**UNIVERSIDAD DE SANTIAGO DE CHILE**

**FACULTAD DE INGENIERÍA**

**DEPARTAMENTO DE INGENIERÍA INFORMÁTICA**

|  |  |
| --- | --- |
| Nombre: | Fernando Cabrera |
| Carrera: | Mag. en Ingeniería Informática |
| Año estimado de egreso: | 2019 |
| E-mail: | fernando.cabrera@usach.cl |
| Profesor: | Mario Inostroza |
| Co-guía | Cristián Sepúlveda |
| Fecha presentación: | 26 de Mayo de 2017 |
| Nombre: | Rubén Cavieres |
| Carrera: | Mag. en Ingeniería Informática |
| Año estimado de egreso: | 2019 |
| E-mail: | ruben.cavieres@usach.cl |
| Profesor: | Mario Inostroza |
| Co-guía | Cristián Sepúlveda |
| Fecha presentación: | 26 de Mayo de 2017 |

**Índice de contenido**

[1](#_30j0zll) Introducción 4

[1.1](#_1fob9te) Descripción del Problema 4

[1.2](#_3znysh7) Formulación 4

[2](#_2et92p0) Marco Teórico 5

[2.1 Metaheuristica Utilizada: 5](#_tyjcwt)

[2.1.1 Simulated Annealing: 5](#_3dy6vkm)

[2.2](#_1t3h5sf) Estado del Arte 6

[3](#_4d34og8) Diseño de la Solución 7

[3.1](#_2s8eyo1) Estructura Base: 7

[3.1.1](#_17dp8vu) Algoritmo de simulated annealing: 7

[3.1.2](#_3rdcrjn) Parámetros: 8

[3.2](#_1ksv4uv) Instancias 8

[3.3](#_44sinio) Herramientas de Desarrollo 9

[4](#_2jxsxqh) Resultados 10

[5](#_3j2qqm3) Conclusiones 11

**INTRODUCCIÓN**

El desarrollo de las técnicas de optimización ha tenido un prolongado avance en el tiempo. Para ello, la aplicación de estas técnicas ha abordado problemas de amplia complejidad que se caracterizan por tener un costo alto en recorrer los espacios de soluciones. Para ello, se han construido variadas metaheurísticas capaces de generar buena calidad de soluciones que reduzcan considerablemente los tiempos orientados a la resolución de problemas de complejidad alta.

Actualmente estas técnicas matemáticas se aplican a muchas áreas y disciplinas, donde gracias al sostenido incremento en la capacidad de cómputo, se han desarrollado estudios en diversas casas de estudios, laboratorios y organizaciones para diseñar algoritmos meta heurísticos que puedan resolver un amplio espectro de problemas que requieren la optimización.

El objetivo de este trabajo es desarrollar el algoritmo para la meta heurística *Simulated Annealing*, para resolver el problema de asignación cuadrática QAP, donde se lidia con la disposición de instalaciones en las mejores ubicaciones, para así asegurar un costo mínimo.

En lo sucesivo, se hará una descripción del problema, la meta heurística de resolución, piezas clave del algoritmo codificado, resultados y conclusiones relevantes.

# Introducción

## Descripción del Problema

Se tiene el problema de asignar un conjunto de instalaciones a un conjunto de ubicaciones, donde el costo depende del flujo entre las instalaciones y la distancia entre las ubicaciones.

El objetivo es asignar a cada ubicación una instalación, de forma tal de minimizar el costo total de la operación.

## Formulación

Dadas dos matrices , donde es el flujo entre las instalaciones y , y la distancia entre las ubicaciones y . Sea n el número de instalaciones y ubicaciones y denotado :

Siendo el conjunto de todas las permutaciones

# Marco Teórico

## 2.1 Metaheuristica Utilizada:

### 2.1.1 Simulated Annealing:

La idea del algoritmo simulated annealing (SA) proviene de un trabajo publicado por Metropolis, en 1953.

El algoritmo en este trabajo simula el enfriamiento del material en un baño de calor. Este es un proceso conocido como annealing. Si se calienta un sólido hasta un punto de fusión y luego se enfría, las propiedades estructurales del sólido dependen de la velocidad de enfriamiento. Si el líquido se enfría lo suficientemente lento, se formarán grandes cristales. Sin embargo, si el líquido se enfría rápidamente (apagado) los cristales contendrán imperfecciones.

El algoritmo de Metropolis simula el material como un sistema de partículas. El algoritmo simula el proceso de enfriamiento reduciendo gradualmente la temperatura del sistema hasta que converge a un estado estable y congelado.

En 1982, Kirkpatrick et al (Kirkpatrick, 1983) tomaron la idea del algoritmo Metropolis y lo aplicaron a problemas de optimización. La idea es utilizar simulated annealing(SA) para buscar soluciones viables y converger a una solución óptima.

A partir de una solución inicial, SA procede en varias iteraciones. En cada iteración, se genera un vecino aleatorio. Si el movimiento mejora la función de costo, siempre se acepta. De lo contario, se selecciona al vecino con una probabilidad dada que depende de la temperatura actual y de la degradación ( de la función objetivo. (representa la diferencia en el valor objetivo(energía) entre la solución actual y la solución vecina generada. Si el algoritmo progresa la probabilidad de aceptar movimientos decrece. Esta probabilidad sigue la distribución de Boltzmann:



*Figura 2: Probabilidad de Boltzmann.*

2.2. Estado del Arte

Después de la década de los 80’s, el paradigma de resolución heurístico cambió. Aparecieron las metaheurísticas con sus principales características. Dentro de esta aparición, volvió a cobrar interés el análisis del problema de asignación cuadrático (QAP). (Loiola, Maia de Abreu, Boaventura-Netto, Hahn, & Querido, 2007)

Dentro de las diversas metaheurísticas basadas en fenómenos naturales, se encuentra Simulated Annealing, donde Burkard and Rendl (1984) propusieron una de las primeras aplicaciones de simulated annealing al QAP. Los siguientes autores fueron mejorando el modelo, incluyendo nuevos elementos como los componentes de equilibrio, el concepto de temperatura óptima, entre otros. (Burkard, Cela, Pardalos, & Pitsoulis, 1998)

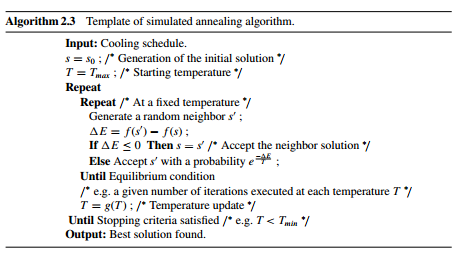
Otras aplicaciones relevantes de Simulated Annealing se encuentran en el campo de las redes neuronales, particularmente con los algoritmos perceptrón multicapa, que utilizan SA para generar entrenamientos que busquen escaparse de los mínimos locales, permitiendo disminuir el radio de error de entrenamiento de los pesos de la red que se esté entrenando. (IBM Knowledge Center, 2012)

# Diseño de la Solución

## Estructura Base:

### Algoritmo de simulated annealing:

El algoritmo construido en R para la solución del problema planteado en la descripción fue diseñado tomando las siguientes características del algoritmo de simulated annealing.



Las variables que son parametrizadas dentro del algoritmo se describen en la sección 3.1.2. La exploración de vecindario escogida en el archivo del algoritmo QAP corresponde al método SWAP.

### Parámetros:

Los parámetros que son considerados son:

* **Temperatura Inicial:** debe ser lo suficientemente caliente para permitir un movimiento a casi cualquier punto del vecindario. Si esto no se hace, la solución final será la misma a la solución inicial.
* **Temperatura Final:** es habitual dejar que la temperatura disminuya hasta que llegue a cero. Sin embargo esto puede provocar que un algoritmo funcione mucho más tiempo, por lo que es una buena acción considerar una temperatura final.
* **Número de iteraciones:** por lo general, es utilizado un número constante de iteraciones a cada temperatura.
* **Decrecimiento de la temperatura:** este parámetro es necesario debido que tenemos que pasar una temperatura inicial dada a una temperatura final y para eso la temperatura debe ir descendiendo. Para esta implementación, consideramos el método lineal para el decrecimiento de la temperatura.

## Instancias

Las instancias que serán utilizadas inicialmente son:

* N. Christofides and E. Benavent (14 instancias)
* C.E. Nugent, T.E. Vollmann and J. Ruml (15 instancias)

Además deben ser consideradas las siguientes instancias para la evaluación de resultados:

* Li and P.M. Pardalos [LiPa:92]

2.4. Herramientas de Desarrollo

El lenguaje de programación escogido fue R, el cual fue utilizado con los siguientes IDEs y software de ejecución de scripts:

* R x64 3.3.3
* RStudio 1.0.143

Este lenguaje es interpretado. Para estructurar el proyecto, se generaron los siguientes archivos:

* **simulated-annealing.R**: implementación de algoritmo de metaheurística. Utiliza qap.R para el cálculo de costos.
* **qap.R**: contiene código base para resolver problema de asignación cuadrática. Contiene las matrices de flujos, distancias, los métodos de obtención de flujos, distancias, ubicaciones de instalaciones y cálculo de costos (función objetivo) de distribución de instalaciones.
* **config.yaml**: configuraciones del algoritmo. Se establecen los valores de todos los parámetros disponibles para ejecutar algoritmo.
* **inputs/**: carpeta que contiene las instancias a utilizar por algoritmo. Cada instancia se compone de dos archivos: **<id\_instancia>-f.txt** y **<id\_instancia>-d.txt**, flujos y distancias correspondientemente.

# Resultados

Las siguientes ejecuciones se realizaron con las instancias de los autores N. Christofides y E. Benavent del sitio QAPLIB, donde se presentan junto a estas instancias los parámetros utilizados para ejecutar algoritmo y sus correspondientes resultados.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instancia | T° max | T° min | Repet. Por T° | Tipo Enfriamiento | Valor Costo Óptimo | Error (%) | Tiempo (Seg.) |
| chr12a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 20532 | 20.5 | 2.11 |
| chr12b | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 18545 | 0.94 | 1.83 |
| chr12c | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 23512 | 1.11 | 1.78 |
| chr15a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 26826 | 1.71 | 2.92 |
| chr15b | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 23915 | 1.99 | 2.78 |
| chr15c | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 29018 | 2.05 | 2.75 |
| chr18a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 27213 | 1.45 | 4.25 |
| chr18b | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 2072 | 0.35 | 3.96 |
| chr20a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 4050 | 0.85 | 4.84 |
| chr20b | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 4985 | 1.17 | 4.97 |
| chr20c | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 39359 | 1.79 | 5.17 |
| chr22a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 5624 | 0.09 | 5.71 |
| chr22b | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 5234 | 0.15 | 5.71 |
| chr25a | 500 | 0.01 | 6 | Lineal (ꞵ = 7.4) | 8574 | 1.26 | 7.41 |

Los resultados arrojan como resultado de error relativo promedio de 2.53% y tiempo promedio de 4.01 segundos.

Los parámetros que influenciaron en gran medida los resultados fue la temperatura inicial y la cantidad de iteraciones, ya que una mezcla de ambos afectaba directamente el tiempo de ejecución y la profundidad en la búsqueda del valor óptimo.

# Conclusiones

El desarrollo del algoritmo de la metaheurística *Simulated Annealing* presentó un desafío tanto en su construcción como en su utilización.

Su construcción significó leer documentación de implementación de sus componentes, los cuales son diversos, dentro de las cuales se puede nombrar la función de enfriamiento, que tiene diversas alternativas (lineal, geométrica, logarítmica, etc.).

Por su utilización, se presenta la complejidad de la cantidad de parámetros de personalización y el efecto que produce un ajuste en alguno de ellos, por lo cual dependiendo de la entrada de datos, evaluar qué valor de parámetro es mejor utilizar.

Por otro lado, se descubre lo interesante que es la estrategia de la metaheurística para evitar caer en óptimos locales, aceptando soluciones malas gracias a un cálculo de una probabilidad, lo que permite dar una posibilidad de investigar posibles mejores soluciones en un campo más amplio.

# Bibliografía

IBM Knowledge Center. (2012). *Alternated Simulated Annealing and Training (multilayer perceptron algorithms)*. Obtenido de https://www.ibm.com/support/knowledgecenter/SSLVMB\_21.0.0/com.ibm.spss.statistics.help/alg\_mlp\_training\_annealing.htm