UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

CENTRO DE FÍSICA APLICADA Y TECNOLOGÍA AVANZADA

MODELADO DE ESCENARIOS PROYECTIVOS DE COSECHA DE BIOMASA UTILIZANDO LA MÉTRICA DE DISTANCIA INVERSA PONDERADA Y CÓMPUTO DE ALTO RENDIMIENTO

TESIS

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN TECNOLOGÍA

Presenta:

Saúl Octavio Pérez Elizondo

Tutor:

Dr. Ulises Olivares Pinto

C F A T A

Juriquilla, Querétaro a enero de 2024

Agradecimientos

A mis padres por la confianza y el apoyo que me brindaron cuando decidí realizar mis estudios lejos de casa, las experiencias y conocimientos que adquirí en estos años son invaluables para mí, nada de esto habría sido posible sin ustedes y siempre estaré profundamente agradecido.

A mis abuelos que con su cariño me han impulsado siempre a perseguir mis metas y por brindarme su sabiduría cuando más me hizo falta.

A mi tía Mayra por darme su apoyo incondicional y por ser un modelo que seguir cuando mis objetivos se hacían confusos.

A Andrea por la confianza que me brindaban sus palabras de aliento, por su paciencia y cariño a lo largo de estos años.

A mi tutor por su dedicación y paciencia a lo largo de estos años, usted estuvo muy presente a lo largo de mi formación académica, formando parte importante de esta historia con sus aportes profesionales e influyendo en la dirección de vida profesional.

Se extiende un profundo agradecimiento a las entidades financiadoras que han hecho posible la realización de este proyecto. Este proyecto fue parcialmente financiado por los proyectos de investigación CONAHCYT: Ciencia de Frontera (191982) (UOP) y UNAM DGAPA-PAPIIT (TA 101022 y IA105920) (UOP).

Resumen

En el presente trabajo, se aborda la problemática derivada de la extracción insostenible de biomasa. Se introduce un enfoque innovador que combina el uso de la métrica de distancia inversa ponderada con técnicas de cómputo de alto rendimiento para modelar escenarios proyectivos de cosecha de biomasa. Este modelo tiene como objetivo proporcionar predicciones precisas sobre el impacto que dicha extracción tendrá en las comunidades de la región en cuestión. La relevancia de este estudio se fundamenta en la implementación del cómputo en paralelo, el cual optimiza significativamente el rendimiento de la métrica de distancia inversa ponderada sin comprometer su precisión. Esta mejora permite la evaluación de áreas geográficas más extensas y democratiza el acceso a la herramienta al facilitar su implementación mediante lenguajes de programación ampliamente utilizados.

Finalmente, se compararon algoritmos implementados en Rust, C++ y Dinámica EGO para modelar la cosecha de biomasa utilizando la métrica de distancia inversa ponderada. Las pruebas se realizaron en mapas de Kenia y Haití con variaciones en el número de comunidades y características geográficas. Los resultados validaron la precisión de los algoritmos propuestos y mostraron que las implementaciones en paralelo en C++ y Rust (propuestas en esta tesis) son significativamente más eficientes que Dinámica EGO, especialmente en escenarios con un mayor número de comunidades, lo que resalta la utilidad del cómputo en paralelo en la mejora del rendimiento sin comprometer la precisión.

Lista de Figuras y Tablas

Acrónimos

Abreviatura	Significado
CD	Costo Distancia
CPU	Unidad Central de Procesamiento
Dinamica EGO	Entorno para Objetos de Geoprocesamiento
GPU	Unidad de Procesamiento Gráfico
IDW	Distancia Inversa Ponderada
MoFuSS	Modeling Fuelwood Saving Scenarios
OpenMP	Open Multiprocessing

Índice

Capítulo 1.
Introducción

1.1 Antecedentes

En la actualidad los combustibles fósiles son la principal fuente para la generación de energía en el mundo, especialmente en países en vías de desarrollo que no tienen acceso a infraestructura que permita la explotación de fuentes sustentables de energía. En consecuencia, cerca del 86% del CO_2 anual que se emite en el planeta es generado por la quema de estos combustibles fósiles (Sánchez Barboza et al., 2018; Global Carbon Budget, 2021). Uno de los principales problemas asociados a este tipo de combustibles, es la cantidad de gases de efecto invernadero que producen, ya que estos contribuyen al cambio climático, esto tiene como consecuencia el incremento de la temperatura global y la afectación de diversos ecosistemas, aumento del nivel del mar y la productividad agrícola. Además de que las emisiones de CO_2 pueden tener un impacto negativo en la biodiversidad y el funcionamiento hidrológico (Romero Salvador, 2010; Useros, 2013).

A pesar de estas desventajas, es una práctica usual generar energía a partir de la quema de *biomasa*, que en este caso se define como la materia orgánica generada de árboles crecidos espontáneamente en tierras no cultivadas (Rodríguez Márquez, 2022). La biomasa puede usarse para obtener calor y electricidad, en cuyo caso, la extracción no renovable de biomasa puede causar deforestación en regiones altamente explotadas (Romero Salvador, 2010).

Esta situación representa una gran problemática en México ya que la mayor parte de las emisiones de CO_2 nacionales se deben a la quema de combustibles fósiles (Saynes Santillán et al., 2016), entre las cuales se encuentra como práctica común la quema de biomasa. Además de esto, la obtención de biomasa desmedida puede causar un problema de deforestación, siendo de especial interés para el país, ya que "cerca de 138.7 millones de hectáreas están cubiertas por algún tipo de vegetación forestal, superficie que corresponde al 70.6% del territorio mexicano" (Comisión Nacional Forestal, 2023), dificultando el control de esta práctica con áreas tan grandes.

Es necesario mencionar que la extracción de biomasa es considerada como un método no renovable cuando la tasa de extracción excede la tasa de recuperación natural de los ecosistemas o de la zona de donde se extrae; de este modo, se debe recurrir al cálculo de la tasa de deforestación que se efectúa mediante la Ecuación 1.

$$D = Tr - Te, \tag{1}$$

donde D es el índice de deforestación, Tr es la tasa de recuperación y Te es la tasa de extracción. Un dominio negativo en D implica que existe deforestación en la zona, mientras que un dominio positivo indica que existe una extracción sustentable de biomasa.

Los patrones de deforestación han sido ampliamente estudiados, especialmente para los casos donde la biomasa se representa a través de leña o carbón vegetal. Esto se debe a que en la mayoría de los países en vías de desarrollo se sigue utilizando estos recursos para cubrir actividades de subsistencia básicas (Rasoulkhani et al., 2018; Sutar et al., 2015).

La evidencia ha concluido que los patrones de deforestación presentan una alta correlación con una tasa de extracción no renovable de biomasa (Conservation & December, 2006; Manolis et al., 2019). Debido a la importancia de esta situación, se han realizado múltiples investigaciones desde distintas perspectivas para reducir los riesgos de deforestación de la zona y aminorar la quema/consumo de biomasa. Para este fin se han investigado desde las características que tienen los árboles a los que debería priorizarse su tala y así generar una nueva legislación cuyas restricciones promuevan una extracción renovable de biomasa vegetal (Manolis et al., 2019), hasta las estufas que se utilizan en la quema de biomasa, analizando el material, el diseño para ventilación y la energía que generan para determinar cuál de estos aspectos es más importante para que se requiera menos combustible a la hora de utilizarlas (Sutar et al., 2015).

Un elemento crucial para realizar estos estudios es el conocimiento de las zonas más propensas a la deforestación y cuáles son las comunidades que más influyen en este fenómeno. Para analizar este proceso es necesario recurrir al modelado de escenarios *proyectivos de biomasa*. Estos modelos predicen el impacto que tienen las comunidades en el medio ambiente en el que se encuentran debido a la recolección de biomasa para satisfacer sus necesidades, entendiendo que una comunidad está compuesta por un conjunto de personas que habitan una misma zona geográfica. Para llevar a cabo estos modelos, es necesario contar con la información real del terreno y de las comunidades existentes, de esta forma es posible obtener un estimado de cómo afectará el consumo de biomasa de estas comunidades en la zona estudiada, facilitando así una acción eficaz. Por este motivo, es imperante cuantificar la cantidad de biomasa no renovable extraída en un ecosistema y así predecir el impacto que esto causará (Ghilardi et al., 2016).

1.2 Planteamiento del problema

El cambio climático es una problemática actual que afecta de manera global, por lo que requiere medidas urgentes contra esto (Gernaat et al., 2021). Para esto se busca combatir la deforestación que es una de las principales actividades que contribuyen al cambio climático (Gatti et al., 2021; Ellwanger et al., 2020).

Por lo tanto, concientizar a las personas sobre las consecuencias de la obtención de energía a partir de fuentes no renovables es una actividad necesaria. Asimismo, debido a la gran extensión forestal que tiene nuestro país, se debe evitar la deforestación y la tala desmedida de árboles. Para conocer el alcance que podría tener esta mala práctica, se puede hacer uso de herramientas de modelado que ayuden a predecir el impacto de la deforestación en escenarios futuros (Ghilardi et al., 2016).

En relación a la prevención de deforestación y planeación de acciones urgentes contra este efecto, se han desarrollado herramientas de cómputo para estimar cuales son las zonas más propensas a sufrir deforestación, a partir de las comunidades de la región y la demanda de biomasa que estas requieren (Ferreira et al., 2019). No obstante, el consumo de energía de las comunidades es de naturaleza dinámica, pues depende de diversos factores como el clima o el aumento de población. Por lo que es necesario que el tiempo de procesamiento de herramientas, como Dinámica EGO, que realizan simulaciones sobre el potencial impacto ambiental en escenarios actuales y futuros, sea el menor posible para así generar resultados lo más acercados a la realidad.

En este sentido, uno de los principales problemas que presentan estas herramientas predictivas es el requerimiento excesivo de recursos y tiempos de cómputo muy elevados (semanas o meses). Dada la naturaleza iterativa del problema, se han buscado formas para optimizar el tiempo de procesamiento del modelo. En este sentido, el cómputo en paralelo representa una opción para el cálculo de la métrica la métrica de Distancia Inversa Ponderada, de ahora en adelante referida como **IDW** por sus siglas en inglés (Inverse Distance Weighted), pues permite realizar el cálculo de esta métrica en diversas comunidades al mismo tiempo. Al generar una herramienta más eficiente aplicando el método de cómputo paralelo, podría disminuirse el tiempo de procesamiento del modelo y es posible evaluar extensiones territoriales más grandes.

1.3 Objetivos

1.3.1 Objetivo general

Implementar un algoritmo eficiente para el cálculo de los algoritmos de costo-distancia y la métrica de Distancia Inversa Ponderada utilizando programación paralela, para reducir los tiempos de procesamiento del algoritmo.

1.3.2 Objetivos específicos

1. Construir una implementación serial en el lenguaje de programación C++ del algoritmo de Costo-Distancia (CD).
2. Desarrollar una versión serial del algoritmo de Distancia Inversa Ponderada (IDW) utilizando el lenguaje de programación C++.
3. Desarrollar una versión paralela de los algoritmos CD e IDW utilizando los lenguajes de programación C++ y RUST.
4. Implementar las funciones de Costo-Distancia e IDW en Dinamica EGO.
5. Evaluar y comparar desempeño de los lenguajes C++, RUST y Dinamica EGO usando estudios de caso reales.
6. Verificar y validar los resultados obtenidos del modelo.

1.4 Hipótesis

En términos de rendimiento, el uso de arquitecturas paralelas de memoria compartida en conjunción con optimizaciones de código disminuirá sustancialmente el tiempo de ejecución de las métricas de Costo Distancia e IDW.

Capítulo 2.

Marco Teórico

Para que el modelo sea lo más preciso posible es necesario tomar en cuenta la geografía de la zona, los requerimientos de biomasa de las comunidades circundantes y las distancias que tendrían que recorrer para obtener la biomasa. Para poder incluir estas variables en el modelo, primero se realizará el cálculo de la métrica de Costo Distancia y posteriormente el cálculo de la métrica de Distancia Inversa Ponderada, este proceso se deberá realizar para cada comunidad en el mapa y finalmente se sumarán para conocer el impacto de todas ellas en el mapa.

2.1 Métrica de Costo Distancia

Para poder calcular la métrica de Costo Distancia, es necesario contar con información muy precisa de la zona, para esto se hace uso de mapas de fricción. Estos mapas contienen información de la topografía de la zona como: pendientes, carreteras, ríos, cuerpos de agua, áreas naturales protegidas, entre otras. Estos mapas se obtienen mediante imágenes satelitales. (Ghilardi et al., 2016).

Una vez que se tienen los mapas de fricción es necesario realizar el cálculo de la métrica de costo distancia, esta métrica proporcionará una información similar a la del mapa de fricción, pero adicional a esto, los valores de cada módulo del mapa dependen de la distancia que hay hacia la comunidad, entre más lejos estén de la comunidad, mayor será el valor de costo distancia debido a los módulos que se tienen que recorrer para acceder a esa zona. Estos valores se obtiene acumulando el costo que supone atravesar cada módulo del mapa de fricción, como se muestra más adelante con las ecuaciones 2A y 2B, de tal forma que en el

mapa resultante se encuentra el esfuerzo que requiere una persona de esa comunidad para acceder a cada módulo del mapa, dada la naturaleza del terreno y el esfuerzo acumulado para llegar a esa zona (Ferreira et al., 2019; Ghilardi et al., 2016).

Esta diferencia entre los mapas de fricción y costo distancia se puede apreciar mejor con las siguientes figuras, para ambos mapas el color claro en verde representa los lugares más difíciles de atravesar, pero la Figura 2.1 estos valores solo dependen de la geografía del terreno, mientras que en la Figura 2.2 los valores dependen de la cercanía con la comunidad, que en este caso se encuentra en la parte superior del mapa.

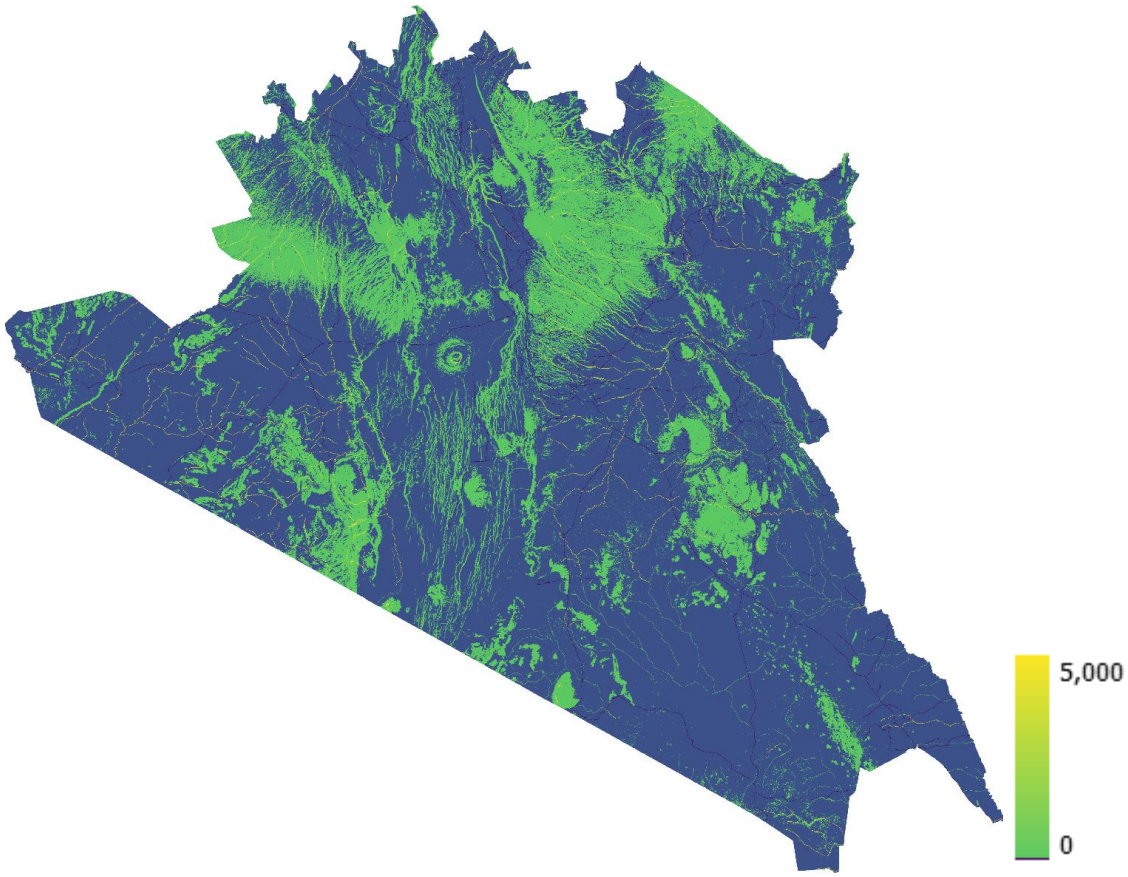


Figura 2.1 *Ejemplo mapa de Fricción Kenia*

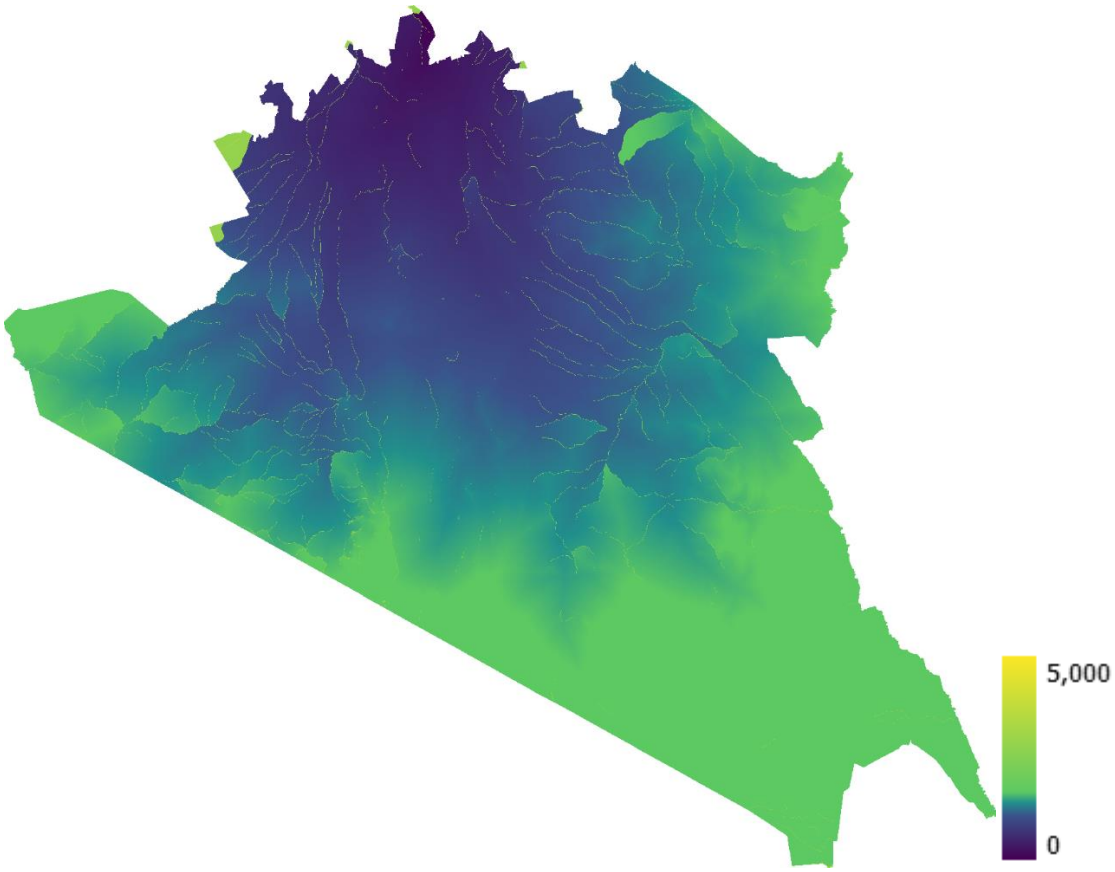


Figura 2.2 *Ejemplo mapa de Costo Distancia Kenia*

2.2 Métrica Distancia Inversa Ponderada

La información previamente calculada en la métrica de Costo Distancia se usa como base para crear un escenario proyectivo de biomasa utilizando la métrica de IDW que calcula la probabilidad de que una comunidad acuda a una región demográfica para satisfacer su requerimiento de madera (Lu & Wong, 2008). Para modelar la influencia que tienen las comunidades en la zona, es necesario realizar el cálculo de IDW para cada una de las comunidades dentro de una zona geográfica. El método de IDW, se basa en los valores asignados a un par de puntos determinados que están relacionados entre sí, pero su similitud

depende de la distancia entre estos. Y la influencia de un punto sobre el otro es menor entre mayor sea la distancia (Lu & Wong, 2008).

De esta forma, la métrica de IDW iniciará el cálculo con la información del costo que le representa a la comunidad obtener biomasa de cada zona del mapa en función de la distancia y la fricción del terreno. A partir de estos datos, el cálculo de la métrica de IDW muestra el efecto que ejerce el consumo de una comunidad sobre el paisaje, esto con el objetivo de caracterizar aquellas comunidades que ejercen más presión sobre un ecosistema y consecuentemente, las zonas con un alto riesgo de deforestación (Ghilardi et al., 2016; Lu & Wong, 2008). Por ejemplo, conociendo los valores nominales de una superficie que representen el esfuerzo necesario para atravesarla, se podría calcular la métrica IDW para estimar la probabilidad de que una persona atraviese determinadas zonas de la superficie, de este modo, entre más alto sea el valor mayor será la probabilidad de que esto suceda (La Torre & Rozo, 2016).

Dada su eficiencia en modelos espaciales predictivos, la métrica de IDW ha sido ampliamente utilizada para hacer predicciones en distintos ecosistemas, por ejemplo, modelar las características de cuencas hidrográficas, como el uso del suelo y la cobertura del suelo, y analizar cómo afectan estas a la propagación de la corriente (Peterson et al., 2017). También se ha usado para analizar cómo afecta la topografía y los patrones espaciales del paisaje a la distribución de la precipitación (Zhang et al., 2017), así como para crear diagnósticos de las condiciones térmicas de los ambientes y mapas de la variabilidad espacial de los índices de confort térmico (Oliveira et al., 2019). En todos estos casos, se declaran las variables que van a afectar al modelo y con ayuda de estas, se establecen puntos de interés

usando mapas espaciales de la zona a estudiar, para que al utilizar la métrica de IDW, se pondere la distancia de estos puntos y se genere una predicción con base en eso.

2.3 Plataforma geoespacial Dinamica EGO

En la actualidad existen múltiples plataformas geoespaciales que permiten modelar el impacto de la extracción no renovable de biomasa, una de ellas es *Dinamica EGO*. A través de esta plataforma es posible crear modelos computacionales para simular la tasa de extracción de biomasa, tomando como base el consumo de recursos en comunidades sobre casos de estudio particulares (Ferreira et al., 2019; Soares et al., 2002; Yi et al., 2012). Fue creada en 1998 y está conformada de un conjunto de componentes implementados en C++ llamados functores, cada uno con distintos operadores básicos, de tal forma que para codificar un algoritmo en Dinamica EGO es necesario seleccionar y unir functores por medio de flechas en la interfaz, a manera de diagrama de flujo como se observa en la Figura 2.3 (Ferreira et al., 2019).

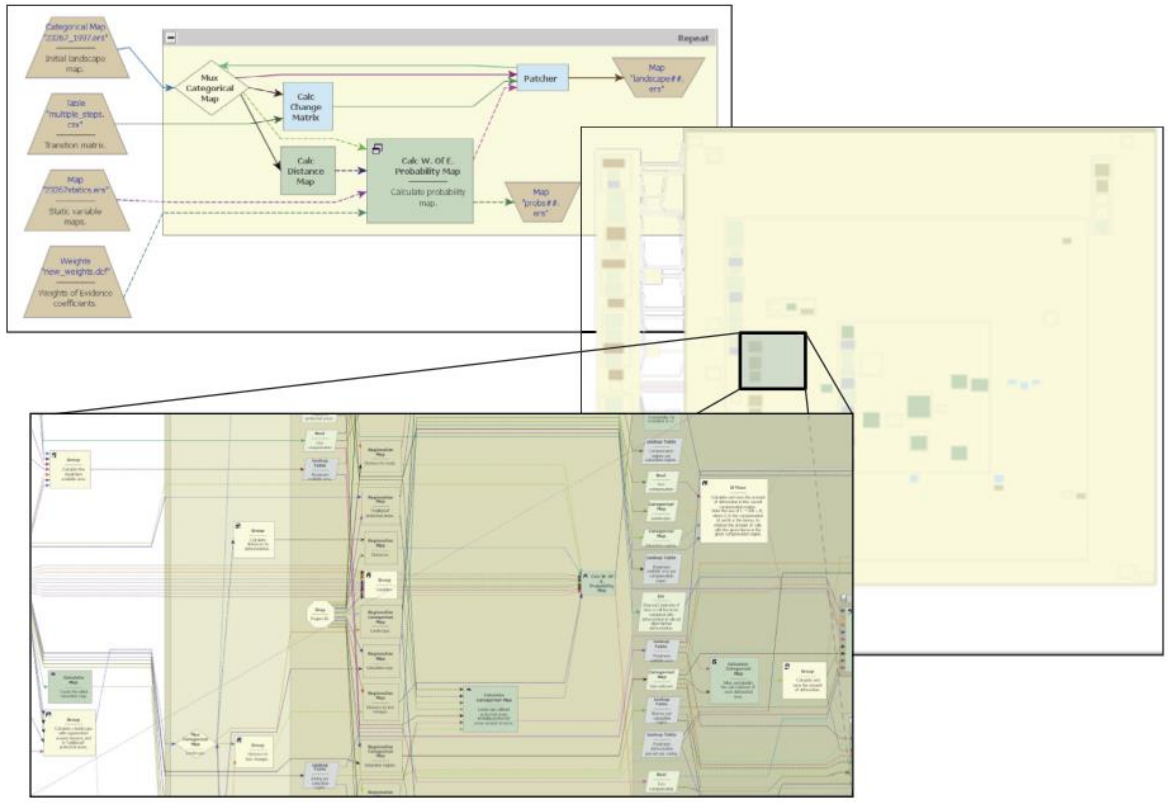


Figura 2.3 *Ejemplo de algoritmo elaborado con functores en Dinamica EGO*

Fuente: Ferreira et al. (2019)

Para calcular la métrica de IDW en Dinamica EGO es necesario cargar los datos de entrada del modelo, primero se importa un mapa de fricción de la zona a evaluar, la ubicación de las comunidades y su consumo de biomasa. Para calcular el mapa de fricción de la zona se utiliza la plataforma MoFuSS, en esta se encuentran imágenes geoespaciales para hacer proyecciones sobre la explotación de recursos forestales (Ghilardi et al., 2016). De MoFuSS se obtienen dos imágenes, una con el mapa de fricción y otra con la ubicación de las comunidades de la zona, con esta información se obtiene para cada comunidad la métrica de Costo Distancia y finalmente se realiza el cálculo de la métrica de IDW.

Además de Dinamica EGO, es posible utilizar otras plataformas geoespaciales para calcular la métrica de IDW como son Surfer, GS+ o QGIS (Oliveira et al., 2019). Haciendo una comparación entre estas, es posible catalogarlas dependiendo de la calibración que le permiten al usuario. En algunos casos, se cargan de forma automática ciertas variables que facilitan el uso de la plataforma, mientras que en otras, es necesario establecer manualmente cada parámetro de la simulación (Jean-François et al., 2010). Es aquí cuando resalta la mayor virtud de Dinamica EGO, su flexibilidad, ya que al estar pensada como un conjunto de módulos (functores) que interactúan entre ellos para generar el modelo, es posible para el usuario ajustar el modelo tanto como sea necesario. Resaltando también que es una herramienta gratuita y que brinda la posibilidad de trabajar en conjunto con una amplia gama de software tales como Fragstats, Arc GIS, Maxent y R (Espinoza, 2016).

2.4 Paralelización

Actualmente en la mayoría de los ámbitos de investigación es necesario llevar acabo el procesamiento de un amplio volumen de datos con tiempos de cómputo muy altos (Acosta & Segura, 2012; Montero & Antunez, 2011; Weinbach, 2008). Por tal motivo, la paralelización se ha convertido en una de las herramientas de cómputo de mayor relevancia; para realizar esto, es necesario contar con equipos de cómputo y lenguajes de programación que soporten las librerías para el desarrollo de aplicaciones en paralelo (Weinbach, 2008).

La evolución del hardware computacional ha dado lugar a arquitecturas de cómputo paralelo, creando procesadores capaces de ejecutar varios procesos simultáneamente (Weinbach, 2008). En sus inicios, los procesadores solo eran capaces de ejecutar una instrucción por cada ciclo de reloj, pero con la creación de procesadores multinúcleo, fue

posible el desarrollo de aplicaciones de alto nivel en paralelo (Montero & Antunez, 2011). Esto resulta tan relevante que en la actualidad los fabricantes de chips ya no se enfocan en aumentar la capacidad de sus componentes por medio de aumentar la velocidad de los ciclos de reloj que tienen, si no que se han agregado núcleos adicionales para poder realizar más tareas simultaneas (Weinbach, 2008).

En relación al software, las tareas de un programa se dividen en procesos paralelos mediante el uso de hilos; un hilo es una sección de código que puede ser planificada para ejecutarse simultáneamente, los hilos comparten un espacio de memoria entre ellos y el número que se puede utilizar al mismo tiempo están relacionado con los núcleos del equipo que se está utilizando (Montero & Antunez, 2011).

Para poder generar un modelo eficiente es importante crear un programa que pueda ser ejecutado sobre distintas arquitecturas paralelas y hacer un buen manejo de los procesos paralelos en el desarrollo del software. En este sentido, la decisión más importante a tomar es la herramienta que se va a utilizar, por esto sería necesario comparar distintos lenguajes de programación, tomando en cuenta el funcionamiento de sus librerías para cómputo paralelo, los tiempos de ejecución y los recursos de cómputo que utilizan.

En la actualidad se están realizando esfuerzos para eficientar el procesamiento de modelos geoespaciales, como la métrica de IDW. Aunque las implementaciones admiten distintas escalas en los mapas iniciales, el cálculo para un dominio a gran escala requiere una cantidad masiva de ciclos, lo que limita su uso (Xia et al., 2011). Los trabajos recientes se enfocan en aprovechar la máxima capacidad de las arquitecturas computacionales modernas, que pueden dividirse en dos enfoques, computación paralela basado en GPU y basado en CPU. Ambos enfoques hacen uso de cómputo paralelo y a pesar de que la GPU fue creada

como una unidad de procesamiento gráfico, actualmente es un procesador paralelo de uso general cuyo rendimiento tiende a ser mejor, pero es necesario un equipo más especializado (Xia et al., 2011; X. Zheng et al., 2016; Hofierka et al., 2017).

2.5 Herramientas de programación

En este sentido, se decidió trabajar con el enfoque en CPU debido a su uso más general, ya que esta no requiere un equipo de cómputo más especializado. Para esto se destacan dos lenguajes compilados C++ y Rust, ya que su rendimiento es muy similar (Brandefelt & Heyman, 2020), además de que ambos cuentan con librerías para paralelizar sus procesos, OpenMP [A] para C++ y Thread channels, mpsc [B] para RUST.

El lenguaje de programación C fue creado a mediados de los años 70, es un lenguaje estructurado que ofrece herramientas de control de flujo, estructuras sencillas y un amplio conjunto de operadores lógicos, uno de los objetivos en el diseño de algoritmos en C es que se requiera poco código para representar elementos de la vida real, y que este corra sin intervención en tiempo de ejecución (Vergara, 2017). Esto hace que los tiempos de ejecución de programas en este lenguaje sean tan eficientes, sin embargo, se presenta la dificulta en el desarrollo del software, ya que es difícil encontrar posibles errores que se presenten en tiempos de ejecución debido a la libertad que este ofrece (Brandefelt & Heyman, 2020).

A partir del lenguaje C se crea una extensión llamada C++ en 1980, la principal diferencia es que C está orientado a la programación estructurada, mientras que C++ está orientada a objetos, esta extensión se usa cuando el rendimiento del programa es importante al ser más rápido y ligero (Vergara, 2017).

El desarrollo en paralelo en C++ se realiza con la librería OpenMP, cuyas siglas en ingles significan Open Multi-Proccessing, el cual se refiere a la ejecución de un multiprocesamiento de código abierto. Esta librería contiene un conjunto de instrucciones que ayudan a controlar el flujo de procesos entre los hilos y la sección de memoria que comparten (Aslla et al., 2011). Esta tarea es fundamental en la programación en paralelo, ya que un error muy común es manejar incorrectamente la concurrencia de los hilos; esto se presenta cuando varios hilos intentan acceder a una localidad de memoria al mismo tiempo, y en consecuencia se producen errores al momento de leer o escribir en la memoria, como sobrescribir e invalidar el proceso de otros hilos (Weinbach, 2008).

A pesar de las ventajas anteriormente mencionadas del lenguaje C y su extensión C++, resultan ser ineficientes al tomar en cuenta las arquitecturas de cómputo paralelo actuales, por lo que es pertinente considerar lenguajes de programación más recientes, como Rust.

Rust se creó en 2010 y posee varias similitudes con C++, como su gestión de memoria, la sintáctica de las instrucciones, polimorfismo, entre otras (Castillo, 2007). Pero entre lo más destacable es que es un lenguaje multiparadigma, haciéndolo adecuado para aplicaciones que requieran un alto rendimiento; además de esto, es un lenguaje fuertemente tipado, con gestión manual de memoria, cuya principal ventaja en programación paralela es evitar errores de concurrencia (Arroyo, 2019).

Rust está diseñado para tener mayor seguridad en el acceso a la memoria, esto también se ve reflejado en la librería Thread channels mpsc, ya que esta brinda herramientas similares a OpenMP, pero con la ventaja de que el compilador se asegura de que exista solo una

referencia a la memoria para escribir pero que esta pueda ser leída en cualquier momento (Sydow et al., 2020).

Con base en los antecedentes previamente expuestos, se considerará utilizar ambos lenguajes de programación para calcular la métrica de IDW. Además de que en la práctica se tiene evidencia de un rendimiento similar entre estos lenguajes, declarando que C++ es un poco más veloz en tiempos de procesamiento, pero siendo Rust el que requiere menos esfuerzo para generar el programa debido a las medidas de seguridad que implementa contra la concurrencia (Castillo, 2007).

Capítulo 3.
Metodología

3.1 Procedimiento

Se obtuvieron los mapas de imágenes satelitales, la información del consumo de biomasa de las comunidades y su ubicación, después con ayuda de la plataforma MoFuSS se calcularon los mapas de fricción de la zona, además de codificar las métricas de Costo Distancia e IDW en Dinámica EGO (Ghilardi et al., 2016).

Posteriormente se codificaron las métricas de costo distancia e IDW en C++ y se compararon con los valores obtenidos en Dinamica EGO. Los resultados eran idénticos para mapas de una cuarta parte del tamaño de los presentados en los resultados con únicamente dos comunidades. A continuación se paralelizo el código en C++ y se realizó exitosamente la misma comparación de los resultados variando el número de hilos usados en el proceso para evaluar posibles errores de concurrencia. Finalmente se repitió el mismo proceso con la codificación en Rust serial y paralelo.

Ya que se tuvo la certeza de la validez del resultado, se tomaron los tiempos de ejecución y se optimizó el código para reducir dichos tiempos en ambos lenguajes, cambiando las estructuras de datos usadas, variando las librerías de programación en paralelo y cambiando la estructura general del código.

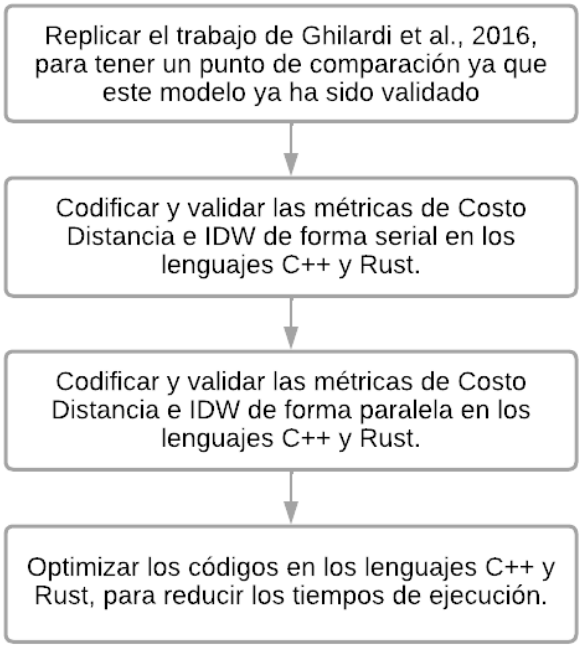


Figura 3.1 *Flujo general del desarrollo y optimización del modelo de IDW*

Para calcular la métrica de IDW se puede dividir el proceso en dos componentes; la obtención de los mapas de información espacial, que a partir de imágenes satelitales se crea el mapa de fricción en la plataforma de Mofuss (Ghilardi et al., 2016), y el modelo de IDW en el a partir de estos mapas calculados anteriormente, se creará un mapa general de IDW con el efecto que ejerce el consumo de biomasa de las comunidades sobre el paisaje. Los mapas de información geoespacial solo se calculan una vez, mientras que el modelo de IDW se calcula para cada comunidad en los mapas.

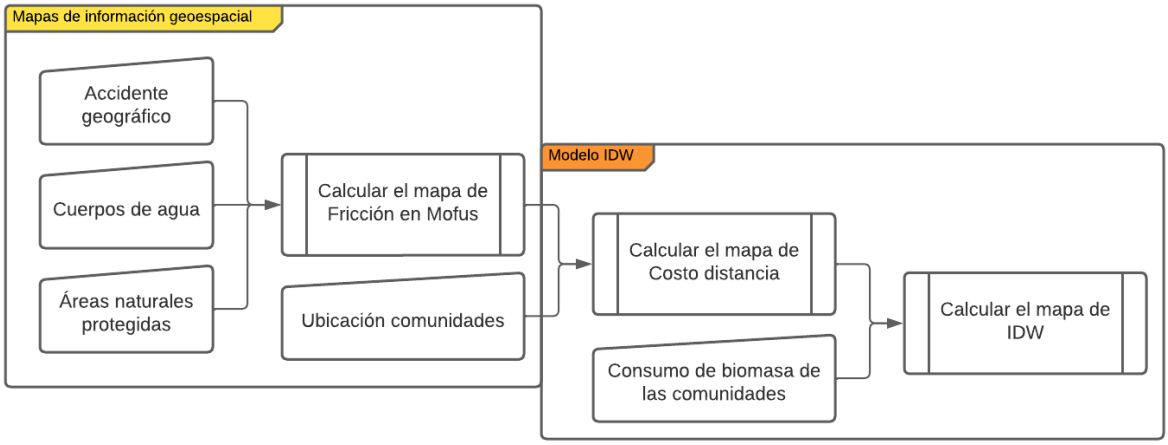


Figura 3.2 *Componentes de la métrica de IDW*

3.2 Adquisición de datos

Antes de iniciar el cálculo de las métricas, se deben importar las imágenes de los mapas de fricción en C++ con ayuda de la librería GDAL (Qin et al., 2014) y en Rust con su comando *tiff decoder* (Klabnik, 2022). Todas las imágenes de entrada deben estar en formato TIFF debido a que la principal ventaja de este formato de compresión es que no tiene perdidas en la calidad de la imagen (Yepes & Gay, 1994), esto es muy importante porque en la imagen del mapa de fricción cada pixel tiene el valor de la fricción en ese cuadrante.

Las imágenes importadas se guardan en una matriz de las mismas dimensiones, donde el valor del píxel se almacena en su correspondiente posición en una celda. Para el caso de la imagen de fricción, cada celda de la matriz resultante contendrá el esfuerzo necesario para atravesar esa zona del mapa, mientras que los lugares que son inaccesibles para las personas (e.g. cuerpos de agua o zonas protegidas) tendrán un valor nulo. En el caso de la imagen con las ubicaciones de las comunidades, en la celda en la que se encuentre cada una se guardará el número de la comunidad y el resto serán valores nulos.

A continuación, se debe implementar la primera optimización al código, ya que dependiendo de cómo se almacena la matriz, el tiempo que le toma al programa leerla puede variar. Si se utiliza en un arreglo de dos dimensiones la lectura será más lenta en comparación a un arreglo lineal, también conocido como matriz vectorizada (Besard et al., 2019).

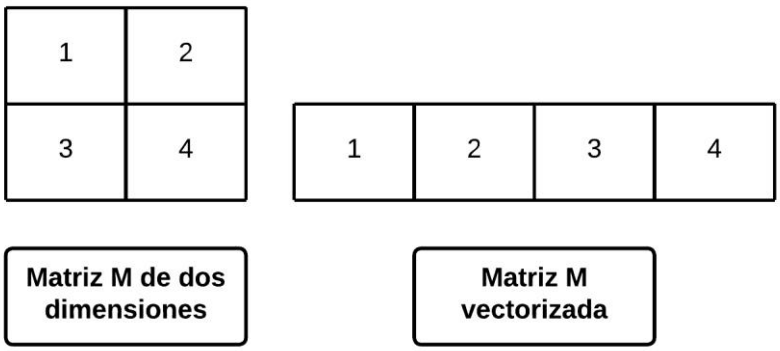


Figura 3.3 *Implementación Matriz vectorizada para almacenamiento de datos*

Además de las imágenes se debe importar un archivo con los requerimientos de biomasa de las comunidades. De esta forma, se cuenta con una matriz con el mapa de fricción y se almacena en una estructura de datos la información de las comunidades, tales como, el número de ésta, su requerimiento de biomasa, la columna y renglón en la que se encuentra.

3.3 Codificación de las métricas

Como se mencionó anteriormente, el cálculo de costo distancia y de IDW se realizan para cada comunidad, de tal forma que el código itera sobre la estructura de datos en la que se almacena la información de las comunidades, donde se encuentra la ubicación y los requisitos de biomasa para todas las comunidades en la región, como se muestra en el diagrama de flujo general de la Figura 3.4.

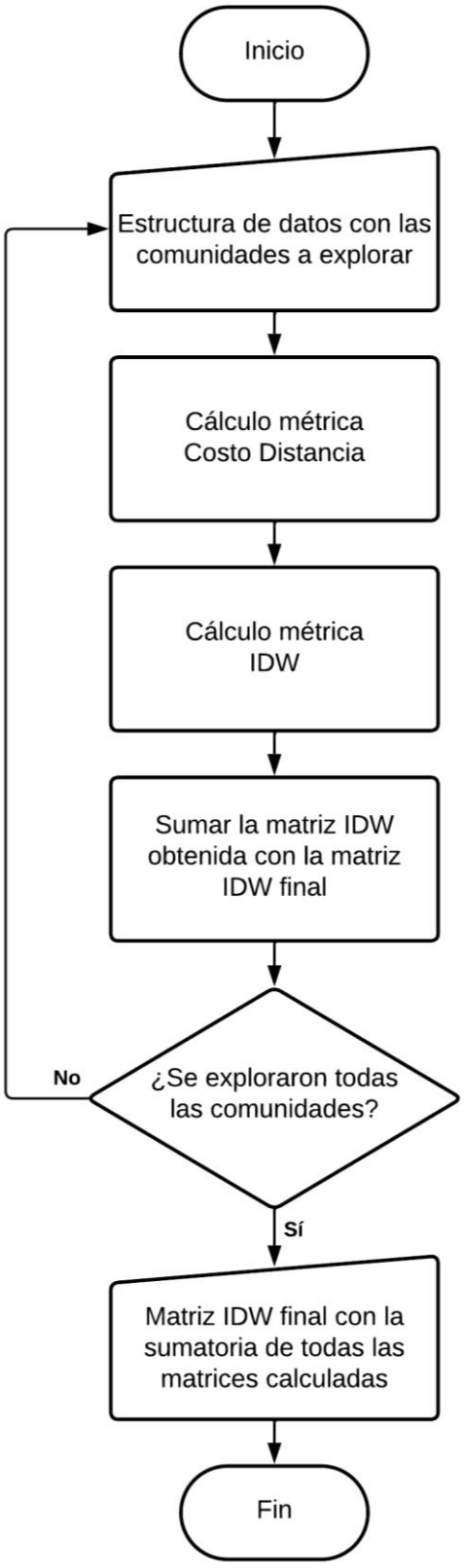


Figura 3.4 *Diagrama de flujo general del programa*

3.3.1 Costo Distancia

Para evaluar la cantidad de biomasa no renovable a través de un modelo, es necesario conocer la ubicación espacial de las comunidades, una demanda asociada a cada comunidad y un mapa de fricción de la zona. Las imágenes con los mapas de fricción que se usarán denotan el esfuerzo (costo) que se requiere para recorrer un píxel de una imagen georreferenciada. Los valores de fricción dependen de la topografía de la zona y se incluyen variables tales como: Pendientes, carreteras, ríos, cuerpos de agua, áreas naturales protegidas, entre otras (Ghilardi et al., 2016).

A partir del mapa de fricción es necesario realizar el cálculo previo de la métrica de costo distancia, esta métrica representa el esfuerzo requerido para obtener los recursos que necesita la comunidad, se obtiene acumulando el costo de los pixeles explorados del mapa de fricción, de tal forma que en cada pixel del mapa resultante se encuentra el esfuerzo acumulando de los pixeles que se tuvieron que explorar para llegar a cada uno (Ferreira et al., 2019; Ghilardi et al., 2016). Para calcular el costo del movimiento a las celdas circundantes se consideraran dos ecuaciones según el tipo de movimiento que se realice, para los movimientos laterales se usará la ecuación 2A, y para los movimientos en diagonal se usará la ecuación 2B (Ghilardi et al., 2016).

$$CD = \frac{Ca+Cn}{2} + CT, \tag{2A}$$

$$CD = \sqrt{2}\frac{Ca + Cn}{2} + CT, \tag{2B}$$

donde CD es el valor de costo distancia de la celda, Ca es el valor de Fricción de la celda actual, Cn es el valor de fricción de la celda circundante y CT es el valor de costo distancia total, que es el acumulado del costo distancia de las celdas ya exploradas, de tal forma que entre más nos alejemos de la comunidad mayor será este valor, teniendo como valor inicial cero.

Esta métrica de costo distancia se calcula para todo el mapa de fricción y para cada comunidad, debido a que los valores de cada pixel dependen de la ubicación de la comunidad; estos se calculan con la ecuación 2A para los movimientos laterales y con la ecuación 2B para los movimientos diagonales. Ambas incluyen el costo acumulado que se tenga al momento de explorar la nueva celda, por lo que primero se realiza el cálculo de costo distancia y posteriormente el de IDW, con el que finalmente obtendremos la probabilidad de que las personas de la comunidad acudan a ciertas zonas para satisfacer su demanda de biomasa.

El método de costo distancia recibe como parámetros de entrada la matriz de fricción y la información de la comunidad en la estructura de datos. A partir de la ubicación de la comunidad en la matriz se comenzará a explorar las celdas circundantes como se muestra en la Figura 3.5.

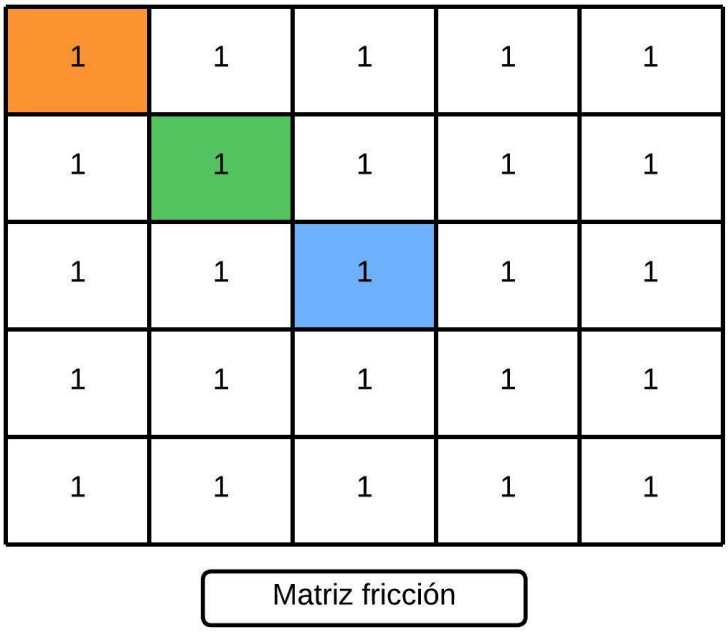


Figura 3.5 *Exploración celdas circundantes con el método costo distancia*

La exploración comienza con la celda marcada en rojo y calcula el valor de costo distancia de las celdas circundantes en sentido horario. Para la primera iteración marcada en azul, el valor de CT será de cero y el valor de Cn será el marcado con el mismo color de la iteración en la matriz de fricción. Para las celdas calculadas con la ecuación 2A, el valor de Costo distancia será de 1, mientras que para las celdas calculadas con la ecuación 2B, el valor de Costo distancia será de $\sqrt{2}$. Para la segunda iteración marcada en verde, se tomará el ultimo valor más grande guardado como CT y su ubicación en la matriz será la celda inicial, el valor de Cn será el marcado con el mismo color de la iteración en la matriz de fricción. Para las celdas calculadas con la ecuación 2A, el valor de Costo distancia será de $\sqrt{2} + 1$, mientras que para las celdas calculadas con la ecuación 2B, el valor de Costo distancia será de $2\sqrt{2}$. Estos nuevos valores solo se almacenarán en caso de que la celda no haya sido explorada o si el nuevo valor calculado es menor al que actualmente tiene asignado esa celda. Para la tercera iteración marcada en naranja se repetirá el mismo proceso, pero como los nuevos valores calculados no son menores a los que ya se tienen almacenados, no habrá cambios en la matriz de costo distancia.

Este procedimiento se realiza para todo el mapa, de tal forma que en la matriz solo se encontraran los menores valores de costo distancia posibles para cada celda. Esta exploración se explica en el siguiente pseudocódigo de la Figura 3.6.

1: $MatrizCD$
2: $CeldasPorExplorar[i] \leftarrow$ Ubicación de la comunidad
3: $CostoAcumulado \leftarrow 0$
4: $CD \leftarrow 0$
5: **while** $CeldasPorExplorar[i]$ no vacío **do**
6: $CostoAcumulado \leftarrow$ Valor fricción de $CeldasPorExplorar[i]$
7: **for** 8 celdas circundantes a $CeldasPorExplorar[i]$ **do**
8: **if** celda circundante no sale del mapa **then**
9: **if** celda circundante no es un valor nulo **then**
10: **if** es un movimiento lateral **then**
11: Calcular

$$CD = \frac{Ca + Cn}{2} + CT \tag{2A}$$

12: **else**
13: Calcular

$$CD = \sqrt{2}\frac{Ca + Cn}{2} + CT \tag{2B}$$

14: **end if**
15: **if** celda no había sido explorada en $MatrizCD$ **then**
16: $MatrizCD \leftarrow CD$
17: $CeldasPorExplorar[i+1] \leftarrow$celda circundante explorada
18: **else**
19: **if** CD es menor al valor almacenado en $MatrizCD$ **then**
20: $MatrizCD \leftarrow CD$
21: **end if**
22: **end if**
23: **end if**
24: **end if**
25: **end for**
26: $CeldasPorExplorar[i] \leftarrow$ Eliminar celda explorada
27: **end while**

Figura 3.6 *Pseudocódigo de la métrica de costo distancia*

El método de costo distancia tendrá como resultado una matriz del mismo tamaño que la matriz de fricción, esta pasará como parámetro de entrada para la métrica de IDW. Posteriormente la información calculada se eliminará para no apartar otra sección de memoria adicional y así usar los recursos de manera más eficiente.

3.3.2 Distancia Inversa Ponderada

Para realizar el cálculo de IDW en el modelo, se toman como valores de entrada el requerimiento de biomasa de la comunidad y el mapa de Costo de Distancia calculado anteriormente. A partir de esto, se calcula la presión ejercida por la comunidad para cada píxel de la imagen y se suman todos los valores de cada comunidad como se muestra en la siguiente ecuación.

$$P = \sum_{i=1}^{n} \frac{C_i}{d_i^k}, \tag{3}$$

donde P denota la presión ejercida por la totalidad de comunidades en una zona geográfica, C_i es el consumo de la iesima comunidad en toneladas de biomasa, d_i^k es el costo necesario para llegar a esa zona desde la comunidad *i*, finalmente *k* es un número real positivo que modula la función de decaimiento de la interpolación (Ghilardi et al., 2016). De tal forma que al final obtendremos una sola imagen en donde cada píxel tendrá la sumatoria de la presión que cada comunidad ejerce en este.

Entre las aplicaciones de este método, encontramos simulaciones espaciales sobre las condiciones de la fertilidad de los suelos agrícolas (Castillo et al., 2007), observación de la variación espacial del dengue con respecto a un punto de ocurrencia (Cadavid et al., 2014) y la elaboración e implementación de un geocodificador Web Predial (La Torre & Rozo, 2016),

entre otros. En este sentido, la métrica de IDW tiene muchas aplicaciones y mejora su rendimiento haciendo uso de tecnologías recientes.

Gracias al desarrollo de nuevas tecnologías, la capacidad de procesamiento ha aumentado considerablemente, dando lugar a la programación en paralelo, que surgió como una alternativa para aprovechar de una forma más eficiente los recursos computacionales y poder manejar un alto volumen de datos (Acosta & Segura, 2012). Al aprovechar el cómputo en paralelo podemos manejar simulaciones de extensiones territoriales más grandes y obtener resultados de estos modelos en menos tiempo.

A partir de la matriz generada con el modelo de costo distancia se calculará el modelo IDW, la celda inicial estará en la ubicación de la comunidad y terminará una vez que haga la exploración de todo el mapa. Para cada celda explorada se calculará la influencia de la comunidad mediante la ecuación 3, de esta forma el valor de la métrica de IDW para cada comunidad estará dada en función de la cantidad de recursos que necesita, la dificultad para acceder a la zona y su exponente de la influencia en la región que depende de su población (Lu & Wong, 2008). El modelo sumará las matrices resultantes de cada comunidad en una matriz final, por lo que mientras más alto sea el valor de las celdas de esta matriz, mayor será la presión que ejercen las comunidades de la región en esa zona. Este proceso se describe en el siguiente pseudocódigo de la Figura 3.7.

1: $MatrizIDWTotal$ ▷ Los valores de IDW de todo el mapa
2: $MatrizIDW$ ▷ Los valores de IDW una comunidad
3: **for** Todas matrices de Costo distancia **do**
4: **for** Todas las celdas de una matriz de Costo distancia **do**
5: **if** el valor de la celda es mayor a cero **then**
6: Calcular valor de IDW de la celda

$$P = \sum_{i=1}^{n} \frac{C_i}{d_i^k} \tag{3}$$

7: $MatrizIDW \leftarrow valorPdeIDWdelacelda$
8: **else**
9: $MatrizIDW \leftarrow NULO$ ▷ No es posible obtener biomasa
10: **end if**
11: **end for**
12: $MatrizIDWTotal \leftarrow MatrizIDW$
13: **end for**

Figura 3.7 *Pseudocódigo de la métrica IDW*

Finalmente, la matriz se exportará en una imagen con extensión TIFF para que sea accesible la visualización de los datos.

3.3.3 Codificación en paralelo

Para paralelizar los algoritmos se realizó el mismo procedimiento previamente descrito, asignando a cada hilo de la computadora el cálculo de ambas métricas de un grupo de comunidades, de esta forma, se podrán hacer cálculos de varias comunidades simultáneamente para reducir el tiempo total de ejecución. Para la paralelización del programa se usarán todos los hilos disponibles, estos dependerán de la arquitectura del equipo de cómputo sobre la cual se ejecute el programa.

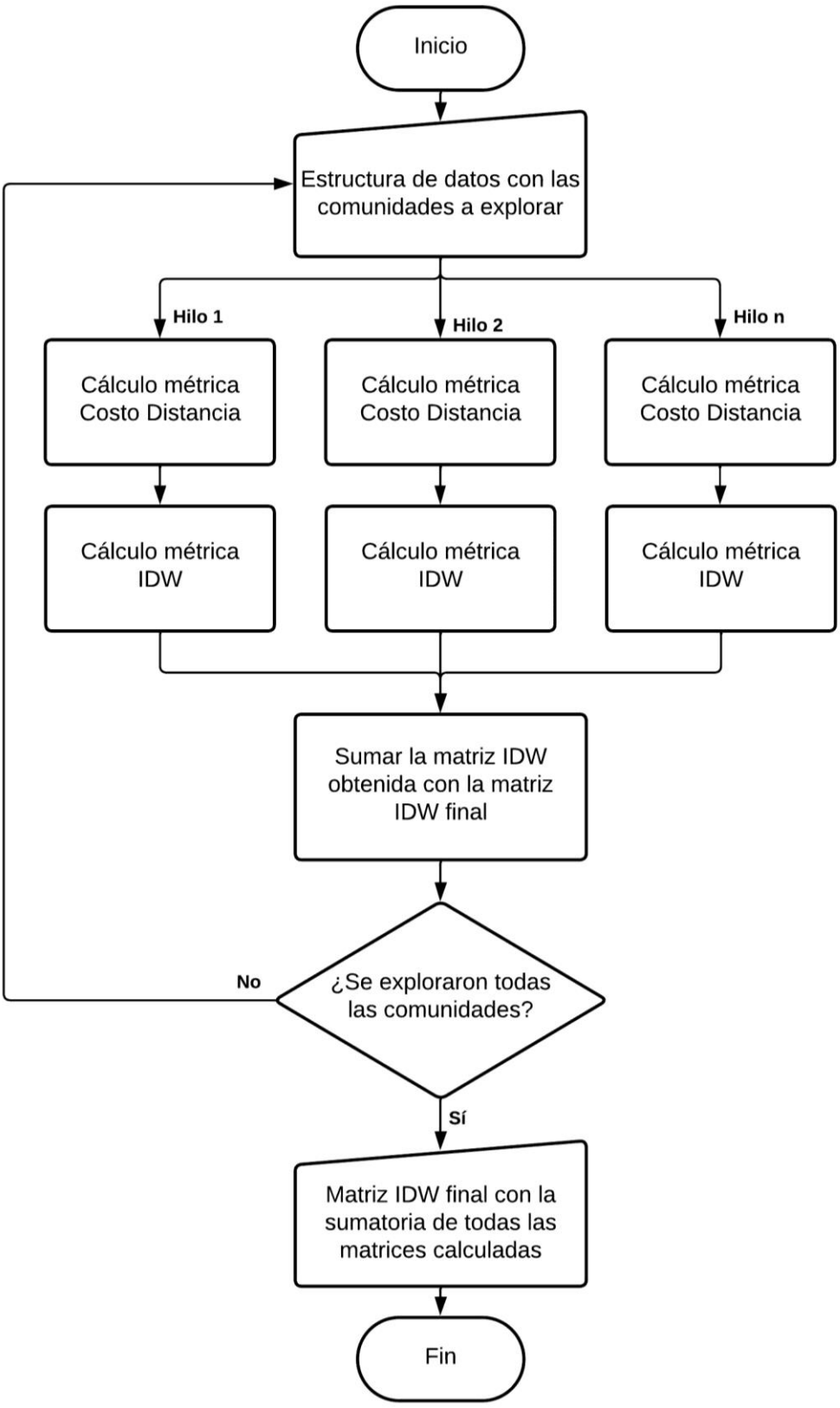


Figura 3.8 *Diagrama de flujo general del programa en paralelo*

De manera similar al proceso anterior al finalizar el cálculo se genera una imagen en formato TIFF. Ambas imágenes deben ser idénticas ya que se utilizaron las mismas métricas, pero con una notable diferencia en los tiempos de ejecución.

3.4 Análisis de Datos

Para comprobar que los resultados del programa son correctos, se debe codificar el modelo sobre la plataforma de Dinamica EGO, herramienta que ya ha sido validada por Ferreira et al. (2019) y Soares et al. (2002). De esta forma, se puede analizar la metodología de la plataforma y poder entregar una herramienta más eficiente.

Una vez implementados ambos métodos en C++ y Rust, se compararon las imágenes de salida de estos con las obtenidas mediante la plataforma de Dinámica EGO. Para esto, se importaron las imágenes en matrices para analizar celda por celda y obtener el error relativo de la celda, si este es menor a 0.01, entonces se considerará que las celdas son iguales. Cuando se tenga certeza que los resultados no presentan cambios significativos, se proseguirá a comparar los tiempos de ejecución.

Capítulo 4.
Resultados

Para poder comparar los algoritmos de Rust, C++ y Dinámica EGO, se generó un mapa con los valores resultantes de la métrica de IDW para tres diferentes escenarios variando el número de comunidades y la zona geográfica, a la par, se tomó el tiempo de ejecución de cada uno para comprobar que lenguaje es más eficiente.

Para realizar este análisis, primero se compararon los resultados de los tres modelos codificados mediante Dinamica EGO, C++ y Rust para corroborar que se obtiene el mismo resultado, durante estas pruebas se utilizarán recortes de un mapa de Kenia con las siguientes características:

- Tamaño de mapa: 3784 x 4356 pixeles
- Escala 100 m^2
- Mapa con 40 y 1940 comunidades.

Recortes de un mapa de Haití con las siguientes características:

- Tamaño de mapa: 3014 x 2309 pixeles
- Escala 100 m^2
- Mapa con 15 comunidades.

Estos modelos se calcularon en una computadora con las siguientes características:

Procesador Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz, 2601 Mhz, 8 procesadores principales, 16 procesadores lógicos 2 X 128 GB DDR4 SODIM DRAM 2133 MHz 2 x SSD 512 GB

Inicialmente se seleccionó un segmento del mapa de Kenia que contenía 40 comunidades, con lo cual se pudo comprobar que todas las imágenes resultantes sean iguales, de esta forma se asegura que el algoritmo obtiene valores válidos; posteriormente se probó el modelo con 1940 comunidades y se comprobó que es posible ejecutarlos sobre un equipo de cómputo comercial de gama media. Posterior a esto, se usará el recorte de 15 comunidades de Haití para analizar cómo se comporta el algoritmo en mapas con terrenos complicados, en donde probamos las rutas marítimas de la zona. Finalmente se tomará la zona geográfica de Kenia y se aumentarán las comunidades en este recorte para comparar los tiempos de ejecución, se tomarán 5 veces el tiempo de ejecución de cada modelo.

4.1 Casos de estudio

En la Figura 4.1 se puede observar el modelo codificado en la plataforma de Dinamica EGO para un mapa con *n* comunidades.

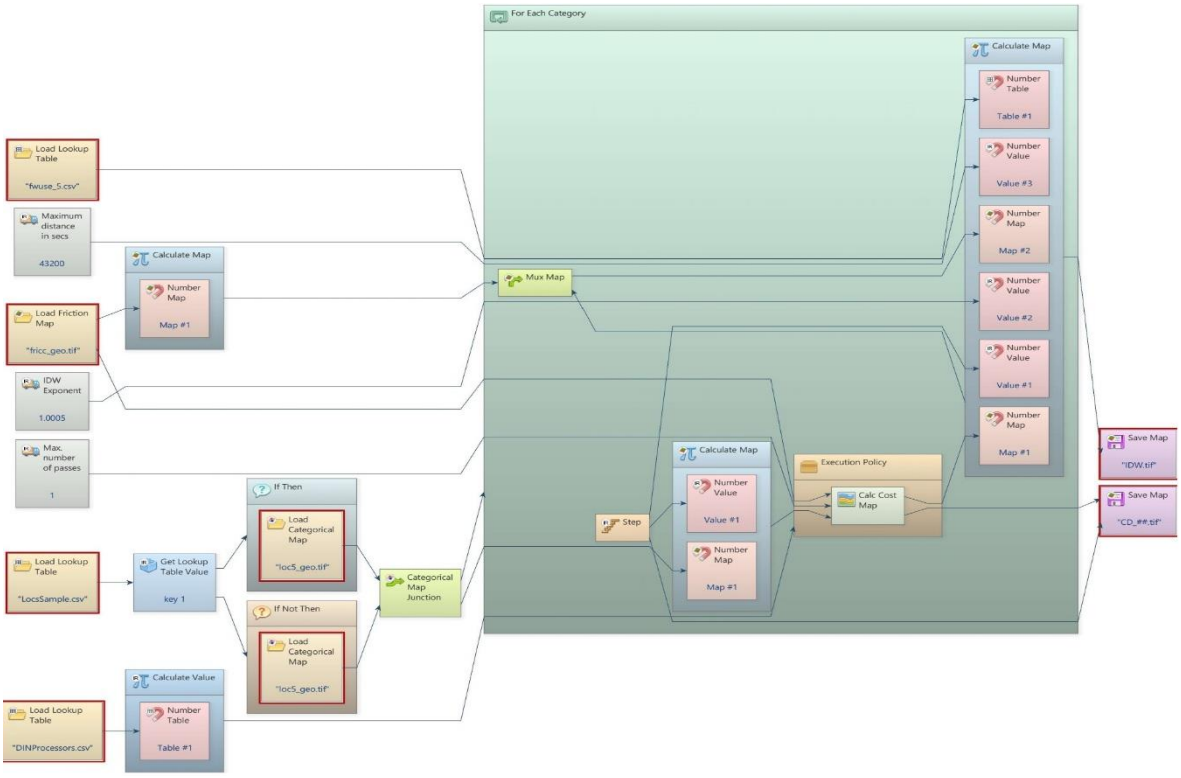


Figura 4.1 *Modelo Costo distancia e IDW en DINAMICA EGO*

Las imágenes resultantes son del mismo tamaño y poseen la misma escala. Como se observa en los mapas de las figuras 4.2, 4.3 y 4.4, las áreas con el color más claro corresponden a zonas en las que la comunidad tiene mayor influencia, es decir, los lugares de los que las personas obtienen los recursos que necesitan.

Podemos ver más clara esta diferencia en las siguientes imágenes, para la Figura 4.2 hay 40 comunidades en el mapa y se encuentran en la zona superior, mientras que para la Figura 4.3 hay 1940 comunidades repartidas a lo largo de toda la zona. En ambas imágenes se puede mostrar un cambio significativo ya que, al tener más demanda por parte de las comunidades, la presión que ejercen aumenta.

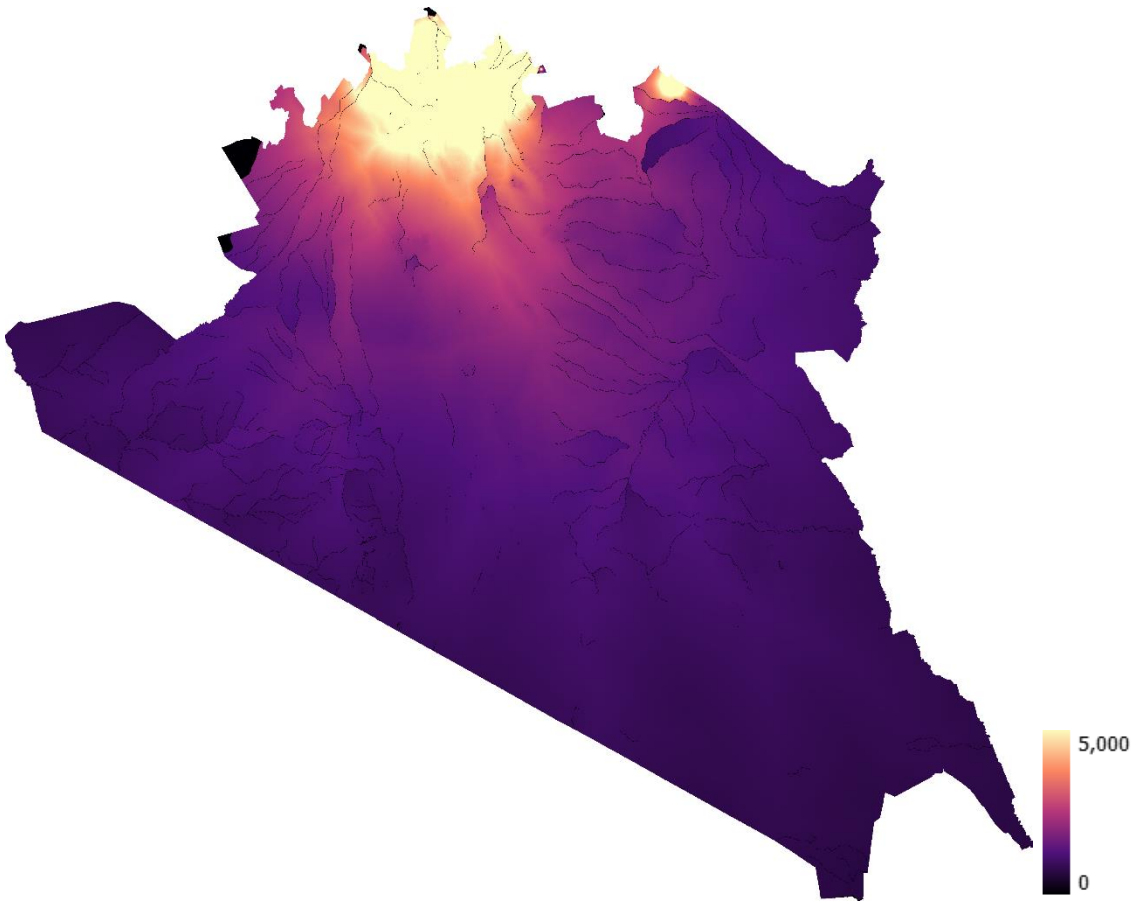


Figura 4.2 *Imagen final modelo IDW en C++ de Kenia, 40 comunidades*

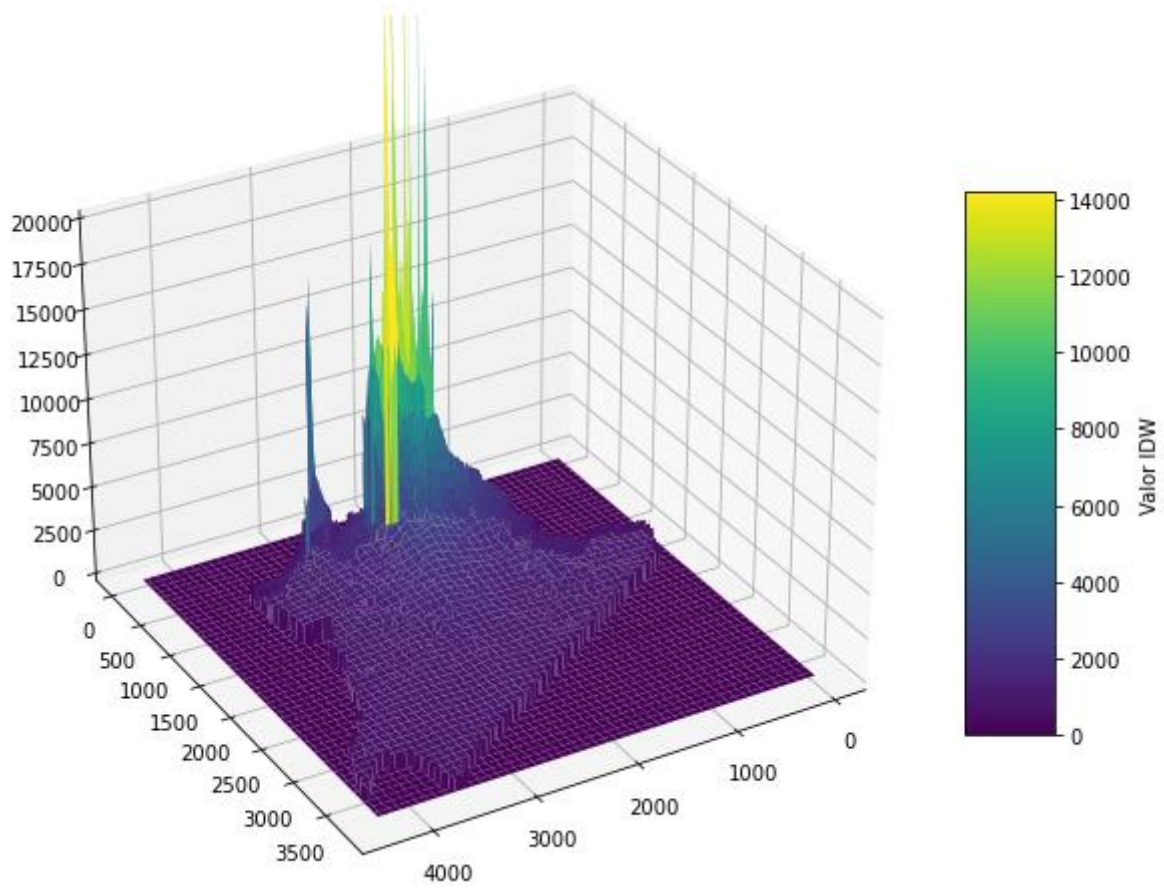


Figura 4.3 *Imagen 3D final modelo IDW en C++ de Kenia, 40 comunidades*

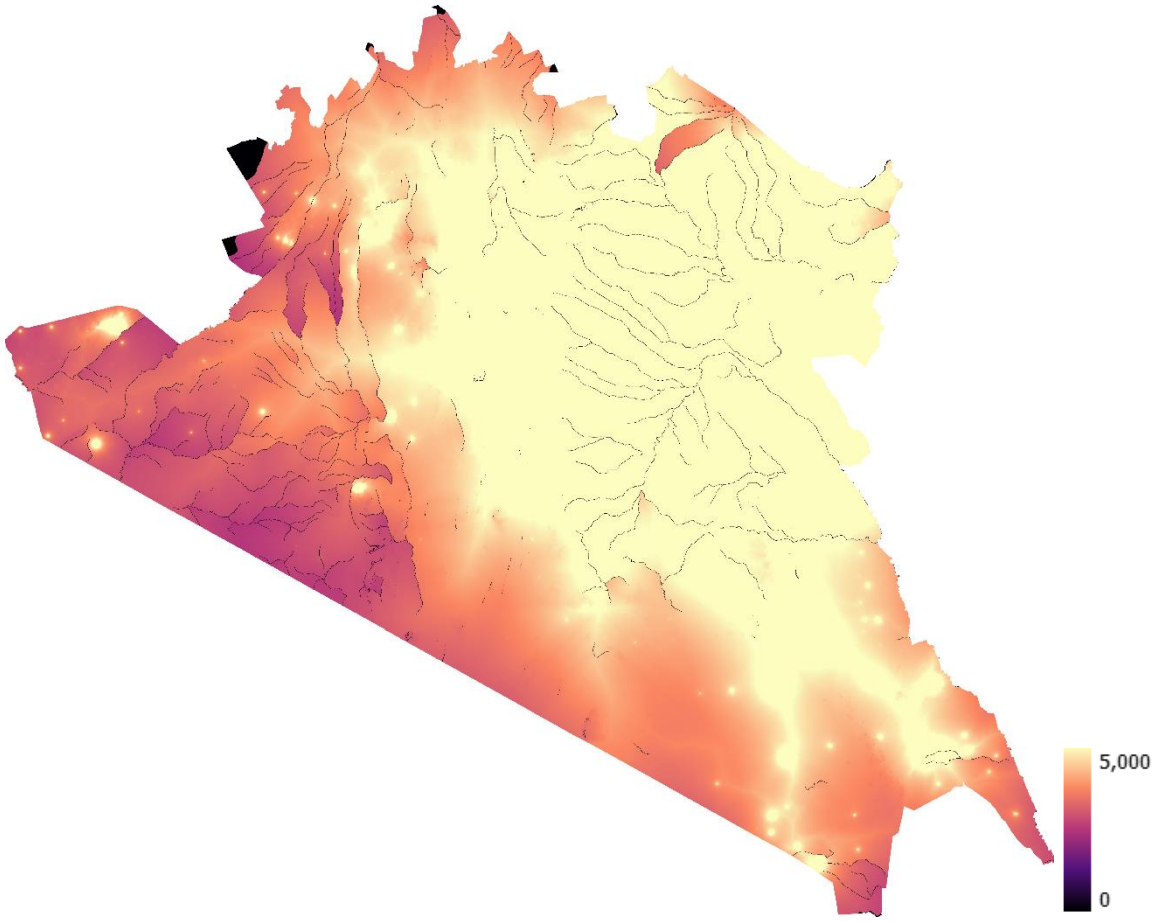


Figura 4.4 *Imagen final modelo IDW en C++ de Kenia, 1940 comunidades*

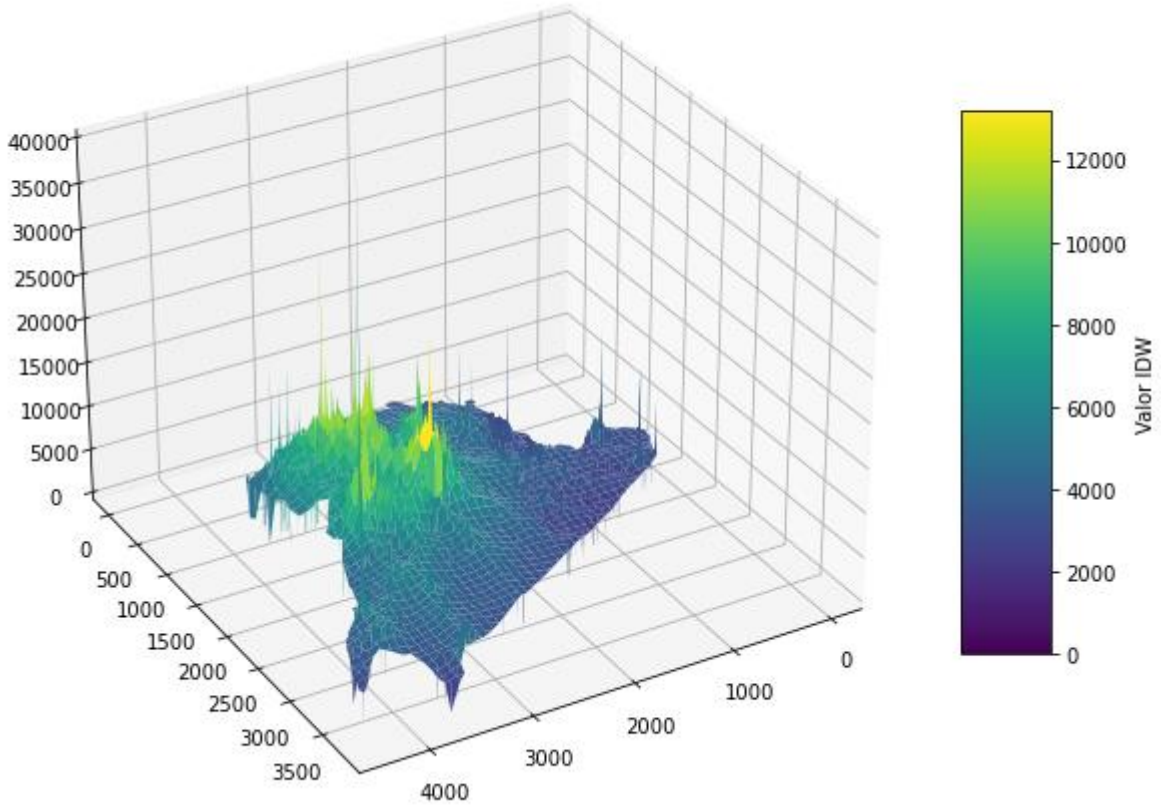


Figura 4.5 *Imagen 3D final modelo IDW en C++ de Kenia, 1940 comunidades*

Además de estos dos recortes se utilizó un mapa de Haití con 15 comunidades como se muestra en la Figura 4.4, en este caso, las líneas oscuras representan rutas marítimas que pueden ser usadas para desplazarse sobre el mapa, pero no suman a la obtención de biomasa.

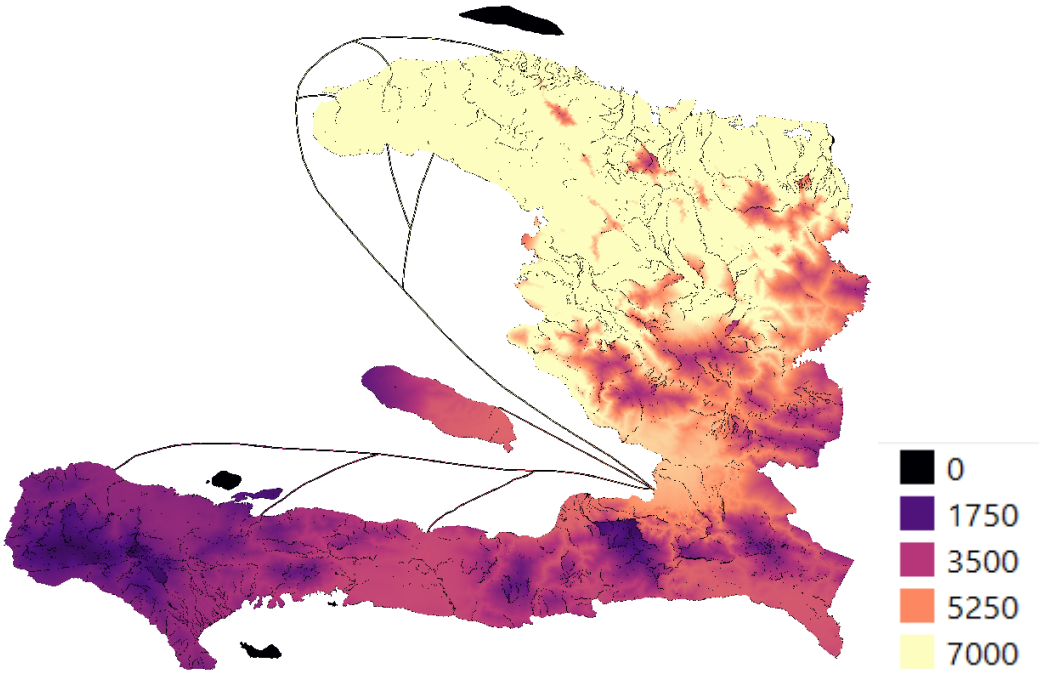


Figura 4.6 *Imagen final modelo IDW en C++ de Haití, 15 comunidades*

Finalmente se compararon los mapas de IDW de Dinamica EGO y C++ donde se encontró una diferencia máxima de $2.17x10^{-7}$en el mapa de 40 comunidades de Kenia como se muestra en la Figura 4.5.

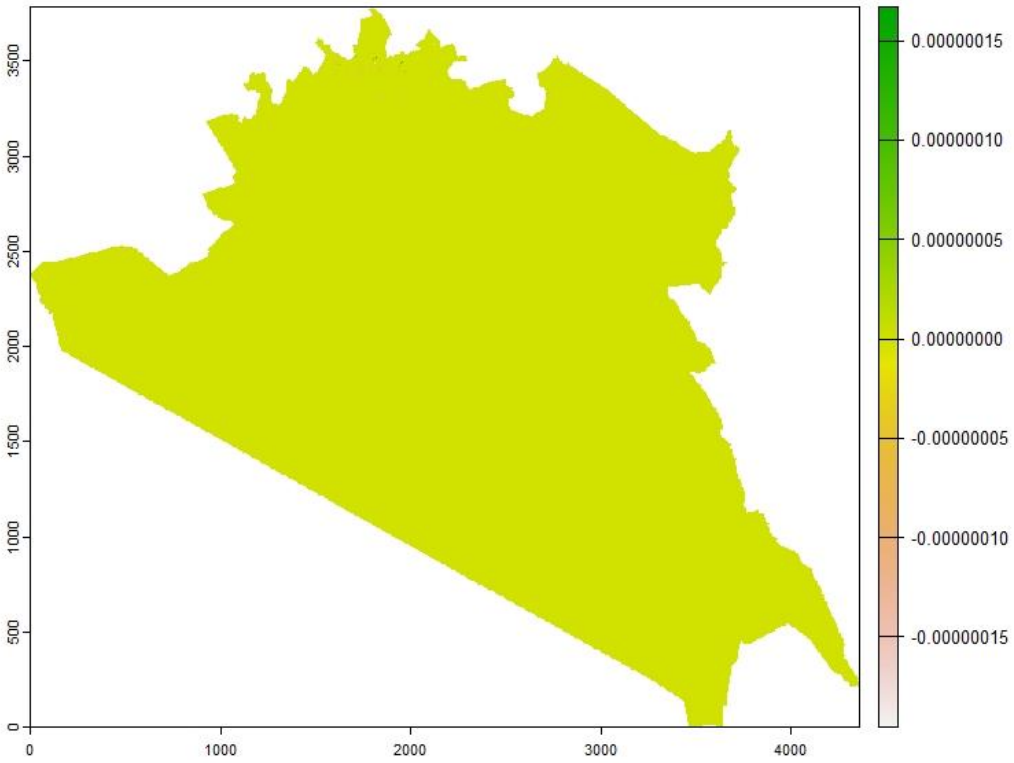


Figura 4.7 *Error porcentual entre Dinamica EGO y C++, modelo IDW 40 comunidades*

4.2 Tiempos de ejecución

Para la toma de tiempos de ejecución se aseguró que el servidor no estuviera ejecutando otros procesos, por lo que todos los modelos se ejecutaron en igualdad de condiciones. Además de esto, todos los modelos fueron ejecutados cinco veces y se reportaron el promedio de estos. En la Tabla 4.1 se puede ver un ejemplo de este proceso, en el que se reportan los tiempos de ejecución obtenidos para la implementación serial en el lenguaje C++ para el recorte de Kenia con distintas comunidades y podemos ver en la Figura 4.6 la desviación estándar en estos tiempos de forma gráfica.

Tabla 4.1 *Tiempos en C++ recorte de Kenia*

Comunidad	Tiempo en segundos
5	17.02
5	16.81
5	16.71
5	16.69
5	17.02
10	33.0
10	33.68
10	33.76
10	31.55
10	32.9
15	49.46
15	49.96
15	50.08
15	49.25
15	49.20
20	66.56
20	66.65
20	66.41
20	66.59
20	66.38
40	132.03
40	131.02
40	132.54
40	132.51
40	132.38

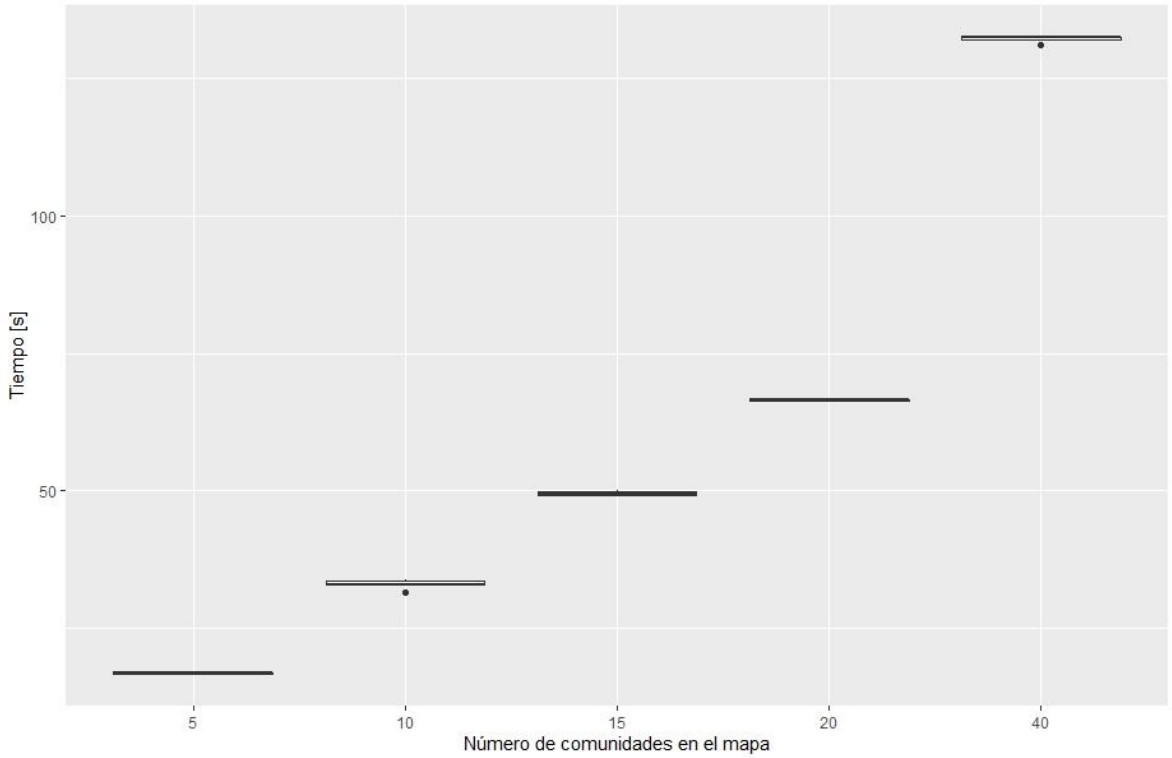


Figura 4.8 *Boxplot tiempos en C++ recorte de Kenia*

En la Tabla 4.2 se presentan el promedio de los tiempos tomados para cada lenguaje de forma serial. Para cada lenguaje se repitió el mismo proceso mostrado en la Tabla 4.1. También se puede ver esta diferencia en la Figura 4.7.

Tabla 4.2 *Tiempos promedio del recorte de Kenia implementación serial*

Comunidad	Dinamica EGO	C++	Rust
5	8806	17.02 ± 0.16	21.20 ± 0.01
10	17396	32.97 ± 0.88	41.70 ± 0.06
15	26132	49.59 ± 0.40	62.31 ± 0.03
20	35257	66.51 ± 0.11	82.82 ± 0.02
40	70597	132.09 ± 0.63	161.52 ± 0.04

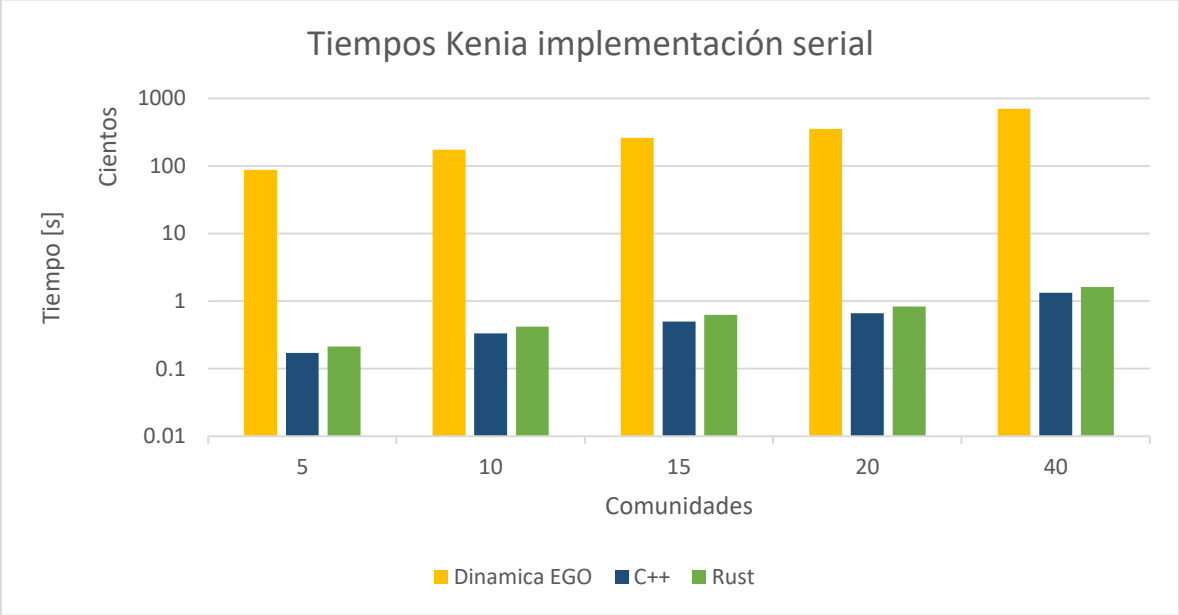


Figura 4.9 *Gráfica de tiempos del recorte de Kenia implementación serial*

Se repitió el procedimiento de toma de tiempos para la implementación en paralelo del recorte de Kenia para las mismas comunidades, los tiempos obtenidos se muestran en la Tabla 4.3.

Tabla 4.3 *Tiempos promedio del recorte de Kenia implementación paralela*

Comunidad	Dinamica EGO	C++	Rust
5	30.23	3.94 ± 0.13	4.51 ± 0.04
10	48.74	4.15 ± 0.04	5.19 ± 0.02
15	61.34	6.23 ± 0.08	7.84 ± 0.03
20	81.33	9.78 ± 0.12	10.59 ± 0.03
40	184.93	20.32 ± 0.03	21.85 ± 0.02

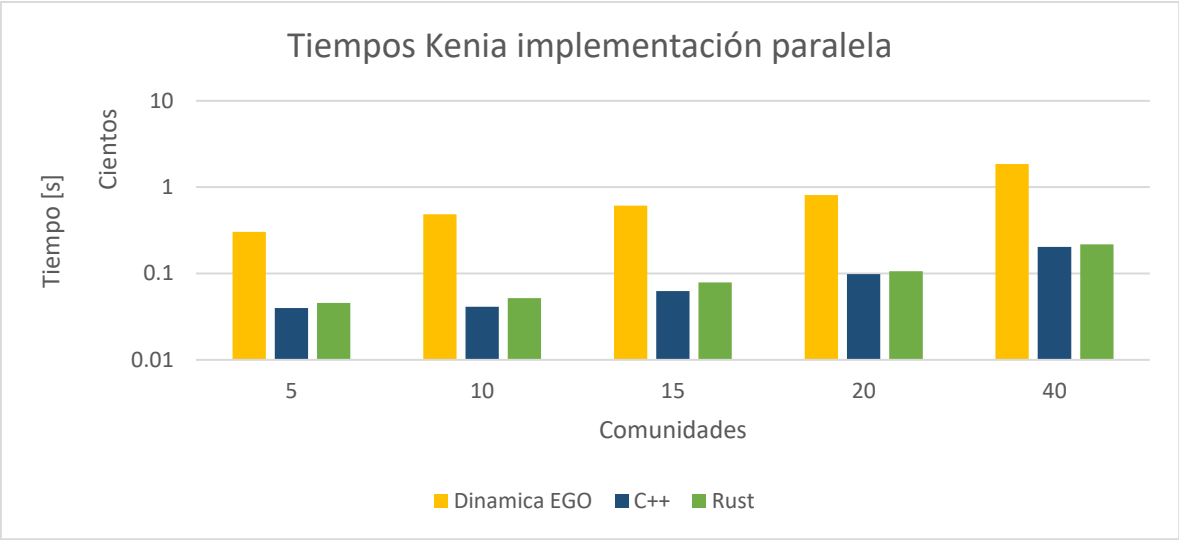


Figura 4.10 *Gráfica de tiempos del recorte de Kenia implementación paralela*

Adicionalmente, se tomaron los tiempos de ejecución para el lenguaje C++ con el mapa de 40 comunidades de Kenia variando los hilos utilizados para el cálculo, este proceso no se repitió para el resto de los lenguajes debido a que no todos permitían limitar la capacidad de procesamiento. Para los tiempos de la Tabla 4.3, se usó la máxima potencia disponible.

Tabla 4.4 *Tiempos promedio por Hilos usados del recorte de Kenia, 40 comunidades*

Hilos	Tiempo [s]
1	132.27 ± 0.04
3	45.93 ± 0.03
6	22.64 ± 0.02
9	21.36 ± 0.02
12	20.16 ± 0.03

Capítulo 5.
Discusión

En este trabajo se implementó la métrica de IDW en dos lenguajes de programación, Rust y C++, mejorando la eficiencia de la plataforma Dinamica Ego. La métrica de IDW calcula la probabilidad de que los habitantes de comunidades cercanas acudan a una zona específica para satisfacer su requerimiento de madera mediante el uso de imágenes satelitales. Para calcular esta métrica, es necesario obtener la ubicación de cada localidad, el consumo de biomasa de esta y un mapa de fricción con información geográfica de la región. Utilizando el mapa de fricción de la zona, se calculó el valor de costo distancia de cada celda metódicamente, el modelo concluye cuando todas las celdas del mapa son exploradas. A partir de esto se calcula el modelo IDW, la celda inicial está en la ubicación de la comunidad y termina una vez que se hizo la exploración de todo el mapa, generando una imagen final donde se puede ver la presión que ejercen todas las comunidades sobre la región.

Se compararon las imágenes resultantes de las tres implementaciones obteniendo valores prácticamente idénticos como se muestra en la Figura 4.5, de tal forma que la única diferencia entre estos fueron los tiempos de ejecución, tal como se muestra en las Tablas 4.2 y 4.3. Este proceso logró reducir los tiempos de procesamiento del algoritmo tanto en la implementación serial como en la paralela. Con respecto a C++, se puede observar que es aproximadamente 18% más rápido que Rust en la implementación serial y aproximadamente 22% más rápido en la implementación paralela. En las Tablas 4.2 y 4.3, se describen con mayor detalle dichos tiempos de ejecución. Una posible explicación a esta diferencia podría

deberse a que Rust tiene protocolos implementados en la librería paralela que evitan la concurrencia de los hilos, ocasionando que tarde más en iniciar el cálculo del modelo. En contraparte, estas medidas facilitan la implementación de la métrica en este lenguaje de programación ya que en el desarrollo del código se presentaron menos errores en la codificación en comparación con C++.

Independientemente del lenguaje de programación, el método de estimación en paralelo redujo los tiempos de procesamiento de las tres implementaciones, demostrando así aprovechar al máximo las capacidades del equipo sobre el cual se ejecute el modelo. Esto hace del cómputo paralelo una herramienta muy adaptable, con la posibilidad de que los tiempos de ejecución puedan reducirse aún más si se usara un procesador más potente.

La principal ventaja del cómputo paralelo es la facilidad de uso que tiene, a nivel de software, el modelo está diseñado para utilizar el mayor número de hilos disponibles del equipo, por lo que cuando se quiere compartir el modelo, siempre estará funcionando a la máxima capacidad disponible. Aunado a lo anterior, el modelo será más rápido simplemente con cambiar las características del entorno en que se ejecute debido a la constante actualización de los equipos de cómputo que surgen año con año.

Por otro lado, el resultado de la estimación se presenta en imágenes similares a mapas de calor, estos pueden ser observados en las Figuras 4.2 - 4.4, las regiones marcadas con el color amarillo son las zonas donde se corre el riesgo de deforestación. Es importante mencionar que en cada mapa se emplearon distintos números de comunidades (15 y 1940 comunidades respectivamente), este cambio se puede modificar directamente en el código. Esto se hizo con la finalidad de poder simular el cambio que habría en el impacto al

ecosistema si ciertas comunidades utilizaran un método alternativo para satisfacer sus necesidades energéticas y poder actuar en consecuencia para evitar la deforestación.

Finalmente, para poder asegurar que el modelo es válido para cualquier zona geográfica, se probó el comportamiento del modelo en regiones con características diferentes. Por tal motivo, se tomó una región en la que se tenía la información de rutas marítimas de la zona, en el mapa de la Figura 4.4 se pueden observar zonas y líneas oscuras, las líneas representan rutas marítimas, por lo que no se puede obtener biomasa de estas, pero si pueden ser usadas para moverse por el mapa. Además de estas líneas, se tienen zonas completamente oscuras que representan islas en la vida real, lo que implica que es imposible llegar a estas zonas a obtener biomasa, debido a que no hay rutas marítimas que las conecten. De esta forma es posible asegurar que el modelo respeta las zonas protegidas y hace uso de las rutas utilizadas en la región.

El presente estudio cuenta con una serie de fortalezas y limitaciones, entre las fortalezas se encuentran: 1) los métodos de estimación y lenguajes de programación empleados, esto permitió la comparación de los tiempos de ejecución de la paralelización y la serialización, y de manera adicional la comparación de C++ y Rust. Aunado a lo anterior, fue posible aprovechar al máximo la capacidad del equipo empleado, haciendo uso de un lenguaje de programación conocido a nivel mundial hace de esta herramienta una opción amigable y flexible para el usuario que desee hacer uso de ella e incluso hacer algunos cambios; 2) los datos empleados, al utilizar imágenes geoespaciales de acceso libre facilita el uso del código empleado y permite que otros investigadores corroboren dichos resultados.

La limitación que encontramos para esta simulación es que a pesar de que las imágenes satelitales requeridas para este modelo son de libre acceso, es imposible obtener de manera exacta el consumo de biomasa de todas las comunidades de la región, debido a que estas dependen de muchos factores que están cambiando constantemente. Se intenta compensar esta limitación creando modelos más eficientes, para poder actualizar el modelo cada que se tengan datos nuevos de los consumos de las comunidades.

Capítulo 6.

Conclusiones

En este trabajo se logró codificar y paralelizar las métricas de Costo Distancia e IDW en dos lenguajes de programación, Rust y C++, estos algoritmos arrojaron los mismos resultados que los obtenidos mediante la plataforma de Dinámica EGO que ya ha sido probada en anteriores trabajos, pero logrando obtener un mejor rendimiento en los tiempos de ejecución en ambos casos.

Por los tiempos de ejecución podemos concluir que el modelo desarrollado en C++ es significativamente más rápido que la plataforma de Dinamica EGO, además de esto, se puede apreciar una significativa diferencia entre la implementación paralela y serial. El cómputo paralelo permite obtener menores tiempos de ejecución, sin embargo, esta mejora depende en cierto grado del equipo sobre el cual se ejecute el modelo, de esta forma se pueden aprovechar el máximo los avances tecnológicos más recientes.

Para poder difundir y facilitar el uso del modelo desarrollado se decidió cargar el código en GitHub, servicio en la nube que permite a desarrolladores colaborar y compartir proyectos, con esto esperamos que pueda ser utilizado para monitorear y predecir la deforestación de los ecosistemas, para poder concientizar a las personas y tomar acciones para evitarlo.

Finalmente, en el contexto del presente trabajo, el cómputo paralelo demostró ser una herramienta esencial. Dada la naturaleza intensiva en cálculos de las métricas de Costo Distancia e IDW, la capacidad de procesar datos simultáneamente ha sido crucial para

mejorar los tiempos de ejecución. Aprovechando múltiples unidades de procesamiento de manera simultánea, ha sido posible computar operaciones complejas con una eficiencia notablemente superior a la programación serial. Esta técnica es fundamental para algoritmos y aplicaciones que utilizan grandes volúmenes de datos. Al dividir y asignar subprocesos a diferentes unidades, no solo fue posible acelerar la ejecución, sino también procesar extensiones territoriales más grandes. Esta adaptación al cómputo paralelo, en conjunto con las implementaciones en Rust y C++, resalta la importancia de integrar avances tecnológicos en investigaciones actuales, permitiendo así enfrentar desafíos computacionales con mayor precisión y rapidez.

Capítulo 7. **Apéndice**

Referencias

Acosta, F. A., & Segura, O. M. (2012). Guía y fundamentos de la programación en paralelo. *Revista de Telecomunicaciones e Informática*, *2*(4), 81–97. https://repository.upb.edu.co/bitstream/handle/20.500.11912/6577/Gu%c3%ada%20y%20fundamentos%20de%20la%20programaci%c3%b3n%20en%20paralelo.pdf?sequence=1&isAllowed=y

Arroyo, M. (2019, 14-18 de octubre). *Implementación Básica de Typestates en Rust*. XXV Congreso Argentino de Ciencias de la Computación. Universidad Nacional de Río Cuarto 727–736. http://sedici.unlp.edu.ar/bitstream/handle/10915/91094/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y

Aslla, A. P., Montalvo, R. A., & Rivera, P. H. (2011). Revisitando la programación en paralelo. *Revista de Investigación de Física*, *14* (1), 1–9. https://doi.org/10.15381/rif.v14i01.8540

Besard, T., Foket, C., & De Sutter, B. (2019). Effective Extensible Programming: Unleashing Julia on GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 30(4), 827–841. https://doi.org/10.1109/TPDS.2018.2872064

Brandefelt, L., & Heyman, H. (2020). A Comparison of Performance & Implementation Complexity of Multithreaded Applications in Rust , Java and C ++. *Degree project in technology, first cycle* https://www.diva-portal.org/smash/get/diva2:1463855/FULLTEXT01.pdf

Cadavid, J. I., Restrepo, C., & Marulanda, E. (2014). Distribución espacial del dengue basado enherramientas del Sistema de Información Geográfica,Valle de Aburrá, Colombia. R*evista Facultad Nacional de Salud Pública*, *32*(1), 7–15. https://doi.org/10.17533/udea.rfnsp.15464

Castillo Ibarra, D., Ruiz Corral, J.A., Flores Garnica, J. G. & González Eguiarte, D. R. (2007). Distribución espacial del contenido de materia orgánica de los suelos agrícolas de Zapopan, Jalisco. *Terra Latinoamericana, 25* (2), 187–194. https://www.redalyc.org/pdf/573/57325211.pdf

Comisión Nacional Forestal, (2023). ¿Cuánto carbono secuestran los ecosistemas forestales? Recuperado de https://www.gob.mx/conafor/es/articulos/cuanto-carbono-secuestran-los-ecosistemas-forestales

Constanzo, M. (2021) *Comparación de rendimiento y esfuerzo de programación entre Rust y C en arquitecturas Multicore. Caso de estudio: Simulación de N cuerpos computacionales*. Trabajo para obtener el grado de Especialista en cómputo de altas prestaciones y tecnología GRID. Universidad Nacional de La Plata.

Ellwanger J. H., Kulmann-leal B., Kaminski L. V. G. DA Veiga A.B., Spilkir. F., Fearnside M. P., Caesar L., Giattigabriel L. L., Wallau L.G., Almeida E.M., Borba R. M., P. Da Hora P. V. & Chies B. J. A. (2020) Beyond diversity loss and climate change: Impacts of Amazon deforestation on infectious diseases and public health. Annals of the Brazilian Academy of Sciences, *92* (1), 1–33. https://doi.org/10.1590/0001-3765202020191375

Espinoza M.V. (2016). *DINAMICA EGO: Una herramienta gratuita para modelar y brindar soporte en el análisis de CCUS.* Boletín del Colegio de Geografos del Perú. 3.

Ferreira, B. M., Soares-Filho, B. S., & Pereira, F. M. Q. (2019). The Dinamica EGO virtual machine. Science of Computer Programming, *Science of Computer Programming, 173*, 3–20. https://doi.org/10.1016/j.scico.2018.02.002

Gatti, L. V., Basso, L. S., Miller, J. B., Gloor, M., Gatti Domingues, L., Cassol, H. L.,Tejada G., Aragão, E. O. C., Nobre, C., Peters, W., Marani, L., Arai, E., Sanches, H. A., Correa, M. S., Anderson, L., Randow, V. C., Correia, S. C., Crispim, S. P. & Neves, R. A. L. (2021). Amazonia as a carbon source linked to deforestation and climate change. Nature, *595*(7867), 388–393. https://doi.org/10.1038/s41586-021-03629-6

Gernaat, D.E.H.J., de Boer, H.S., Daioglou, V., Yalew, G. S., Müller, C. & van Vuuren D. P. (2021) *Climate change impacts on renewable energy supply*. Nature Climate Change. 11, 119–125. https://doi.org/10.1038/s41558-020-00949-9

Ghilardi, A., Bailis, R., Mas, J. F., Skutsch, M., Elvir, J. A., Quevedo, A., Masera, O., Dwivedi, P., Drigo, R., & Vega, E. (2016). Spatiotemporal modeling of fuelwood environmental impacts: Towards improved accounting for non-renewable biomass. *Environmental Modelling and Software*, *82*, 241–254. https://doi.org/10.1016/j.envsoft.2016.04.023

Global Carbon Budget (2021), CO2 emissions rebound towards pre-COVID. Recuperado de chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.globalcarbonproject.org/global/images/carbonbudget/Infographic_Emissions2021.pdf

Hernández Montero, L. & Ramón Antunez, R. (2011). *Programación en paralelo: definiciones, inconvenientes y mecanismos.* Serie Científica de la Universidad de las Ciencias Informáticas, 4 (7),1-8. http://publicaciones.uci.cu/index.php/SC

Hofierka, J., Lacko, M., & Zubal, S. (2017). *Parallelization of interpolation, solar radiation and water flow simulation modules in GRASS GIS using OpenMP*. Computers & Geosciences, 107, 20–27. doi:10.1016/j.cageo.2017.07.007

Jean-François Mas, Melanie Kolb, Thomas Houet, Martin Paegelow, Maria T. Camacho Olmedo. *Una comparación de diferentes enfoques de modelación de cambios de cobertura/uso del suelo.* XIV Simposio Internacional SELPER 2010 , Nov2010 ,Guanajuato, México. pp.CD. halshs-01063482

Kumar, R. & Shahabuddin, G. (2006) Consequences of Rural Biomass Extraction for Bird Communities in an Indian Tropical Dry Forest and the Role of Vegetation Structure. *Conservation & Society*, *4* (4), 562-591https://www.jstor.org/stable/26392862.

Klabnik, S. & Nichols, C. (2022). The Rust programmings laguage. https://doc.rust-lang.org/book/

La Torre Gonzalez, D. F. , & Rozo Rozo, S. N. (2016). Elaboración e implementación de un geocodificador predial – web basado en el método inverso de la distancia ponderada para la aproximación de coordenadas. plan piloto municipio de chía. Tesis para obtener el grado de Ingeniero Catastral y Geodesta 133. http://repository.udistrital.edu.co/bitstream/11349/14694/1/LatorreDavidRozoSergio2016.pdf

Lu, G. Y., & Wong, D. W. (2008). An adaptive inverse-distance weighting spatial interpolation technique, *34*(9), 1044–1055. https://doi.org/10.1016/j.cageo.2007.07.010

Manolis, E. N., Zagas, T. D., Karetsos, G. K., & Poravou, C. A. (2019). *Ecological restrictions in forest biomass extraction for a sustainable renewable energy production*. Renewable and Sustainable Energy Reviews, 110 (October 2018), 290–297. https://doi.org/10.1016/j.rser.2019.04.078

Multi-producer, single-consumer (2022). Module std::sync::mpsc. https://doc.rust-lang.org/std/sync/mpsc/index.html

Oliveira Júnior, A. J. de, Souza, S. R. L. de, Dal Pai, E., Rodrigues, B. T., & Souza, V. C. de. (2019). *Aurora: Mobile application for analysis of spatial variability of thermal comfort indexes of animals and people, using IDW interpolation. Computers and Electronics in Agriculture,* 157, 98–101. doi:10.1016/j.compag.2018.12.029

OpenMp (2020). OpenMP 5.2 API Syntax Reference Guide. https://www.openmp.org/wp-content/uploads/OpenMPRefCard-5-2-web.pdf

Peterson, E. E., & Pearse, A. R. (2017). *IDW-Plus: An ArcGIS Toolset for Calculating Spatially Explicit Watershed Attributes for Survey Sites*. JAWRA Journal of the American Water Resources Association, 53(5), 1241–1249. doi:10.1111/1752-1688.12558

Qin, C. Z., Zhan, L. J., & Zhu, A. X. (2014). How to Apply the Geospatial Data Abstraction Library (GDAL) Properly to Parallel Geospatial Raster I/O? *Transactions in GIS*, 18(6), 950–957. https://doi.org/10.1111/tgis.12068

Rasoulkhani, M., Ebrahimi-Nik, M., Abbaspour-Fard, M. H., & Rohani, A. (2018). Comparative evaluation of the performance of an improved biomass cook stove and the traditional stoves of Iran. *Sustainable Environment Research*, *28* (6), 438–443. https://doi.org/10.1016/j.serj.2018.08.001

Rodríguez Márquez J. D. (2022) Aprovechamiento de la biomasa para obtener energía eléctrica en el municipio barinas estado barinas. *Revista Ambientellanía*, *5* (1), 42-50. http://revistas.unellez.edu.ve/index.php/ambientellania/article/view/1759

Romero Salvador, A. (2010). Aprovechamiento de la biomasa como fuente de energía alternativa a los combustibles fósiles. *Revista de la Real Academia de Ciencias Exactas, Fisicas y Naturales (Espana)*, *104* (2), 331–345. http://www.rac.es/ficheros/doc/00979.pdf

Sánchez Barboza, L., Pérez Pineda, R. E., & Vásquez Stanescu, C. (2018). Eficiencia de países desarrollados en el control del uso de combustibles fósiles para generar energía. *Revista Científica Ecociencia*, *4* (2), 58–71. https://doi.org/10.21855/ecociencia.42.28

Saynes Santillán, V., Etchevers Barra, J. D., Paz Pellat, F., & Alvarado Cárdenas, L. O. (2016). Emisiones de gases de efecto invernadero en sistemas agrícolas de México . *Terra Latinoamericana*, *34*, 83–96. https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0187-57792016000100083#:~:text=En%20el%20caso%20de%20M%C3%A9xico,de%20las%20emisiones%20nacionales%20totales.

Soares-Filho B. S., Coutinho Cerqueira, G. & Lopes Pennachin, C. (2002). DINAMICA - a stochastic cellular automata model designed to simulate the landscape dynamics in an Amazonian colonization frontier. *Ecological Modelling*, *154*, 217–235. https://doi.org/10.1016/S0304-3800(02)00059-5

Sutar, K. B., Kohli, S., Ravi., M.R. & Ray, A. (2015). Biomass cookstoves: A review of technical aspects. *Renewable and Sustainable Energy Reviews*, *41*, 1128–1166. https://doi.org/10.1016/j.rser.2014.09.003

Sydow, S., Nabelsee, M., Glesner, S., & Herber, P. (2020). Towards profile-guided optimization for safe and efficient parallel stream processing in rust. Proceedings - Symposium on Computer Architecture and High Performance Computing, 2020-Septe, 289–296. https://doi.org/10.1109/SBAC-PAD49847.2020.00047

Useros Fernández, J. L. (2013). El Cambio Climático: sus causas y efectos medioambientales. *Real Academia de Medicina y Cirugía de Valladolid*, 50, 71–98. https://dialnet.unirioja.es/servlet/articulo?codigo=4817473#:~:text=El%20incremento%20de%20la%20temperatura,al%20hielo%2C%20a%20los%20sistemas

Vergara, A. (2017). Diferencias entre C, C# y C++. FacilCloud. https://www.facilcloud.com/noticias/diferencias-del-lenguaje-c/

Weinbach, N. L (2008). Paradigmas de Programación en Paralelo: Paralelismo Explícito y Paralelismo Implícito. Tesis de magister en ciencias de la computación. Universidad Nacional Del Sur.

X. Zheng, Y. Xue, J. Guang and J. Liu, "*Remote sensing data processing acceleration based on multi-core processors,"* 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Beijing, China, 2016, pp. 641-644, doi: 10.1109/IGARSS.2016.7729161.

Xia, Yj., Kuang, L. & Li, Xm. *Accelerating geospatial analysis on GPUs using CUDA.* J. Zhejiang Univ. - Sci. C 12, 990 – 999 (2011). https://doi.org/10.1631/jzus.C1100051

Yepes, A. L., & Gay, F. S. (1994). Fototecas digitales en prensa: formatos gráficos, entornos y sistemas informáticos. *Cuadernos de Documentación Multimedia*, 3, 57. http://redocom.hst.ucm.es/CDM/Documentos compartidos/58_CDM_Vol_3.pdf

Yi, W., Gao, Z., & Chen, M. (2012). Dynamic modelling of future land-use change: a comparison between CLUE-S and Dinamica EGO models. *Remote Sensing and Modeling of Ecosystems for Sustainability IX*, *8513*, 1-7 . https://doi.org/10.1117/12.927781

Zhang, L., He, C., Li, J., Wang, Y., & Wang, Z. (2017). C*omparison of IDW and Physically Based IDEW Method in Hydrological Modelling for a Large Mountainous Watershed, Northwest China.* River Research and Applications, 33(6), 912–924. doi:10.1002/rra.3147

Apéndice A - Código serial empleado.

Figura A1

Cargar mapa de fricción, la ubicación y consumo de las comunidades en Rust

```rust
let inicio=Instant::now();
let mut rows=0;
let mut cols=0;
let fric_img = match leer_img("/root/modelos/Kenia/fricc_singeo0.tif",&mut rows,&mut cols){
    tiff::decoder::DecodingResult::F32(v) => v,
    _ => panic!("panicccccc"),
};
let locs_img = match leer_img("/root/modelos/Kenia/20_comunidades/mapa_locs_20.tif",&mut rows,&mut cols){
    tiff::decoder::DecodingResult::F32(v) => v,
    _ => panic!("panicccccc "),
};
let mut localidades = leer_csv("/root/modelos/Kenia/20_comunidades/fwuse_20.csv");
for (i,val) in locs_img.iter().enumerate().filter(|&(_,&val)| val !=-9999.0){
    if let Some(elem) = localidades.get_mut(*val as usize-1){
        let col=i%cols;
        let row=(i-col)/cols;
        elem.row=row;
        elem.col=col;
    }
}
```

Nota. Las imágenes se guardan en matrices y el consumo en una estructura de datos con el numero de la comunidad y su consumo.

Figura A2

Métrica de Costo distancia en Rust

```rust
cd_map.push(pos_cell.clone());//comunidad inicial
let mut cont = 1;
let mut row_temp;
let mut col_temp;
let mov =[(1,0),(1,1),(0,1),(-1,1),(-1,0),(-1,-1),(0,-1),(1,-1)];
//let mov: [[isize;8];2] =[[1,1,0,-1,-1,-1,0,1],[0,1,1,1,0,-1,-1,-1]];
//----------------------------------------------------------------inicia calculo
while let Some(Cell {row,col, friccion:costo_acumulado, key:_ }) = cd_map.pop(){
    for (col_mov,row_mov) in mov.iter(){
        let mut pos_cell=pos_cell.clone();
        row_temp=row_mov+row as isize;
        col_temp=col_mov+col as isize;
        //row_temp = mov[1][i - 1] + row as isize;
        //col_temp = mov[0][i - 1] + col as isize;
        let it =((cols*row_temp)+col_temp) as usize;
        if row_temp < rows && row_temp >= 0 && col_temp < cols && col_temp >= 0 {
            if fric_matrix[((cols*row_temp)+col_temp) as usize] > 0.0 {
                if col_mov*row_mov  == 0{//si es un movimiento lateral
                    pos_cell.friccion = costo_acumulado + OrderedFloat(fric_matrix[it]);
                }
                else{//si es un movimiento diagonal
                    pos_cell.friccion = costo_acumulado + OrderedFloat(fric_matrix[it] * 2f32.sqrt());
                }
                //se busca el menor valor de CD, es posible que se escriba varias veces en una celda
                if OrderedFloat(cd_matrix[it]) > pos_cell.friccion {
                    pos_cell.row = row_temp as usize;
                    pos_cell.col = col_temp as usize;
                    pos_cell.key=cont;
                    cont+=1;
                    let OrderedFloat(fric_mov) = pos_cell.friccion;
                    cd_matrix[it] = fric_mov;
                    cd_map.push(pos_cell);
                }
            }
        }
    }
}
return cd_matrix;
```

Nota. Se establecen los movimientos a seguir en la búsqueda y las condiciones para evitar que la exploración se salga del mapa o que se extraiga biomasa de zonas protegidas.

Figura A3

Métrica de IDW en Rust

```rust
fn idw_met (comunidad: &Cell,cd_matrix: &Vec<f32>, idw_matrix: &mut Vec<f32>){
    let exp=1.005;
    let OrderedFloat(req)=comunidad.friccion;
    for (val_cd,val_idw)  in cd_matrix.iter().zip(idw_matrix.iter_mut()){
        if *val_cd<=0.0{
            *val_idw=-9999.0;
        }
        else{
            *val_idw += req / val_cd.powf(exp);
        }
    }
}
```

Nota. Se calcula los valores de IDW respecto a la matriz de costo distancia ya que en esta se consideró la fricción de la zona y la distancia respecto a la comunidad inicial.

Figura A4

Cargar mapa de fricción, la ubicación y consumo de las comunidades en C++

```cpp
//-------------------------------------------------------------------------------------------------------carga de archivos
    //mapa fricción
    fric_map = objrast.read_tif_matrix("/home/otavio/Documentos/Proyecto_OpenMP/Serial/fricc_w.tif", M, N, scale,cell_null_2);
    printf("--------No. columnas: %d\n", M);
    printf("--------No. renglones: %d\n", N);
    // mapa localidades
    map_local = objrast.read_tif_matrix("/home/otavio/Documentos/Proyecto_OpenMP/locs_10_int.tif", M_i, N_i,scale,cell_null);
    // guardamos los requisitos de las comunidades en un mapa
    objrast.carga_requisitos("/home/otavio/Documentos/Proyecto_OpenMP/fwuse.csv",bio_local);
    //objrast.carga_requisitos("fwuse.csv",bio_local);
    //obtenemos el numero de comunidades
    num_com = objrast.contar_comunidades(map_local,M_i,N_i,cell_null);
    // guardamos las localidades en un mapa para ordenarlas
    objrast.leer_localidades(map_local,M_i,N_i,localidades,cell_null,num_com);
```

Nota. Las imágenes se guardan en matrices y el consumo en una estructura de datos con el numero de la comunidad y su consumo.

Figura A5

Métrica de Costo distancia en C++

```
const int mov[2][8]={{1,1,0,-1,-1,-1,0,1},{0,1,1,1,0,-1,-1,-1}};
for(row_temp=0;row_temp<rows;row_temp++)
    for(col_temp=0;col_temp<cols;col_temp++)
        CD_matrix[(cols*row_temp)+col_temp]=std::numeric_limits<float>::max();
position inicial;
CD_costos.push(array);
//-----------------------------------------------------------------inicia calculo
while(!CD_costos.empty()){
    inicial=CD_costos.top();
    CD_costos.pop();
    for(i=1;i<9;i++){
        row_temp = mov[1][i - 1] + inicial.row;
        col_temp = mov[0][i - 1] + inicial.col;
        if (row_temp < rows && row_temp >= 0 && col_temp < cols && col_temp >= 0 && fric_matrix[(cols*row_temp)+col_temp]>0.0) {
            if (i % 2 != 0)//si es un movimiento lateral
                array.val_fricc = inicial.val_fricc + (fric_matrix[(cols*row_temp)+col_temp]);
            else//si es un movimiento diagonal
                array.val_fricc =  inicial.val_fricc + sqrt(2) * (fric_matrix[(cols*row_temp)+col_temp]);
            //se busca el menor valor de CD, es posible que se escriba varias veces en una celda
            if (CD_matrix[(cols*row_temp)+col_temp]>array.val_fricc) {
                array.row = row_temp;
                array.col = col_temp;
                array.key=key;
                key++;
                CD_matrix[(cols*row_temp)+col_temp] = array.val_fricc;
                CD_costos.push(array);
            }
        }
    }
}
return CD_matrix;
```

Nota. Se establecen los movimientos a seguir en la búsqueda y las condiciones para evitar que la exploración se salga del mapa o que se extraiga biomasa de zonas protegidas.

Figura A6

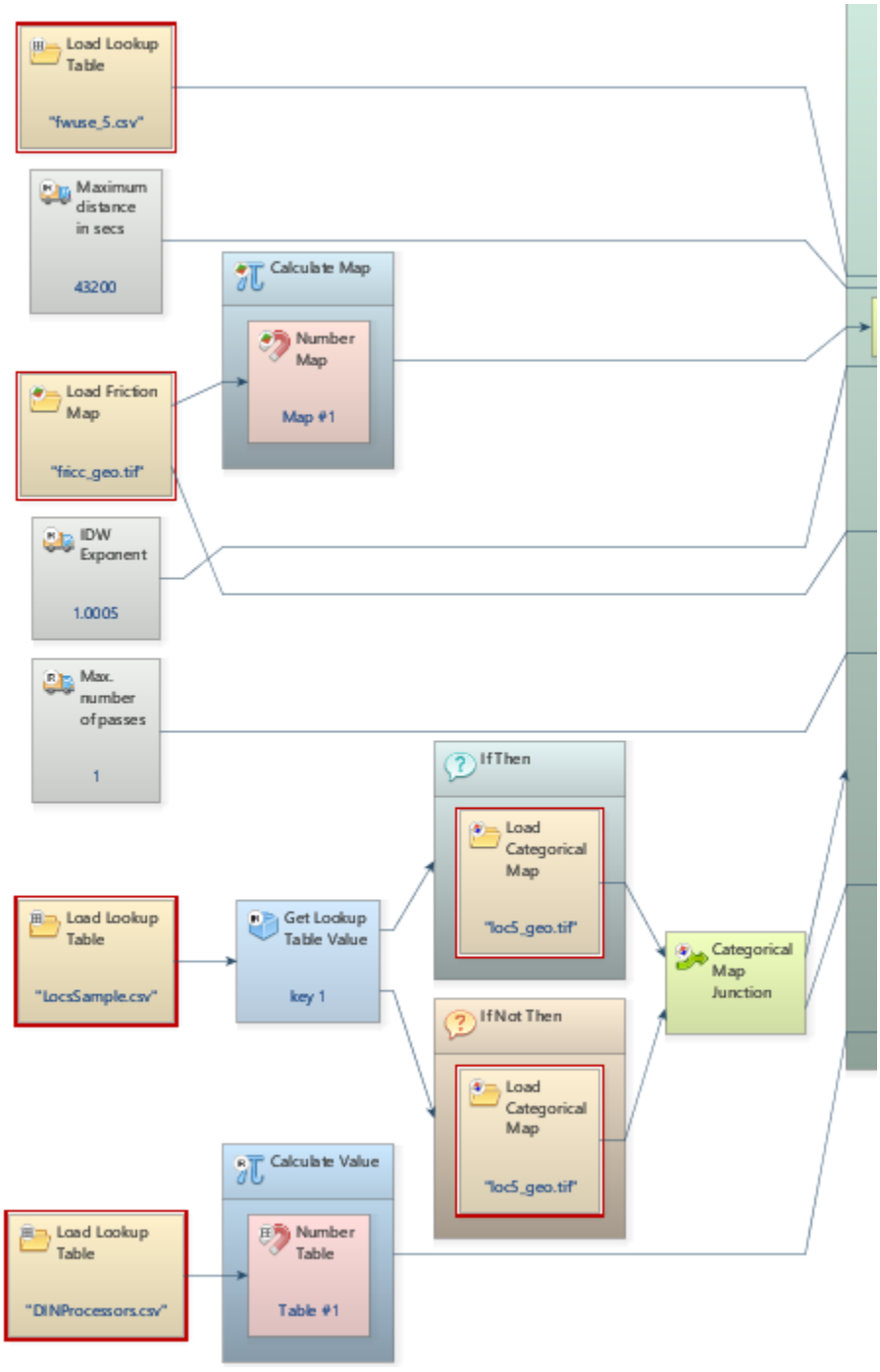
Métrica de IDW en C++

```
int row,col;
for(row = 0; row < rows; row++) {
    for (col = 0; col < cols; col++) {
        if (CD_matrix[(cols * row) + col] <= 0) {
            IDW_matrix[(cols * row) + col] = cell_null;
        } else {
            IDW_matrix[(cols * row) + col] += biomass->second / pow(CD_matrix[(cols * row) + col], exp);
        }
    }
}
```

Nota. Se calcula los valores de IDW respecto a la matriz de costo distancia ya que en esta se consideró la fricción de la zona y la distancia respecto a la comunidad inicial.

Figura A7

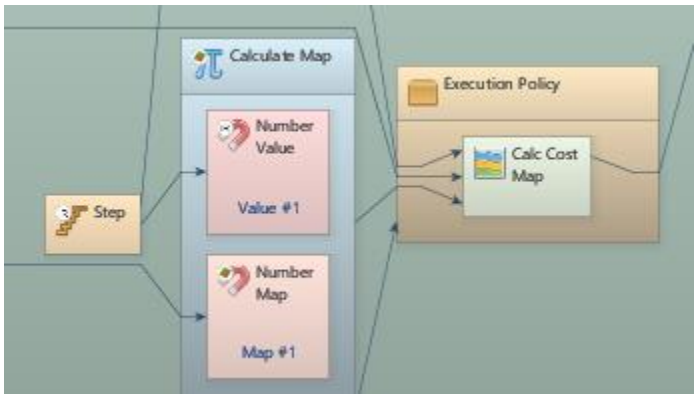
Cargar mapa de fricción, la ubicación y consumo de las comunidades en Dinamica EGO



Nota. Las imágenes se guardan en matrices y el consumo en una estructura de datos con el numero de la comunidad y su consumo.

Figura A8

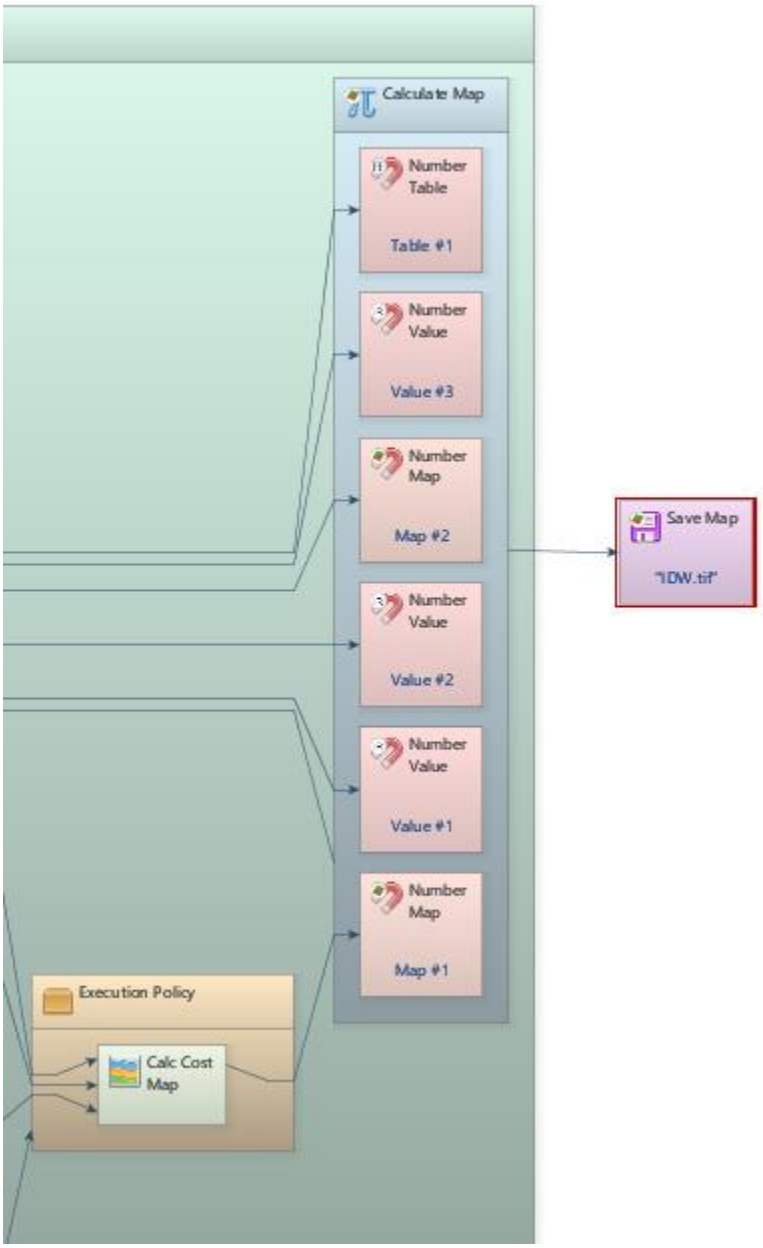
Métrica de Costo distancia en Dinamica EGO



Nota. Se establecen los movimientos a seguir en la búsqueda y las condiciones para evitar que la exploración se salga del mapa o que se extraiga biomasa de zonas protegidas.

Figura A9

Métrica de IDW en Dinamica EGO



Nota. Se calcula los valores de IDW respecto a la matriz de costo distancia ya que en esta se consideró la fricción de la zona y la distancia respecto a la comunidad inicial.

Apéndice B - Código paralelo empleado.

Figura B1

Paralelización código en C++

```
const int mov[2][8]={{1,1,0,-1,-1,-1,0,1},{0,1,1,1,0,-1,-1,-1}};
//omp_set_num_threads(10);
#pragma omp parallel for default(shared) private(ubicacion,biomass,array)
for(i=start;i<=end;i++) {
    if (biomass_requerida.find(i) != biomass_requerida.end()) {//existe la comunidad con ese numero?
        biomass = biomass_requerida.find(i);
        if (biomass->second != 0) {//requisitos distintos a cero
            if (localidades.find(i) != localidades.end()) { //existe la comunidad con ese numero?
                ubicacion = localidades.find(biomass->first);//buscar ubicacion de la localidad
                //ubicaion inicial
                array.row = ubicacion->second.row;
                array.col = ubicacion->second.col;
                array.val_fricc = 0;
                array.key=0;
                cont++;//localidades calculadas
                float *CD_matrix = new float[rows*cols];
                //CD_matrix=objMeth.cost_distance(fric_matrix, rows, cols, array, CD_matrix);
                priority_queue<position> CD_costos;
                int key = 1;
                int row_temp,col_temp,h;
                for(row_temp=0;row_temp<rows;row_temp++)
                    for(col_temp=0;col_temp<cols;col_temp++)
                        CD_matrix[(cols*row_temp)+col_temp]=std::numeric_limits<float>::max();
                position inicial;
                CD_costos.push(array);
```

Nota. Es necesario establecer variables privadas y compartidas, las variables privadas crean una copia de ellas para cada hilo, esto ayuda a evitar la concurrencia

Figura B2

Paralelización código en Rust

```
let (tx,rx)=mpsc::channel();

while let Some(com) = localidades.pop(){
    let fric_img = Arc::clone(&fric_img);
    let tx1=tx.clone();
    thread::spawn(move ||{
        let cd_matrix=cd_met(&com,rows, cols,&fric_img);
        let idw_parcial=idw_met(&com,&cd_matrix);
        tx1.send(idw_parcial).unwrap();
    });
}
drop(tx);
for idw_parcial in rx{
    for a in 0..idw_final.len(){
        idw_final[a]+=idw_parcial[a];
    }
}
```

Nota. A diferencia de C++, se tiene que crear dos canales, uno que transmite y otro recibe, el resultado de los hilos se guarda en el canal de transmisión hasta que se terminan de evaluar todas las comunidades, en ese momento, se guardan todas las matrices de IDW de cada hilo en una sola en el canal de recepción para exportarla como imagen.