**Proyecto Análisis y Diseño de Algoritmos 2021**

**Introducción**

Diseñar y analizar algoritmos con garantías de rendimiento comprobables permite la resolución eficiente de problemas de optimización en diferentes dominios de aplicación, por ejemplo, redes de comunicación, transporte, economía y fabricación por citar algunas.

Durante muchas décadas, los investigadores han estado desarrollando algoritmos cada vez más sofisticados para resolver problemas de optimización difíciles. Estos algoritmos incluyen enfoques de programación matemática, programación dinámica, entre otros.

Este proyecto discute cómo se pueden caracterizar adecuadamente las particularidades de una instancia de problema que tienen un impacto en la dificultad en términos de rendimiento algorítmico, y cómo se pueden definir y medir tales características para un cierto tipo de problemas de optimización.

**Planteamiento del problema**

Se entiende por Optimización Combinatoria a la rama de la matemática enfocada en el diseño y análisis de algoritmos eficientes para encontrar objetos óptimos en conjuntos finitos. Más formalmente se considera un conjunto finito DOM y una función f : DOM 🡪R, y se buscan los elementos x ∈ arg max(f) o x ∈ arg mın(f).

Considerando problemas de la forma anterior, es claro que estos siempre tienen solución dada la finitud del conjunto DOM. Pese a esto, un problema clave que aparece es el hecho que |DOM| es enorme, por lo que son demasiadas opciones a revisar. Es por esto que interesa diseñar un algoritmo que encuentre esta solución eficientemente. El gran tamaño del conjunto DOM nos llevará a describirlo de forma implícita.

El objetivo de este proyecto es conocer y aplicar a problemas teóricos y del mundo real, el paradigma de solución basado en la optimización combinatoria. El paradigma de solución de la optimización combinatoria descansa en fundamentos teóricos para modelar y representar problemas, y en algoritmos de búsqueda local para resolver dichos problemas. El fundamento teórico de la optimización combinatoria se apoya en la teoría de la complejidad computacional y en el uso de objetos matemáticos para representar problemas de optimización combinatoria. Los objetos matemáticos que se presentan incluyen; grafos, árboles, permutaciones, particiones, cadenas de crecimiento restringido, polinomios diagonales, polinomios caja, polinomios “mayor que”. Es altamente relevante el énfasis que se hace sobre la representación, generación, y enumeración. Los algoritmos de búsqueda local usan la modelización/representación basada en objetos matemáticos para resolver instancias de problemas teóricos y prácticos. Los algoritmos de búsqueda local incluyen: algoritmos de mutación/selección; algoritmo de recocido simulado; algoritmos genéticos; búsqueda tabú; y algoritmo de colonia de hormigas. Se busca modelar problemas de la vida real usando, por ejemplo, los objetos matemáticos estudiados, y resolver dichos problemas usando algoritmos de optimización combinatoria básicos o mezclas de ellos.

En particular para este proyecto, tratarán el problema de las particiones: bi-particiones (y posiblemente k-particiones). Este problema se estudiará en el marco de una solución ya establecida que se presentará más adelante en el documento.

**Objetivos**

Los objetivos del proyecto son:

-Caracterizar el o los conjuntos de datos, que se usarán para el problema.

-Comprender el planteamiento propuesto por PyPhi toolbox al problema de la partición que cumple con una función de optimización(la partición con el mínimo valor).

-Revisar soluciones planteadas a este problema desde la matemática y desde la algoritmia.

-Establecer comparativos entre las soluciones propuestas.

-Revisar como mejorar las limitaciones.

**Parte I**

**Tareas iniciales**

* Recolección y entendimiento del tipo de datos a manejar, a partir de ejemplos desarrollados.
* Puesta a punto de las soluciones a estudiar.
* Realización de pruebas y análisis de resultados.

**Generalidades**

En esta parte del proyecto introducimos PyPhi, un software en Python que implementa análisis causal y despliega la estructura causa-efecto de sistemas dinámicos discretos de elementos binarios. El software permite a los usuarios fácilmente estudiar estas estructuras y puede ser aplicado en investigación sobre complejidad, que es nuestro interés, entre otras cosas. La idea es revisar la funcionalidad empleando un sistema ejemplo, y luego entender los aspectos de diseño e implementación más relevantes.

PyPhi puede ser instalado en sistemas Linux y macOS con Python 3.4 o versiones superiores. Se puede ejecutar en Windows usando la distribución Anaconda de Python y además instalando PyPhi con Conda[[1]](#footnote-1). PyPhi es open-source y licenciado bajo GPLv3. El código fuente está en GitHub https://github.com/wmayner/pyphi

En el siguiente enlace está la fuente de toda la información:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6080800/>

He aquí lo que deben tener en cuenta:

1) El enlace lleva a un documento que describe todo el aplicativo que se va a probar. Aunque no es relevante la fundamentación sobre la información integrada, si es importante leer el documento desde la parte que empieza a describir el software para conocer los módulos que tiene y su arquitectura general.

2)Al inicio del documento en el enlace *Data Availability* encontraran el enlace en GitHub  para descargar el código y este enlace <https://pypi.python.org/pypi/pyphi>, sobre la librería de Python para calcular la información integrada.

3) Al final del documento está la sección *Supporting information,* y ahí podrán encontrar información sobre descargas, documentación, explicaciones y pruebas (que irán siendo revisadas progresivamente).

**Procedimiento**

Como en esta primera parte es necesario comprender tanto las características del problema a resolver, como el software presentado para ello, ya que ustedes trabajaran sobre un módulo en particular, es fundamental documentar cada paso del proceso para tener el software disponible y a punto para poder probarlo.

* Solución 1:
* Escojan el sistema operativo en el que trabajarán.
* Instalen todo el paquete de pyphi correspondiente
* En el material de estudio que esta en la pagina en "*Supporting* *information*",  el ***s1******file*** es el código de Pyphi que no está actualizado y tiene un  archivo de python de demostración donde indica cómo se obtiene la estructura causa-efecto, la partición mínima y otras cosas. Y en "Data Availability", está el enlace al código en github de PyPhi pero ya actualizado pero no presenta un archivo de demostración. Ustedes deben usar la versión actualizada y basarse en la demostración del ***s1******file,*** con el fin de puedan comprender mucho mejor como se hacen las particiones y como se consigue la partición de mínima información (MIP).



* Solución 2 (Matlab):
* Revisen cuidadosamente la información contenida en el siguiente enlace: <https://figshare.com/articles/software/phi_toolbox_zip/3203326/10>.
* Este enlace corresponde al de github

<https://sites.google.com/a/g.ecc.u-tokyo.ac.jp/oizumi-lab/codes>

* Con esta solución se debe seguir el mismo proceso que con la anterior, ya que con todas las aplicaciones luego se hará un comparativo.

1. Conda install -c wmayner pyphi [↑](#footnote-ref-1)