Modelos y algoritmos para asignación de usos de suelo para un desarrollo sustentable

Francisco Paroli
Lourdes Cairelli

Informe de Proyecto de Grado
Ingeniería en Computación

Instituto de Computación, Facultad de Ingeniería,
Universidad de la República,
2018

Supervisor: Antonio Mauttone

Tabla de Contenido

Capítulo 1 - Introducción

El uso que se da a los suelos (ya sea de propiedad pública o privada) en un contexto rural, generalmente está regulado por políticas a nivel estatal y determinado por la aptitud del suelo para cada uso. Generalmente existe una agencia encargada de formular estas políticas, las que deben contemplar consideraciones productivas y ambientales, entre otros aspectos.

Cuando el área de decisión se enmarca en una región con límites conocidos (por ejemplo la cuenca de un río), un problema típico es el encontrar un balance entre la maximización de la productividad de la tierra y la minimización del impacto ambiental. Las decisiones implican determinar el uso que se asigna a cada unidad espacial del área de estudio (por ejemplo una hectárea), que dependerá del potencial productivo que tenga, así como del impacto ambiental que pueda generar la actividad allí desarrollada. Un ejemplo de este problema es la determinación de usos de suelo en la cuenca de una laguna (que pueden ser por ejemplo el desarrollo de agricultura, ganadería o forestación) de forma de maximizar la aptitud para la producción y minimizar el impacto ambiental, en el sentido del vertido de agroquímicos a la laguna, producto del desarrollo de cada una de las actividades. El vertido de estos agroquímicos tiene como resultado un aumento del grado de eutrofización de la laguna.

La eutrofización es la acumulación de residuos orgánicos, nitrógeno y fósforo en un lago, laguna, embalse, etc., que causa la proliferación de ciertas algas que disminuyen drásticamente la calidad del agua (Echarri, 1998).

Para minimizar este impacto es necesario fijar un nivel de exportación de fósforo máximo permitido, denominado umbral, ya que el fósforo es uno de los agroquímicos más empleados en las actividades agropecuarias y tiene un alto impacto en la eutrofización. Además de contemplar estos objetivos, las decisiones deben tener en cuenta restricciones espaciales, de forma que las asignaciones de usos no resulten dispersas en el área geográfica y queden de forma compacta. Este proyecto plantea el modelado y resolución de este problema en un contexto de optimización combinatoria multiobjetivo, a través del modelado matemático y la resolución utilizando métodos aproximados (heurísticas). Las metodologías desarrolladas se probarán con un caso de estudio real relativo a la cuenca de la Laguna de Rocha (**Figura 1**). El proyecto está enmarcado en una línea de colaboración entre el Departamento de Investigación Operativa del Instituto de Computación y el Polo de Desarrollo Universitario de Ecología Funcional de Sistemas Acuáticos del Centro Universitario Regional del Este, sede Rocha.

Figura 1: Imagen aérea de la Laguna de Rocha (Google Maps, 2018)

1.1 Antecedentes

La línea de investigación en que se enmarca este Proyecto de Grado incluye diferentes instancias en donde se destacan las siguientes actividades específicas y publicaciones.

Actividades específicas:

- Proyecto CSIC I+D “Planificación ambiental: aplicación de herramientas interdisciplinarias para el desarrollo sustentable” , 2012.
- Lorena Rodríguez-Gallego. “Eutrofización de las lagunas costeras de Uruguay: impacto y optimización de los usos del suelo”. Tesis de doctorado en Ciencias Biológicas – Limnología. Universidad de la República - PEDECIBA, 2011.

- Carolina Cabrera. - “Optimización de usos del suelo para prevenir floraciones nocivas de fitoplancton en la Laguna de Rocha, Uruguay”. Tesis de Maestría en Geociencias. Universidad de la República - PEDECIBA, 2015
- Antonella Barletta. “Modelos de optimización y multiatributo para la asignación de usos del suelo en la cuenca de la Laguna de Rocha”. Tesina de Licenciatura en Ciencias Biológicas. Universidad de la República, 2017.

Publicaciones:

- Lorena Rodríguez-Gallego, Marcel Achkar, Daniel Conde. “Land suitability assessment in the catchment area of four Southwestern Atlantic coastal lagoons: multicriteria and optimization modeling”. Environmental Management. 50(1):140-152, 2012.
- Mariana Nin, Alvaro Soutullo, Lorena Rodríguez-Gallego, Enrico Di Minin. "Ecosystem services-based land planning for environmental impact avoidance", Ecosystem Services, 17:172-184, 2016.
- Antonella Barletta, Carolina Cabrera, Antonio Mauttone, Lorena Rodríguez-Gallego. "Multi-objective optimization for land use allocation in the basin of Laguna de Rocha". First International Conference on Agro Big Data and Decision Support Systems in Agriculture, 2017.
- Antonio Mauttone, Lourdes Cairelli, Francisco Paroli, Lorena Rodríguez-Gallego, Antonella Barletta, Carolina Cabrera. "Land use optimization considering compact allocations". Second International Conference on Agro Big Data and Decision Support Systems in Agriculture, 2018.
- Antonio Mauttone, Lourdes Cairelli, Francisco Paroli, Lorena Rodríguez-Gallego. "Metaheuristic approach to land use optimization balancing productivity and environmental protection”. XIX Latin-Iberoamerican Conference on Operations Research, 2018.

El antecedente más cercano a este proyecto es la tesina de Barletta (2017), en la que se busca maximizar la aptitud y minimizar el aporte de nutrientes a la cuenca. En dicha tesina se utilizaron distintas metodologías para la resolución exacta de un modelo de optimización lineal entero multiobjetivo. Primero se utilizó el método de suma ponderada y posteriormente el método de ε−restricción. Para la metodología de resolución de ε−restricción se realizaron pruebas con diferentes valores de umbral de fósforo. Se consideraron tres usos de suelo: agricultura, ganadería y forestación. Las soluciones encontradas para umbrales de fósforo más bajos no fueron compactas, obteniéndose una distribución de los usos del suelo por franjas entre el uso del suelo agricultura y el uso del suelo ganadería.

Motivado por dichos resultados, que evidencian una debilidad en el modelo de optimización propuesto y aplicado, en el presente proyecto se busca encontrar soluciones que no presenten una distribución de usos en franjas, sino que la distribución quede de forma compacta.

1.2 Objetivos del proyecto

Objetivos generales:

- Modelar y resolver el problema de optimización de usos de suelo utilizando conceptos de optimización combinatoria.
- Validar las metodologías desarrolladas con un caso real.

Objetivos específicos:

- Continuar la línea de investigación de actividades anteriores agregando el concepto de compacidad.
- Desarrollar algoritmos para resolver el problema.
- Realizar pruebas para evaluar el comportamiento de los algoritmos.

1.3 Estructura del informe

En el **capítulo 2** se introduce el marco conceptual, definiendo los conceptos básicos necesarios para comprender el proyecto. Luego, en el **capítulo 3** se detalla el modelo de optimización propuesto para la resolución del problema. En el **capítulo 4** se presentan los algoritmos propuestos para llegar a la solución. En el **capítulo 5** se reportan las pruebas realizadas con el objetivo de probar la robustez, eficiencia y eficacia de los algoritmos. Por último, en el **capítulo 6** se formulan las conclusiones y se delinea el trabajo a futuro.

Capítulo 2 - Marco conceptual

En este capítulo se presentan los conceptos y notaciones necesarios para comprender el resto del informe.

2.1 Conceptos básicos

Optimización combinatoria (Blum y Roli *, 2003*)

Un problema de optimización combinatoria $P = (S,f)$ puede ser definido por:

- Un conjunto de variables $X = \{x_1 , ..., x_n\}$;
- Dominios de las variables $D_1, ..., D_n$;
- Restricciones sobre las variables;
- Una función objetivo f a ser minimizada, donde $f : D_1 \times ... \times D_n \rightarrow \Re^+$;

El conjunto de todas las soluciones factibles es $S = \{s = \{(x_1, v_1), ...(x_n, v_n)\} \mid v_i \, \varepsilon \, D_i, \; s \; satisface \; todas \; las \; restricciones \}$. S es usualmente llamado el espacio de búsqueda, dado que cada elemento del conjunto puede verse como una solución candidata.

Para resolver un problema de optimización combinatoria se debe encontrar una solución $s^* \, \varepsilon \, S$ que minimice (sin perder generalidad) el valor de la función objetivo, es decir, s^* es el óptimo global.

Vecindario (Blum y Roli *, 2003*)

Dado un punto factible $s \, \varepsilon \, S$ en un problema particular, es de utilidad en muchas situaciones definir un conjunto $N(s)$ de puntos “cercanos” a s .

Formalmente, dado un problema de optimización con instancias (S, c) ,donde S es un conjunto dado (dominio de puntos factibles) y c es la función de costo, un vecindario es un mapeo $N : S \rightarrow 2^S$ definido para cada instancia, donde $N(s)$ es el conjunto de soluciones vecinas de s, que se obtienen a través de un cambio sistemático (movida).

Óptimo local (Blum y Roli *, 2003*)

Dado problema de optimización combinatoria $P = (S,f)$, se define óptimo local con respecto a un vecindario N, a $s^* \varepsilon S$ tal que $f(s^*) \leq f(s) \ \forall \ s \varepsilon N(s^*)$. Se llama s^* solución estrictamente local si $f(s^*) < f(s) \ \forall \ s \varepsilon N(s^*)$.

Búsqueda local (Blum y Roli *, 2003*)

Los algoritmos de búsqueda local comienzan con una solución inicial e intentan reemplazar iterativamente la solución actual por una mejor solución en un vecindario apropiadamente definido de la solución actual.

Heurística y Metaheurística (Reeves, 1993 ; Blum y Roli *, 2003*)

"Heurística" deriva del griego heuriskein, que significa "encontrar" o "descubrir". Es una técnica que busca soluciones buenas a un precio computacional razonablemente bajo, tratando de acercarse al óptimo global. La heurística es específica de un problema.

Las heurísticas se pueden clasificar del siguiente modo:

- Métodos Constructivos: generan una solución (desde una solución vacía) agregando componentes hasta completar la solución, son métodos rápidos, generalmente de baja calidad en los resultados.
- Métodos de Búsqueda local: comienzan desde una solución e iterativamente van reemplazando la solución actual con alguna solución parecida (según un vecindario específico) de mejor calidad.

Una metaheurística es "una heurística de alto nivel", con los siguientes objetivos:

- Encontrar soluciones factibles de buena calidad (valor cercano al óptimo global).
- Encontrar rápidamente soluciones factibles.
- Recorrer el espacio de soluciones sin quedar atrapados en una determinada región u óptimo local.

Existen varias clasificaciones de metaheurísticas. Las mismas pueden ser basadas o no en la naturaleza (algoritmos genéticos, búsqueda dispersa), aleatorias o determinísticas, basadas en individuo o en poblaciones, con memoria o sin memoria, entre otras.

Optimización Multiobjetivo (Ehrgott y Gandibleux, 2004)

Un problema de optimización multiobjetivo se puede definir de la siguiente manera:

$$min_{x\varepsilon X}(f_1(x), f_2(x), ...,f_p(x)) \quad (MOP)$$

donde $X \subset \Re^n$ es un conjunto de soluciones factibles y $f : \Re^n \rightarrow \Re^p$ el vector valor objetivo de la función. $Y = f(x) \subset \Re^p$ es la imagen en el conjunto factible dentro del espacio de las restricciones.

$x \varepsilon X$ es considerado eficiente, si no existe $x' \varepsilon X$ donde $f_k(x') \leq f_k(x)$ para todo $k = 1, ..., p$ y $f_j(x') < f_j(x)$ para algún j. En otras palabras no existe otra solución que sea mínima para todas las restricciones y estrictamente la mínima para una de las funciones.

GRASP (Feo y Resende, 1995; Pitsoulis y Resende, 2002)

GRASP (Greedy Randomized Adaptive Search Procedures) es una metaheurística que combina procedimientos constructivos y de búsqueda local. El aspecto clave de GRASP es que la construcción es aleatoria, por lo tanto, varias ejecuciones independientes generan diferentes soluciones.

Es un procedimiento iterativo en dos fases: una de construcción de la solución (heurística) **Figura 2** y otra de mejora F**igura 3**. La mejor solución encontrada se devuelve al finalizar el proceso de búsqueda (Resende y Ribeiro, 2010).

```
procedure GreedyRandomizedAlgorithm(Seed)
1.  S ← ∅;
2.  Initialize the candidate set: C ← E;
3.  Evaluate the incremental cost c(e) for all e ∈ C;
4.  while C ≠ ∅ do
5.      Build a list with the candidate elements having the smallest incremental costs;
6.      Select an element s from the restricted candidate list at random;
7.      Incorporate s into the solution: S ← S ∪ {s};
8.      Update the candidate set C;
9.      Reevaluate the incremental cost c(e) for all e ∈ C;
10. end;
11. return S;
end.
```

Figura 2 - Pseudocódigo de generador de soluciones iniciales

```
procedure GRASP(MaxIterations, Seed)
1.  Set f* ← ∞;
2.  for k = 1,..., MaxIterations do
3.      S ← GreedyRandomizedAlgorithm(Seed);
4.      if S is not feasible then
5.          S ← RepairSolution(S);
6.      end;
7.      S ← LocalSearch(S);
8.      if f(S) < f* then
9.          S* ← S;
10.         f* ← f(S);
11.     end;
12. end;
13. return S*;
end.
```

Figura 3 - Pseudocódigo GRASP

2.2 Modelado espacial para la optimización de usos de suelo

Los usos de suelo son los distintos tipos de asignación que se pueden realizar en cada unidad espacial. En el contexto del problema a resolver en este proyecto corresponden a A, G y F (agricultura, ganadería y forestación, respectivamente).

La aptitud refiere a la capacidad que tiene un lugar específico para desarrollar los distintos usos de suelo.

La exportación de fósforo es la cantidad de fósforo que exportan los distintos usos de suelo por unidad de superficie, dando como resultado el aumento de fósforo en la laguna.

La compacidad refiere a qué tan agrupados están los distintos usos de suelo. Por lo tanto hay tres tipos de medidas en el contexto del problema estudiado en este proyecto. La aptitud es una medida económica productiva, la exportación de fósforo es una medida ambiental y la compacidad es una medida espacial.

Cuanto mayor sea la cantidad de usos de suelo del mismo tipo que se encuentran en unidades espaciales contiguas, más grande va a ser la medida de compacidad. Esta medida es necesaria para poder dar soluciones en las que la asignación de usos de suelo sea por zonas y no distintas asignaciones para cada unidad espacial en una misma zona.

La definición informal de compacidad admite muchas definiciones formales. Para comprender mejor el concepto se agregan a continuación las imágenes de una asignación compacta y una que no lo es (**Figura 4**).

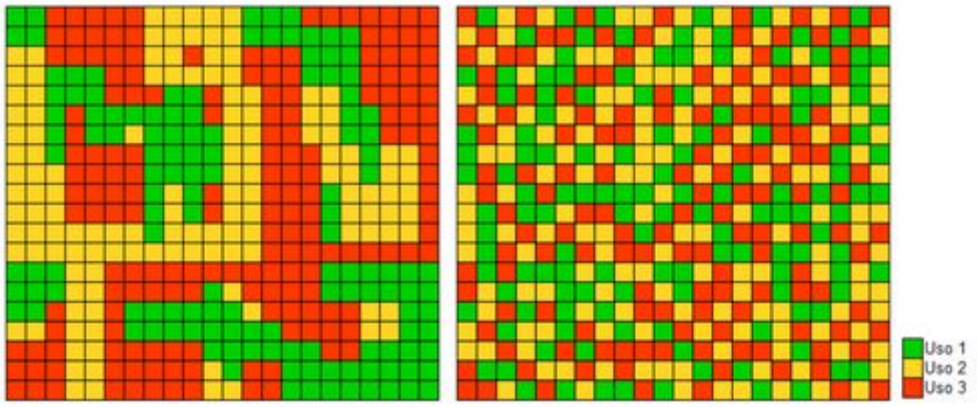


Figura 4: Asignación compacta vs asignación fragmentada

2.3 Representación espacial

Para abordar el problema de optimización de usos de suelo, se dividió el área total del mapa de la Laguna de Rocha en unidades espaciales (píxeles), donde cada pixel está numerado y corresponde a un área de media hectárea.

Los usos de suelo corresponden a las actividades que se pueden asignar a los píxeles, pero no todos son asignables, ya que algunos corresponden a áreas no aptas para asignación, es decir, territorio donde por diferentes motivos no se toman decisiones de cambios de uso de suelo.

Los distintos usos a asignar son **agricultura**, **ganadería** y **forestación**.
Cada uso se representa con un color para facilitar la visualización del mapa. A lo largo de este informe, el **verde** representa la agricultura, el **rojo** la ganadería, el **marrón** la forestación y el **gris** la tierra no apta para asignar un uso de suelo.

El procesamiento de la información relativa a la cuenca de la Laguna de Rocha fue realizado por parte de los equipos de Facultad de Ciencias y CURE Rocha. Dicho procesamiento incluye modelado en Sistema de Información Geográfica y estimaciones de aptitudes y exportaciones de fósforo. Los modelos y algoritmos desarrollados en este proyecto, toman datos generados a partir de dichos procesamientos.

Capítulo 3 - Modelo de optimización

En este capítulo se presenta el modelo de optimización propuesto para resolver el problema.

El mismo es de naturaleza multiobjetivo, donde se reconoce un problema de maximización de dos funciones f_1 y f_2, y minimización de una función f_3 . Las mismas las podemos mapear de la siguiente forma: $f_1 = \textit{función de aptitud}$, $f_2 = \textit{función de compacidad}$, $f_3 = \textit{umbral de fósforo}$.

Consideramos la siguiente notación para el modelo:

- $J = \{1, 2, 3\}$ que corresponde a los posibles usos de suelo que puede tomar cada pixel (agricultura, forestación y ganadería respectivamente).
- I = {1...nxn} es el conjunto de píxeles asignables que divide la región de estudio.
- La matriz $A = (a_{ij})$, es la aptitud correspondiente a asignarle al pixel $i \in I$ el uso $j \in J$.
- La matriz $E = (e_{ij})$, es la exportación de fósforo que produce el uso $j \in J$ en el pixel $i \in I$.
- Constante $B =$ umbral de fósforo.
- Variable x_{ij} toma el valor 1 si se asigna el uso j al píxel i.

Se partió del modelo exacto propuesto en (Barletta, 2017):

max $$\sum_{i \in I} \sum_{j \in J} a_{ij} . x_{ij} \quad (1)$$

s.a.

$$\sum_{i \in I} \sum_{j \in J} e_{ij} . x_{ij} \leq B \quad (2)$$

$$\sum_{j \in J} x_{ij} = 1 \ \forall i \in I \quad (3)$$

$$x_{ij} \leq a_{ij} + 1 - a_{min} \ \forall i \in I, j \in J \quad (4)$$

$$x_{ij} \in \{0, 1\} \ \forall i \in I, j \in J \quad (5)$$

En las ecuaciones (1-5) se formula el modelo exacto para el problema planteado, donde la función objetivo (1) maximizar la aptitud. Como restricción se establece un umbral de fósforo el cual no se puede superar (2). La ecuación (3) indica que solo se puede realizar una actividad en cada pixel, tomando todas las variables de decisión valores booleanos (5). En la ecuación (4) a_{min} es la menor aptitud mayor

que cero que puede ser asignada en un píxel i dado, $a_{min} = min\ \{a_{ij},\ \forall j \in J \mid aij > 0\}$. Esto quiere decir que no se puede asignar a un pixel un uso con aptitud cero, si $x_{ij} = 0$ entonces $a_{ij} = 0$

3.1 Primera aproximación a la resolución del problema

Como primera aproximación a resolución del problema se consideró el modelo que se presenta a continuación. Para este modelo se considera una grilla de n*n píxeles, donde cada pixel tiene un número de posición y hasta cuatro vecinos.

Se considera la siguiente notación para el modelo:

- I = {1...nxn}, se asumió una matriz cuadrada para simplificar los cálculos de v_1 y v_2, pero podría generalizarse para una matriz nxm.
- $K = \{1...2n(n-1)\}$ corresponde a los bordes entre los píxeles adyacentes, tomando adyacencias horizontales y verticales (**Figura 5**).
- $v_1(k)$ definido de la siguiente forma

  $k\ mod\ (2n\ -1)\ +\ [(k\ div\ (2n-1))\ *\ n]\ \ sii\ k\ mod\ (2n-1)\ \varepsilon\ \{1\ ..\ (n-1)\}$

  $(k\ mod\ (2n\ -1))\ -\ (n-1) + [(k\ div\ (2n-1))\ *\ n]\ \ sii\ k\ mod\ (2n-1)\ \varepsilon\ \{n\ ..\ (2n-1)\}$
- $v_2(k)$ definido como

  $v_1(k) + 1\ \ \ sii\ k\ mod\ (2n-1)\ \varepsilon\ \{1\ ..\ (n-1)\}$

  $v_1(k) + n\ \ \ sii\ k\ mod\ (2n-1)\ \varepsilon\ \{n\ ..\ (2n-1)\}$
- $\delta \in [0,1]$ es un parámetro que pondera la compacidad en la función objetivo.

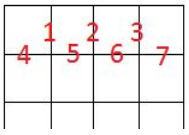


Figura 5: Ejemplo utilizado para la definición de compacidad

Se formula la primera aproximación del problema con las ecuaciones (6)-(12).

Se introduce la función de compacidad local, en donde solo se tienen en cuenta los cuatros vecinos para cada pixel. La misma está ponderada por un parámetro δ que junto con la función aptitud corresponden a la función objetivo que se quiere maximizar.

Las ecuaciones (7), (8), (9) y (12) son análogas a las ecuaciones (2), (3), (4) y (5) respectivamente del modelo anterior.

K refiere a las adyacencias de los píxeles, tanto horizontales como verticales (en la **Figura 5** se muestra un ejemplo de esto).

La variable y_k toma el valor 1 si el pixel $x_{v_1(k)}$ tiene el mismo uso de suelo que el pixel $x_{v_2(k)}$, siendo y_k la intersección de los mismos. De esta forma al sumar todos los valores que toma, se puede tener una idea de qué tan compacto es el resultado. Este método utiliza un concepto de compacidad local, ya que solo tienen en cuenta para cada pixel sus vecinos inmediatos. Esto motivó a incluir otro método que haga una consideración global de la compacidad.

Modelo

maximizar $\sum_{i \in I} \sum_{j \in J} a_{ij} . x_{ij} + \delta \sum_{k \in K} y_k$ (6)

s.a

$$\sum_{i \varepsilon I} \sum_{j \varepsilon J} e_{ij} . x_{ij} \leq B \quad (7)$$

$$\sum_{j \varepsilon J} x_{ij} = 1 \quad \forall i \in I \quad (8)$$

$$x_{ij} \leq a_{ij} + 1 - a_{min} \quad \forall i \in I, j \in J \quad (9)$$

$$2y_k \leq x_{v_1(k)j} + x_{v_2(k)j} \quad \forall j \in J, k \in K \quad (10)$$

$$y_k \in \{0, 1\} \quad \forall k \in K \quad (11)$$

$$x_{ij} \varepsilon \{0, 1\} \quad \forall i \in I, \quad j \in J \quad (12)$$

3.2 Segunda aproximación a la resolución del problema

Para este segundo modelo también se considera una grilla de n*n píxeles, donde cada pixel tiene un número de posición y hasta cuatro vecinos.

Se define como *cluster*, a un conjunto de píxeles contiguos con un mismo uso de suelo. Entonce se introduce en este modelo otra componente de la función de compacidad en la que se tienen en cuenta el tamaño de los clusters generados.

Se considera la siguiente notación para el modelo:

- C_{min} cantidad mínima de píxeles contiguos del mismo uso desde la cual se comienza a considerar que es un cluster.

- W_x conjunto de clusters determinado por una asignación de x que se generan de tamaño mayor o igual a C_{min}

Se plantea la formulación del modelo propuesto en esta segunda instancia, por las ecuaciones (13)-(19). La diferencia sustancial con la formulación (6)-(12) anterior es que se introduce un segundo componente a la función de compacidad, que es la compacidad relacionada con el tamaño de los clusters que se generan. La suma de ambas componentes determinan la función de compacidad. Esta se pondera con un parámetro $(1-\alpha)$, que junto con la función aptitud (ponderada con el parámetro α) corresponden a la función objetivo que se quiere maximizar (13).

Tanto la función de aptitud como las de compacidad están normalizadas.

Modelo

maximizar $\alpha \sum_{i \varepsilon I} \sum_{j \varepsilon J} a_{ij} . x_{ij} + (1-\alpha)((\sum_{k \varepsilon K} y_k + \sum_{w \varepsilon W_x} |w|)/2)$ (13)

s.a

$$\sum_{i \varepsilon I} \sum_{j \varepsilon J} e_{ij} . x_{ij} \leq B \quad (14)$$

$$\sum_{j \varepsilon J} x_{ij} = 1 \quad \forall i \in I \quad (15)$$

$$x_{ij} \leq a_{ij} + 1 - a_{min} \; \forall i \in I, j \in J \quad (16)$$

$$2 y_k \leq x_{v_1(k)j} + x_{v_2(k)j} \quad \forall j \in J \quad (17)$$

$$y_k \in \{0, 1\} \quad \forall k \in K \quad (18)$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \; j \in J \quad (19)$$

Capítulo 4 - Algoritmos

En este capítulo se presentan los algoritmos desarrollados para resolver el problema de optimización de usos de suelo especificado por la formulación (13)-(19).

El lenguaje utilizado para crear los algoritmos fue c++. Se eligió este lenguaje ya que presenta un manejo de estructuras potente.

Se comenzó creando un algoritmo heurístico con el objetivo de maximizar la aptitud, para luego poder ir agregando búsquedas locales que lleven a soluciones compactas. Para esto, primero se realizó un estudio de los datos de entrada del caso de la cuenca de la Laguna de Rocha, donde se observó que el uso de suelo que genera más exportación de fósforo también es el que tiene mejor aptitud (agricultura), por esto la relación aptitud/exportación es baja comparado con los otros usos.

Viendo los datos de entrada se pudo observar que si se asigna sólo el uso de suelo ganadería, nunca se supera el umbral de fósforo, por lo tanto en la función generadora de soluciones iniciales se asignan inicialmente a todos los píxeles el uso ganadería y luego se van modificando los píxeles por los usos de mayor aptitud hasta llegar al límite del umbral.

4.1 Elección de la heurística

La primera heurística que se consideró fue de naturaleza constructiva, según los siguientes pasos:

- En una estructura auxiliar guardar la relación aptitud/fósforo por uso, quedándose con la mayor para cada pixel.
- Ordenar la estructura auxiliar de forma decreciente.
- Asignar como solución inicial a todos los píxeles el uso que exporta menos fósforo.
- Ir modificando píxeles de la solución (tomados en el orden según lo indicado en el paso dos), teniendo en cuenta que la exportación de fósforo no supere el umbral.

Al comparar la solución que se obtuvo con la obtenida en (Barletta, 2017), se observó que los resultados de la heurística propuesta se alejaban considerablemente del óptimo, tanto el valor de aptitud como el fósforo exportado. El problema con la heurística fue que no se estaba asignando nunca el uso agricultura y el fósforo no era una limitante. Esto se debió a que el uso agricultura tiene una aptitud alta pero una exportación de fósforo también alta, por esto la relación aptitud/fósforo quedaba al final de la lista y no era asignada.

Luego se optó por utilizar la siguiente variante:

- En una estructura auxiliar guardar el uso con mayor aptitud para cada pixel.
- Ordenar la estructura auxiliar de forma decreciente.
- Asignar como solución inicial a todos los píxeles el uso que exporta menos fósforo.
- Ir modificando píxeles de la solución (tomados en el orden según lo indicado en el paso dos), teniendo en cuenta que la exportación de fósforo no supere el umbral.

4.2 Elección de la metaheurística

Se observó que el problema era parecido al problema del mochilero múltiple elección (multiple-choice knapsack problem) (Martello y Toth, 1990) en donde se tiene que llenar una mochila maximizando el valor, sujeto a un peso máximo, con objetos de diferente valor y peso.

El problema del mochilero es un problema de optimización combinatoria muy referenciado en la literatura de Investigación de Operaciones, tanto por sus aplicaciones como por su estructura, que lo hace ideal para la evaluación del desempeño de métodos de búsqueda inteligente en problemas de optimización combinatoria (Sandoya, 2014).

Buscando las diferentes metodologías para encarar este problema se observó que la metodología GRASP es muy eficiente para la resolución de problemas de optimización combinatoria en general (Resende y Ribeiro, 2010) y en particular para el problema del mochilero (Sandoya, 2014). Notar que GRASP se basa en una construcción greedy, y el problema de la mochila admite una resolución greedy eficiente.

Por lo tanto se decidió tomar como base esta metodología para resolver el problema de este proyecto.

4.3 GRASP instanciado a nuestro problema

En la implementación de GRASP realizada (Figura 6), en vez de tener una sola búsqueda local, se utilizaron tres búsquedas donde cada una prioriza un aspecto diferente del problema: en una de las búsquedas se prioriza la aptitud, en otra la cantidad de píxeles vecinos con el mismo uso y en la última la cantidad y tamaño de clusters. Estas se seleccionan aleatoriamente con igual peso cada una. Asimismo, la generación de soluciones se realiza de dos formas diferentes: una que obtiene soluciones de alta aptitud pero difíciles de modificar en las búsquedas locales y otra que no tiene alta aptitud pero tiene altas posibilidades de ser mejorable en las búsquedas locales.

Pseudocódigo

```
Procedure Grasp
Para i Hasta MAX_ITERACIONES hacer
        solucion = Generar_Solucion_Inicial()
        nro_aleatorio = Sortear_Numero_Aleatorio(0..2)
        Si ((α = 1) || ((α > 0) && (nro_aleatorio == 0))) entonces
                solucio_busqueda = Busqueda_Local_Priorizando_Aptitud()
        Fin
        Si (nro_aleatorio == 1) entonces
                solucio_busqueda = Busqueda_Local_Priorizando_Compacidad_Local()
        Fin
        Si (nro_aleatorio == 2) entonces
                solucio_busqueda = Busqueda_Local_Priorizando_Cluster()
        Fin
        solucion = Mejor_Solucion(solucio_busqueda, solucion)
Fin Para

FinGrasp
```

Figura 6 - Pseudocódigo GRASP instanciado al problema del proyecto

4.4 Construcciones de soluciones iniciales

Soluciones Iniciales

Para la implementación de GRASP se utilizaron dos generadores de soluciones iniciales.

- El generador descrito en la **Figura 7** recibe como parámetro dos matrices de píxeles, una ordenada de mayor a menor por aptitud y la otra que contiene la posición de cada píxel en la matriz original. El otro generador descrito en la **Figura 8** también recibe dos matrices, una ordenada de mayor a menor según la relación aptitud/fósforo y la otra que tiene la posición de cada píxel en la matriz original.
- El primer generador está basado en encontrar una solución inicial con buena aptitud y el segundo generador busca mejorar la relación aptitud/fósforo para de esta forma dejar cierta holgura en el fósforo y que las búsquedas locales puedan realizar más cambios por cada corrida.
- La solución inicial basada en aptitud asigna a cada píxel con una probabilidad 0,5 el uso con mayor aptitud. Para esto, primero se obtiene una matriz que contiene los píxeles ordenados de mayor a menor aptitud y su uso correspondiente.
- En caso de no elegirse un uso de la matriz ordenada, se elige un uso de forma aleatoria, siempre y cuando ese uso tenga aptitud positiva para el píxel a asignar.

- Se deja una holgura de fósforo para luego tener un margen de modificación en las búsquedas locales.
- El segundo generador busca mejorar la relación aptitud/fósforo de los píxeles. El mismo recibe como parámetro de entrada la matriz generada por el primer generador. Luego, a partir de una posición tomada de forma aleatoria, se modifican los usos de los píxeles por el uso que tenga la mejor relación aptitud/fósforo. De esta forma se logra tener una mejor holgura de fósforo para que luego las búsquedas locales puedan realizar más modificaciones y no estén tan acotadas por el fósforo.

Pseudocódigos

```
Procedure Generador 1
  se_supero_umbral_fosforo = false
  probabilidad = Defino_probabilidad()
  contador = 1
  Mientras (!Recorrieron_Todos_Pixeles(contador) && (!se_supero_umbral_fosforo)) Hacer
   aleatorio = Sorteo_Numero_aleatorio()
    Si (aleatorio < probabilidad) Entonces
          tipo = Tomo_Tipo_Matriz_Aptitud(contador)
      Switch(tipo)
        case Agricultura:
          Si(!Asignando_Pixel_Agricultura_Supero_Umbral(contador) Entonces
              Se_Asigna_Pixel_Agricultura(contador)
          SiNo
             se_supero_umbral_fosforo = true
           FinSi

        case Forestacion:
          Si(!Asignando_Pixel_Forestacion_Supero_Umbral(contador) Entonces
              Se_Asigna_Pixel_Forestacion(contador)
          SiNo
              se_supero_umbral_fosforo = true
          FinSi

        case Ganaderia:
          Asigno_Pixel_Ganaderia(contador)

    FinSwitch
    Si !se_supero_umbral_fosforo Entonces
        contador++
    FinSi
  FinMientras

FinGenerador1
```

Figura 7 - Pseudocódigo generador 1

```
Procedure Generardor2
  pos = Sorteo_Numero_aleatorio(0,max_pixeles);
  Mientras (pos < max_pixeles) Hacer
    Switch(Tipo(pos))
      case Agricultura:
        Si Relacion_Aptitud_Contra_Fosforo_Ganaderia(pos)>=
             Relacion_Aptitud_Contra_Fosforo_Agricultura(pos)  &&
             !Asignando_Pixel_Ganaderia_Supero_Umbral(pos) Entonces
          Se_Asigna_Pixel_Ganaderia(pos)
        SiNo
          Si(Relacion_Aptitud_Contra_Fosforo_Forestacion(pos)>=
               Relacion_Aptitud_Contra_Fosforo_Agricultura(pos)  Entonce
            Se_Asigna_Pixel_Forestacion(pos)
          FinSi
        FinSi
      case Forestacion:
        Si (Relacion_Aptitud_Contra_Fosforo_Ganaderia(pos) >=
            Relacion_Aptitud_Contra_Fosforo_Forestacion(pos) Entonces
          Se_Asigna_Pixel_Ganaderia(pos)
        SiNo
          Si(Relacion_Aptitud_Contra_Fosforo_Agricultura(pos) >=
               Relacion_Aptitud_Contra_Fosforo_Forestacion(pos) &&
               !Asignando_Pixel_Agricultura_Supero_Umbral(pos) Entonces
            Se_Asigna_Pixel_Agricultura(pos)
          FinSi
        FinSi

      case Ganaderia:
        Si (Relacion_Aptitud_Contra_Fosforo_Agricultura(pos) >=
            Relacion_Aptitud_Contra_Fosforo_Ganaderia(pos) &&
            !Asignando_Pixel_Agricultura_Supero_Umbral(pos) Entonce
         Se_Asigna_Pixel_Agricultura(pos)
        SiNo
          Si(Relacion_Aptitud_Contra_Fosforo_Forestacion(pos) >=
              Relacion_Aptitud_Contra_Fosforo_Ganaderia(pos) &&
              !Asignando_Pixel_Forestacion_Supero_Umbral(pos) Entonces
            Se_Asigna_Pixel_Forestacion(pos)
          FinSi
        FinSi
     FinSwitch
    pos++
FinMientras

FinGenerador2
```

Figura 8 - Pseudocódigo generador 2

4.5 Búsquedas locales

Dado que se necesita maximizar la aptitud y la compacidad de forma simultánea, se decidió realizar diferentes búsquedas locales atacando cada uno de estos aspectos. La primera busca obtener un máximo local teniendo en cuenta únicamente la aptitud. Para la compacidad se proponen dos búsquedas locales. La primera maximiza la compacidad tomando en cuenta ésta como la suma de las intersecciones de píxeles vecinos del mismo uso (variables y_k). La segunda búsqueda local maximiza la compacidad tomando en cuenta esta como la cantidad de píxeles contiguos del mismo uso, pero contando estos píxeles solamente si la cantidad de píxeles contiguos del mismo uso supera un mínimo (C_{min}). Esta búsqueda local se realiza modificando los píxeles vecinos a un pixel origen en un radio de valor “profundidad”. La profundidad es un parámetro de la búsqueda local que indica hasta qué radio se desean modificar los usos de los píxeles partiendo de un pixel origen. La modificación de píxeles se hace de forma recursiva disminuyendo la profundidad en cada entrada hasta llegar a profundidad cero.

Pseudocódigo

```
Procedure BusquedaLocalAptitud

 pos = Sorteo_Numero_aleatorio(0,max_pixeles)
 Mientras (pos < max_pixeles) Hacer
   cant_de_vecinos_tipo_agricutura = Cantidad_Vecinos_Tipo(Agricultura)
   cant_de_vecinos_tipo_ganaderia = Cantidad_Vecinos_Tipo(Ganaderia)
   cant_de_vecinos_tipo_forestacion = Cantidad_Vecinos_Tipo(Forestacion)
   aptitud_inicial = Calcular_Aptitud()
   aptitud_modificada = aptitud_inicial
   modificado = true
   Switch(Tipo(pos))
     case Agricultura:
         Si Mejora_Aptitud_Cambiando_Forestacion() Entonces
           Se_Asigna_Pixel_Forestacion(pos)
           modificado = true
           Calcular_Compacidad()
           aptitud_modificada = Calcular_Aptitud()
         SiNo
           Si (Mejora_Aptitud_Cambiando_Ganaderia() Entonces
             Se_Asigna_Pixel_Ganaderia(pos)
             modificado = true
             Calcular_Compacidad()
             aptitud_modificada = Calcular_Aptitud()
           FinSi
         FinSi
     case Ganaderia:
```

```
        Si Mejora_Aptitud_Cambiando_Forestacion() Entonces
          Se_Asigna_Pixel_Forestacion(pos)
          modificado = true
          Calcular_Compacidad()
          aptitud_modificada = Calcular_Aptitud()
        SiNo
          Si Mejora_Aptitud_Cambiando_Agricultura() Entonces
            Se_Asigna_Pixel_Agricultura(pos)
            modificado = true
            Calcular_Compacidad()
            aptitud_modificada = Calcular_Aptitud()
          FinSi
        FinSi
    case Forestacion:
      Si Mejora_Aptitud_Cambiando_Agricultura() Entonces
        Se_Asigna_Pixel_Agricultura(pos)
        modificado = true
        Calcular_Compacidad()
        aptitud_modificada = Calcular_Aptitud()
      SiNo
        Si Mejora_Aptitud_Cambiando_Ganaderia() Entonces
          Se_Asigna_Pixel_Ganaderia(pos)
          modificado = true
          Calcular_Compacidad()
          aptitud_modificada = Calcular_Aptitud()
        FinSi
      FinSi
    FinSwitch

    Si modificado Entonces
      Si aptitud_modificada > aptitud_inicial Entonces
        aptitud_inicial = aptitud_modificada
        ActualizarValores()
      SiNo
        CambiosParaAtras()
      FinSi
    FinSi
    Si (pos == max_pixeles &&
          !Supera_Umbral_Fosforo(ValorFosforo() + fosforo_mas_alto) ) Entonces
      pos = Aleatorio(0,max_pixeles)
    SiNo
      pos++
    FinSi
  FinMientras

FinBusquedaLocalAptitud
```

Figura 9 - Pseudocódigo *búsqueda local aptitud*

```
Procedure busquedaCompacidadLocal
  pos = Sorteo_Numero_aleatorio(0,max_pixeles)
  Mientras (pos < max_pixeles) Hacer
    cant_de_vecinos_tipo_agricutura = Cantidad_Vecinos_Tipo(Agricultura)
    cant_de_vecinos_tipo_ganaderia = Cantidad_Vecinos_Tipo(Ganaderia)
    cant_de_vecinos_tipo_forestacion = Cantidad_Vecinos_Tipo(Forestacion)
    compacidad_inicial = Calcular_Compacidad()
    compacidad_modificada = compacidad_inicial
    modificado = true
    Switch(Tipo(pos))
      case Agricultura:
          Si Mejora_Cantidad_Vecinos_Cambiando_Forestacion() Entonces
            Si Aptitud_Forestacion_Positivo(pos) &&
                 !Asignando_Pixel_Forestacion_Supero_Umbral(pos) Entonces
              Se_Asigna_Pixel_Forestacion(pos)
              modificado = true
              compacidad_modificada = Calcular_Compacidad()
            FinSi
          SiNo
        Si Mejora_Cantidad_Vecinos_Cambiando_Ganaderia() Entonces
          Se_Asigna_Pixel_Ganaderia(pos)
          modificado = true
          compacidad_modificada = Calcular_Compacidad()
        FinSi
      case Forestacion:
        Si Mejora_Cantidad_Vecinos_Cambiando_Agricultura() Entonces
          Si Aptitud_Agricultura_Positivo(pos) &&
              !Asignando_Pixel_Agricultura_Supero_Umbral(pos) Entonces
            Se_Asigna_Pixel_Agricultura(pos)
            modificado = true
            compacidad_modificada = Calcular_Compacidad()
          FinSi
       SiNo
          Si Mejora_Cantidad_Vecinos_Cambiando_Ganaderia() Entonces
            Se_Asigna_Pixel_Ganaderia(pos)
            modificado = true
            compacidad_modificada = Calcular_Compacidad()
          FinSi

      case Ganaderia:
        Si Mejora_Cantidad_Vecinos_Cambiando_Agricultura() Entonces
          Si Aptitud_Agricultura_Positivo(pos) &&
              !Asignando_Pixel_Agricultura_Supero_Umbral(pos) Entonces
            Se_Asigna_Pixel_Agricultura(pos)
            modificado = true
            compacidad_modificada = Calcular_Compacidad()
          FinSi
        SiNo
```

```
          Si Mejora_Cantidad_Vecinos_Cambiando_Forestacion() Entonces
            Si Aptitud_Forestacion_Positivo(pos) &&
                !Asignando_Pixel_Forestacion_Supero_Umbral(pos) Entonces
              Se_Asigna_Pixel_Forestacion(pos)
              modificado = true
              compacidad_modificada = Calcular_Compacidad()
            FinSi
          FinSi
        FinSi
    FinSwitch
    Si modificado Entonces
      Si compacidad_modificada > compacidad_inicial Entonces
        compacidad_inicial = compacidad_modificada
        ActualizarValores()
        pos = Aleatorio(0, max_pixeles)
      SiNo
        CambiosParaAtras()
      FinSi
    SiNo
      pos++
    FinSi
  FinMientras

FinbusquedaCompacidadLocal
```

Figura 10 - Pseudocódigo *búsqueda local por compacidad local*

```
Procedure BusquedaCompacidadGlobal
  Para i=0 hasta iteraciones hacer
    pos = Aleatorio(0,MAX_PIXELES)
    Mientras (pos < MAX_PIXELES && !VecinoDistinto(pos)) Hacer
      Para i = Vecino(pos) hasta TodosLosVecinos(pos)
        Si(Tipo(i) != Tipo(pos))
          CambiarPixel(i, Tipo(i), PROFUNDIDAD)
        FinSi
      FinPara
    Fin Mientras
  FinPara

FinBusquedaCompacidadGlobal
```

Figura 11 - Pseudocódigo *búsqueda local por compacidad global*

```
Procedure CambiarPixel(pixel, tipo, profundidad)
  Si (Aptitud(pixel, tipo) > 0 && !Asignando_Pixel_Supero_Umbral(pixel, tipo) Entonces
    Se_Asigna_Pixel_Tipo(pixel,tipo)
    profundidad --
  FinSi
  compacidad= Calcular_Compacidad()
  compacidad_modificada = compacidad
  Para i = 0 Hasta Todos_Los_Vecinos (pixel) Hacer
    Si ( Tipo(i) != Tipo(pixel) && profundidad > 0) Entonces
      CambiarPixel(i, Tipo(vecino), tipo, profundidad)
      volver profundidad a valor anterior a la recursión
    FinSi
    Si(!Vecinos_Modificar(i)) Entonces
      compacidad_modificada =  Calcular_Compacidad()
      Si(compacidad_modificada < compacidad) Entonces
        CambiosParaAtras()
      SiNo
        parar = true
        compacidad = compacidad_modificada
      FinSi
    FinSi
  FinPara
FinCambiarPixel
```

Figura 12 - Pseudocódigo *cambiar pixel*

Capítulo 5 - Pruebas computacionales

En este capítulo se muestran las pruebas computacionales del algoritmo.

5.1 Plan de pruebas

Se realizaron distintos tipos de pruebas. Las mismas se explican a continuación.

Diagrama de Pareto
En estas pruebas se busca ver la relación que hay entre la aptitud que se obtiene heurísticamente con el algoritmo implementado contra la aptitud del modelo exacto.
Se muestran dos resultados:

- Primer resultado sin tener en cuenta la compacidad. Esto se hace para que la comparación sea justa, es decir en las mismas condiciones.
- Segundo resultado teniendo en cuenta distintos valores de compacidad, esto se logra variando el peso que se da a la compacidad.

Los resultados mostrados tienen como objetivos:

1. Mostrar la calidad de aproximación de la heurística con respecto al modelo exacto cuando no se considera la compacidad.
2. Observar cuánto difieren las soluciones cuando se comienza a exigir más compacidad.

Las soluciones se presentan gráficamente en el espacio de los objetivos aptitud y fósforo, transformando el problema (13)-(19) en uno multi-objetivo donde se busca maximizar aptitud y minimizar fósforo.

Pruebas de tamaño de cluster
El tamaño del cluster se utiliza para la segunda medida de compacidad, en donde se cuentan la cantidad de píxeles del mismo uso dentro de un cluster. Un cluster se considera como tal solo solo si su tamaño es mayor a un tamaño dado; dicho tamaño se ingresa como parámetro al algoritmo. El objetivo de estas pruebas es ir variando el tamaño de los clusters y analizar cómo afecta esto a la compacidad del problema, si es que tiene una relación directa.

Pruebas de performance
En estas pruebas se analizan los tiempos de ejecución del algoritmo heurístico, modificando los valores de algunos de sus factores. Los parámetros que se van a variar son: α y umbral de fósforo B, siendo 1 - α el valor de ponderación de la compacidad.

Se van a registrar los tiempos obtenidos en cada prueba para hacer un análisis de performance y comprobar si los tiempos del algoritmo son afectado al modificar las variables mencionadas anteriormente.

Pruebas de búsquedas locales

El objetivo de estas pruebas es comprobar que las búsquedas locales mejoran las soluciones iniciales. Se va a mostrar en una tabla comparativa para distintos valores de α y umbrales de fósforo, cual fue la mejor solución inicial, cuál fue la solución inicial a partir de la cual se obtuvo la solución final y cómo mejoró la búsqueda local la solución inicial de la cual se partió.

Pruebas de robustez

Se busca ver cómo reacciona el algoritmo ante algunos cambios en los datos de entrada. Esto es imprescindible ya que muchas veces pueden haber errores en los datos de entrada y eso no debería afectar la solución devuelta por el algoritmo. El objetivo de esta prueba es comprobar que cambios pequeños no produce resultados drásticamente distintos.

Para ello, se alteran los datos de entrada mediante la modificación de la aptitud según:

aptitud = aptitud + (número aleatorio entre -0,5 y 0,5) * (aptitud * 0.2)

Por lo tanto la aptitud se altera entre 0 y 10%.

Esta prueba se realiza variando α en tres valores y para cada uno de ellos se utilizan tres umbrales de fósforo distintos.

Mapas

El objetivo de los mapas es mostrar de forma gráfica los resultados, que muchas veces son difíciles de apreciar en números, principalmente lo referente a la compacidad.

5.2 Resultados y análisis

Para realizar las pruebas de decidió fijar las iteraciones GRASP en 100 y la profundidad en 100. La razón por la que se tomaron estos valores fue porque luego de realizar varias pruebas con distintos valores, se observó que el tiempo de respuesta del algoritmo variaba significativamente, pero sin embargo los resultados no se veían muy afectados.

En las pruebas la aptitud exacta refiere a la aptitud obtenida con CPLEX (CPLEX, 2017). Los datos correspondientes a la aptitud exacta son tomados de la tesina de Antonella Barletta (Barletta, 2017).

Iteraciones GRASP	**100**
Profundidad	**100**

Se considera la siguiente nomenclatura en el detalle de las pruebas:

- AE Aptitud exacta
- AH Aptitud heurística, obtenida con el algoritmo propuesto en este trabajo
- CO Compacidad obtenida con el algoritmo
- B Umbral de fósforo
- FH Fósforo correspondiente a la heurística
- TMC Tamaño mínimo que puede tomar cada Cluster.

5.2.1 Diagrama de Pareto

Las pruebas fueron realizadas considerando TMC = 100

Para este caso se va a mantener $\alpha = 1$, el objetivo aquí es no tener en cuenta la compacidad. Se quiere comparar la aptitud heurística con la aptitud exacta al ir variando el umbral de fósforo.

En la **Figura 13** se muestra el resultado de las pruebas, que como se puede apreciar para los diferentes valores de umbral de fósforo la aptitud heurística y la aptitud exacta están muy cercanas. Esto muestra que el algoritmo es capaz de acercarse a la solución exacta. Para comprobarlo numéricamente se calcula el GAP entre la aptitud heurística y la aptitud exacta, aplicando la siguiente fórmula:

$$GAP = \left(\frac{(AE - AH)}{AE}\right) * 100$$

El promedio del GAP es de 0,48 , con un valor máximo de 1,78 y un mínimo de 0.

En la **Tabla 1** se muestran todos los resultados obtenidos en las pruebas que no son reflejados en gráfica, como es la medida de compacidad obtenida y el fósforo que se exporta en cada caso.

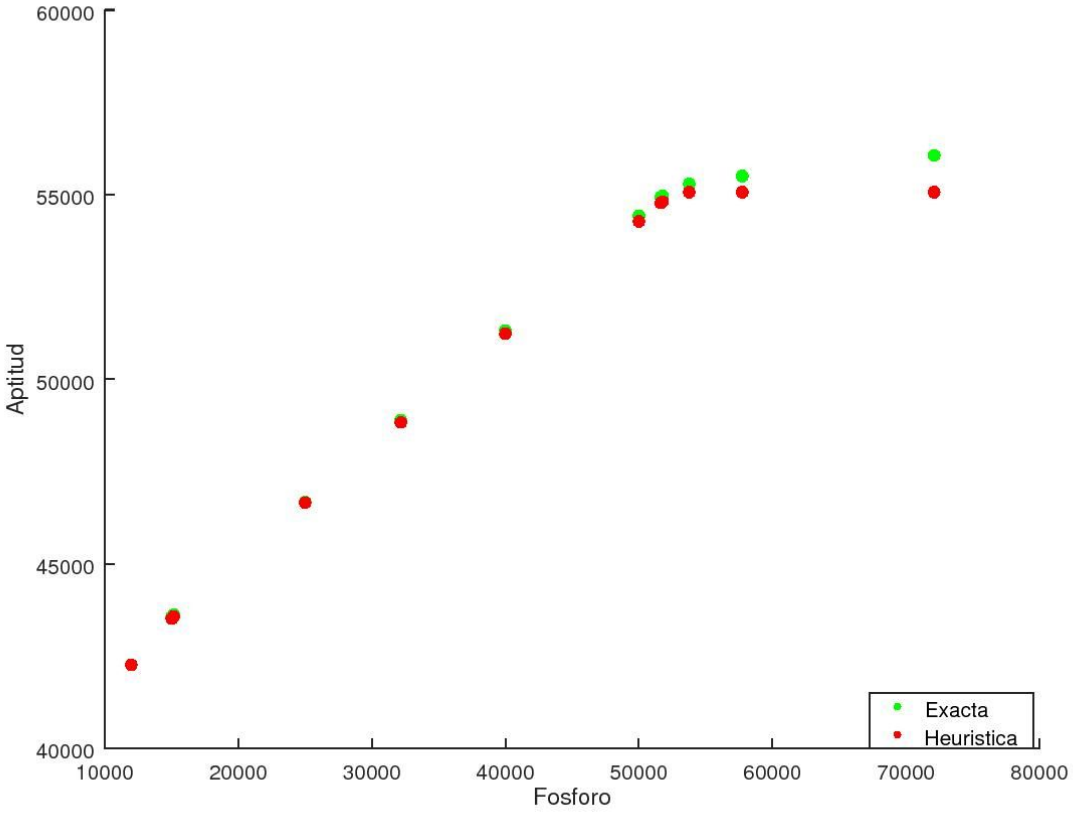


Figura 13 - Gráfica de Pareto Aptitud Exacta vs Actitud Heurística

α	B	AH	CO	FH	AE	GAP
1	**11982**	42262	0,80	11982	**42262**	**0,00**
1	**15020**	43510	0,77	15019	**43565**	**0,13**
1	**15170**	43575	0,77	15169	**43621**	**0,11**
1	**25000**	46656	0,72	25000	**46670**	**0,03**
1	**32174**	48824	0,67	32173	**48894**	**0,14**
1	**40000**	51233	0,70	40000	**51322**	**0,17**
1	**50000**	54273	0,64	50000	**54424**	**0,28**
1	**51645**	54770	0,64	51644	**54933**	**0,30**
1	**51779**	54810	0,63	51778	**54965**	**0,28**
1	**53770**	55067	0,59	53243	**55288**	**0,40**
1	**57741**	55064	0,60	52934	**55501**	**0,79**
1	**72108**	55064	0,60	52934	**56062**	**1,78**
Promedio						**0,48**

Tabla 1 - **Aptitud Heurística vs Aptitud Exacta con** $\alpha = 1$ **y TMC = 100**

Aunque para estas pruebas no se esté considerando la compacidad, podemos ver que esta no es cero, ya que cada una de las configuraciones tienen un grado de compacidad.

Se puede apreciar que en general a medida que el umbral de fósforo aumenta, la compacidad se hace menor. En la medida que hay más holgura de fósforo, este ya no es una limitante, sin embargo en este caso al ponderar más las aptitud con $\alpha = 1$, vemos que la compacidad decrece. Esto se puede explicar analizando los datos de entrada, que al no estar limitados por fósforo ni compacidad, se va asignando a cada pixel el uso de suelo que aporta más aptitud, y esto no mantiene una relación con los usos de suelo, por lo tanto se pierde la compacidad.

También se aprecia un aumento significativo en el GAP, lo que significa una solución más alejada de la exacta.

A continuación se muestran las tablas y gráficas correspondientes variando α:

α	B	AH	CO	FH	AE	GAP
0,5	**11982**	42229	0,80	11981	**42262**	**0,08**
0,5	**15020**	43334	0,79	15019	**43565**	**0,53**
0,5	**15170**	43347	0,79	15169	**43621**	**0,63**
0,5	**25000**	46535	0,77	25000	**46670**	**0,29**
0,5	**32174**	48722	0,76	32173	**48894**	**0,35**
0,5	**40000**	51186	0,72	40000	**51322**	**0,26**
0,5	**50000**	54276	0,65	50000	**54424**	**0,27**
0,5	**51645**	54770	0,64	51644	**54933**	**0,30**
0,5	**51779**	54807	0,64	51779	**54965**	**0,29**
0,5	**53770**	54810	0,64	51779	**55288**	**0,86**
0,5	**57741**	55060	0,60	52592	**55501**	**0,79**
0,5	**72108**	55060	0,60	52592	**56062**	**1,79**
Promedio						**0,63**

Tabla 2 - **Aptitud Heurística vs Aptitud Exacta con** $\alpha = 0,5$ **y TMC = 100**

α	B	AH	CO	FH	AE	GAP
0	**11982**	26960	0,96	11530	**42262**	**36,21**
0	**15020**	36095	0,82	14620	**43565**	**17,15**
0	**15170**	35777	0,82	14821	**43621**	**17,98**
0	**25000**	42020	0,80	12027	**46670**	**9,96**
0	**32174**	42095	0,80	12470	**48894**	**13,91**
0	**40000**	42007	0,80	12347	**51322**	**18,15**
0	**50000**	42215	0,80	12757	**54424**	**22,43**
0	**51645**	42215	0,79	12757	**54933**	**23,15**
0	**51779**	42215	0,79	12757	**54965**	**23,20**
0	**53770**	42215	0,79	12757	**55288**	**23,65**
0	**57741**	42215	0,79	12757	**55501**	**23,94**
0	**72108**	42046	0,79	12205	**56062**	**25,00**
Promedio						**21,52**

Tabla 3 - **Aptitud Heurística vs Aptitud Exacta con** $\alpha = 0$ **y TMC = 100**

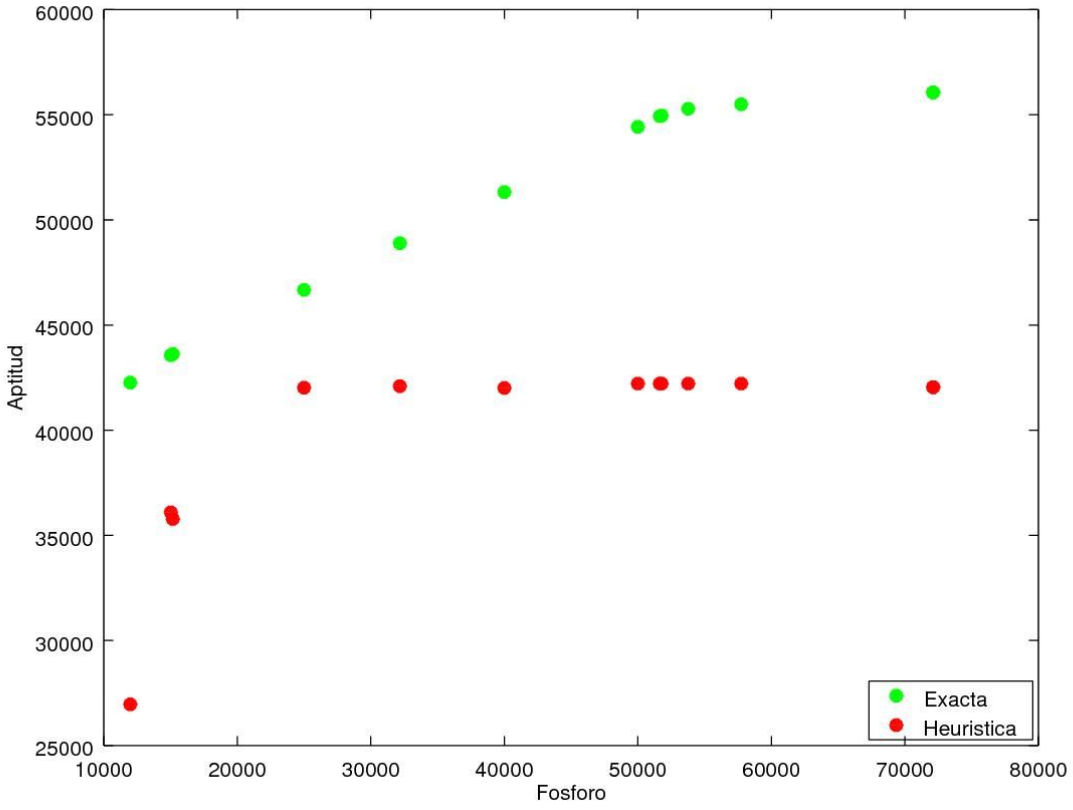


Figura 14 - Aptitud Heurística vs Aptitud Exacta con $\alpha = 0$ y TMC = 100

α	B	AH	CO	FH	AE	GAP
0	**11982**	26960	0,96	11530	**42262**	**36,21**
0	**32174**	42095	0,80	12470	**48894**	**13,91**
0,5	**11982**	42229	0,80	11981	**42262**	**0,08**
0,5	**32174**	48722	0,76	32173	**48894**	**0,35**
1	**11982**	42262	0,80	11982	**42262**	**0,00**
1	**32174**	48824	0,67	32173	**48894**	**0,14**

Tabla 4 - Comparativa de los resultados anteriores

Como se pueden ver en los resultados de las tablas anteriores (resumidas en la **Tabla 4),** a medida que disminuye α (más peso a la compacidad) la aptitud comienza a quedar más lejana de la aptitud exacta. Esto se da porque se generan soluciones mucho más compactas, las que deben sacrificar aptitud. También se puede observar que para $\alpha = 1$ y $\alpha = 0,5$ el fósforo resultante del algoritmo, si bien nunca excede el umbral (porque está implementado así), siempre se aproxima. En cambio para $\alpha = 0$ se puede apreciar que el valor de fósforo resultante está muy alejado del

umbral. Analizando los valores de entrada, se puede ver que cuando no se le da peso a la aptitud se asigna a más píxeles el uso de ganadería, que hacen la solución más compacta pero son los que aportan menos aptitud. Asimismo, se puede observar que para $\alpha = 0$ las soluciones son más compactas cuando el umbral de fósforo es menor. Esto se debe a que en los generadores no se tiene en cuenta la compacidad para generar soluciones iniciales, entonces cuando el umbral de fósforo es muy bajo, las soluciones tienen una asignación de píxeles de uso ganadería muy grande, por lo tanto quedan más compactas. Este comportamiento es el esperado por el algoritmo.

5.2.2 Pruebas de tamaño de cluster

Para las siguientes pruebas se utiliza la nomenclatura:

- #C Cantidad de Clusters generados.
- #CA Cantidad de Cluster generados con uso Agricultura
- #CF Cantidad de Cluster generados con uso Forestación
- #CG Cantidad de Cluster generados con uso Ganadería.

Otra de las entradas configurables del algoritmo es el tamaño de cada cluster, este refiere a la cantidad de píxeles contiguos con el mismo uso asignado que definen lo que se considera un cluster. Se va a ir variando dicha entrada y observando qué cantidad de clusters de cada uso se forma. También se va a realizar el cálculo del tamaño medio del cluster y su desviación estándar por cada uso de suelo en cada una de las configuraciones. El cálculo se realizó de la siguiente manera:

$$\text{Tamaño medio } \mu = \sum_{i=1}^{\#c} TamCi \,/\#c$$

$$\text{Desviación estándar } \sigma = \sqrt{\sum_{i=1}^{\#C} (\mu - TamCi)^2 \,/(\#C - 1)}$$

donde μ y σ se calculan para una solución y para cada uno de los usos A, G y F.

Con esto veremos cuánto varían los tamaños de los distintos clusters. Se muestran los resultados variando α.

$\alpha = 1$

TMC	#CA	$media_A$	σ_A	#CF	$media_F$	σ_F	#CG	$media_G$	σ_G
1	2486	6,88	29,21	4207	4,46	426,36	4810	12,49	288,32
10	375	33,86	69,22	276	36,39	5379,39	564	87,66	838,81
100	15	283,13	227,78	16	239,19	45197,10	44	839,77	2929,21
300	5	537,40	238,30	4	501,75	89675,58	12	2636,75	5356,40
500	2	784,50	149,20	1	949,00	-	7	4233,29	6742,58
1000	0	-	-	0	-	-	5	5602,60	7744,82
5000	0	-	-	0	-	-	1	19438,00	-

Tabla 5 - Cantidad de Clusters con $\alpha = 1$

$\alpha = 0,5$

TMC	#CA	$media_A$	σ_A	#CF	$media_F$	σ_F	#CG	$media_G$	σ_G
1	1193	14,34	276,96	4207	4,46	426,36	4058	14,80	274,18
10	46	36,39	5379,38	276	36,39	5379,38	378	135,92	890,35
100	4	3467,75	3809,96	16	239,19	45197,09	50	860,86	2340,56
300	3	4552,33	3836,11	4	501,75	89675,58	20	1897,20	3498,39
500	3	4552,33	3836,11	1	949,00	-	13	2714,31	4160,52
1000	3	4552,33	3836,11	0	-	-	8	3994,12	4979,16
5000	0	-	-	0	-	-	1	51712,00	-

Tabla 6 - Cantidad de Clusters con $\alpha = 0.5$

$\alpha = 0$

TMC	#CA	$media_A$	σ_A	#CF	$media_F$	σ_F	#CG	$media_G$	σ_G
1	141	1,40	0,87	4207	4,46	426,36	3878	20,92	603,00
10	0	-	-	276	36,39	5379.39	345	200,30	1964,40
100	0	-	-	16	230,56	36744,52	33	1892,36	7961,43
300	0	-	-	4	479,75	59026,92	12	4909,17	12983,76
500	0	-	-	1	864,00	-	6	9730,50	20494,90
1000	0	-	-	0	-	-	3	18748,33	28391,70
5000	0	-	-	0	-	-	1	51532,00	-

Tabla 7 - Cantidad de Clusters con $\alpha = 0$

En base a lo observado en las tablas 5, 6 y 7 se puede concluir que al aumentar el TMC, disminuye la cantidad de clusters generados (para todo alfa).

α	TMC	#CA	$media_A$	σ_A	#CF	$media_F$	σ_F	#CG	$media_G$	σ_G
0	100	0	-	-	16	230,56	36744,52	33	1892,36	7961,43
0,5	100	4	3467,75	3809,96	16	239,19	45197,09	50	860,86	2340,56
1	100	15	283,13	227,78	16	239,19	45197,10	44	839,77	2929,21

Tabla 8 - Comparativa de resultados para TMC=100

α	TMC	#CA	$media_A$	σ_A	#CF	$media_F$	σ_F	#CG	$media_G$	σ_G
0	10	0	-	-	276	36,39	5379.39	345	200,30	1964,40
0,5	10	46	36,39	5379,38	276	36,39	5379,38	378	135,92	890,35
1	10	375	33,86	69,22	276	36,39	5379,39	564	87,66	838,81

Tabla 9 - Comparativa de resultados para TMC=10

Se puede concluir que cuanto más peso se le de a la compacidad (valores decrecientes de alfa) menos clusters de agricultura se forman y de menor tamaño. Esto es debido a que el uso ganadería puede ser asignado en cualquiera de los píxeles asignables y que el umbral de fósforo no es un limitante para este uso.

5.2.3 Pruebas de performance

Las variables que se dejan fijas toman el valor que se utilizó para las otras pruebas, salvo α que se decidió fijar en 0,5. La variable que se va modificando se marca con negrita.

α	**Iteraciones GRASP**	**Umbral de fósforo**	**Tamaño mínimo de clusters**	**Tiempo de respuesta (segundos)**
0	100	32174	100	249,8
0,5	100	32174	100	175,6
1	100	32174	100	1,6
0,5	**10**	32174	100	11,3
0,5	**100**	32174	100	175,6
0,5	**1000**	32174	100	2712,8
0,5	100	**11982**	100	73,9
0,5	100	**32174**	100	175,6
0,5	100	**62348**	100	193,5
0,5	100	32174	**100**	175,6
0,5	100	32174	**1000**	160,9
0,5	100	32174	**10000**	156,8

Tabla 10 - Pruebas de performance

Como se puede apreciar en la **Tabla 10**, las variables que tienen más impacto en la performance de la heurística son la cantidad de iteraciones GRASP, el umbral de fósforo y α.

El impacto de las iteraciones GRASP era previsible, ya que por cada iteración vuelve a generar una solución inicial y a hacerse una búsqueda local.

Al aumentar el umbral de fósforo las búsquedas locales pueden realizar una mayor cantidad de cambios que con un umbral más pequeño, y esto lleva a tener un tiempo de ejecución más largo.

Para $\alpha = 1$ la ejecución es más performante que para otros valores de α. Esto es debido a que al no tener en cuenta la compacidad no se realiza ninguna búsqueda local para mejorar la compacidad. Estas búsquedas son las que más

tiempo consumen en ejecutarse ya que hay muchas distribuciones diferentes que pueden dar una buena compacidad.

El tamaño mínimo de los clusters no afecta la performance, solo tiene impacto en la medida de compacidad.

5.2.4 Pruebas de búsquedas locales

Para entender los resultados de las tablas se explica el significado de las nuevas columnas.

Mejor aptitud solución inicial (MSI): Contiene el valor objetivo de la solución inicial que se generó con mayor valor de aptitud.

Mejora con búsqueda local (MBL): Esta columna contiene la mejora (valor objetivo) que se obtuvo al aplicar la búsqueda local a la solución inicial con mejor aptitud.

Aptitud solución inicial (AI) : Esta columna contiene el valor de la aptitud de la solución inicial que llevó a obtener la solución final.

Aptitud final (AF): Esta columna contiene el valor de la aptitud luego de realizadas las búsquedas locales a partir de la solución inicial que llevó a obtener la solución final.

$\alpha = 0$

B	MSI	MBL	AI	AF	Compacidad	FH	Exacto (Aptitud)
11982	42261,86	42261,86	26960,32	26960,32	0,96	11530,12	**42262**
32174	44744,73	44744,73	42400,00	42095,25	0,80	12469,61	**48894**
50000	44045,26	43447,09	42496,65	42215,57	0.80	12757,33	**54424**

Tabla 11 - Búsquedas locales para $\alpha = 0$

Al no tener en cuenta la compacidad se puede apreciar que por más que exista una solución inicial con buena aptitud, esta no es la que genera el resultado.

$\alpha = 0,5$

B	MSI	MBL	AI	AF	Compacidad	FH	Exacto (Aptitud)
11982	42261,86	42261,86	38794,63	38866,026	0,80	11981,99	**42262**
32174	44744,73	48631,65	43226,81	48722,38	0,76	32172,88	**48894**
50000	44045,26	44045,26	43165,58	54275,94	0,65	49999,15	**54424**

Tabla 12 - Búsquedas locales para $\alpha = 0,5$

Cuando se tienen en cuenta tanto la aptitud como la compacidad se ve que la solución inicial que lleva al mejor resultado tiene una aptitud cercana a la mejor aptitud inicial.

$\alpha = 1$

B	MSI	MBL	AI	AF	Compacidad	FH	Exacto (Aptitud)
11982	42261,86	42261,86	42261,86	42261,86	0,80	11981,90	**42262**
32174	44744,73	48631,65	43564,81	48824,34	0,67	32172,88	**48894**
50000	44045,26	52951,34	42917,13	54272,95	0,64	49999,14	**54424**

Tabla 13 - Búsquedas locales para $\alpha = 1$

Si no se tiene en cuenta la compacidad, la solución inicial que lleva al mejor resultado tiene una aptitud prácticamente igual a la mejor aptitud inicial.

5.2.5 Pruebas de robustez

Para estas pruebas se utiliza la notación definida en el punto anterior.
Se modificaron las aptitudes de todos los píxeles de la siguiente forma:

aptitud = aptitud + (número aleatorio entre -0,5 y 0,5) * (aptitud * 0,2)

Esta perturbación modela las variabilidades y los eventuales errores que puede tener la estimación de la aptitud de cada media hectárea de terreno.

Todas las pruebas fueron realizadas para el umbral de fósforo 32173,85, que es el valor identificado en Barletta (2017), además las pruebas se hacen variando el parámetro α para observar si esta perturbación de datos tiene alguna influencia considerable tanto en la aptitud como en la compacidad de la solución.

$\alpha = 0$

MSI	MBL	AI	AF	Compacidad	FH	Datos Modificados
48776.93	48776.93	42268,82	42268,82	0,75	12004,32	SI
44744,73	44744,73	42400,00	42095,25	0,79	12469,61	NO

Tabla 14 - Comparativo entre datos originales y modificados para $\alpha = 0$

Datos	Aptitud	Compacidad	Fósforo
Originales	422	18354	77174
Modificados	22	18764	77164
Valor absoluto de diferencia	400	410	10

Tabla 15 - Asignaciones de usos para datos originales y modificados para $\alpha = 0$

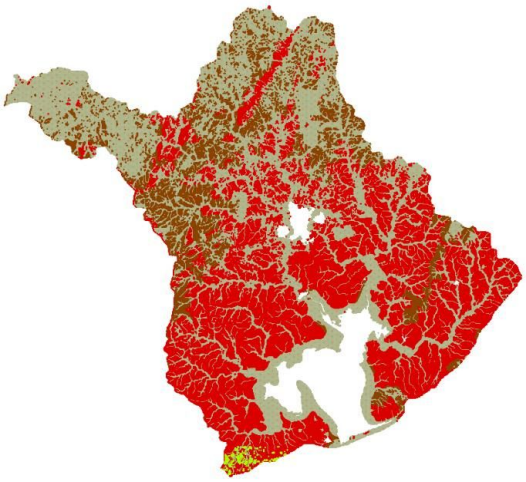


Figura 15 - Datos originales

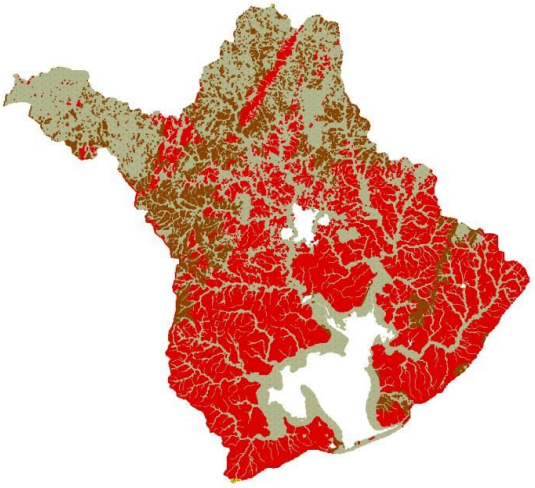


Figura 16 - Datos modificados

Para $\alpha = 0$ se observa que la alteración de datos no tiene influencia significativa en los resultados. Esto tiene sentido ya que no se tiene en cuenta la aptitud, por lo tanto las diferencias se dan sólo en las soluciones iniciales. Después los algoritmos de búsquedas locales van llevando a las soluciones a tener resultados similares en cuanto a la aptitud y compacidad.

A pesar de esto, para $\alpha = 0$ es cuando las cantidades de asignaciones por uso difieren más.

$\alpha = 0,5$

MSI	MBL	AI	AF	Compacidad	FH	Datos Modificados
48776,93	48879,83	46184,89	48285,53	0,64	32173.18	SI
44744,73	48631,65	43226,81	48722,38	0,74	32172,88	NO

Tabla 16 - Comparativo entre datos originales y modificados para $\alpha = 0,5$

Datos	Agricultura	Forestación	Ganadería
Originales	17113	18764	60073
Modificados	17116	18764	60070
Valor absoluto de diferencia	3	0	3

Tabla 17 - Asignaciones de usos de datos originales y modificados para $\alpha = 0,5$

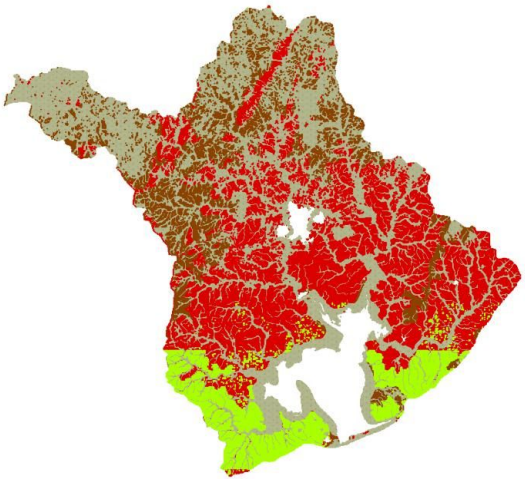


Figura 17 - Datos originales

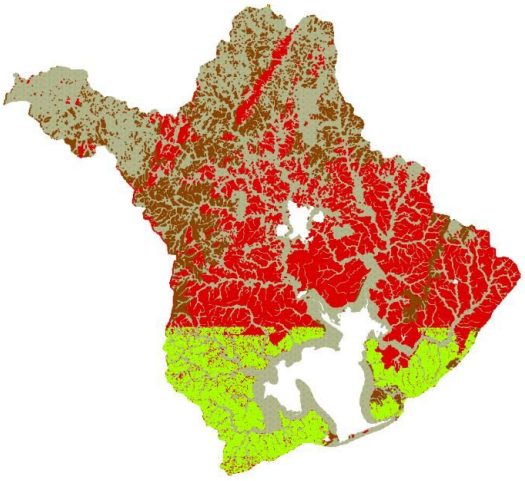


Figura 18 - Datos modificados

Para $\alpha = 0,5$, al tener en cuenta tanto la compacidad como la aptitud y partir de soluciones iniciales diferentes, las búsquedas locales generan una salida variada. Aunque la cantidad de píxeles por uso sea prácticamente igual, la distribución de esos píxeles es diferente, como se puede apreciar en la **Figura 17 y Figura 18**. Esto hace que la aptitud y compacidad final sean distintas.

En la **Figura 18** se puede ver como en los datos modificados quedan varios píxeles con el uso ganadería entre los píxeles de agricultura. Esto se puede ver más claramente teniendo en cuenta los datos de la **Tabla 17,** en donde los pixeles de agricultura en los datos modificados difieren con los de los datos originales sólo en tres y en los mapas se ve que el área verde es mayor en la **Figura 18** que en la **Figura 17**. Esto es porque en la **Figura 18** entre los píxeles de agricultura hay pixeles de ganadería.

$\alpha = 1$

MSI	MBL	AI	AF	Compacidad	FH	Datos Modificados
48634,63	48934,60	48634,63	48957,25	0,57	32173,37	SI
44744,73	48631,65	43564,81	48824,34	0,65	32172,88	NO

Tabla 18 - Comparativo entre datos originales y modificados para $\alpha = 1$

Datos	Agricultura	Forestación	Ganadería
Originales	17112	18764	60074
Modificados	17115	18764	60071
Valor absoluto de diferencia	3	0	3

Tabla 19 - Asignaciones de usos de datos originales y modificados para $\alpha = 1$

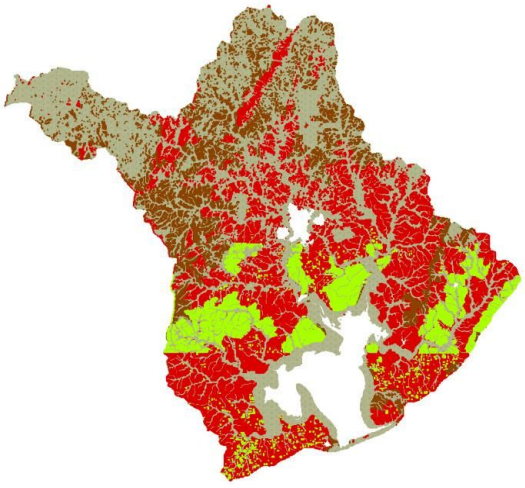


Figura 19 - Datos originales

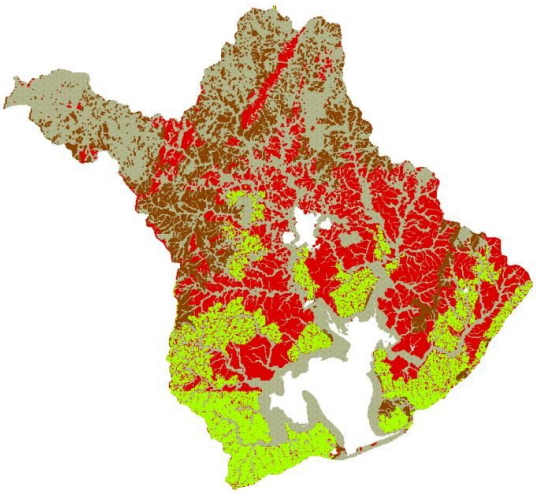


Figura 20 - Datos modificados

Al no tener en cuenta las búsquedas locales de compacidad, se puede apreciar cómo las soluciones finales tienen una aptitud similar. Aunque la asignación de píxeles de cada uso sea prácticamente igual, tal como ocurre para $\alpha = 0,5$, la distribución de los píxeles es diferente y esto lleva a que la medida de compacidad varíe considerablemente.

Para $\alpha = 1$ es el caso que se ven más diferencias en los mapas, esto tiene sentido porque no se tiene en cuenta la compacidad.

α	Aptitud en datos modificados	Aptitud en datos originales
0	42268,82	42095,25
0,5	48285,53	48722,38
1	48957,25	48824,34

Tabla 20 - Aptitudes de datos originales y modificados para 3 valores de α

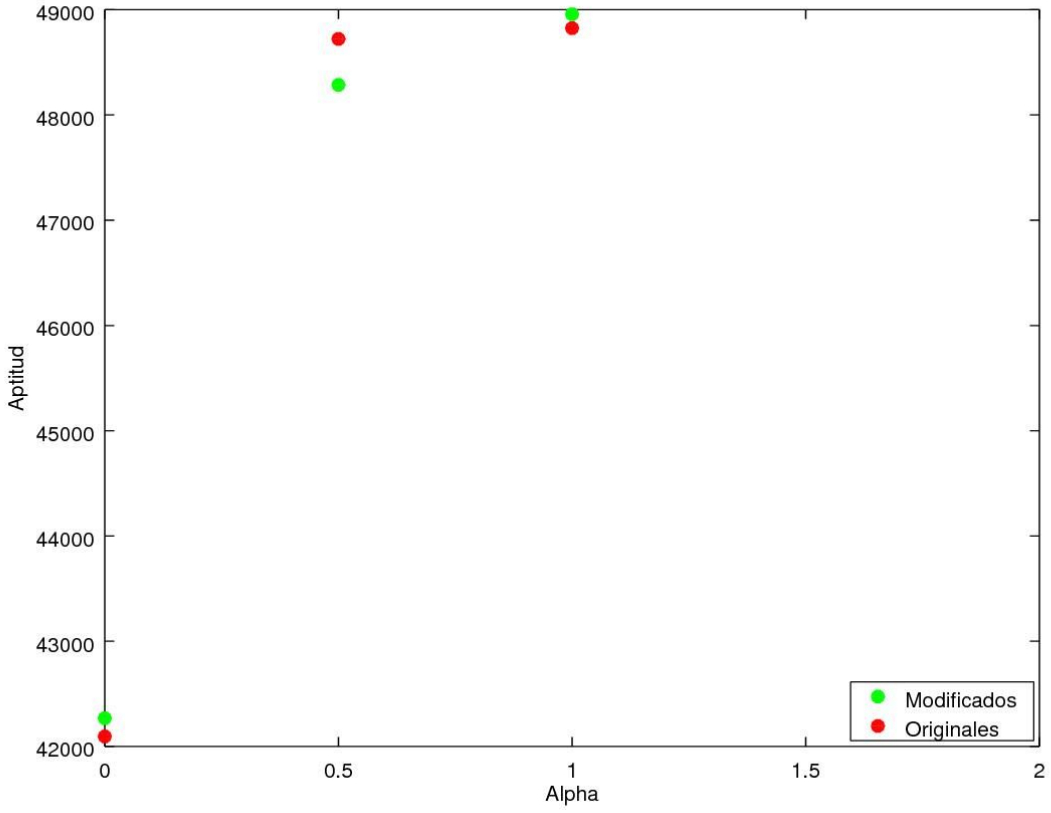


Figura 21 - Gráfica de aptitud de datos originales y modificados variando α

En la **Figura 21** se grafican las aptitudes al variar el valor de α para los datos originales y los datos modificados.

Como se puede observar, por más que se varíen levemente los datos de entrada, los resultados son similares en lo que a aptitud refiere. Por lo tanto, la variabilidad en la estimación del valor de aptitud en algunos píxeles no tiene influencia significativa en el resultado final. Esto implica que en un escenario real, en donde existe variabilidad, la solución entregada por el algoritmo va a ser similar en distintos escenarios.

5.2.6 Mapas

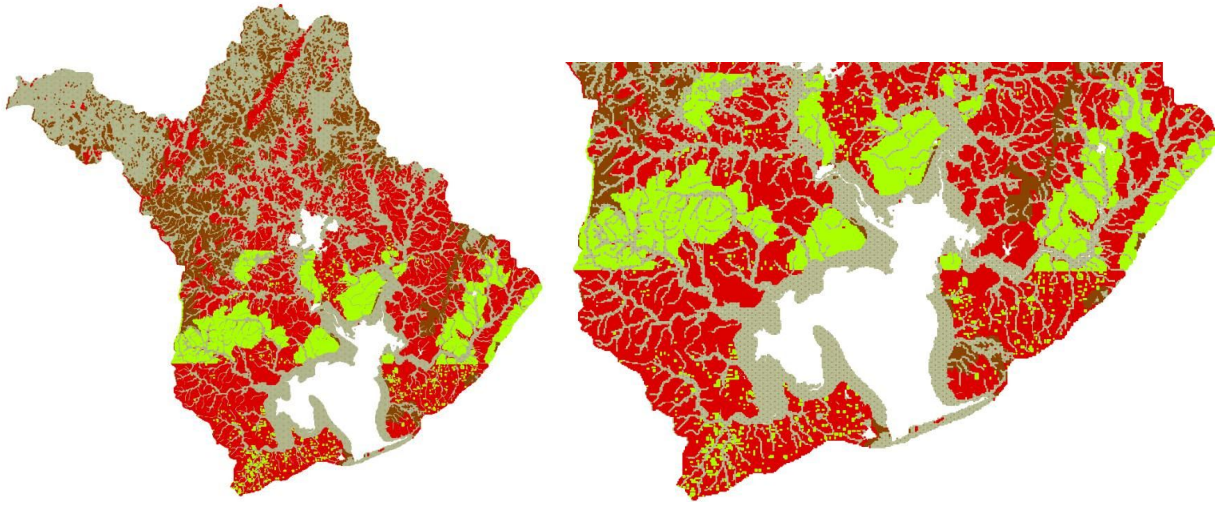
Para realizar los mapas se utilizó el programa QGIS (QGIS, 2016) con los archivos shapefile y dbf facilitados por Lorena Rodríguez-Gallego.

Para esas pruebas se varía el parámetro alfa, mientras que el umbral de fósforo fijado para las pruebas es de 32173,85 y TMC=100.

En los mapas mostrados a continuación, el verde refiere a la agricultura, el rojo a la ganadería y el marrón a la forestación.

Dado que en las búsquedas locales se parte de una posición aleatoria, pero luego a partir de esa posición se realizan las modificaciones aumentando en uno la posición, se ve que la asignación queda separada en franjas horizontales.

$\alpha = 1$



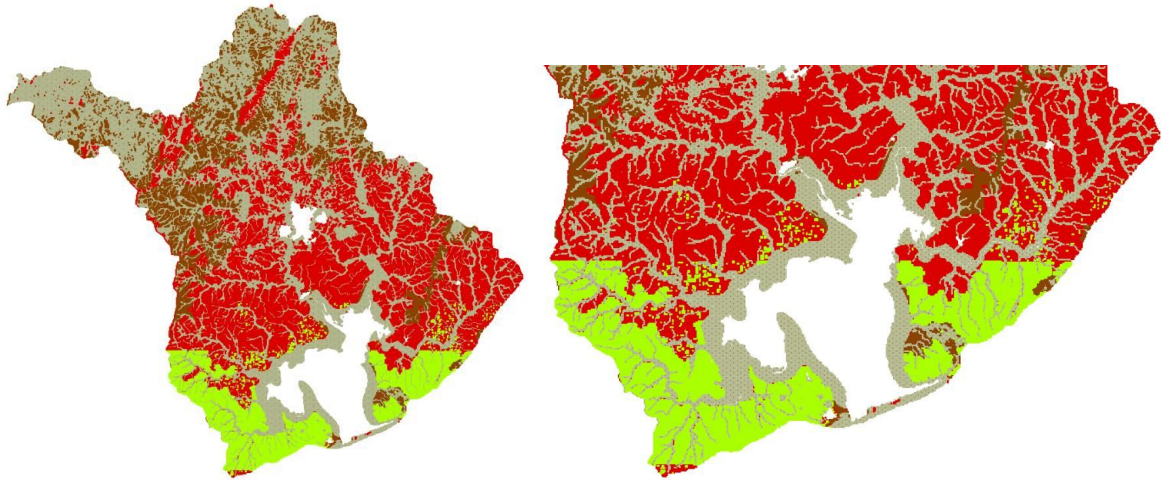
(a) **(b)**

Figura 22 - Mapas generados con $\alpha = 1$

Al no tener en cuenta la compacidad, la solución tiene muchos píxeles de agricultura dispersos, sobre todo en la parte inferior del mapa. Esto es porque el uso agricultura es el que tiene generalmente mayor aptitud.

En la **Figura 22 b** que una ampliación de la **Figura 22 a**, se puede apreciar como los píxeles de agricultura en la parte inferior de la imagen están dispersos.

$\alpha = 0,5$

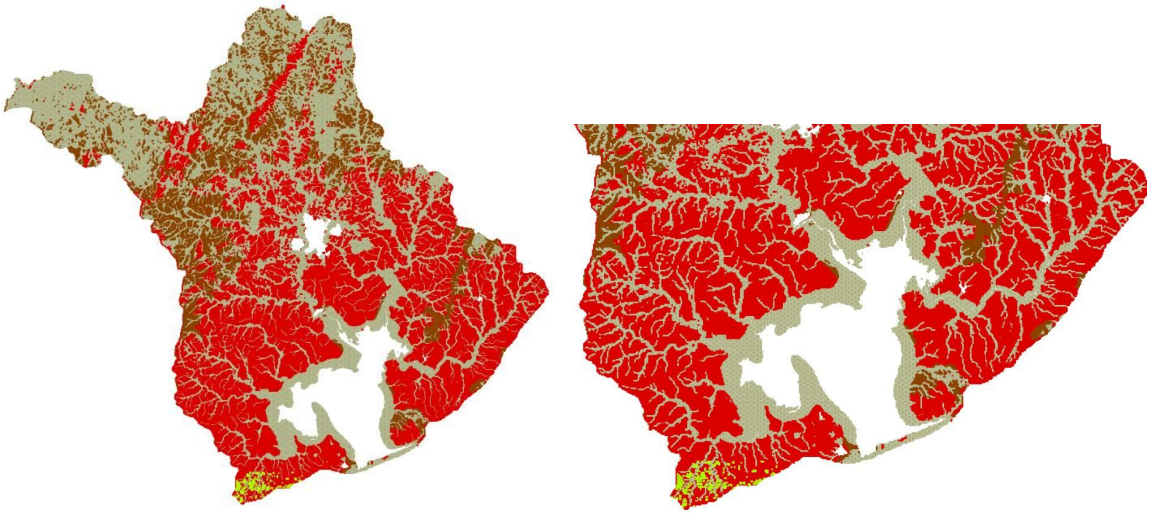


(a) **(b)**

Figura 23 - Mapas generados con $\alpha = 0,5$

Al tener en cuenta la compacidad y la aptitud, se puede apreciar como resulta una franja compacta de uso agricultura y algunos otros píxeles de agricultura dispersos. Esto sucede porque la aptitud también tiene peso, y por lo tanto es necesario que se asigne una buena cantidad de píxeles de agricultura.

$\alpha = 0$



(a) **(b)**

Figura 24 - Mapas generados con $\alpha = 0$

Cuando no se tiene en cuenta la aptitud, quedan muy pocos píxeles de uso agricultura y la mayoría de la asignación queda con el uso ganadería, ya que este uso es el único que se puede asignar completamente sin alcanzar el umbral de

fósforo y que es asignable en todos los píxeles.

En la **Figura 24 b** se pueden ver los pocos píxeles de uso agricultura que resultan. Aunque no se tenga en cuenta la aptitud no se llega a un 100% de compacidad. Esto se debe a que para considerar un grupo de píxeles como compacto se consideró que como mínimo sean 100 píxeles contiguos, y en la parte superior de los mapas se puede ver cómo hay grupos de menos de 100 píxeles separados por áreas no utilizables (marcadas en gris).

Capítulo 6 - Conclusiones y trabajo a futuro

6.1 Conclusiones

Se logró avanzar la investigación al siguiente nivel, que es obtener soluciones con una aptitud cercana a la óptima pero respetando una asignación espacial de forma compacta. Esto se logró mediante un algoritmo heurístico que incluye dos funciones de compacidad, una de alcance local y otra de alcance global. Los resultados dependen de la prioridad que se le da a los tres objetivos principales (aptitud, exportación de fósforo y compacidad).

Se observó un aumento de la compacidad sobre todo para valores de α bajos. Para valores de α altos la aptitud es cercana a la óptima, lo que demuestra un correcto funcionamiento del algoritmo. La aptitud se ve acotada por el umbral de fósforo. Cuanto más grande sea el umbral, mayor es la aptitud de la solución.

El tiempo de ejecución es razonable para un algoritmo de estas características.

Una de las dificultades encontradas fue la de cuantificar la compacidad. El principal desafío que presenta la compacidad es obtener una solución que tenga una cantidad de usos de suelo distribuida y no toda la compacidad dada por una asignación casi total de usos de suelo de tipo ganadería, ya que al medir la compacidad no se tiene en cuenta cuántos clusters hay de cada uso.

Mediante las pruebas realizadas se puede concluir que el algoritmo reacciona de forma correcta ante la variación de los datos de entrada, lo que muestra su robustez.

Para realizar este proyecto fue necesario primero una fase de formación, que llevó prácticamente el primer año entero. En esta etapa se realizó una primer aproximación al algoritmo final, en base a un prototipo con un conjunto acotado de datos.

Luego se realizó la implementación del algoritmo para el caso de estudio, donde se analizaron funciones de compacidad más complejas que en la primer aproximación. Esta etapa tuvo una duración aproximada de dos años.

La tercer etapa fue la de generación del informe. La misma comenzó durante la implementación del algoritmo y duró aproximadamente un año y medio.

6.2 Trabajo a futuro

Como trabajo a futuro se podría implementar el algoritmo con otras metaheurísticas (por ejemplo, Algoritmos Genéticos) y luego comparar los resultados.

Analizar el comportamiento del algoritmo en un entorno diferente al de la Laguna de Rocha. Los entornos diferentes podrían ser:

- La cuenca de otra laguna en donde se analicen los mismos usos de suelo.
- Otros usos de suelo en la misma cuenca. Por ejemplo diferenciando los distintos tipos de agricultura.

Investigar otras medidas de compacidad que tengan en cuenta una asignación más equitativa de los usos de suelo. Hacer pruebas con distintas ponderaciones para la compacidad local y global.

Considerar otros factores ambientales que puedan incluirse en las restricciones del modelo (además de la exportación de fósforo) y analizar las posibles modificaciones en el algoritmo para adaptarlo a esto.

Modificar el orden de recorrida de los pixeles en las búsquedas locales para que la asignación no quede en franjas horizontales.

Sería interesante realizar una interfaz más amigable al usuario, donde se pueda introducir los valores configurables y se obtenga como respuesta el mapa de asignación de uso de suelo, tablas y gráficas establecidas que sean de utilidad para el usuario final.

Referencias

Barletta A (2017) Modelos de optimización y multiatributo para la asignación de usos del suelo en la cuenca de la Laguna de Rocha. Tesina de Licenciatura en Ciencias Biológicas, Universidad de la República.

Blum C, Roli A (2003) Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys 35(3): 268–308.

CPLEX (2017). Ilog ampl cplex System, Version 8.0. https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

Echarri L (1998) Ciencias de la tierra y del medio ambiente. Barcelona : Editorial Teide, S.A.

Ehrgott M, Gandibleux X (2004) Approximative Solution Methods for Multiobjective Combinatorial Optimization. Top: Revista de la Asociación Española de Estadística e Investigación Operativa 12(1):1–89, 2004.

Feo T, Resende M (1995) Greedy randomized adaptive search procedures. Journal of Global Optimization 6:109–133.

Google Maps (2018)
https://www.google.com.uy/maps/place/Laguna+de+Rocha/@-34.62016,-54.364127 7,19918m/data=!3m2!1e3!4b1!4m5!3m4!1s0x957

Nin M, Soutullo A, Rodríguez-Gallego L, Di Minin E (2016) Ecosystem Services-based land planning for environmental impact avoidance. Ecosystem Services. 17: 172-184.

Martello S, Toth P (1990) Knapsack Problems - Algorithms and computer implementations

Pitsoulis L, Resende M (2002) Greedy randomized adaptive search procedure. In Handbook of Applied Optimization, P. Pardalos and M. Resende, Eds. Oxford University Press, 168–183.

QGis, D. (2016). Quantum gis geographic information system. Open Source

Geospatial Foundation Project. http://www.qgis.org/es/site/, 45.

Reeves C (1993) Modern Heuristic Techniques for Combinatorial Problems. Blackwell Scientific Publishing, Oxford, England.

Resende M, Ribeiro C (2010) Greedy Randomized Adaptive Search Procedures. In M. Gendreau and J.M. Potvin, editors,Handbook of Metaheuristics, International Series in Operations Research & Management Science, pp 283-319.

Rodríguez-Gallego L, Achkar M, Conde D (2012) Land suitability assessment in the catchment area of four Southwestern Atlantic coastal lagoons: multicriteria and optimization modeling. Environmental Management.

Sandoya F (2014) Matemática : Una publicación de FCNM – ESPOL
2014, Vol. 12, No.2 El problema de la mochila, complejidad, cotas y métodos de búsqueda eficientes.