Manual de usuario para el aplicativo en Python

Kevin J. Culma - 20182020008

Sergio E. Suarique - 20161020526

Julian A, Sanchez - 20181020169

Repositorio:  
<https://github.com/INderKev/IO-Repo>

Universidad Distrital Francisco José de Caldas

Facultad de Ingeniería

Ingeniería de sistemas

Albert Acosta López

2020

# **Tabla de contenido**

INCIDES

[**Tabla de contenido** 2](#_Toc59004610)

[**Introducción.** 3](#_Toc59004611)

[**Descripción:** 3](#_Toc59004612)

[**Funcionalidad.** 3](#_Toc59004613)

[**Versión Actual** 3](#_Toc59004614)

[**Requisitos** 4](#_Toc59004615)

[**Conocimientos mínimos** 4](#_Toc59004616)

[**Tecnicismos** 4](#_Toc59004617)

[**Mecanismos de acceso** 4](#_Toc59004618)

[**Descarga** 5](#_Toc59004619)

[**Modo de uso** 6](#_Toc59004620)

[**Restricciones:** 6](#_Toc59004621)

[**Ingreso de datos** 6](#_Toc59004622)

[**Referencias** 8](#_Toc59004623)

# **Introducción.**

## **Descripción:**

El aplicativo generado para el desarrollo de la clase de “Investigación de Operaciones I” se basa en la solución de problemas con la característica de ser problemas orientados a la programación entera pura, y para poner en contexto al lector de este manual, el aplicativo que se desarrollo en Python soluciona problemas de mayor complejidad en referente a los problemas de programación lineal, pues la programación entera pura o PEP, considera la enteridad de las variables tomadas para el uso y el resultado de la función a optimizar.

Es decir que, como una de las condiciones de uso encontramos que solo es posible abordar, a partir del aplicativo creado, problemas donde sea estrictamente necesario que las variables y el resultado no pueda tomar valores flotantes, problemas donde las cantidades sean discretas, como ejemplo alguno, si el problema abordado corresponde a un problema de cantidades para cocinar un plato, donde las variables corresponden a la cantidad de algún ingrediente medida en peso. Y el resultado es un plato con ciertas restricciones de cantidad, calculado en peso.

El problema por cantidades podría solucionarse por PEP, haciendo que los pesos solo sean enteros, es posible pero no es muy útil, pues sus cantidades podrían variar fácilmente y su optimización poco relevante, y si el usuario no está interesado en que sean cantidades enteras, el aplicativo creado no será realmente funcional. Por otro lado, si el problema es discreto y finito, su efectividad será total, pues las restricciones de enteridad son parte obligatoria del programa, pues si no seria un problema de programación lineal y no es lo que se busca.

## **Funcionalidad.**

* Solución a problemas de programación entera pura.
  + Basándose en el método Simplex.

## **Versión Actual**

PEP 1.0, actualmente el aplicativo se basa exclusivamente en el método de acotación y ramificación para la solución a problemas en PEP, usando Python exclusivamente para su desarrollo y una serie de librerías apoyo.

# **Requisitos**

## **Conocimientos mínimos**

Como se mencionó en la descripción de este documento, el programa se basa solamente y exclusivo en la programación entera pura, es por ello que el usuario que desee implementar el aplicativo generado, es recomendable que el potencial usuario maneje conceptos como lo es:

* Programación Lineal
* Método Simplex
* Programación Entera Pura.
* Métodos de solución para PEP.

## **Tecnicismos**

Este apartado trae buenas noticias, pues al ser un aplicativo Python su compatibilidad con los diferentes sistemas operativos que ofrece el mercado es bastante optima, Python brinda un entorno de desarrollo flexible y sencillo, siendo así que cualquiera con un IDLE de Python o un ordenador con conectividad a la internet pueda descargar el ejecutable y disfrutar de dicho programa, es por ello que no se hará un apartado de requerimientos técnicos o software obligatorio.

* Software recomendado: Python 3.8.

## **Mecanismos de acceso**

En referente a las plataformas para las cuales esta disponible, debido a que es un prototipo sencillo, donde realmente la importancia y el causal de creación, es el curso anteriormente mencionado de “Investigación de Operaciones I”, pues su mecanismo actual y único es:

* Ordenador.

Pero fácilmente podría ser un aplicativo web, creando así las diferentes interfaces de usuario haciendo uso del código de la actual versión, adicionalmente extenderse a un aplicativo móvil utilizando algún entorno de desarrollo para ello que ofrezca el mercado.

# **Descarga**

La actual versión del aplicativo generado en Python corresponde a código libre disponible en GitHub, en el repositorio en el cual se encuentra en la portada de este documento ([Selecciones aquí para ir](#Repoo)) aunque para su edición se hace necesario la implementación de librerías como:

* **Tkinter**: Utilizado para generar la parte grafica de programa. [[1]](#REF1)
* **NumPy**: El cual ofrece toda la parte referente a cálculos cálculo numérico, a través, de su manejo y creación de vectores n-dimensionales. [[2]](#REF2)
* **CVXOPT**, Es una biblioteca gratuita de optimización convexa para Python, parte fundamental para la solución de problemas. [[3]](#REF3)

De las anteriores librerías, podrás encontrar su documentación en las referencias y en el respectivo orden mencionado en los links a continuación:

* **Tkinter**: <https://docs.python.org/es/3/library/tk.html>
* **Numpy**: <https://numpy.org/doc/>
* **CVXOPT**: <https://cvxopt.org/>

Actualmente solo personas con un IDLE de Python pueden utilizar el aplicativo generado, lo bueno de Python es que podemos generar ejecutables del programa para evitar la edición del programa, es decir que, muy sencillo (aunque no esta realizado) generar un ejecutable para que personas sin conocimiento en programación puedan usar dicho aplicativo, aunque actualmente no esté disponible en GitHub.

# **Modo de uso**

## **Restricciones:**

Al aplicativo generado se le brindo una importancia elevada a la parte de funcionalidad en comparación con la parte visual, es por ello que por el momento es necesario indicar las siguientes instrucciones:

* La aplicación solo recibe problemas de maximización, si desea resolver un problema de minimización debe poner los coeficientes de la función objetivo con signo contrario.
* Todas las restricciones están predefinidas como ≤, todavía no se tiene soporte para los demás tipos de desigualdad o igualdad, recuerde que puede invertir una desigualdad ≥ en ≤ invirtiendo todos los signos de los coeficientes de la desigualdad.

## **Ingreso de datos**

Al ejecutar el programa en el IDLE de Python, se brinda una interfaz la selección de los diferentes aspectos y valores del problema que se desea solucionar, la cual puede visualizarse en la Figura 1 que es presentada a continuación:

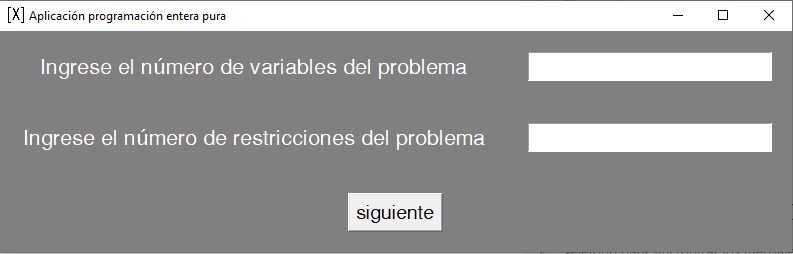


Figura 1. Interfaz para las características del problema

En la primera casilla el usuario ingresa la cantidad de variables que va usar, teniendo en cuenta que debe ser mayor a 1. Posteriormente se ingrese la cantidad de restricciones que van a limitar la función objetivo.

Una vez ingresados los datos de cantidad de variables y restricciones, se provee al usuario otra interfaz la cual se puede visualizar en la Figura 2, esta genera casillas iguales a las variables para definir la función objetivo, y las n restricciones por fila que el usuario ingreso anteriormente.

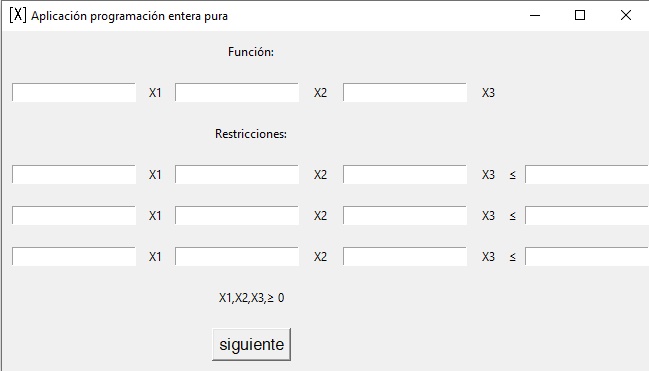


Figura 2. Definición de función objetivo y asignación de valores a las restricciones.

Tenga presente que las restricciones de enteridad no son asignables, se abordan solamente problemas de programación entera pura, si no se asignaran de manera automática podría generar problemas de interpretación. Al oprimir siguiente, el programa brinda la solución al problema brindado, vea la Figura 3 a continuación:

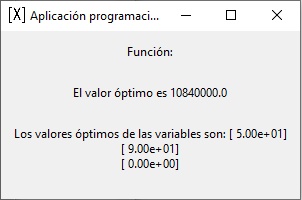


Figura 3. Interfaz de solución al problema por programación entera pura

Como podemos observar en la Figura 3, se brinda el valor optimo, y en el orden de las variables se brinda el valor de cada una para la solución del dicho problema.

# **Referencias**

[[1]](#Marca1) Python Software Fundation (2001-2020). Tkinter. Recuperado de:   
<https://docs.python.org/3/library/tk.html>

[[2]](#Marca2) NumPy (2008-2020) NumPy Documentation. Recuperado de: <https://numpy.org/doc/>

[[3]](#Marca3) Andersen. M. y Vandenberghe. L. (s.f.), CVXOPT. Recuperado de: <https://cvxopt.org/index.html>