*/
23.         vector<vector<double> > R;
24.         vector<double> B;
25.         vector<char> signos;
26.         vector<double> resultado;
27.         //funciones para mostrar lai
28.
      QString mostrar_z1_lai(vector<double> x_pis,vect
    or<double> x_nis);
29.
      QString mostrar_z2_lai(vector<double> x_pis,vect
    or<double> x_nis);
30.
      QString mostrar_z3_lai(vector<double> x_pis,vect
    or<double> x_nis);
31.         QString mostrar_alfa_lai(vector<double>);
32.
33.
      QString mostrar_variable(QString var, vector<dou
    ble> L);
34.     private:
35.         Ui::equations *ui;
36.     };
37.     #endif // EQUATIONS_H
```

Code:

```
1. #include "equations.h"
2. #include "ui_equations.h"
3. equations::equations(QWidget *parent) :
4.     QWidget(parent),
5.     ui(new Ui::equations)
6. {
7.     ui->setupUi(this);
8.     this->setWindowTitle("Resultados Lai");
9. }
10.     equations::~equations()
```

```
11.     {
12.         delete ui;
13.     }
14.     QString equations::mostrar_z1_lai(vector<doub
  le> x_pis,vector<double> x_nis)
15.     {
16.         QString S,T;
17.         QString temp;
18.         S= "Z1pis = max ";
19.         for(size_t i=0; i<min_z1.size()-1; i++)
20.         {
21.             S = S + temp.setNum(max_z1[i])+"x";
22.
  S = S + T.setNum(i+1)+((max_z1[i+1]<=0)?"    ":"
   +  ");
23.         }
24.         S= S + temp.setNum(max_z1[max_z1.size()-
  1])+"x"+T.setNum(max_z1.size())+"\n";
25.
26.         S = S + mostrar_variable("X1_pis",x_pis);
27.         S= S+ "  Z1nis = min ";
28.         for(size_t i=0; i<max_z1.size()-1; i++)
29.         {
30.             S = S + temp.setNum(max_z1[i])+"x";
31.             S = S+T.setNum(i+1)+"
   "+((max_z1[i+1]<=0)?"   ":"  +  ");
32.         }
33.         S= S + temp.setNum(max_z1[max_z1.size()-
  1])+"x"+T.setNum(max_z1.size())+"\n";
34.       S = S + mostrar_variable("X1_nis",x_nis);
35.         S= S + "Z1pis = " +T.setNum(this-
  >z1_pis) + "\n";
36.         S= S + "Z1nis = " + T.setNum(this-
  >z1_nis) + "\n";
37.         ui->text_prim->insertPlainText(S);
38.         return S;
39.     }
40.     QString equations::mostrar_z2_lai(vector<doub
  le> x_pis,vector<double> x_nis)
41.     {
42.         QString S,T;
43.         QString temp;
44.         S= "Z2pis = min ";
45.         for(size_t i=0; i<max_z2.size()-1; i++)
46.         {
```

```
47.
   S = S + temp.setNum(max_z2[i])+"x"+T.setNum(i+1)+(
   (max_z2[i+1]<=0)?"   ":"  +  ");
48.         }
49.         S= S + temp.setNum(max_z2[max_z2.size()-
   1])+"x"+T.setNum(max_z2.size())+"\n";
50.
   S = S + mostrar_variable("X2_pis",x_pis);
51.         S= S+ "  Z2nis = max ";
52.         for(size_t i=0; i<min_z2.size()-1; i++)
53.         {
54.
   S = S + temp.setNum(max_z2[i])+"x"+T.setNum(i+1)+(
   (max_z2[i+1]<=0)?"   ":"  +  ");
55.         }
56.         S= S + temp.setNum(max_z2[max_z2.size()-
   1])+"x"+T.setNum(max_z2.size())+"\n";
57.
58.
   S = S + mostrar_variable("X2_nis",x_nis);
59.         S= S + "Z2pis = " + temp.setNum(this-
   >z2_pis) + "\n";
60.         S= S + "Z2nis = " + temp.setNum(this-
   >z2_nis) + "\n";
61.         ui->text_seg->insertPlainText(S);
62.         return S;
63.     }
64.     QString equations::mostrar_z3_lai(vector<doub
   le> x_pis,vector<double> x_nis)
65.     {
66.         QString S, T;
67.         QString temp;
68.         S= "Z3pis = min ";
69.         for(size_t i=0; i<max_z3.size()-1; i++)
70.         {
71.
   S = S + temp.setNum(max_z3[i])+"x"+T.setNum(i+1)+(
   (max_z3[i+1]<=0)?"    ":"  +  ");
72.         }
73.         S= S + temp.setNum(max_z3[max_z3.size()-
   1])+"x"+T.setNum(max_z3.size())+"\n";
74.         S = S + mostrar_variable("X3_pis",x_pis);
75.         S= S+ "  Z3nis = max ";
76.         for(size_t i=0; i<min_z3.size()-1; i++)
77.         {
```

```
78.
   S = S + temp.setNum(max_z3[i])+"x"+T.setNum(i+1)+(
   (max_z3[i+1]<=0)?"    ":"  +  ");
79.         }
80.         S= S + temp.setNum(max_z3[max_z3.size()-
   1])+"x"+T.setNum(max_z3.size())+"\n";
81.         S = S + mostrar_variable("X3_nis",x_nis);
82.         S= S + "Z3pis = " + temp.setNum(this-
   >z3_pis) + "\n";
83.         S= S + "Z3nis = " + temp.setNum(this-
   >z3_nis) + "\n";
84.         ui->text_ter->insertPlainText(S);
85.         return S;
86.     }
87.     QString equations::mostrar_alfa_lai(vector<do
   uble> Xs)
88.     {    QString T,N,P;
89.         T="Maximizar Z: alpha \n ";
90.         T= T+ "X"+P.setNum(R[0].size())+" =
   alpha\n\n";
91.         for(size_t i=0; i<R.size(); i++)
92.         {
93.             for(size_t j=0; j<R[i].size(); j++)
94.             {
95.
   T= T + N.setNum(R[i][j])+"X"+P.setNum(j+1);
96.                 if(j<(R[i].size()-1))
97.                     T=T+ ((R[i][j+1]>=0)?"  +
   ":"   ");
98.             }
99.             if(signos[i]=='<')
100.                T= T+" <=   ";
101.            else if(signos[i]=='>')
102.                T= T+" >=   ";
103.            else
104.                T= T+" =    ";
105.
106.            T = T+P.setNum((B[i]))+"\n";
107.        }
108.        T=T+"\n"+mostrar_variable("X",Xs);
109.       T=T+mostrar_variable("Minimo :
   ",resultado);
110.       ui->text_cua->insertPlainText(T);
111.       return T;
112.   }
```

```
113.    QString equations::mostrar_variable(QString v
  ar, vector<double> L)
114.    {
115.        QString S;
116.        QString temp;
117.        S= var+"= {";
118.        for(unsigned int i=0; i< L.size()-1; i++)
119.             S = S +temp.setNum(L[i])+",";
120.        S = S+temp.setNum(L[L.size()-
  1])+"}\n\n";
121.        return S;
122.    }
```

8. Clase Problema

Header

```
1. #ifndef PROBLEMA_H
2. #define PROBLEMA_H
3. #include <QWidget>
4. #include "datos.h"
5.
6. namespace Ui {
7.     class problema;
8. }
9. class problema : public QWidget
10.     {
11.         Q_OBJECT
12.
13.     public:
14.         explicit problema(QWidget *parent = 0);
15.         ~problema();
16.         void set_data(datos *S );
17.     private:
18.         Ui::problema *ui;
19.     };
20.      #endif // PROBLEMA_H
```

Code

```
1. #include "problema.h"
2. #include "ui_problema.h"
3. #include "datos.h"
4. #include <vector>
5. using namespace std;
```

```
6.
7. problema::problema(QWidget *parent) :
8.     QWidget(parent),
9.     ui(new Ui::problema)
10.     {
11.         ui->setupUi(this);
12.         this->setWindowTitle("Problema a
   resolver");
13.     }
14.     problema::~problema()
15.     {
16.         delete ui;
17.     }
18.     void problema::set_data(datos *S)
19.     {
20.         vector<ferti> L;
21.         QString temp;
22.         L = S->List;
23.         /*datos que fueron seleccionados*/
24.         for(size_t i=0; i<L.size(); i++)
25.         {
26.             ui->fertilizante-
   >insertColumn(i); //insertamos una columna
27.             ui->fertilizante-
   >setItem(0,i, new QTableWidgetItem(L[i].nombre.c_s
   tr())); //nombre del fertilizante
28.             ui->fertilizante-
   >setItem(1,i, new QTableWidgetItem("X"+temp.setNum
   (i+1))); //Variable
29.             ui->fertilizante-
   >setItem(2,i, new QTableWidgetItem(temp.setNum(L[i
   ].nitrato))); //%N
30.             ui->fertilizante-
   >setItem(3,i, new QTableWidgetItem(temp.setNum(L[i
   ].fosforo))); //%P
31.             ui->fertilizante-
   >setItem(4,i, new QTableWidgetItem(temp.setNum(L[i
   ].potasio))); //%K
32.              ui->fertilizante-
   >setItem(5,i, new QTableWidgetItem(temp.setNum(L[i
   ].cost_optimo))); //nombre costo optimo
33.             ui->fertilizante-
   >setItem(6,i, new QTableWidgetItem(temp.setNum(L[i
   ].cost_medio))); //nombre costo medio
```

```
34.             ui->fertilizante-
  >setItem(7,i, new QTableWidgetItem(temp.setNum(L[i
  ].cost_pesimo))); //nombre costo pesimo
35.         }
36.         //set de variables del problema
37.         vector<vector<double> > M;
38.         vector<double> B;
39.         vector<char> cad;
40.         M = S->get_restricciones();
41.         B = S->get_B();
42.         cad = S->get_signos();
43.         QString T,N,P;
44.         T="Maximizar  ";
45.         for(size_t i=0; i<M[0].size(); i++)
46.         {
47.
  T= T+"X"+P.setNum(i+1)+"(C_o,C_m,C_p)"+P.setNum(i+
  1)+(i<(M[0].size()-1)?"  + ":" ");
48.         }
49.         T= T+"\n\n";
50.         for(size_t i=0; i<M.size(); i++)
51.         {
52.             for(size_t j=0; j<M[i].size(); j++)
53.             {
54.
  T= T + N.setNum(M[i][j])+"X"+P.setNum(j+1);
55.                 if(j<(M[i].size()-1))
56.                     T=T+ ((M[i][j+1]>=0)?"  +
  ":"   ");
57.             }
58.             if(cad[i]=='<')
59.                 T= T+" <=   ";
60.             else if(cad[i]=='>')
61.                 T= T+" >=   ";
62.             else
63.                 T= T+" =    ";
64.             T = T+P.setNum((B[i]))+"\n";
65.         }
66.
67.        T=T+"\n\n";
68.        ui->texto->appendPlainText(T);
69.     }
```

9. Clase tableview

   Header

```
1. #ifndef TABLEVIEW_H
2. #define TABLEVIEW_H
3. #include <QWidget>
4. #include  "database.h"
5. namespace Ui {
6.     class tableview;
7. }
8. class tableview : public QWidget
9. {
10.         Q_OBJECT
11.     public:
12.         explicit tableview(QWidget *parent = 0);
13.         ~tableview();
14.          bool load_data(QString );
15.     private:
16.         Ui::tableview *ui;
17.         vector<ferti> backup(); //carga todos los
   datos de la tabla
18.     private slots:
19.         void on_guardar_but_clicked();
20.         void on_agregar_but_clicked();
21.     };
22.
23.     #endif // TABLEVIEW_H
```

Code:

```
1. #include "tableview.h"
2. #include "ui_tableview.h"
3. #include <iostream>
4. using namespace std;
5. tableview::tableview(QWidget *parent) :
6.     QWidget(parent),
7.     ui(new Ui::tableview)
8. {
9.     ui->setupUi(this);
10.         //this->load_data("ferti");
11.     }
12.     tableview::~tableview()
13.     {
14.         delete ui;
15.     }
16.     void tableview::on_agregar_but_clicked()
17.     {
18.         int row=ui->tableWidget->rowCount();
19.         QString S;
```

```
20.         ui->tableWidget->insertRow(row);
21.         ui->tableWidget-
  >setItem(row,0, new QTableWidgetItem(S));
22.
23.         ui->tableWidget-
  >setItem(row,1, new QTableWidgetItem(S));
24.         ui->tableWidget-
  >setItem(row,2, new QTableWidgetItem(S));
25.         ui->tableWidget-
  >setItem(row,3, new QTableWidgetItem(S));
26.         ui->tableWidget-
  >setItem(row,4, new QTableWidgetItem(S));
27.         ui->tableWidget-
  >setItem(row,5, new QTableWidgetItem(S));
28.         ui->tableWidget-
  >setItem(row,6, new QTableWidgetItem(S));
29.     }
30.     void tableview::on_guardar_but_clicked()
31.     {
32.         QSqlQuery query;
33.
34.               cout<<query.exec("DELETE  FROM
  ferti ")<<endl;
35.         vector<ferti> datos;
36.         ferti temp;
37.         int num;
38.         QString toParse;
39.         datos= backup();
40.         num= datos.size();
41.         for(int i=0;i<num;i++){
42.         temp= datos[i];
43.         query.prepare("INSERT INTO ferti VALUES
   (?,?,?,?,?,?,?)");
44.         toParse=(temp.nombre.c_str());
45.         query.addBindValue(QVariant(toParse));
46.
  query.addBindValue(QVariant(temp.nitrato));
47.
  query.addBindValue(QVariant(temp.fosforo));
48.
  query.addBindValue(QVariant(temp.potasio));
49.
  query.addBindValue(QVariant(temp.cost_optimo));
50.
  query.addBindValue(QVariant(temp.cost_medio));
```

```
51.
  query.addBindValue(QVariant(temp.cost_pesimo));
52.        query.exec();
53.         }
54.       //  padre->L=datos;
55.         datos.clear();
56.     }
57.     vector<ferti> tableview::backup()
58.     {
59.         vector<ferti> L;
60.         ferti temp;
61.         int num;
62.         num=ui->tableWidget->rowCount();
63.         for(int i=0;i<num;i++)
64.         {
65.             temp.nombre=ui->tableWidget-
  >item(i,0)->text().toStdString();
66.             temp.nitrato=ui->tableWidget-
  >item(i,1)->text().toDouble();
67.             temp.fosforo=ui->tableWidget-
  >item(i,2)->text().toDouble();
68.             temp.potasio=ui->tableWidget-
  >item(i,3)->text().toDouble();
69.             temp.cost_optimo=ui->tableWidget-
  >item(i,4)->text().toDouble();
70.             temp.cost_medio=ui->tableWidget-
  >item(i,5)->text().toDouble();
71.             temp.cost_pesimo=ui->tableWidget-
  >item(i,6)->text().toDouble();
72.             if(temp.nombre.size())
73.             L.push_back(temp);
74.         }
75.         return L;
76.     }
77.     bool tableview::load_data(QString tab)
78.     {
79.             QSqlQuery query;
80.             query.exec("SELECT * FROM "+tab);
81.             ui->tableWidget->clear(); //limpiar
  vista
82.             int row=ui->tableWidget->rowCount();
83.             QString S;
84.
85.
86.                 cout<<row<<endl;
87.           while(query.next())
```

```
88.             {  ui->tableWidget->insertRow(row);
89.                S=query.value(0).toString();
90.                ui->tableWidget-
  >setItem(row,0, new QTableWidgetItem(S));
91.                S=query.value(1).toString();
92.                ui->tableWidget-
  >setItem(row,1, new QTableWidgetItem(S));
93.                S=query.value(2).toString();
94.                ui->tableWidget-
  >setItem(row,2, new QTableWidgetItem(S));
95.                S=query.value(3).toString();
96.                ui->tableWidget-
  >setItem(row,3, new QTableWidgetItem(S));
97.                S=query.value(4).toString();
98.                ui->tableWidget-
  >setItem(row,4, new QTableWidgetItem(S));
99.                S=query.value(5).toString();
100.               ui->tableWidget-
  >setItem(row,5, new QTableWidgetItem(S));
101.               S=query.value(6).toString();
102.               ui->tableWidget-
  >setItem(row,6, new QTableWidgetItem(S));
103.               row=ui->tableWidget->rowCount();
104.          }
105.       return true;
106.   }
```

10. Clase Mainwindow

Header:

```
1. #ifndef MAINWINDOW_H
2. #define MAINWINDOW_H
3. #include <QMainWindow>
4. #include <vector>
5. #include <string>
6. #include "database.h"
7. #include "datos.h"
8. #include "equations.h"
9. #include "simplex.h"
10.    #include "suelo.h"
11.    #include "tableview.h"
12.    #include "lai.h"
13.    #include "leberling.h"
14.    #include "problema.h"
```

```
15.     #include "equations.h"
16.     #include "xml.h"
17.     #include "document.h"
18.     #include "report.h"
19.     #include <QPrinter>
20.     namespace Ui {
21.         class MainWindow;
22.     }
23.     class MainWindow : public QMainWindow
24.     {
25.         Q_OBJECT
26.     public:
27.         explicit MainWindow(QWidget *parent = 0);
28.         ~MainWindow();
29.      /*variables relacionadas al problema*/
30.         Lai* lai;
31.         Leberling *leb;
32.         ferti fertilizante;
33.         cultivo cult;
34.         Database* BD;
35.         datos *data_suelo;
36.         suelo * Suelo;
37.         tableview *filas_BD;
38.         problema * pr;
39.         equations * eq;
40.         tableview * tab;
41.         Xml * archivo;
42.         document* docs;
43.         report * rep;
44.         vector<string> ferti_list;
45.         vector<string> cult_list;
46.         vector<ferti> ferti_escogidos;
47.     private:
48.         Ui::MainWindow *ui;
49.         void init(); //inicializa todo
50.         void clear();//limpiar todo
51.
     void load_ferti_win(vector<string>); //metodo
   para cargar fertilizantes en la pantalla
52.         vector<string> get_seleccionados();
53.
   vector<ferti> seleccion_ferti(vector<string>); //l
   ista de fertilizantes con su data
54.
     void load_cult_win(vector<string>); //metodo par
   cargar cultivos en la pantalla
```

```
55.         cultivo seleccion_cultivo();//cultivo
  seleccionado
56.         void extra_suelo(); //calcula valores
   del suelo usados en el problema
57.         void carga_panel(); //carga la data del
  GUI
58.         void genera_reporte();//genera reporte
59.     private slots:
60.
    void on_actionModificar_cultivo_triggered();
61.         void on_actionVista_Previa_triggered();
62.         void on_actionSobre_Qt4_triggered();
63.         void on_actionSobre_FertiDif_triggered();
64.         void on_actionGuia_triggered();
65.
    void on_actionGenerar_triggered(); //genera y
  escribre el reporte en un archivo
66.         void on_actionCerrar_triggered();
67.         void on_actionImprimir_triggered();
68.         void on_actionGuardar_triggered();
69.         void on_actionAbrir_triggered();
70.
    void on_actionModificar_triggered(); //acion de
  modificar en BD
71.
    void on_actionExec_triggered(); //ejecutar el
  prblema
72.
    void on_actionLeberling_triggered(); //resolver
  por Leberling
73.
    void on_actionLai_Hwang_triggered(); //resolver
  por Lai
74.         void on_actionLoad_triggered(); //cargar
  data para el problema
75.         void on_ida_clicked(); //accion de
  seleccion de fertilizante izq >> der
76.         void on_regreso_clicked();//accion de
  seleccion de fertilizante  der << izq
77.         void previewPrinter(QPrinter *printer);
78.     };
79.     #endif // MAINWINDOW_H
```

Code:

```
1. #include "mainwindow.h"
```

```
2. #include "ui_mainwindow.h"
3. #include <algorithm>
4. #include <iostream>
5. #include "equation_leb.h"
6. #include <QFileDialog>
7. #include <QPrinter>
8. #include <QPrintDialog>
9. #include <QMessageBox>
10.     #include <QPrintPreviewDialog>
11.     using namespace std;
12.     MainWindow::MainWindow(QWidget *parent) :
13.         QMainWindow(parent),
14.         ui(new Ui::MainWindow)
15.     {
16.         ui->setupUi(this);
17.         lai = new Lai();
18.         leb = new Leberling();
19.         BD = new Database();
20.         data_suelo = new datos();
21.         Suelo = new suelo();
22.         filas_BD = new tableview();
23.         archivo = new Xml();
24.         docs = new document();
25.         rep = new report();
26.         //1. abrir las bases de datos de
  fertilizantes y cultivos
27.           //1.1 fertilizantes
28.           BD->load_fertilizante("ferti.db");
29.         init();
30.         connect(ui-
  >actionSobre_Qt4,SIGNAL(triggered()),qApp,SLOT(abo
  utQt()));
31.         connect(ui-
  >actionCerrar,SIGNAL(triggered()),qApp,SLOT(closeA
  llWindows()));
32.     }
33.     MainWindow::~MainWindow()
34.     {
35.         delete lai ;
36.         delete  leb ;
37.         delete BD ;
38.         delete  data_suelo ;
39.         delete  Suelo;
40.         delete  filas_BD;
41.         delete ui;
42.     }
```

```
43.     void MainWindow::init()
44.     {
45.         //desactivamos botenes
46.         ui->actionLai_Hwang->setEnabled(false);
47.         ui->actionLeberling->setEnabled(false);
48.         ui->actionGenerar->setEnabled(false);
49.         ui->actionImprimir->setEnabled(false);
50.         ui->actionVista_Previa-
  >setEnabled(false);
51.         //0. limpiamos los vectores
52.        ferti_list.clear();
53.         cult_list.clear();
54.         //1.1.1 extraemos la lista de nombres
55.         ferti_list = BD->get_fert("ferti");
56.         this-
  >load_ferti_win(ferti_list); //cargamos a la
  pantalla los fertilizantes
57.         //1.2  cultivos
58.      //   BD->load_cultivo("cultivo.sqlite");
59.         cult_list = BD->get_cultivo("cultivo");
60.         this-
  >load_cult_win(cult_list); //cargamos a la
  pantalla los cultivos
61.     }
62.     void MainWindow::load_cult_win(vector<string>
    L)
63.     {
64.         //limpiamos lista de cultivos
65.         sort(L.begin(), L.end());
66.        for(size_t i =0; i<L.size(); i++)
67.         ui->cultivo_box->addItem(L[i].c_str());
68.     }
69.     void MainWindow::load_ferti_win(vector<string
  > L)
70.     {
71.         //limpiamos los disponibles y
  seleccionados
72.         ui->disponible->clear();
73.         ui->seleccionado->clear();
74.         sort(L.begin(),L.end());
75.         for(size_t i=0; i<L.size(); i++)
76.           ui->disponible->addItem(L[i].c_str());
77.     }
78.     void MainWindow::on_regreso_clicked()
79.     {
80.         QListWidgetItem* select;
```

```
81.         select = ui->seleccionado->takeItem(ui-
  >seleccionado->currentRow());
82.         if(select!=NULL)
83.         {
84.             ui->disponible->addItem(select-
  >text());
85.            delete select;
86.
87.         }
88.     }
89.     void MainWindow::on_ida_clicked()
90.     {
91.         QListWidgetItem* select;
92.         select = ui->disponible->takeItem(ui-
  >disponible->currentRow()); //tomando el item
  actual
93.         if(select!=NULL)
94.         {
95.             ui->seleccionado->addItem(select-
  >text());
96.            delete select;
97.         }
98.     }
99.     /*
100.     extraemos los nombres de los fertilizantes
  seleccionados*/
101.    vector<string> MainWindow::get_seleccionados(
  )
102.    {
103.        vector<string> L;
104.        for( int i=0; i< ui->seleccionado-
  >count(); i++)
105.        {
106.            L.push_back(ui->seleccionado-
  >item(i)->text().toStdString());
107.        }
108.        return L;
109.    }
110.    vector<ferti> MainWindow::seleccion_ferti(vec
  tor<string> L)
111.    {
112.        vector<ferti> lista_ferti;
113.        ferti F; //variable de fertilizantes
114.        for(size_t i=0; i < L.size(); i++)
115.        {
```

```
116.            F = BD-
  >consult_fert("ferti",QString::fromStdString( L[i]
  ));
117.            lista_ferti.push_back(F);
118.        }
119.        return lista_ferti;
120.    }
121.    void MainWindow::extra_suelo()
122.    {
123.        double temp[3];
124.        // o: nitrogeno 1: fosforo 2: potasio
125.       temp[0]= ui->ex_nitro->text().toDouble();
126.       temp[1]= ui->ex_fosfo->text().toDouble();
127.       temp[2]= ui->ex_pot->text().toDouble();
128.       /*asginamos lo extraido del suelo*/
129.       Suelo->set_ext_suelo(temp);
130.       temp[0]= ui->ni_f1->text().toDouble();
131.       temp[1]= ui->ni_f2->text().toDouble();
132.       temp[2]= ui->ni_f3->text().toDouble();
133.       /*asignamos coeficientes para nitrogenso
  en f1,f2 y f3*/
134.       Suelo->set_coef_us_nit(temp);
135.       temp[0]= ui->fo_f1->text().toDouble();
136.       temp[1]= ui->fo_f2->text().toDouble();
137.       temp[2]= ui->fo_f3->text().toDouble();
138.       /*asignamos coeficientes para fosforo en
  f1,f2 y f3*/
139.       Suelo->set_coef_us_fos(temp);
140.       temp[0]= ui->po_f1->text().toDouble();
141.       temp[1]= ui->po_f2->text().toDouble();
142.       temp[2]= ui->po_f3->text().toDouble();
143.       /*asignamos coeficientes para fosforo en
  f1,f2 y f3*/
144.       Suelo->set_coef_us_pot(temp);
145.       double anali_temp[5];
146.      anali_temp[0] = ui->an_ph-
  >text().toDouble();
147.      anali_temp[1] = ui->an_mo-
  >text().toDouble();
148.      anali_temp[2] = ui->an_n-
  >text().toDouble();
149.      anali_temp[3] = ui->an_P_Disp-
  >text().toDouble();
150.      anali_temp[4] = ui->an_k_disp-
  >text().toDouble();
151.      Suelo->set_anali_suelo(anali_temp);
```

```
152.      Suelo->calc_aport(); //calculamos aporte
153.      Suelo->calc_abona(); //calculamos abonar
154.    }
155.    void MainWindow::on_actionLoad_triggered()
156.    {
157.
158.        this->extra_suelo();
159.
  ferti_escogidos = seleccion_ferti( get_seleccionad
  os());//fertilizantes seleccionados
160.        if(ferti_escogidos.size()==0)
161.            return;
162.        //activar
163.        ui->actionLai_Hwang->setEnabled(true);
164.        ui->actionLeberling->setEnabled(true);
165.        data_suelo->set_fert(ferti_escogidos); //
  cargando lista de fertilizantes al objeto de la
  clase suelo
166.
167.        data_suelo->set_suelo(*Suelo); //set de
  Suelo
168.        data_suelo->genera();
169.        docs->clear();
170.        pr = new problema();
171.        docs->problema = pr-
  >set_data(data_suelo);
172.        pr->setVisible(true);
173.       // docs->clear();
174.        docs->add_fertilizante(ferti_escogidos);
175.        docs->cliente.cliente = ui->cli_text-
  >text();
176.        docs->cliente.direccion = ui->dir_text-
  >text();
177.        docs->cliente.RUC = ui->ruc->text();
178.        docs->cliente.tel = ui->tel->text();
179.        //extraccion suelo
180.        docs->ext_data.push_back(ui->ex_nitro-
  >text());
181.        docs->ext_data.push_back(ui->ex_fosfo-
  >text());
182.        docs->ext_data.push_back(ui->ex_pot-
  >text());
183.        //nitrogeno
184.        docs->coef_data[0].push_back(ui->ni_f1-
  >text());
```

```
185.        docs->coef_data[0].push_back(ui->ni_f2-
  >text());
186.        docs->coef_data[0].push_back(ui->ni_f3-
  >text());
187.        //fosforo
188.        docs->coef_data[1].push_back(ui->fo_f1-
  >text());
189.        docs->coef_data[1].push_back(ui->fo_f2-
  >text());
190.        docs->coef_data[1].push_back(ui->fo_f3-
  >text());
191.        //potasio
192.        docs->coef_data[2].push_back(ui->po_f1-
  >text());
193.        docs->coef_data[2].push_back(ui->po_f2-
  >text());
194.        docs->coef_data[2].push_back(ui->po_f3-
  >text());
195.        //cultivo
196.        docs->cultivo = ui->cultivo_box-
  >currentText();
197.        //analisis de suelo
198.        docs->anal_su_data.push_back(ui->an_ph-
  >text());
199.        docs->anal_su_data.push_back(ui->an_mo-
  >text());
200.        docs->anal_su_data.push_back(ui->an_n-
  >text());
201.        docs->anal_su_data.push_back(ui-
  >an_P_Disp->text());
202.        docs->anal_su_data.push_back(ui-
  >an_k_disp->text());
203.    }
204.    void MainWindow::on_actionLai_Hwang_triggered
  ()
205.    {
206.        //habilitamos resportes e impresion
207.        ui->actionGenerar->setEnabled(true);
208.        ui->actionImprimir->setEnabled(true);
209.        ui->actionVista_Previa->setEnabled(true);
210.            vector<vector<double> > M;
211.            vector<double> B;
212.            vector<double> X, min;
213.            vector<char> signos;
214.            vector<double> Z1_min,Z1_max;
215.            vector<double> Z2_min, Z2_max;
```

```
216.            vector<double> Z3_min, Z3_max;
217.            M = data_suelo->get_restricciones();
218.            signos = data_suelo->get_signos();
219.            B = data_suelo->get_B();
220.            //Z1
221.            Z1_min = data_suelo->min_z1();
222.            Z1_max = data_suelo->max_z1();
223.            cout<<"OBjetivos Z1_min \n";
224.            for(size_t i=0;i<Z1_min.size();i++)
225.                cout<<Z1_min[i]<<"  ";
226.            cout<<endl;
227.            cout<<"OBjetivos Z1_max \n";
228.            for(size_t i=0;i<Z1_max.size();i++)
229.                cout<<Z1_max[i]<<"  ";
230.            cout<<endl;
231.            //Z2
232.            Z2_min = data_suelo->min_z2();
233.            Z2_max = data_suelo->max_z2();
234.            cout<<"OBjetivos Z2_min \n";
235.            for(size_t i=0;i<Z2_min.size();i++)
236.                cout<<Z2_min[i]<<"  ";
237.            cout<<endl;
238.            cout<<"OBjetivos Z2_max \n";
239.            for(size_t i=0;i<Z2_max.size();i++)
240.                cout<<Z2_max[i]<<"  ";
241.            cout<<endl;
242.            //Z3
243.            Z3_min = data_suelo->min_z3();
244.            Z3_max = data_suelo->max_z3();
245.            cout<<"OBjetivos Z3_min \n";
246.            for(size_t i=0;i<Z3_min.size();i++)
247.                cout<<Z3_min[i]<<"  ";
248.            cout<<endl;
249.            cout<<"OBjetivos Z3_max \n";
250.            for(size_t i=0;i<Z3_max.size();i++)
251.                cout<<Z3_max[i]<<"  ";
252.            cout<<endl;
253.
254.
255.             {
256.                lai->set_constrains(M,B,signos);
257.
258.              X=lai-
  >calcular(Z1_min,Z1_max,Z2_min,Z2_max,Z3_min,Z3_ma
  x);
259.              if(!X.size())
```

```
260.                    return;
261.                min = lai-
  >min_resul(ferti_escogidos,X);
262.              }
263.       cout<<"\n\n minimos : ";
264.
265.           for(size_t k=0; k<min.size(); k++)
266.              cout<<min[k]<<"  ";
267.            cout<<endl;
268.        eq = new equations();
269.        eq->resultado = min;
270.        eq->min_z1 = Z1_min; eq->z1_nis = lai-
  >z1_nis;
271.        eq->max_z1 = Z1_max; eq->z1_pis = lai-
  >z1_pis;
272.        eq->min_z2 = Z2_min; eq->z2_nis = lai-
  >z2_nis;
273.        eq->max_z2 = Z2_max; eq->z2_pis = lai-
  >z2_pis;
274.        eq->min_z3 = Z3_min; eq->z3_nis = lai-
  >z3_nis;
275.        eq->max_z3 = Z3_max; eq->z3_pis = lai-
  >z3_pis;
276.
277.        docs->lai_z1 = eq->mostrar_z1_lai(lai-
  >X_z1_pis,lai->X_z1_nis);
278.        docs->lai_z2 = eq->mostrar_z2_lai(lai-
  >X_z2_pis,lai->X_z2_nis);
279.        docs->lai_z3 = eq->mostrar_z3_lai(lai-
  >X_z3_pis,lai->X_z3_nis);
280.        eq->R = lai->R;
281.        eq->B = lai->B;
282.        eq->signos = lai->signos;
283.        docs->lai_alpha = eq-
  >mostrar_alfa_lai(X);
284.        docs->agregar_valor_lai_leb(X,true);
285.        eq->show();
286.    }
287.    void MainWindow::on_actionLeberling_triggered
   ()
288.    {
289.        //habilitamos resportes e impresion
290.        ui->actionGenerar->setEnabled(true);
291.        ui->actionImprimir->setEnabled(true);
292.        ui->actionVista_Previa-
  >setEnabled(true);
```

```
293.        cout<<"\nejecutar Lai"<<endl;
294.            vector<vector<double> > M;
295.            vector<double> B;
296.            vector<double> X, min;
297.            vector<char> signos;
298.            vector<double> Z1_min,Z1_max;
299.            vector<double> Z2_min, Z2_max;
300.            vector<double> Z3_min, Z3_max;
301.            M = data_suelo->get_restricciones();
302.            signos = data_suelo->get_signos();
303.            B = data_suelo->get_B();
304.            //Z1
305.            Z1_max = data_suelo->max_z1();
306.
307.            cout<<"OBjetivos Z1_max \n";
308.            for(size_t i=0;i<Z1_max.size();i++)
309.                cout<<Z1_max[i]<<"  ";
310.            cout<<endl;
311.            //Z2
312.            Z2_min = data_suelo->min_z2();
313.            cout<<"OBjetivos Z2_min \n";
314.            for(size_t i=0;i<Z2_min.size();i++)
315.                cout<<Z2_min[i]<<"  ";
316.            cout<<endl;
317.            //Z3
318.            Z3_min = data_suelo->min_z3();
319.            cout<<"OBjetivos Z3_min \n";
320.            for(size_t i=0;i<Z3_min.size();i++)
321.                cout<<Z3_min[i]<<"  ";
322.            cout<<endl;
323.         {
324.                leb->set_constrains(M,B,signos);
325.                X=leb-
  >calcular(Z1_max,Z2_min,Z3_min);
326.                if(!X.size())
327.                    return;
328.                min=leb-
  >min_resul(ferti_escogidos,X);
329.            }
330.           cout<<"\n\n minimos : ";
331.            for(size_t k=0; k<min.size(); k++)
332.              cout<<min[k]<<"  ";
333.            cout<<endl;
334.        equation_leb* E= new equation_leb();
335.        E->z110 = leb->z110;
336.        cout<<"leberling salida"<<E->z110<<endl;
```

```
337.        E->z1m  = leb->z1m;
338.        E->z210 = leb->z210;
339.        E->z2m  = leb->z2m;
340.        E->z310 = leb->z310;
341.        E->z3m  = leb->z3m;
342.        //matriz
343.        E->R = leb->R;
344.        E->B = leb->B;
345.        E->signos = leb->signos;
346.        E->x10 = leb->x10;
347.        E->x20 = leb->x20;
348.        E->x30 = leb->x30;
349.        E->resultado = min;
350.        docs->leb_z0= E-
  >mostrar_todos(Z1_max,Z2_min,Z3_min);
351.        docs->leb_z1m = E->mostrar_calculos();
352.        docs->leb_alpha = E->mostrar_final(X);
353.        docs->agregar_valor_lai_leb(X,false);
354.        E->show();
355.    }
356.    void MainWindow::on_actionExec_triggered()
357.    {
358.    }
359.    void MainWindow::on_actionModificar_triggered
  ()
360.    {
361.        tab = new tableview();
362.        tab->load_data("ferti");
363.        tab->show();
364.    }
365.    void MainWindow::on_actionAbrir_triggered()
366.    {
367.
     QString filename = QFileDialog::getOpenFileName(
    this,tr("Open file dialog"),
368.
          QDir::currentPath(), "FertDif files
  (*.fd)");
369.       // QFile file(filename);
370.        if(!archivo->open_xml(filename))
371.            return;
372.          init();
373.        ui->cli_text->setText(archivo-
  >cliente.cliente);
374.        ui->dir_text->setText(archivo-
  >cliente.direccion);
```

```
375.        ui->ruc->setText(archivo->cliente.RUC);
376.        ui->tel->setText(archivo->cliente.tel);
377.        ui->ex_nitro->setText(archivo->e_sol[0]);
378.        ui->ex_fosfo->setText(archivo->e_sol[1]);
379.        ui->ex_pot->setText(archivo->e_sol[2]);
380.        //Ni
381.        ui->ni_f1->setText(archivo-
  >coef[0][0]);  ui->ni_f2->setText(archivo-
  >coef[0][1]);
382.        ui->ni_f3->setText(archivo->coef[0][2]);
383.        //F
384.        ui->fo_f1->setText(archivo-
  >coef[1][0]);  ui->fo_f2->setText(archivo-
  >coef[1][1]);
385.        ui->fo_f3->setText(archivo->coef[1][2]);
386.        //K
387.        ui->po_f1->setText(archivo-
  >coef[2][0]);  ui->po_f2->setText(archivo-
  >coef[2][1]);
388.        ui->po_f3->setText(archivo->coef[2][2]);
389.        ui->cultivo_box->setCurrentIndex(ui-
  >cultivo_box->findText(archivo->cultivo));
390.        int ind;
391.        for(int i=0; i<archivo-
  >list_fert.size(); i++)
392.        {
393.            ind=-1;
394.            for(int j=0; j< ui->disponible-
  >count(); j++)
395.            {
396.                if(ui->disponible->item(j)-
  >text()==archivo->list_fert[i])
397.                {
398.                    ind = j;
399.                    break;
400.                }
401.            }
402.            if(ind==-1)
403.                continue;
404.            ui->disponible->setCurrentRow(ind);
405.            this->on_ida_clicked();
406.        }
407.    }
408.    void MainWindow::on_actionGuardar_triggered()
409.    {
```

```
410.
    QString filename = QFileDialog::getSaveFileName(
  this,tr("Save file dialog"),
411.
         QDir::currentPath(), "FertDif files
  (*.fd)");
412.       dat_client cliente;
413.       vector<QString> e_sol; //extraccion de
  suelo
414.
  vector<vector<QString> > coef; //coeficiente con
  N,K, P
415.       QString cultivo; //cultivo
416.       vector<QString> anal; //analisis
417.       vector<string> list_fert; //lista de
  fertilizante
418.       //cliente
419.     cliente.cliente = ui->cli_text->text();
420.     cliente.direccion = ui->dir_text->text();
421.     cliente.RUC = ui->ruc->text();
422.     cliente.tel = ui->tel->text();
423.       //extraccion de suelo
424.     e_sol.push_back(ui->ex_nitro-
  >text()); //nitro
425.     e_sol.push_back(ui->ex_fosfo-
  >text());  //fosforo
426.     e_sol.push_back(ui->ex_pot-
  >text()); //potasio
427.     //coeficientes
428.     vector<QString> Ni;
429.     Ni.push_back(ui->ni_f1-
  >text()); Ni.push_back(ui->ni_f2-
  >text()); Ni.push_back(ui->ni_f3->text());
430.     vector<QString> F;
431.     F.push_back(ui->fo_f1-
  >text()); F.push_back(ui->fo_f2-
  >text()); F.push_back(ui->fo_f3->text());
432.     vector<QString> K;
433.     K.push_back(ui->po_f1-
  >text()); K.push_back(ui->po_f2-
  >text()); K.push_back(ui->po_f3->text());
434.     coef.push_back(Ni);
435.     coef.push_back(F);
436.     coef.push_back(K);
437.     cultivo = ui->cultivo_box->currentText();
438.     //ph mo n pd kd
```

```
439.      anal.push_back(ui->an_ph->text());
440.      anal.push_back(ui->an_mo->text());
441.      anal.push_back(ui->an_n->text());
442.      anal.push_back(ui->an_P_Disp->text());
443.      anal.push_back(ui->an_k_disp->text());
444.      //fertilizantes
445.      list_fert = this->get_seleccionados();
446.      archivo-
  >save_xml(filename,cliente,e_sol,coef,cultivo,anal
  ,list_fert);
447.    }
448.    void MainWindow::on_actionImprimir_triggered(
  )
449.    {
450.     QPrinter print;
451.    // print.setOrientation();
452.     QPrintDialog dlg(&print,this);
453.     if(dlg.exec()==QDialog::Accepted)
454.     {
455.         this->genera_reporte();
456.       rep->imprimir(&print);
457.     }
458.    }
459.
460.    void MainWindow::on_actionCerrar_triggered()
461.    {
462.    }
463.    void MainWindow::on_actionGenerar_triggered()
464.    {
465.            this->genera_reporte();
466.
    QString filename = QFileDialog::getSaveFileName(
  this,tr("Save file dialog"),
467.
             QDir::currentPath(), "Open Document
  Text (*.odt)");
468.          if(filename.size()>0)
469.           rep->write(filename);
470.          // delete rep;
471.    }
472.    void MainWindow::genera_reporte()
473.    {
474.        rep->limpiar();
475.     //archivo->
476.       rep->datos(docs->cliente.cliente,docs-
  >cliente.direccion,
```

```
477.                 docs->cliente.RUC,docs-
  >cliente.tel);
478.       //extraccion de suelo
479.       rep->insertar_tabla("Extracción de
  Suelo\n",docs-
  >ext_head,vector<vector<QString> > (1,docs-
  >ext_data));
480.       //Coeficientes del uso de nutriente
481.       rep->insertar_tabla("\nCoeficientes del
  uso de nutriente\n",docs->coef_head,docs-
  >coef_data);
482.       //Analisis de suelo
483.       rep->insertar_tabla("\nAnalisis de
  suelo\n",docs-
  >anal_su_head,vector<vector<QString> > (1,docs-
  >anal_su_data));
484.       //Seleccion de fertilizantes
485.       rep->insertar_tabla("\nSeleccion de
  fertilizantes\n",docs->fert_head,docs-
  >fertilizante);
486.       //problema
487.       rep->insertar_texto("\nProblema a
  resolver\n",docs->problema);
488.       //Lai
489.           //z1
490.           rep->insertar_texto("Lai","\n"+docs-
  >lai_z1+docs->lai_z2+docs->lai_z3+docs-
  >lai_alpha);
491.           rep->insertar_tabla("Resultados
  Lai\n",docs->res_head,docs->lai_res);
492.       //Leberling
493.           //z0
494.           rep-
  >insertar_texto("Leberling","\n"+docs-
  >leb_z0+docs->leb_z1m+docs->leb_alpha);
495.
496.           rep->insertar_tabla("Resultados
  Leberling\n",docs->res_head,docs->leb_res);
497.    }
498.    void MainWindow::on_actionGuia_triggered()
499.    {
500.    }
501.    void MainWindow::on_actionSobre_FertiDif_trig
  gered()
502.    {
```

```
503.        QMessageBox::about(this,tr("About
  FertiDif"),
504.                           tr("FertiDif 1.0\n"
505.                              "(c) 2011 Esmelin
  Niquin Alayo\n"
506.                              "Developed by Marks
  Calderon Niquin"));
507.    }
508.    void MainWindow::on_actionSobre_Qt4_triggered
  ()
509.    {
510.    //QMessageBox::aboutQt();
511.    }
512.    void MainWindow::on_actionVista_Previa_trigge
  red()
513.    {
514.    #ifndef QT_NO_PRINTER
515.
    QPrinter printer(QPrinter::HighResolution);
516.
    QPrintPreviewDialog preview(&printer, this);
517.
    connect(&preview, SIGNAL(paintRequested(QPrinter
   *)), SLOT(previewPrinter(QPrinter *)));
518.        preview.exec();
519.    #endif
520.    }
521.    void MainWindow::previewPrinter(QPrinter *pri
  nter)
522.    {
523.    #ifdef QT_NO_PRINTER
524.        Q_UNUSED(printer);
525.    #else
526.        this->genera_reporte();
527.      rep->imprimir(printer);
528.    #endif
529.    }
530.
531.    void MainWindow::on_actionModificar_cultivo_t
  riggered()
532.    {
533.     }
```

11. Clase XML

Header

```
1. #ifndef XML_H
2. #define XML_H
3. #include <QDomDocument>
4. #include <QFile>
5. #include "database.h"
6. #include "datos.h"
7. #include "problema.h"
8. #include "suelo.h"
9. #include <vector>
10.     #include <QString>
11.     #include <string>
12.     using std::vector;
13.     using std::string;
14.     class Xml
15.     {
16.     public:
17.         Xml();
18.      dat_client cliente;
19.      vector<QString> e_sol; //extraccion de suelo
20.      vector<vector<QString> > coef; //coeficiente
   con N,K, P
21.      QString cultivo; //cultivo
22.      vector<QString> anal; //analisis
23.      vector<QString> list_fert; //lista de
   fertilizante
24.       bool open_xml(QString);
25.       bool save_xml(QString path, dat_client
   cl, vector<QString> extr_suelo,
26.
   vector<vector<QString> > coeficientes, QString cul
   t,
27.
   vector<QString> analisis,vector<string> lista_fert
   ilizantes);
28.     private:
29.       QDomDocument *doc;
30.       QFile *file;
31.       QDomElement docElem;
32.       QDomNode node;
33.       QDomNodeList *list;
34.
35.       void clear(); //limpiar variables
36.     };
```

```
37.     #endif // XML_H
```

Code

```
1. #include "xml.h"
2. #include <iostream>
3. #include <QFile>
4. #include <QTextStream>
5. using namespace std;
6. Xml::Xml()
7. {
8.     doc = new QDomDocument();
9.      list = new QDomNodeList();
10.     }
11.     bool Xml::open_xml(QString path)
12.     {
13.         file=new QFile(path);
14.        if(!file->open(QIODevice::ReadOnly))
15.        {
16.            return false;
17.        }
18.        if(!doc->setContent(file))
19.        {
20.            file->close();
21.            return false;
22.        }
23.        file->close();
24.        docElem = doc->firstChildElement();
25.
    QDomNodeList nod = docElem.childNodes(); //
  extraemos todos los nodos del XML en total son 6
26.        if(nod.count()!=6)
27.            return false;
28.
     QDomNodeList data = nod.at(0).childNodes(); //ah
  ora extraemos del subarbol data
29.
  cliente.cliente = data.at(0).toElement().text();
30.
  cliente.direccion = data.at(1).toElement().text();
31.
  cliente.RUC = data.at(2).toElement().text();
32.
  cliente.tel = data.at(3).toElement().text();
33.         /*datos sobre suelo*/
```

```
34.
    QDomNodeList suelo = nod.at(1).childNodes();
35.         //nitrogeno
36.
    QDomNodeList Ni = suelo.at(0).childNodes();
37.         vector<QString> l_Ni;
38.         for(int i=0;i<Ni.count();i++)
39.
  l_Ni.push_back(Ni.at(i).toElement().text());
40.         //fosforo
41.
    QDomNodeList F = suelo.at(1).childNodes();
42.         vector<QString> l_F;
43.         for(int i=0;i<F.count();i++)
44.
  l_F.push_back(F.at(i).toElement().text());
45.         //potasio
46.
    QDomNodeList K = suelo.at(2).childNodes();
47.         vector<QString> l_K;
48.         for(int i=0;i<K.count();i++)
49.
  l_K.push_back(K.at(i).toElement().text());
50.         coef.push_back(l_Ni);
51.         coef.push_back(l_F);
52.         coef.push_back(l_K);
53.         //analisis
54.
    QDomNodeList analisis = nod.at(2).childNodes();
55.         for(int i=0;i<analisis.count();i++)
56.
  anal.push_back(analisis.at(i).toElement().text());

57.         //seleccion de fertilizantes
58.
    QDomNodeList seleccion = nod.at(3).childNodes();
59.         for(int i=0;i<seleccion.count();i++)
60.
   list_fert.push_back(seleccion.at(i).toElement().t
  ext());
61.         //cultivo
62.         cultivo = nod.at(4).toElement().text();
63.         //extraccion de suelo N,K,P
64.
    QDomNodeList ext_suelo = nod.at(5).childNodes();
```

```
65.         for(int i=0; i<ext_suelo.count();i++)
66.
  e_sol.push_back(ext_suelo.at(i).toElement().text()
  );
67.         return true;
68.     }
69.     void Xml::clear()
70.     {
71.         cliente.cliente.clear();
72.         cliente.direccion.clear();
73.         cliente.RUC.clear();
74.         cliente.tel.clear();
75.         e_sol.clear();
76.         coef.clear();
77.        cultivo.clear();
78.         anal.clear();
79.         list_fert.clear();
80.     }
81.     bool Xml::save_xml(QString path, dat_client
  cl, vector<QString> extr_suelo,
82.
    vector<vector<QString> > coeficientes, QString cu
  lt,
83.
    vector<QString> analisis, vector<string> lista_fe
  rtilizantes)
84.     {
85.         QDomDocument doc("myfile");
86.
     QDomElement root = doc.createElement("root");
87.         //insertamos el hijo
88.         doc.appendChild(root);
89.
    QDomElement datos = doc.createElement("datos");
90.          //cliente
91.
     QDomElement cliente = doc.createElement("cliente
  ");
92.
  cliente.appendChild(doc.createTextNode(cl.cliente)
  );
93.          //direccion
94.
    QDomElement dir = doc.createElement("direccion");
```

```
95.
    dir.appendChild(doc.createTextNode(cl.direccion))
   ;
96.          //ruc
97.
    QDomElement ruc = doc.createElement("ruc");
98.
   ruc.appendChild(doc.createTextNode(cl.RUC));
99.          //telefono
100.
    QDomElement tel = doc.createElement("telefono");
101.
    tel.appendChild(doc.createTextNode(cl.tel));
102.         datos.appendChild(cliente);
103.         datos.appendChild(dir);
104.         datos.appendChild(ruc);
105.         datos.appendChild(tel);
106.
    QDomElement suelo = doc.createElement("suelo");
107.
    QDomElement ni = doc.createElement("Ni");
108.         for(int i=0;i<3;i++)
109.         {
110.
    QDomElement f_ni = doc.createElement("f");
111.             f_ni.setAttribute("id",i);
112.
    f_ni.appendChild(doc.createTextNode(coeficientes.
   at(0)[i]));
113.             ni.appendChild(f_ni);
114.         }
115.         QDomElement f = doc.createElement("F");
116.         for(int i=0;i<3;i++)
117.         {
118.
    QDomElement f_f = doc.createElement("f");
119.             f_f.setAttribute("id",i);
120.
    f_f.appendChild(doc.createTextNode(coeficientes.a
   t(1)[i]));
121.             f.appendChild(f_f);
122.         }
123.         QDomElement k = doc.createElement("K");
124.         for(int i=0;i<3;i++)
125.         {
```

```
126.
    QDomElement f_k = doc.createElement("f");
127.             f_k.setAttribute("id",i);
128.
    f_k.appendChild(doc.createTextNode(coeficientes.a
  t(2)[i]));
129.             k.appendChild(f_k);
130.         }
131.         suelo.appendChild(ni);
132.         suelo.appendChild(f);
133.         suelo.appendChild(k);
134.
    QDomElement anal = doc.createElement("analisis");
135.         for(int i=0;i<5;i++)
136.         {
137.
    QDomElement a_l = doc.createElement("a");
138.             a_l.setAttribute("id",i);
139.
    a_l.appendChild(doc.createTextNode(analisis[i]));

140.             anal.appendChild(a_l);
141.         }
142.
    QDomElement seleccion = doc.createElement("selecc
  ion");
143.
    for(int i=0;i<lista_fertilizantes.size();i++)
144.         {
145.
    QDomElement sel = doc.createElement("fertilizante
  ");
146.             sel.setAttribute("id",i);
147.
    QString qs = lista_fertilizantes[i].c_str();
148.
    sel.appendChild(doc.createTextNode(qs));
149.             seleccion.appendChild(sel);
150.         }
151.
    QDomElement cultivo = doc.createElement("cultivo"
  );
152.
    cultivo.appendChild(doc.createTextNode(cult));
```

```
153.
    QDomElement ext_suelo = doc.createElement("ext_su
   elo");
154.         for(int i=0;i<3;i++)
155.         {
156.
    QDomElement ext = doc.createElement("f");
157.             ext.setAttribute("id",i);
158.
    ext.appendChild(doc.createTextNode( extr_suelo[i]
   ));
159.             ext_suelo.appendChild(ext);
160.         }
161.         root.appendChild(datos);
162.         root.appendChild(suelo);
163.         root.appendChild(anal);
164.         root.appendChild(seleccion);
165.         root.appendChild(cultivo);
166.         root.appendChild(ext_suelo);
167.     QFile file(path);
168.     if(!file.open(QIODevice::WriteOnly))
169.         return false;
170.     QTextStream text(&file);
171.     doc.save(text,1);
172.     file.close();
173.        return true;
174.    }
```

12. Clase report

Header

```
1. #ifndef REPORT_H
2. #define REPORT_H
3. #include <QString>
4. #include <QTextDocument>
5. #include <QTextCursor>
6. #include <QTextCharFormat>
7. #include <QString>
8. #include <vector>
9. #include <QPrinter>
10.     using std::vector;
11.     class report
12.     {
13.     public:
14.         report();
```

```
15.         void write(const QString &fileName);
16.
     void datos(QString nombre,QString dir, QString r
  uc,QString tel);
17.
     void insertar_tabla(QString titulo,vector<QStrin
  g> head,vector<vector<QString> > data);
18.
19.
     void insertar_texto(QString titulo,QString texto
  );
20.         void limpiar();
21.          void imprimir(QPrinter *print);
22.     private:
23.         QTextDocument * const m_document;
24.         QTextCursor m_cursor;
25.         QTextCharFormat* plainFormat;
26.     };
27.     #endif // REPORT_H
```

Code

```
1. #include "report.h"
2. #include <QTextDocumentWriter>
3. report::report()
4.     : m_document(new QTextDocument()),
5.     m_cursor(m_document)
6. {
7.     m_cursor.movePosition(QTextCursor::Start);
8.
    plainFormat = new QTextCharFormat(m_cursor.charFo
  rmat());
9.      QTextCharFormat headingFormat = *plainFormat;
10.
  headingFormat.setFontWeight(QFont::Bold);
11.             headingFormat.setFontPointSize(16);
12.
  m_cursor.insertText(QString("\t\t\tReporte de
  Fertilización"),headingFormat);
13.         m_cursor.insertBlock();
14.      }
15.     void report::write(const QString &fileName)
16.     {
17.         QTextDocumentWriter writer(fileName);
18.         writer.write(m_document);
19.     }
```

```
20.    void report::datos(QString nombre, QString di
  r, QString ruc, QString tel)
21.    {
22.        QTextCharFormat subFormat = *plainFormat;
23.        subFormat.setFontWeight(QFont::Bold);
24.        subFormat.setFontPointSize(14);
25.         m_cursor.insertText(QString("I.
  Datos\n"),subFormat); m_cursor.insertBlock();
26.
  m_cursor.insertText("Cliente:\t"+nombre,*plainForm
  at); m_cursor.insertBlock();
27.
  m_cursor.insertText("Direccion:\t"+dir,*plainForma
  t); m_cursor.insertBlock();
28.
  m_cursor.insertText("R.U.C:\t"+ruc,*plainFormat);
  m_cursor.insertBlock();
29.
  m_cursor.insertText("Telefono:\t"+tel,*plainFormat
  ); m_cursor.insertBlock();
30.    }
31.    void report::insertar_tabla(QString titulo, v
  ector<QString> head, vector<vector<QString> > data
  )
32.    {
33.        QTextCharFormat subFormat = *plainFormat;
34.        subFormat.setFontWeight(QFont::Bold);
35.        subFormat.setFontPointSize(14);
36.        QTextCharFormat negrita = *plainFormat;
37.        negrita.setFontWeight(QFont::Bold);
38.        m_cursor.movePosition(QTextCursor::End);
39.        m_cursor.insertBlock();
40.
  m_cursor.insertText(titulo,subFormat); m_cursor.in
  sertBlock();
41.        QTextTableFormat tableFormat;
42.        tableFormat.setCellPadding(5);
43.        tableFormat.setHeaderRowCount(1);
44.
  tableFormat.setBorderStyle(QTextFrameFormat::Borde
  rStyle_Solid);
45.
  tableFormat.setWidth(QTextLength(QTextLength::Perc
  entageLength, 50));
46.        tableFormat.setBorder(0.5);
```

```
47.
  m_cursor.insertTable(data.size()+1 , head.size());

48.         //header
49.         for(unsigned int i=0; i<head.size();i++)
50.         {
51.
  m_cursor.insertText(head[i],subFormat);
52.
  m_cursor.movePosition(QTextCursor::NextCell);
53.         }
54.         //data
55.         for(unsigned int i=0; i< data.size();i++)
56.         {
57.
    for(unsigned int j=0; j< data[i].size(); j++)
58.             {
59.
  m_cursor.insertText(data[i][j],*plainFormat);
60.
  m_cursor.movePosition(QTextCursor::NextCell);
61.             }
62.         }
63.         m_cursor.insertBlock();
64.     }
65.     void report::insertar_texto(QString titulo, Q
  String texto)
66.     {
67.         QTextCharFormat subFormat = *plainFormat;
68.         subFormat.setFontWeight(QFont::Bold);
69.         subFormat.setFontPointSize(14);
70.         QTextCharFormat negrita = *plainFormat;
71.         negrita.setFontWeight(QFont::Bold);
72.         m_cursor.movePosition(QTextCursor::End);
73.         m_cursor.insertBlock();
74.         m_cursor.insertText(titulo,subFormat);
75.         m_cursor.insertBlock();
76.         m_cursor.setVerticalMovementX(4);
77.         m_cursor.insertText(texto,*plainFormat);
78.         m_cursor.insertBlock();
79.     }
80.     void report::imprimir(QPrinter *print)
81.     {
82.         m_document->print(print);
83.     }
84.     void report::limpiar()
```

```
85.     {
86.         m_document->clear();
87.
  m_cursor.movePosition(QTextCursor::Start);
88.
    plainFormat = new QTextCharFormat(m_cursor.charFo
  rmat());
89.
     QTextCharFormat headingFormat = *plainFormat;
90.
  headingFormat.setFontWeight(QFont::Bold);
91.             headingFormat.setFontPointSize(16);
92.         m_cursor.insertText(QString("\t\tReporte
  de Fertilización\n"),headingFormat);
93.         m_cursor.insertBlock();
94.     }
```

13. Clase document

Header

```
1. #ifndef DOCUMENT_H
2. #define DOCUMENT_H
3. #include <vector>
4. #include <QString>
5. #include "database.h"
6. #include "suelo.h"
7. using std::vector;
8. class document
9. {
10.     public:
11.         document();
12.         vector<QString> fert_head;
13.         vector<vector<QString> > fertilizante;
14.         QString problema;
15.         QString lai_z1;
16.         QString lai_z2;
17.         QString lai_z3;
18.         QString lai_alpha;
19.         QString leb_z0;
20.         QString leb_z1m;
21.         QString leb_alpha;
22.         //data de GUI vectorizada
23.         dat_client cliente;
24.         //extraccion suelo
```

```
25.         vector<QString> ext_head;
26.         vector<QString> ext_data;
27.         //coeficiente nutriente
28.         vector<QString> coef_head;
29.         vector<vector<QString> > coef_data;
30.         //cultivo
31.         QString cultivo;
32.         //Analisis de suelo
33.         vector<QString> anal_su_head;
34.         vector<QString> anal_su_data;
35.         //resultados para lai y leberling
36.         vector<QString> res_head;
37.         vector<vector<QString> > lai_res;
38.         vector<vector<QString> > leb_res;
39.
    void agregar_valor_lai_leb(vector<double> L, boo
  l flag);
40.         void clear();
41.         void add_fertilizante(vector< ferti> L);
42.     };
43.     #endif // DOCUMENT_H
```

Code

```
1. #include "document.h"
2. #include <vector>
3. document::document()
4. {
5.   vector<QString> temp;
6.   //data para fertilizante
7.   fert_head.push_back("Fertilizante");
8.   fert_head.push_back("%N");
9.   fert_head.push_back("%P");
10.       fert_head.push_back("%K");
11.       fert_head.push_back("Costo Optimo");
12.       fert_head.push_back("Costo Medio");
13.       fert_head.push_back("Costo Pesimo");
14.
  ext_head.push_back("Nitrogeno");  ext_head.push_ba
  ck("Fosforo");  ext_head.push_back("Potasio");
15.       //coeficiente nutriente
16.
  coef_head.push_back(""); coef_head.push_back("F1")
  ; coef_head.push_back("F2"); coef_head.push_back("
  F3");
```

```
17.
  coef_data.push_back(vector<QString> (1,"Nitrogeno"
  ));
18.
  coef_data.push_back(vector<QString> (1,"Fosforo"))
  ;
19.
  coef_data.push_back(vector<QString> (1,"Potasio"))
  ;
20.       //Analisis de suelo
21.       anal_su_head.push_back("P.H");
22.       anal_su_head.push_back("%M.O.");
23.       anal_su_head.push_back("%N");
24.       anal_su_head.push_back("P Disp");
25.       anal_su_head.push_back("K Disp");
26.       //resultados
27.       res_head.push_back("Variable");
28.       res_head.push_back("Nombre");
29.       res_head.push_back("Valor");
30.     }
31.     void document::clear()
32.     {
33.         fertilizante.clear();
34.         problema.clear();
35.         lai_z1.clear();
36.         lai_z2.clear();
37.         lai_z3.clear();
38.         lai_alpha.clear();
39.         leb_z0.clear();
40.         leb_z1m.clear();
41.         leb_alpha.clear();
42.         ext_data.clear();
43.         for(size_t i=0;i<3;i++)
44.
  coef_data[i].erase(coef_data[i].begin()+1,coef_dat
  a[i].end());
45.         anal_su_data.clear();
46.         lai_res.clear();
47.         leb_res.clear();
48.     }
49.     void document::add_fertilizante(vector<ferti>
   L)
50.     {
51.         QString temp;
52.         size_t j;
53.             for(j=0; j<L.size(); j++)
```

```
54.             {
55.
56.                 vector<QString> fert;
57.
   fert.push_back(QString(L[j].nombre.c_str()));
58.
   fert.push_back(temp.setNum(L[j].nitrato));
59.
   fert.push_back(temp.setNum(L[j].fosforo));
60.
   fert.push_back(temp.setNum(L[j].potasio));
61.
   fert.push_back(temp.setNum(L[j].cost_optimo));
62.
   fert.push_back(temp.setNum(L[j].cost_medio));
63.
   fert.push_back(temp.setNum(L[j].cost_pesimo));
64.                  fertilizante.push_back(fert);
65.                vector<QString> row;
66.
    row.push_back("X"+temp.setNum(j+1)); row.push_bac
   k(QString(L[j].nombre.c_str())); row.push_back("0"
   );
67.                this->lai_res.push_back(row);
68.                this->leb_res.push_back(row);
69.            }
70.             //agregamos alpha
71.             vector<QString> row;
72.
   row.push_back("X"+temp.setNum(j+1)); row.push_back
   ("Alpha"); row.push_back("0");
73.             this->lai_res.push_back(row);
74.             this->leb_res.push_back(row);
75.     }
76.     void document::agregar_valor_lai_leb(vector<d
   ouble> L, bool flag)
77.     {
78.         QString temp;
79.         if(flag){
80.             for(size_t i=0; i<L.size();i++)
81.             {
82.                 lai_res[i][2]=temp.setNum(L[i]);
83.             }
84.         }
85.         else
86.         {
```

```
87.             for(size_t i=0; i<L.size();i++)
88.             {
89.                 leb_res[i][2]=temp.setNum(L[i]);
90.             }
91.         }
92.     }
```

ANEXO 2

TABLA 6: ANÁLISIS DE SUELOS										
Lote	Textura			Clase textural	Ph	% MO	% N	P_2O_5 ppm	K_2O ppm	CIC meq/100gr
	% Arena	% Limo	% Arcilla							
L1	28	22	50	Arcilloso	7.99	1.407	0.070	33	98	9.60
L2	35	25	40	Arcilloso	7.89	1.675	0.084	41	105	6.54
L3	54	26	20	Franco arenoso	7.21	1.675	0.084	43	115	8.91
L4	51	24	25	Franco	7.64	1.541	0.077	32	270	9.99
L5	48	28	24	Franco	6.26	2.814	0.141	6	110	6.75
L6	72	18	10	Franco arenoso	6.99	1.005	0.05	22	260	7.30
L7	64	22	14	Franco arenoso	7.26	0.402	0.02	25	100	7.61
L8	63	25	12	Franco arenoso	5.89	1.930	0.096	43	98	6.78
L9	54	21	25	Franco arcilloso arenoso	6.61	1.608	0.080	29	100	7.82
L10	68	19	13	Franco arenoso	5.46	0.737	0.037	33	73	3.50

Fuente: Laboratorio de análisis de suelos de la Facultad de Ciencias Agrarias de la UNASAM

TABLA 7: COMPOSICIÓN PROMEDIO DE NUTRIENTES EN FERTILIZANTES														
FERTILIZANTE	% N	% P_2O_5	%K_2O_5	%Ca	%Mg	%S	%Cl	%Cu	%Mn	%Zn	%B	costo por kg		
Acido fosfórico		52										0.89	1.12	1.34
Amoniaco anhidro	82											0.90	1.13	1.36
Amoniaco líquido	16											0.88	1.10	1.32
Bayomix	11	22	11									0.72	1.11	1.08
Bicarbonato de potasio			45									0.91	1.14	1.37
Borax											11.6	0.70	0.80	0.90
Carbonato de potasio			60									0.86	1.08	1.30
Chalaza de algodón	6	2.6	1.4	0.2	0.4	0.3	0.06			0.02		0.72	0.90	1.08
Ciamida cálcica	21			38.5	0.06	0.3	0.2	0.02	0.04			0.72	0.90	1.08
Cloruro de amonio	28											0.82	1.03	1.23
Cloruro de potássio			60									1.04	1.30	1.56
Compost	0.5	0.5	0.5		0.3							0.88	1.10	1.32

Escorias Thomas		8										0.79	0.99	1.18
estiércol de caballo	12	0.3	0.4									0.80	1.00	1.20
estiércol de pollo (seco)	2	1.5	1									0.70	0.88	1.05
estiércol de vacuno	o,4	0.2	0.6		0.1							0.68	0.85	1.02
Fosfato de potássio		40	35									0.89	1.12	1.34
Fosfato di-amónico	18	46										1.00	1.15	1.30
Fosfato di-cálcico		53										0.90	1.13	1.35
Fosfato mono-amónico	11	48	0.2	1.1	0.3	2.2	0.1	0.02	0.03	0.03	0.02	0.92	1.15	1.38
Fosfato nítrico	14	10	0.1	8	0.1	0.2	0.1	0.02	0.2	0.02	0.03	0.88	1.11	1.33
Fertimix	11	22	11			12						0.81	1.02	1.22
Fosfato-sulfato de amonio	13	20	0.2	0.3	0.1	15.4	0.1	0.02	0.2	0.02	0.03	0.84	1.05	1.26
Guano de islas	10	10	2									0.80	1.00	1.20
Guano de isla rico	15	9	2									0.96	1.20	1.44
Guano de lombriz	2	0.5	0.5		0.2							0.70	0.88	1.05
Harina de hueso	2	22	0.2	22	0.4	0.1	0.2			0.02		0.70	0.97	1.15
Harina de sangre	15	1.5	0.8									0.68	0.86	1.03

Hidróxido de potasio			75									1.00	1.25	1.50
Humus de lombriz	0.6	0.6	0.6									0.81	1.02	1.22
Metafosfato de potasio		60	40									1.08	1.35	1.62
Musgo	0.3	0.3	0.3									0.81	1.02	1.22
Nitrato de amonio	34									0.01		0.95	1.32	1.60
Nitrato de amonio-cal	20.5			7.3	4.4	0.4	0.4					0.75	0.94	1.12
Nitrato de calcio	15.5			19.4	1.5	0.02	0.2					0.71	0.89	1.06
Nitrato de potasio	14		44	0.4	0.2	0.3	1.1				0.09	1.04	1.32	1.56
Nitrato de sodio	16		0.2	0.1	0.05	0.07	0.4	0.07			0.01	1.13	1.42	1.70
Nuriato de potasio			60			0.1	47				0.03	0.92	1.15	1.38
Oxido de cobre								75				0.70	0.98	1.20
Oxido de magnesio			1.1	56.1	0.2	0.5						0.70	0.88	1.00
Oxido de zinc										77.2		0.90	1.15	1.59
Purin de orina	0.3	0.06	0.45									0.92	1.15	1.38
Rainita			12									0.70	0.88	1.05
Roca fosfatada		27										0.89	1.12	1.34

Roca fosfórica		30										0.92	1.15	1.38
Salitre potássico	15		14									0.98	1.23	1.47
Salitre sódico	16											0-91	1.14	1.36
Sales potásicas			20									0.85	1.07	1.28
Sulfato de amônio	21			0.3		24	0.5	0.03		0.01		0.89	1.12	1.34
Sulfato de cobre						12.8		24.9		0.55		1	1.10	1.30
Sulfato de manganeso				6.6	1.9	14.5		0.05	25.1	0.08	0.3	0.98	1.04	1.45
Sulfato de potasio			50	0.5	0.7	17	2.1					1.07	1.34	1.60
Sulfato de potasio y magnesio			22		11.2	22.7	1.5					1.15	1.42	1.98
sulfato de zinc					0.06	13.6		0.02		27.8		0.75	0.97	1.35
Sulfato-nitrato de amônio	26					15.1						0.71	0.89	1.06
Superfosfato amoniacal	3	13		17.2		12					0.02	0.88	1.11	1.33
Superfosfato concentrado		42	0.4	13.6	0.3	1.4		0.01	0.01		0.01	0.90	1.16	1.55
Superfosfato simple		18	0.2	20.4	0.2	11.9	0.3			0.01		0.91	1.14	1.36
Superfosfato triple de cálcio		46										0.81	1.02	1.22
Superguano	20	20	15			5						1	1.20	1.50

Supernitro	33											1.18	1.48	1.77
Sulpomag			22		18	22						1	1.26	1.51
Tallos de tabaco	2	0.7	6	3.6	0.4	0.4	1.2	0.01	0.03		0.02	0.55	0.68	0.80
Urea	42			1.5	0.7	0.02	0.2			0.02		1.24	1.55	1.86
Urea-azufre	40					10						1.04	1.30	1.56
Urea-formaldehido	36											0.98	1.25	1.47
Yeso			0.5	22.5	0.4	16.8	0.3					0.15	0.25	0.50

Fuente: dirección General de Información Agraria – MINAG.

Notas del curso de Fertilidad de Suelos – Facultad . de Ciencias Agrarias - UNASAM

TABLA 8: CULTIVOS												
Cultivo	%M.O.	P.H.	Tipo de fertilización	N kg/Ha			P_2O_5 Kg/Ha			K_2O Kg/Ha		
				Min.	Media	Max.	Min.	Media	Max.	Min.	Media	Max.
Patata	< 2%	< 6.8	Alto			160			120			160
	2 - 4	6.8 - 7.8	Medio		120			80			120	
	> 4%	> 7.8	Bajo	80			40			80		
Arroz	< 2%	< 6.8	Alto			200			120			120
	2 - 4	6.8 - 7.8	Medio		140			80			80	
	> 4%	> 7.8	Bajo	80			40			40		280
Espárrago	< 2%	< 6.8	Alto			220			100		240	
	2 - 4	6.8 - 7.8	Medio		180			60		200		
	> 4%	> 7.8	Bajo	140			40					
Maíz duro	< 2%	< 6.8	Alto			240			160			180
	2 - 4	6.8 - 7.8	Medio		220			140			160	

	> 4%	> 7.8	Bajo	200			120			140		
Tomates	< 2%	< 6.8	Alto			180			160			170
	2 - 4	6.8 - 7.8	Medio		160			150			150	
	> 4%	> 7.8	Bajo	140			140			130		

Fuente: AS-F02 Fertilidad de suelos; Facultad de Ciencias Agrarias – UNASAM

TABLA 9: CULTIVOS				
CULTIVO	**NIVEL**	**N kg/Ha**	**P_2O_5 kg/Ha**	**K_2O kg/Ha**
ESPARRAGO	MÁXIMO	220	180	160
	MÍNIMO	180	140	120
ALCACHOFA	MÁXIMO	200	180	160
	MÍNIMO	160	140	120
TOMATE	MAXIMO	180	120	170
	MINIMO	150	100	140
MAIZ DURO	MAXIMO	240	160	180
	MINIMO	200	130	140
PATATAS	MAXIMO	100	50	220
	MINIMO	95	45	210

Fuente: AS-F02 Fertilidad de suelos; Facultad de Ciencias Agrarias – UNASAM