Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Matemática

MA55G-1 Laboratorio de Modelamiento Matemático I

Optimización Multiobjetivo para Scheduling del LSST

Laboratorio de Astroinformática - CMM

Primavera 2015

Investigador a cargo
Francisco Förster.
Estudiante
Andrés Cristi.

Índice

1. Introducción

El presente documento tiene por objetivo registrar el trabajo desarrollado en el marco del curso MA55G-1 Laboratorio de Modelamiento Matemático I, que consistió en trabajar en el Laboratorio de Astroiformática del CMM, en particular en la Optimización del Scheduling del LSST.

El LSST es un telescopio que se está construyendo en el norte de Chile, en la región de Coquimbo, cuya particularidad es el gran campo angular que es capaz de observar, con una alta resolución. Dicha característica le permite en pocos días fotografiar por completo el cielo observable y, por lo tanto, muchas veces (alrededor de 1000) durante los 10 años que dura el proyecto.

El proyecto además busca cumplir con una serie de objetivos científicos, entre los cuales no hay una jerarquía, por lo que se hace necesario trabajar con optimización multiobjetivo. Además, la optimización está sujeta a la incertidumbre que impone el clima. Por lo anterior, el principal objetivo fue el probar el desempeño en este problema de la metaheurística Ant Colony Optimization, la cual es sensible a cambios.

2. Descripción del problema

El problema es generar planes de observación de 10 años para el LSST que sean pareto eficientes para las funciones objetivo de los objetivos científicos del proyecto. Se requiere además que se adapte de manera eficiente a los cambios de escenario dados por el clima. Es importante enunciar la dificultad que genera el que los puntos del cielo se mueven y por lo tanto su aporte a las funciones objetivo y su factibilidad cambien con el tiempo.

2.1. Funciones Objetivo

Los planes de observación deben satisfacer varios objetivos científicos de distintos grupos, los cuales no están expresados en forma funcional, pero dependen fuertemente de dos características importantes:

a) Observar del orden de 1000 veces cada punto, homogéneamente en el tiempo. Esto se traduce en observar aproximadamente cada 4 días cada punto factible.

b) Generar imágenes de buena calidad (atravesar poca atmósfera). Esto se traduce en que los puntos estén cerca del cenit y que no haya nubes que interfieran la observación.

Importante es aquí notar que por el tiempo de exposición que requiere cada observación y la velocidad de movimiento del telescopio, es preferible observar puntos cercanos del cielo para aumentar la cantidad de observaciones.

2.2. Factibilidad

Los puntos a observar deben cumplir las siguientes restricciones, las cuales cambian con el tiempo:

a) Estar 45^o por sobre el horizonte para cumplir un mínimo de calidad en la imagen.

b) Estar a cierta distancia de la luna, dependiendo de la fase, debido a su alta luminosidad.

2.3. Impredictibilidad

El clima (humedad y nubosidad) afectan las observaciones y por tratarse de fenómenos de naturaleza impredecible, introducen aleatoriedad al problema. Este efecto puede incluirse en la calidad de las observaciones o en la factibilidad de los puntos. Bajo ambas alternativas, es importante que exista un mecanismo con el cual los planes se adapten a dichos cambios. Este aspecto del problema no fue abordado en el trabajo desarrollado, sin embargo el objetivo de probar la aplicabilidad de Ant Colony Optimization es su capacidad de adaptación a los cambios en el escenario.

3. Abordaje del problema

En un comienzo se hizo la siguiente planificación para abordar el problema:

1) Generación de planes factibles para una noche, sin incertidumbre.

2) Programar y probar heurística para optimización multiobjetivo: Algoritmos genéticos, Ant Colony Optimization.

3) Generar planes para 10 años.

4) Añadir incertidumbre.

De estas tareas, las que se abordaron durante el semestre fueron las dos primeras.

3.1. Simplificación del problema

Para poder trabajar computacionalmente el problema con los recursos e información disponibles, se hicieron las siguientes simplificaciones:

- Dos funciones objetivo a maximizar:
  - Suma de ángulos de altitud (calidad de la imagen) de cada vez que se visita un punto.

  - Suma de función creciente de tiempo desde última visita (2^t) de cada vez que se visita un punto.

- Discretización del cielo en coordenadas (RA,DEC), mediante el método Healpix. El nivel de la discretización para el ángulo de observación del telescopio es npix= 32. Sin embargo, para probar el desempeño de A.C.O. se usó npix= 16 (un nivel más bajo) para poder procesar en tiempo razonable.
- Discretización de la noche en intervalos de tiempo para las distancias, factibilidad y calidad. Aunque los métodos desarrollados funcionan para cualquier cantidad de intervalos, se usó siempre 10.
- No se consideró la fase de la luna. Siempre se consideró que un punto debía estar a al menos 40^o del centro de la luna.

3.2. Generación de planes factibles

Para la generación de planes factibles, se trabajó en tres funciones importantes que se encuentran en los archivos `factibles5.py` y `functions.py`: `conversion`, `factibles` y `Time_Dist`.

- `conversion`: Transforma para un observador dado coordenadas ecuatoriales (DEC,RA) en coordenadas horizontales (AZ,ALT).
- `factibles`: Dado un arreglo de puntos en coordenadas ecuatoriales (DEC,RA) y un observador, decide cuáles son factibles. En `factibles5.py` devuelve las coordenadas de los puntos, y en `functions.py` devuelve un arreglo del mismo largo que el original con ceros en los puntos infactibles y unos en los factibles.
- `Time_Dist`: Recibe un punto y un arreglo de puntos, ambos en coordenadas ecuatoriales (DEC,RA), y entrega un arreglo con el tiempo que se demora el telescopio en moverse desde punto a cada punto del arreglo de puntos. Tiene como parámetros las velocidades del telescopio.

Con lo anterior, se programó la función `Factible_Schedule` para generar planes factibles en una noche, relativamente buenos para las funciones objetivo definidas. Recibe un arreglo de puntos en (DEC,RA), un observador y los tiempos desde la última visita, y entrega un plan en (DEC,RA). También entrega las coordenadas en (AZ,ALT) al momento de observación de cada punto. Para ello se consideró un análogo al algoritmo de Nearest Neighbour (estrategia glotona), usando la función:

$$\text{Valor}_{ij} = \frac{\mathbb{1}_{T_j \geq 3{,}65} T_j}{\text{ZEN}_j + \text{DIST}_{ij} + 1}$$

Que es una función creciente sobre el tiempo desde la última visita y decreciente en el ángulo cenital (creciente en la altitud) y en la distancia (en tiempo de movimiento). Para generar una población de soluciones factibles, el algoritmo elige al azar una entre las mejores 6 alternativas para Valor_{ij}.

Para el observador en todas las funciones se usó una instancia de `observer` de la librería `pyephem` que incluye una posición y un tiempo. El tiempo está medido en días para ser consistente con `pyephem`. Esta librería permite además calcular la posición de la luna en cada momento y los tiempos de salida y puesta del sol.

3.3. Visualización

Se programó además el script `animacion2.py` que dado un arreglo de puntos en coordenadas (AZ,ALT) muestra una animación graficando dichos puntos en un gráfico de coordenadas polares (ALT en el radio, AZ en el giro). Esta herramienta fue muy importante para chequear la correctitud de lo que se estaba haciendo.

3.4. Ant Colony Optimization

Se utilizó la metaheurística Ant Colony Optimization, de la cual se puede encontrar más información en [1] y [2]. En particular se usó una versión adaptada a multiobjetivo de Ant Colony System, que se programó en la clase `ACOSchedule` que se encuentra en el archivo `ACO1.py`. Este incluye comentarios que detallan el funcionamiento y sintaxis de los métodos programados.

El algoritmo sigue el siguiente esquema de funcionamiento. En todo momento hay una lista de planes no dominados y matrices de feromonas:

1. Se genera una cierta cantidad de planes factibles. Se guardan los planes no dominados.
2. Se corre una iteración de $m = 10$ hormigas. Se guardan si es que hay los planes no dominados (se eliminan los que resulten dominados).
3. Se actualizan las matrices de feromonas según los planes no dominados.
4. Se vuelve al segundo paso.

A continuación se detalla cómo funciona cada paso y los elementos que se usan:

Matrices de feromonas: Son matrices que relacionan los puntos del cielo con un cierto valor. Hay una por cada función objetivo y por cada intervalo de la discretización de la noche.

Factibilidad, Distancias y Calidad: Se calculan al inicializar un objeto de la clase. Son arreglos y vectores por cada intervalo de la discretización de la noche.

Planes factibles: Se utiliza el método anteriormente explicado, reprogramado en la función `ACOSchedule.SuperAnts`.

Hormigas: Se encuentran programadas en la función `ACOSchedule.Ants`. Recorren puntos del cielo factible hasta acabarse la noche, usando la siguiente regla: desde el punto i pasan al punto

$$j = \begin{cases} \text{argmax}_k p_{ik} & \text{si } q \leq q_0 \\ J \sim (p_{ik})_k & \text{si } q > q_0 \end{cases}$$

con

$$p_{ij} = \frac{\tau_{ij}\eta_{ij}^{\beta}}{\sum_{k \text{ Factible}} \tau_{ik}\eta_{ik}^{\beta}}$$

Donde $q_0 = 0{,}9, \beta = 2$ son constantes heurísticas tomadas de [1], q es una v.a. uniforme en $[0, 1]$, τ es la matriz de feromonas, la cual es elegida entre ambas en cada paso usando una v.a. bernoulli($\lambda = 0{,}5$) y η es un valor heurístico dado por la información que se tiene de los puntos. Este último se tomó en general de forma parecida a la función Valor_{ij}.

Luego de que una hormiga pasa por cada arista, descuenta una cierta cantidad de feromonas para aumentar la exploración, de la forma (con $\tau_0 = 1, \xi = 0{,}1$):

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0$$

Actualización de feromonas Luego de cada iteración, se actualizan ambas matrices de feromonas de la siguiente forma, para cada solución no dominada BS encontrada hasta el momento:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau^{BS}, \text{ para } (i, j) \in BS$$

Donde $\rho = 0{,}1$ es un parámetro heurístico y $\Delta\tau^{BS}$ es el coeficiente entre el valor de la función objetivo de BS correspondiente a la matriz de feromonas considerada, y el de la primera solución encontrada por el algoritmo.

4. Resultados

4.1. Plan factible

La figura (1) muestra una captura del video de un plan generado con el script `factibles5.py`:

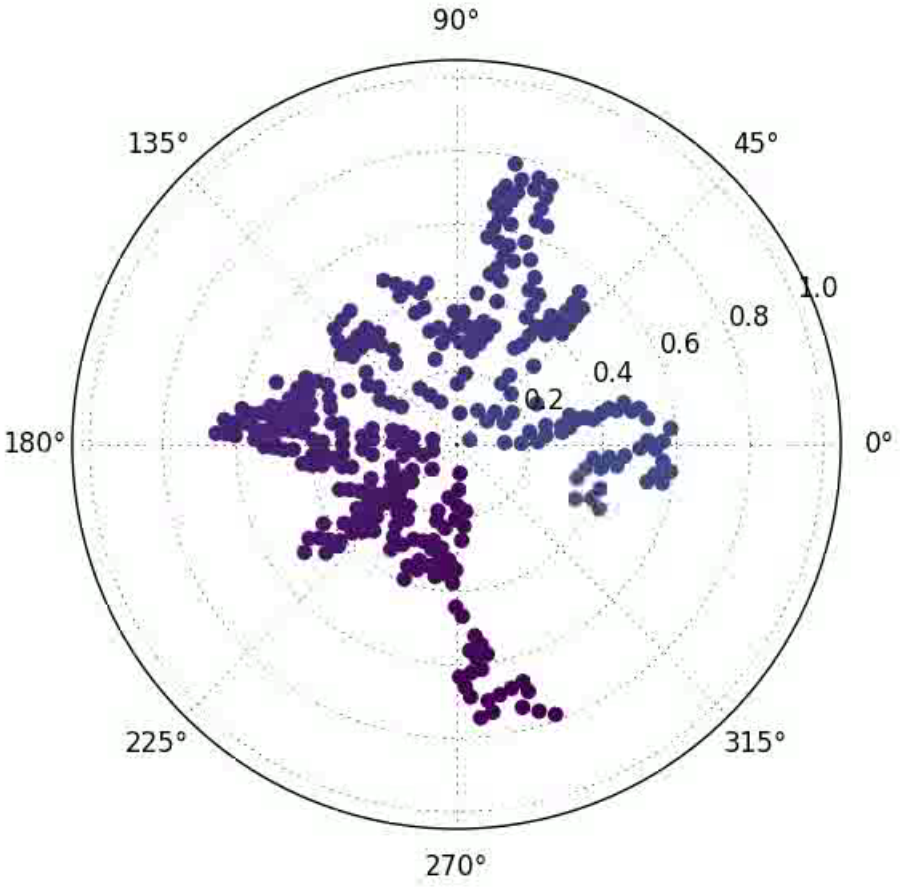

Figura 1: Plan factible generado con `factibles5.py`.

4.2. Ant Colony Optimization

A continuación se muestra el historial de las soluciones no dominadas encontradas después de mil iteraciones de $m = 10$ hormigas, bajo distintas combinaciones de los parámetros. Se ha variado el parámetro ρ y el valor η. En todos se usó una discretización de Healpix nside= 16 y 10 intervalos de tiempo. El color indica el orden en que aparecieron las soluciones, de azul a rojo y el gráfico muestra la calidad de observación en las abscisas y el tiempo entre observaciones en las ordenadas.

Se buscó una combinación que generara mejores soluciones.

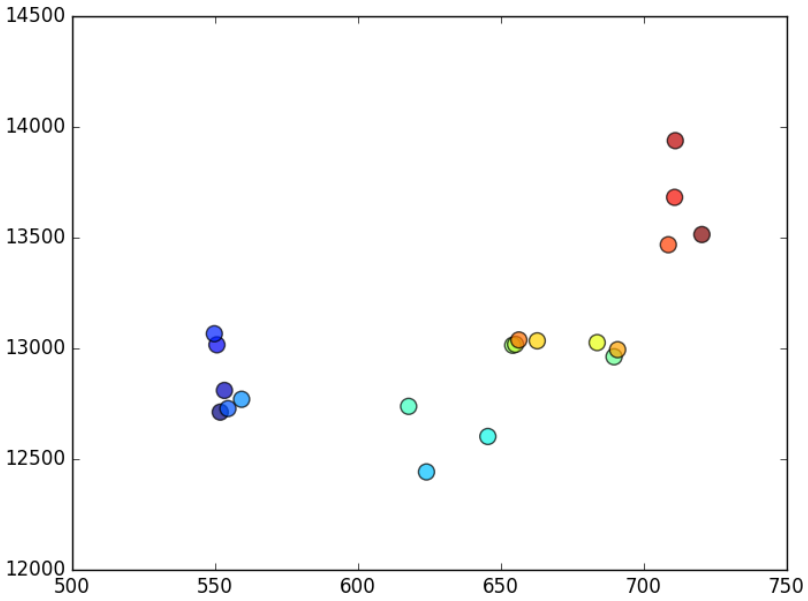

Figura 2: Soluciones no dominadas con $\eta = 1/D$, $\rho = 0{,}1$.

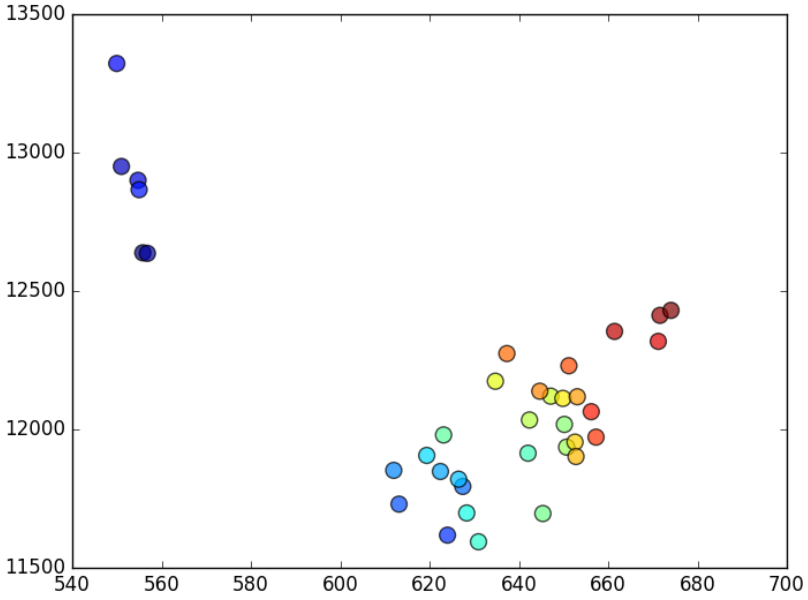

Figura 3: Soluciones no dominadas con $\eta = T/(D \cdot ZEN), \rho = 0{,}1$.

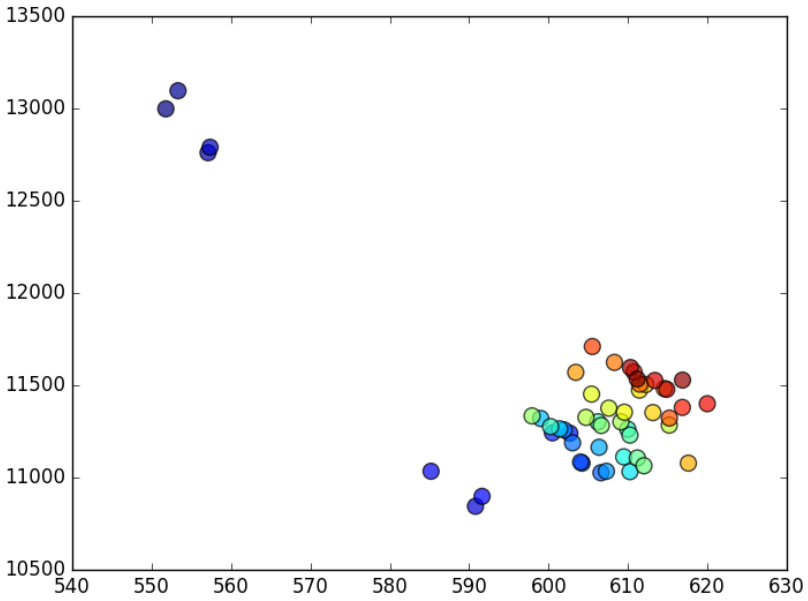

Figura 4: Soluciones no dominadas con $\eta = 2^T/D^2 \cdot ZEN$.

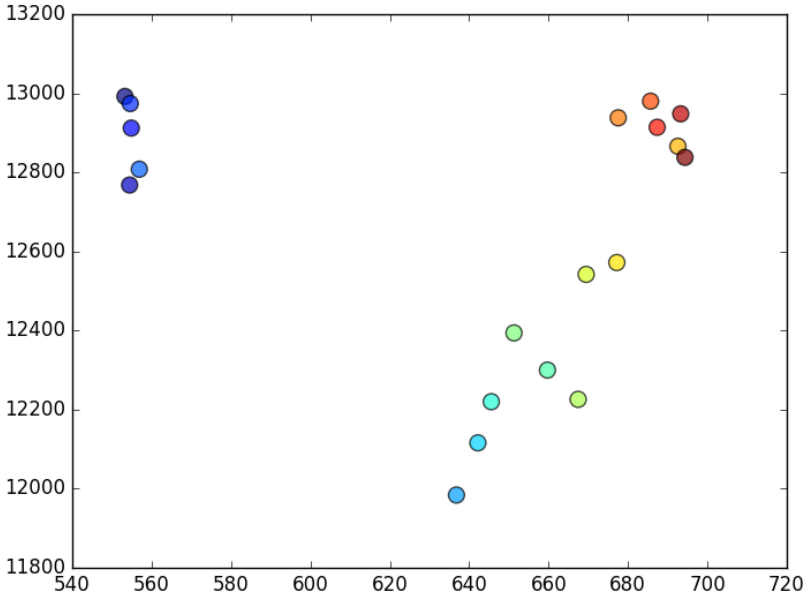

Figura 5: Soluciones no dominadas con $\eta = \frac{(2^T+128)}{192} \cdot \frac{3\pi}{2(ZEN+\pi)} \cdot \frac{1}{D}, \rho = 0{,}1$.

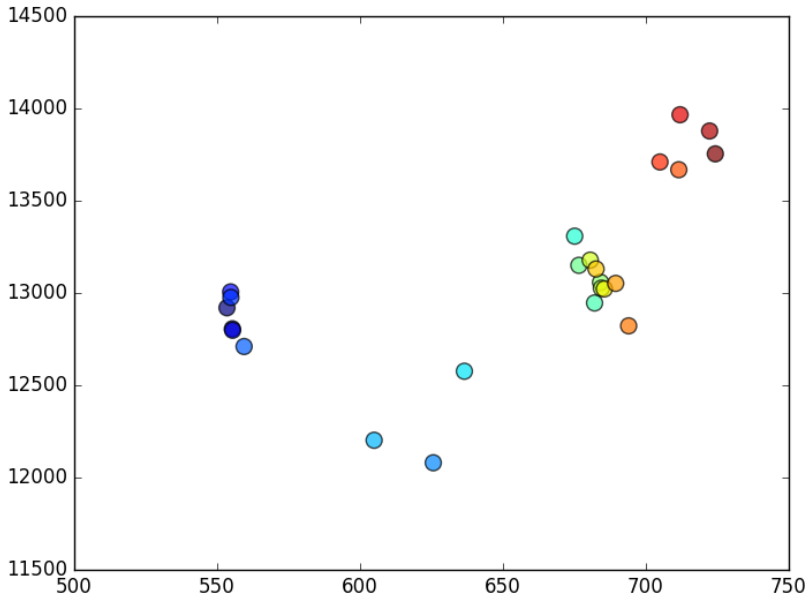

Figura 6: Soluciones no dominadas con $\eta = \frac{(2^T+128)}{192} \cdot \frac{1}{D}, \rho = 0{,}3$.

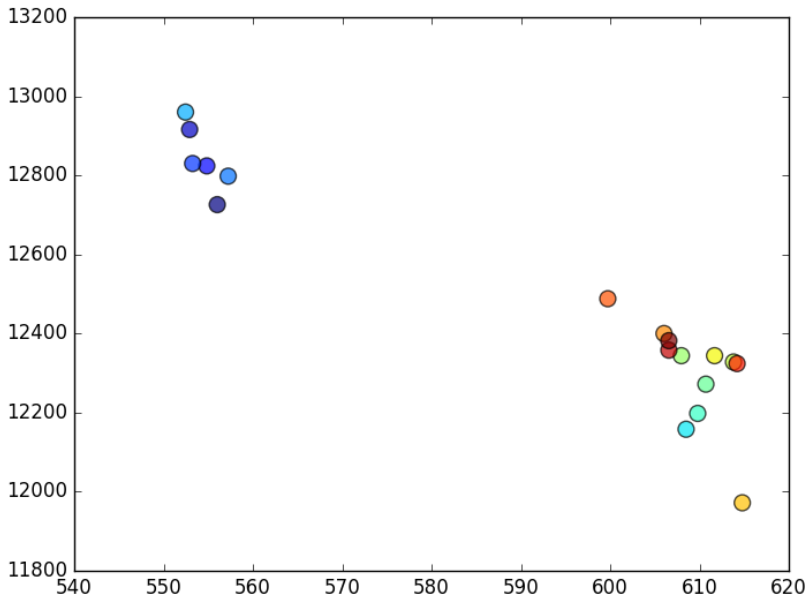

Figura 7: Soluciones no dominadas con $\eta = 1, \rho = 0{,}1$.

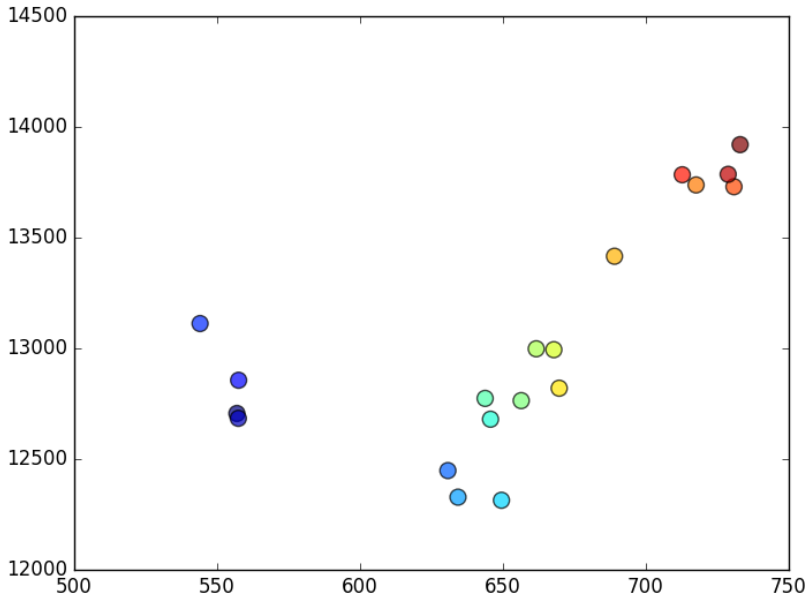

Figura 8: Soluciones no dominadas con $\eta = \frac{(2^T+128)}{192} \cdot \frac{1}{D}, \rho = 1$.

4.3. Análisis

Podemos notar que las formas de agregar la calidad que se probaron sesgan de una manera no deseada el proceso. Además, que el efecto más notable lo hace el sesgar con la distancia, probablemente porque permite hacer más observaciones.

El variar el parámetro ρ no parece alterar demasiado los resultados. Sí se notó que en la última prueba prontamente (en los primeros cientos de iteraciones) dejaron de aparecer nuevas soluciones no dominadas y al reiniciar el proceso guardando las soluciones se vuelve a encontrar más. Esto sugiere que podría aportar cada tanto el reiniciar las matrices de feromonas.

Se nota en los gráficos que se logra aumentar en 50 % el valor alcanzado en la calidad de observación. En la función de tiempo entre visitas hay un aumento considerable aunque no de la misma magnitud que la calidad.

4.4. Discusión

Entre los límites de estos resultados se encuentran:

- Se debe usar la siguiente discretización de Healpix, que contempla alrededor de 6000 puntos en lugar de los alrededor de 1500 usados.

- Sólo dos funciones objetivo podrían no reflejar la verdadera complejidad del problema de tener varias funciones objetivo.
- Las funciones objetivo usadas siguen valores locales, lo que no necesariamente se cumple con las verdaderas.
- No es tan claro cómo afecta la extensión a 10 años a los métodos utilizados.

Algunas extensiones de lo hecho en este trabajo son:

- Probar nuevas configuraciones de parámetros, variando los que no se variaron esta vez.
- Usar un lenguaje de más bajo nivel para optimizar el uso de los recursos computacionales.
- Probar modificaciones del algoritmo, como reiniciar cada cierta cantidad de iteraciones las matrices de feromonas.
- Variar con el tiempo el valor de λ para poder explorar mejor el frente de Pareto.

5. Conclusiones

- Discretizar la noche es un paso importante para aumentar la velocidad del programa.
- ACS para el esquema de una noche sin incertidumbre permite encontrar buenas soluciones.

Referencias

[1] M. Dorigo and T. Stützle. *Ant Colony Optimization*. A Bradford book. BRADFORD BOOK, 2004.

[2] Manuel López-Ibáñez and Thomas Stützle. Automatic configuration of multi-objective aco algorithms. In *Swarm Intelligence*, pages 95–106. Springer, 2010.