CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL
DEPARTAMENTO DE COMPUTACIÓN

# Técnicas evolutivas multiobjetivo aplicadas en el diseño de rutas en vehículos espaciales

Tesis que presenta

**Mario Augusto Ramírez Morales**

Para obtener el grado de

**Maestro en Ciencias**

En la especialidad de

**Computación**

Director de la Tesis: **Dr. Carlos A. Coello Coello**

México, D. F., a 21 de Noviembre del 2007

II

Great people talk about ideas,
average people talk about things,
small people talk ... about other people.

# Agradecimientos

Agradezco ..

# Índice general

# Índice de figuras

# Indice de tablas

# Lista de Algoritmos

## Resumen

La solución de problemas con múltiples objetivos, mejor conocidos como multiobjetivo, es un área en constante desarrollo. Ésta ha representado un reto, en numerosos y muy diversos sentidos, para los expertos en la materia por un largo tiempo. La computación evolutiva engloba a un conjunto de heurísticas inspiradas en el principio de la "supervivencia del más apto" de la teoría de la evolución de Darwin. Los algoritmos evolutivos se han usado para resolver problemas de optimización en una amplia variedad de aplicaciones, que van desde minimizar la cantidad de material utilizado en la fabricación de un producto, hasta el diseño de vehículos espaciales. El uso de algoritmos evolutivos para resolver problemas multi-objetivo se ha vuelto muy popular, sobre todo en los últimos 10 años.

Una misión espacial involucra un conjunto de sistemas que interactúan para poder realizar el objetivo para el cual fueron creadas. Un problema en particular de esa larga lista, que además resulta ser de vital importancia y a la vez muy complicado es el diseño de las trayectorias que deben seguir los vehículos espaciales. Otro problema de singular importancia está relacionado con la utilización de los recursos limitados con los que cuentan todas las misiones espaciales. Por lo expuesto anteriormente es necesario optimizar los recursos con los que se cuenta en el vehículo, como son el alimento y el combustible.

El problema de encontrar la ruta óptima en un vehículo espacial de bajo impulso, consiste en determinar cuál es el camino que requerirá menor tiempo y consumo de combustible para lograr nuestro objetivo, el cual es llegar desde una órbita en particular a otra. La dificultad para hacer esto radica en la complejidad para realizar una sola evaluación de la función objetivo, además de la alta dimensionalidad del problema planteado.

En el presente documento de tesis se plantea un mecanismo para encontrar buenas soluciones al problema de diseño de trayectorias de vehículos espaciales utilizando un algoritmo genético multi-objetivo paralelo. Se optó por una solución paralela debido a la complejidad computacional que plantea encontrar buenas soluciones en un espacio de búsqueda considerablemente grande.

## Abstract

Multiobjective optimization problems is a field that change and development. It represents a challenge in many ways to scientists and experts in this field. Evolutionary Computating is the collective name for a range of problem-solving techniques based on principles of biological evolution, such as natural selection and genetic inheritance, proposed by Darwin. Evolutionary algorithms has been used to solve a numerous set of optimization problems in an extensive variety of applications, such as design and manufacturing products to spacecraft design. Evolutionary algorithms becomes very popular to solve multiobjective optimization problems since 1990's.

An space mission involves a set of very complicated systems that interacts between themselves to achieve de goal for which they were designed. One of this systems, so far complicated but extremly complicated, is spacecraft's trayectories. Another important problem is related with the consume of limited resources. For these reasons, its necesary optimizing resources as food and propellant mass.

Finding optimal trayectory of low-thrust spacecraft is very complicated in many and diferent ways, such as objective funtion evaluation time and high dimensional data. This problem is a trade-off between propellant mass consumption and flight time in order to acheive de goal of de espace mision which is arrive to a diferent eliptical orbit.

Present document explain the mechanism to deal and find good solutions to the problem of low-thrust orbit transfer. Because high dimensional data, function time evaluation and complexity of problem we used parallel model of multiobjective genetic algorithm to solve it.

# Introducción

Desde hace varias décadas, los vuelos interplanetarios se han convertido en uno de los proyectos más ambiciosos de todo el mundo, en el cual se involucran gobierno, empresas privadas e instituciones educativas desarrollando nuevas tecnologías para el logro cabal de los objetivos de las misiones espaciales.

Esto no sería posible de no tener un objetivo superior en mente, en el cual se encuentran interesados de diferente forma cada una de las partes involucradas. En algunos casos, los objetivos son meramente de caracter científico. Sin embargo, de algunos años atrás a la fecha se han propuesto proyectos relacionados con la utilización de nuevos materiales, generación de cultivos en condiciones adversas y prueba de nuevos sistemas de propulsión para todo tipo de vehículos, los cuales tendrán un beneficio económico para el que pueda desarrollarlos.

Los sistemas de navegación para vehículos espaciales son muy complejos y costosos, sin embargo son indispensables ya que se deben usar los recursos de manera eficiente y racionarlos en todo momento que esto sea posible. Es por ello que se han desarrollado nuevos sistemas de propulsión que no están basados en compuestos químicos, lo cual reduce el consumo del combustible necesario para realizar sus tareas básicas.

Un ejemplo de estas nuevas tecnologías son los sistemas de propulsión de bajo impulso, basados en motores iónicos. Los motores de bajo impulso, son máquinas que tienen un empuje menor a $0{,}01$g (g=constante de gravitación de la Tierra, $9{,}81\frac{m}{s^2}$). Estas máquinas han demostrado recientemente su utilidad en misiones espaciales. El lanzamiento exitoso del *Deep Space 1* fue el primer proyecto que usó este tipo de tecnología como medio de propulsión principal el 24 de Octube de 1998. Fue diseñado para probar las nuevas tecnologías, las cuales incluían un sistema de propulsión de bajo impulso de Xenón. Dicho sistema estuvo en uso durante 677 días y fue utilizado de manera continua durante varios meses para que alcanzara la misma aceleración que un motor de propulsión química. Actualmente existen

otras misiones en puerta, como son el caso del lanzamiento de las misiones Dawn y Jimo, las cuales contarán con sistemas de propulsión de bajo impulso. La primera de ellas tiene como objetivo caracterizar las condiciones y procesos del sistema solar en sus inicios, investigando en detalle dos de los más grandes protoplanetas que aún se mantienen intactos desde su formación. Estos cuerpos son conocidos como Ceres y Vesta y se encuentran ubicados en el cinturón de asteroides localizado entre los planetas Marte y Júpiter junto con muchos otros cuerpos de menor tamaño. Mientras que la misión espacial JIMO fue diseñada con la finalidad de poder explorar las lunas congeladas de Júpiter. En particular se tenía como objetivo Europa, ya que existía la sospecha de que contaba con un océano en donde es probable que hubiera vida. Ahora se consideran también Ganímedes y Calisto ya que se piensa que existen océanos salados debajo de sus superficies congeladas, por lo que también se consideran como objetivos de interés para la comunidad científica.

La distancia media de la órbita de la Tierra a Júpiter es de 588 millones de kilómetros, por lo que es indispensable optimizar el uso del combustible. Otro asunto que resulta muy importante son las condiciones en las que se encuantren los operadores del vehículo. Mientras más tiempo se encuentren en el espacio es más probable que sufran de trastornos físicos y psicológicos que pueden poner en peligro su vida y la de sus compañeros, por lo que se deben realizar estas misiones en el menor tiempo posible. De los dos puntos planteados surge el problema de propulsión para vehículos de bajo impulso el cual será abordado en este documento.

En el capítulo 1 se da una explicación breve de los conceptos básicos de la Computación Evolutiva. En el capítulo 2 se explican conceptos básicos indispensables para poder entender el problema de rutas de vehículos espaciales. El capítulo 3 explica los conceptos relacionados de los problemas multiobjetivo. En el capítulo 4 se explica el problema a atacar. El capítulo 5 es la recapitulación de los capítulos anteriores donde se plantea la solución propuesta. En el capítulo 6 se muestran los resultados obtenidos con dicha propuesta. Finalmente, se presentan las conclusiones y algunas de las posibles rutas a seguirse para trabajo futuro.

# Capítulo 1

# Computación evolutiva

La amplia gama de problemas de optimización que existen en el mundo actual, han hecho necesario el desarrollo de nuevos y más poderosos métodos de optimización. Dentro de estas nuevas técnicas, el uso de metaheurísticas ha tenido un enorme auge en los años recientes [7].

Dentro de las metaheurísticas, la computación evolutiva se ha destacado, debido a su simplicidad de uso y a su enorme aplicabilidad. Esto la ha ubicado en una situación privilegiada, convirtiéndola en una de las técnicas modernas de optimización más populares de la actualidad.

En este capítulo se intenta dar un bosquejo general de la computación evolutiva, incluyendo sus orígenes, las diferentes áreas que la nutren de problemas muy complicados de resolver, las ventajas que brinda y algunas de las desventajas que la aquejan, así como una revisión de los conceptos principales que se requieren para hacer este documento auto-contenido.

## 1.1. Introducción al cómputo evolutivo

La computación evolutiva es un área de las ciencias de la computación en la que se estudian y aplican diversos principios basados en el mecanismo de la selección natural que rige la evolución de las especies como una técnica para resolver problemas del mundo real [13].

No es una coincidencia que los científicos hayan escogido precisamente la evolución como fuente de inspiración, ya que dicho mecanismo ha demostrado, a lo largo de los años, su efectiva aplicabilidad en una amplia gama de problemas. La idea principal sobre la que descansa la computación evolutiva es la manera en la que los organismos vivos se van adaptando de manera progresiva a su ambiente, dependiendo de los cambios que en éste

ocurran y de sus propias necesidades para poder subsistir [27].

Podemos considerar el proceso evolutivo como un mecanismo de adaptación y supervivencia de los individuos más aptos de una población dada, dentro de cierto medio ambiente. Para lograr esto, los individuos de dicha especie deben competir entre sí para determinar cuáles de ellos son los mejores, y por ende los que pasarán su información genética a sus descendientes. La competencia consiste en la evaluación de qué tan bueno o exitoso es un individuo dentro de su nicho, y nos dice, de igual forma, cual es su probabilidad de poder reproducirse y sobrevivir [10, 13].

El proceso de reproducción es una constante lucha entre los miembros de una población y puede o no incluir la cruza entre individuos de la misma especie (reproducción sexual). Pongamos como ejemplo el caso de los elefantes marinos, los cuales se encuentran divididos en 4 grupos que se distribuyen en diferentes partes del mundo. Durante la temporada de celo, cientos de machos se agrupan en las costas con un solo objetivo: preñar al mayor número de hembras que les sea posible, y así continuar con la interminable cadena que han seguido sus antecesores durante cientos de años. Sin embargo, hacerlo no es sencillo ya que para el elefante marino, uno de los animales más territoriales del planeta, esto requiere de una gran fuerza y agresividad para proteger a su grupo de hembras. La selección natural a lo largo del tiempo ha favorecido las características de fuerza, tamaño y vigor sexual para dar una ventaja competitiva a los individuos de dicha especie.

Figura 1.1: Elefantes marinos del Antártico.

Si tuviéramos que hacer una analogía entre la computación evolutiva y la evolución de las especies podríamos decir que el problema es el medio ambiente. Los individuos de la especie son los vectores de las soluciones encontradas y la aptitud de los individuos para sobrevivir y reproducirse sería la función a evaluar que nos dice qué tan buena es una solución candidata.

## 1.2. Un vistazo a la historia

Pocas son las ocasiones en las que una idea puede cambiar de manera tan sustancial la concepción que tenemos del mundo que nos rodea; éste es precisamente el caso de la evolución de las especies. Pero, ¿qué es una especie? Una especie es una unidad funcional de reproducción en la que están divididos los seres vivos. Las especies actuales son las predecesoras de las que existieron en el pasado y el proceso que tuvieron que recorrer para transformarse desde sus orígenes hasta nuestros días es conocido como **"Proceso Evolutivo"**. De esta forma, podemos decir que los organismos vivos actuales debieron tener ancestros comunes. Por ejemplo, el hombre comparte un ancestro común con los grandes primates, el cual vivió aproximadamente hace unos 13 millones de años; este esqueleto encontrado cerca de Barcelona, conocido como "Pierolapithecus Catalaunicus", perteneció a un hombre mono de unos 35 kgs. de peso y 1.20 m. de altura. La principal evidencia que nos dejó acerca de su relación con los chimpancés, orangutanes, gorilas y el hombre mismo, son su tórax ancho y aplanado, los omóplatos en la espalda y no a los costados como los monos. Asimismo, la sección final de la columna vertebral es corta y rígida, y tiene una cara corta, entre otras características que sólo comparten los grandes primates [10, 36].

Figura 1.2: Evidencia de la bipedestación en el ser humano (Pierolapithecus Catalanicus).

El párrafo anterior sólo se puede entender bajo los principios de la *Evolución*. Es decir, no podríamos comprender las propiedades que distinguen a una especie, ni la forma en que se han adaptado a su entorno los individuos de ésta si hacemos caso omiso de la historia [3, 33].

Aunque las ideas de Darwin no eran completamente desconocidas en el

siglo XIX, no fue sino hasta que este notable personaje escribió la obra por la cual es recordado, "El origen de las especies", que se sentaron definitivamente las bases de lo que hoy conocemos como "teoría de la evolución". Previo a él, existen los trabajos realizados por un sinnúmero de personajes de los cuales sólo destacaremos los que a nuestro parecer tuvieron las aportaciones más importantes, sin demeritar las investigaciones hechas por sus colegas.

Empezaremos esta breve reseña histórica hablando del que es considerado como uno de los fundadores de la biología. Nos referimos a Aristóteles, uno de los más grandes filósofos del mundo antiguo. Aristóteles propuso a la generación espontánea como el origen de las especies a partir de ciertas fuerzas (Entelequia) capaces de brindar vida a todo aquello que no la tenía. Esta idea no la compartían la mayoría de las personas de su época ya que se contraponía en cierta forma a las doctrinas expuestas por la iglesia. Sin embargo fue sostenida en ciertos círculos intelectuales por casi 20 siglos sin muchos cambios, hasta el siglo XVIII cuando los pensadores modernos empezaron a hacer su aparición [1].


Figura 1.3: Aristóteles de Estargia

Para 1679, un comerciante y científico holandés llamado Anton van Leeuwenhoek, después de haber introducido ciertas mejoras en la fabricación de microscopios, pudo observar los gametos masculinos (espermatozoides). De esta forma, y no antes de realizar ciertas observaciones y llegar a sus propias conclusiones, estuvo en condiciones de encarar la teoría de la

generación expontánea, al indicar que ciertos animales como las pulgas y escarabajos de la familia Curculionidae provenían de pequeñísimos huevos y no surgían naturalmente de los granos de avena y trigo, como se creía en ese entonces [19].


Figura 1.4: Anton van Leeuwenhoek.

A mediados del siglo XVIII, Georges Louis Leclerc, naturalista, matemático, biólogo, cosmólogo y escritor francés mejor conocido como el conde de Buffon, puso en jaque a la iglesia al hablar sobre la inexplicable similitud entre los primates y el hombre, y la posibilidad de tener un ancestro en común. A pesar de sus ideas revolucionarias, las cuales no estaban del todo incorrectas, no pudo dar una explicación coherente a los cambios ocurridos en los seres vivos, pero consideraba que el medio ambiente afectaba de manera directa a los organismos.

Unos cuantos años después, aparece el ilustre fundador de la paleontología de los invertebrados y creador del término *biología* o ciencia de los seres vivos; nos referimos a Jean Baptiste Pierre Antoine de Monete mejor conocido como el caballero de Lamarck. Este personaje concibe a la vida como un fenómeno natural en el cual la materia se encuentra dispuesta de una forma muy particular. Ahora sabemos que dicha aseveración no es del todo correcta, ya que a pesar de que los organismos vivos que habitamos la Tierra estamos compuestos por proteínas y ácidos nucleicos basados en el carbono, no nos ha sido posible crear vida a partir de los componentes básicos que la conforman. Bajo este principio, Lamarck concibe una teoría acerca de la evolución de los organismos vivos; por un lado explica cómo dichos organismos incrementan gradualmente su propia complejidad

Figura 1.5: Georges Louis Leclerc, Conde de Buffon.

y por otro habla sobre la adaptación de estos organismos a los elementos que los rodean y cómo, a través de la herencia, esta adaptación se transmite de padres a hijos.

Figura 1.6: Jean Baptiste Pierre Antoine de Monete, Caballero de Lamarck.

El concepto de *Selección Natural*, fue desarrollado por separado por dos ilustres científicos ingleses, Alfred Russel Wallace y Charles Robert Darwin quien después de recibir un ensayo escrito por el primero confirmó las sospechas que tenía y las cuales presentaron en conjunto ante la *Linnean Society of London*. Sin embargo, hubo un hombre que vaticinó esta idea casi 30 años

antes, del cual no se sabe mucho y no se le da el crédito que le corresponde. Nos referimos a Patrick Matthew, agricultor de frutas de origen escocés que propone a la selección natural como mecanismo de la evolución [4]. En 1831, publica un libro acerca del crecimiento de árboles de excelente calidad para la construcción de navíos. Podríamos resumir sus ideas, en sus propias palabras, de la siguiente manera:

- Las especies varían en forma y la seleción natural las mejora. "Así como la naturaleza, en todas las vertientes de la vida, tiene el poder de incrementar lo que se necesita para ocupar el lugar de lo que cae por influencia del tiempo. Aquellos individuos que no poseen las condiciones de fuerza, rapidez y astucia, mueren prematuramente sin reproducirse, ya sea en garras de sus predadores o simplemente caen por alguna enfermedad, generalmente inducida por lo que toman por alimento. De esta forma, sus lugares son ocupados por individuos más aptos de su misma clase presionados por la subsistencia".

- La selección natural puede producir nuevas especies. "Es improbable que gran parte de la diversificación pertenezca a la mezcla de especies cercanas, ya que todos los cambios creados de esta forma serían muy limitados, y confinados dentro de los límites de lo que es llamado especie; la progenie de los mismos padres, bajo diferentes circunstancias, quizás después de varias generaciones, darán como resultado distintas especies incapaces de reproducirse entre sí".

- La migración es un proceso esencial de la selección natural para eliminar las características no aptas para sobrevivir. "En la agitación que acompaña la migración, el más apto en cuerpo y mente, es decir, la más poderosa variedad de la raza será puesta en su lugar natural de líder, estampando la marca de sus propias características con el tiempo; mientras los más débiles o los menos aptos morirán bajo dificultades incidentales".

- Los cambios evolutivos tienen lugar justamente después de catástrofes mientras que entre un par de ellas, una especie no cambia; de esta forma la selección natural estabiliza dicha especie, sin alterarla. "Ha existido una continuidad particular de las especies durante al menos 40 siglos. Los geólogos han descubierto especies de fósiles a través de una profunda disposición de cada época, pero de igual manera han encontrado una casi completa diferencia entre las especies de una época y cualquier otra".

Figura 1.7: Patrick Matthew.

A diferencia de Matthew, Darwin y Wallace conciben la evolución como un proceso continuo en el que se acumulan cambios graduales favorables que llevan a la adaptación de una especie y en la que puede darse la extinción de toda una clase de organismos sin necesidad de que haya sido provocada por una catástrofe. Dichas ideas reflejan un mecanismo primordial de la evolución en el cual las condiciones de cierto medio ambiente determinan algunos aspectos de los organismos que lo habitan, para poder sobrevivir y reproducirse. Esta teoría se basa en dos premisas, la primera de ellas dice que debe haber una variación aleatoria no determinista en las características de la prole de un organismo y la segunda nos dice que dicha variación puede repercutir o no en el éxito que tendrán dichos individuos para sobrevivir y reproducirse, lo cual da como resultado que estas propiedades se extiendan en la población y en un momento dado este cambio se vea reflejado en el surgimiento de una nueva especie. Este proceso ocurre cuando la naturaleza elige de los individuos más aptos las características que les dan cierta ventaja competitiva frente a sus congéneres. Existen ciertos factores externos que hacen que una característica particular de un ser tienda al éxito, a los cuales se encuentra supeditada, y en la mayoría de los casos estas características son introducidas por el medio que lo rodea. Ejemplos de esto son las fuentes de alimento, las particularidades del medio físico y los depredadores naturales de esta especie, entre otros [10].

Supongamos un conjunto de individuos de cierta especie que, debido a la capacidad con la que cuentan para adaptarse a las condiciones cambiantes del medio, se extiende de manera vertiginosa sobre una amplia región. Debido a esto se crean subpoblaciones que se ven influenciadas por diferentes condiciones ambientales y esto lleva a cada subpoblación a desarrollar


Figura 1.8: Alfred Russel Wallace.

ciertas peculiaridades propias para adaptarse mejor a su entorno. Después de cierto tiempo, las características desarrolladas por cada grupo están tan alejadas que se consideran de diferente tipo. De aquí se desprende el hecho de cómo el camino de una especie se bifurca y toma diversos senderos dando pie al surgimiento de una nueva especie.

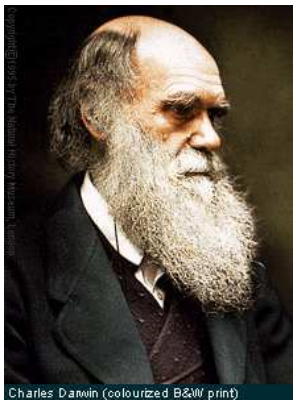



Figura 1.9: Charles Robert Darwin.

Para l865, el monje y naturalista de origen austriaco Gregor Johann Mendel, presenta los resultados que obtuvo al realizar experimentos sobre híbridos de plantas y enuncia lo que hoy conocemos como las leyes de la herencia de Mendel o de la genética [5], las cuales dicen más o menos lo siguiente:

- De la segregación. Durante la formación de los gametos cada miembro de los pares de alelos de un gen se separan para formar el nuevo gameto.
- De la independencia. Durante la formación de los gametos, la segregación del par de alelos es independiente de la segregación de cualquier otro par de alelos.
- De la uniformidad. Cada característica del nuevo individuo proviene de una heredada de alguno de los padres, y esto determina si un gen es dominante o recesivo.

Figura 1.10: Gregor Johann Mendel.

A finales del siglo XVIII y principios del XIX, el biólogo alemán Friedrich Leopold August Weismann, propone la teoría del germoplasma, en la que describe cómo los organismos multicelulares consisten de células germinales que contienen la información transmitida de padres a hijos y células somáticas que solamente llevan consigo información ordinaria relacionada con las funciones corporales. Las células germinales no se ven influenciadas por el medio que las rodea ni por cambios aprendidos o morfológicos que tienen lugar durante el tiempo de vida de un organismo, por lo que esta información no se transmite a sus descendientes y se pierde después de dicha generación.

La unión de la teoría de la evolución de Darwin, la genética de Mendel y el seleccionismo de Weismann, se conoce hoy en día como Neodarwinismo o teoría sintética de la evolución, integrada por Theodosius Dobzhansky, Ernest Mayr y George Simpson [12]. Esta teoría nos dice que la evolución paulatina de las especies se explica por medio de la aparición de cambios

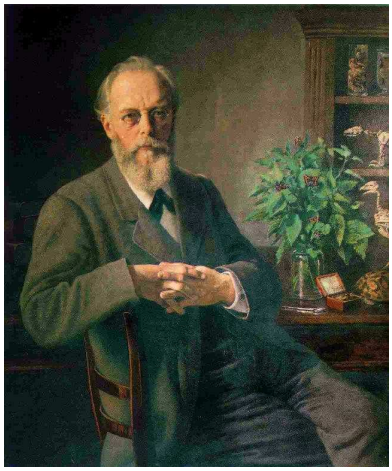


Figura 1.11: Friedrich Leopold August Weismann.

fortuitos (mejor conocidos como mutaciones), la selección natural, la competencia de los individuos y, por supuesto, la reproducción. Además, puede explicar otra serie de manifestaciones de la evolución como son la macroevolución y la especiación a partir de los mismos procedimientos.

Figura 1.12: Theodosius Dobzhansky.

## 1.3. Algoritmos Evolutivos

Existen una gran variedad de *algoritmos evolutivos* (AE); sin embargo, existen ciertas características comunes que comparten todos ellos. Las ideas que encontramos detrás de cualquiera de ellos se describen a continuación; dada una población de individuos dentro de cierto medio ambiente, se generará cierta presión dando como resultado la selección natural o la supervivencia del más apto, lo que se verá reflejado en una mejor aptitud de la población en general. Para determinar si un individuo es más apto se considera una función que nos dice la calidad de dicha solución. Generamos una población de individuos de manera aleatoria a la cual le aplicamos la función de aptitud lo que nos arrojará una medida de qué tan buenas son las soluciones generadas. Basados en la calidad de las soluciones, se selecciona a las mejores para crear a partir de ellas mismas la siguiente generación de soluciones candidatas por medio de ciertos operadores, como son la recombinación y la mutación. Consideramos la recombinación como un operador sobre dos o más individuos seleccionados de la población (padres) y que dan como resultado una o más nuevas soluciones (hijos); esto contrasta con la mutación que se aplica sobre una solución candidata para generar un hijo. Si llevamos a cabo la recombinación y mutación sobre una población, obtendremos un nuevo conjunto (descendientes) los cuales podrían o no competir con sus padres en función de su aptitud para sobrevivir a la siguiente generación. Dicho proceso puede ser iterado por el tiempo suficiente hasta el momento en que se considera que se ha encontrado una solución con la calidad requerida [13].

Durante el proceso descrito con anterioridad se considera a la selección, recombinación y mutación, como los operadores responsables de forzar las soluciones a tener mejor calidad y son las bases de un sistema evolutivo [42].

El conjuntar la diversidad y la selección de los mejores individuos, nos llevará de manera inevitable a mejorar la aptitud en poblaciones consecutivas. Es sencillo observar cómo un proceso, como la evolución, se encarga de optimizar o al menos se acerca por medio de aproximaciones sucesivas de manera gradual a su objetivo. La evolución también se puede ver como un proceso de adaptación en el cual la aptitud no se considera como una función objetivo a ser optimizada sino como una expresión que describe los requerimientos que debe cumplir un individuo dentro del medio ambiente. El proceso evolutivo hace que la población incremente su aptitud y que se vaya adaptando de mejor manera a su entorno.

Muchas de las unidades funcionales del proceso evolutivo son aleato-

rias; por ejemplo la generación de la población inicial y la probabilidad de que un individuo sea seleccionado; aquí cabe destacar que existen esquemas en los cuales incluso los entes más débiles cuentan con una pequeña pero no nula probabilidad de sobrevivir. Inclusive son fortuitas las partes de los individuos que se tomarán para ser recombinadas y mutadas.

**Algoritmo 1**: Pseudocódigo del esquema general de un algoritmo evolutivo

```
1  begin
2      Inicializar( población )
3      Evaluar( población )
4      while No se satisfaga la CONDICION de paro do
5          padres ← Seleccionar( población )
6          hijos ← Recombinar( padres )
7          Mutar ( hijos )
8          Evaluar ( hijos )
9          población ← SeleccionarNuevaGeneración( padres, hijos );
10     end
11 end
```

En el algoritmo 1, podemos observar el esquema general del algoritmo evolutivo básico descrito en los párrafos anteriores. Este tipo de algoritmos pueden ser vistos como un esquema de generación y prueba de las soluciones encontradas. La función a evaluar, la cual llamaremos aptitud, representa una estimación de la calidad de las soluciones, mientras que la búsqueda de las mejores soluciones es guiada por los cambios aleatorios ocurridos en la población y el operador de selección.

Los algoritmos evolutivos se basan en el manejo de una colección o conjunto de soluciones candidatas, mejor conocidas como población, las cuales intercambian información gracias al operador de recombinación para generar nuevas soluciones, además de darle cierto grado de aleatoriedad al introducir un operador de mutación, el cual modifica dichas soluciones candidatas con cierta probabilidad (figura 1.13).

### 1.3.1. Conceptos básicos

Como se mencionó con anterioridad, la computación evolutiva tiene una estrecha relación con la teoría del Neodarwinismo, por lo que es sumamente importante introducir ciertos conceptos biológicos.


Figura 1.13: Diagrama de flujo del esquema general de un algoritmo evolutivo.

Comenzaremos por definir lo que es un *gene*; un gene es una secuencia lineal de nucleótidos de *ADN* que es esencial para una función celular específica. Dicho de otro modo, es la unidad fundamental por la cual las características de los padres son heredadas a su progenie. ADN son las siglas de ácido desoxirribonucleico, el cual es el componente básico del material génetico de igual forma que el *ARN* o ácido ribonucleico, (figura 1.14).

Un *cromosoma* es una unidad estructural en el núcleo de la célula donde se encuentra la información genética. Consiste de una sola cadena de ADN y proteínas.

De esta forma, un cromosoma está formado por genes. Dichos genes toman una posición o *locus* particular dentro del cromosoma. Un *alelo* es el valor de un gene. Es decir, si partimos de la definición de gene como un

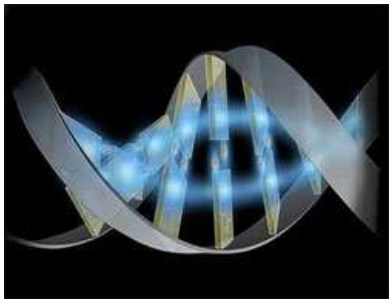


Figura 1.14: Hélice de ADN.

segmento de una cadena genética en particular, los alelos de ese gene serían todas las posibles variantes que puede tomar ese segmento de la cadena.

El *genoma* es el conjunto completo de cromosomas que se encuentran en las células de determinada especie. Por ejemplo, el genoma humano está formado por 46 cromosomas al igual que el de una liebre mientras que un simio cuenta con 48 cromosomas.

El *genotipo* es el contenido genético de un individuo. No lo podemos observar a simple vista, pero éste hace único a cada individuo de cierta especie.

El *fenotipo* es el resultado de decodificar el genotipo; es decir, es la parte que se manifiesta a simple vista en un individuo (figura 1.15).

Figura 1.15: Diversas características fenotípicas.

Una *población* es un conjunto de individuos de la misma especie que interactúan dentro de cierto *ecosistema*. Un *individuo* es la unidad elemental de una población. Un ecosistema es el medio en el cual se desenvuelven los individuos de una población.

### 1.3.2. Elementos de un Algoritmo Evolutivo

Ahora que sabemos lo que es un algoritmo evolutivo, nos daremos a la tarea de describir los elementos que lo conforman de manera más detallada, los cuales podemos englobar en alguna de las siguientes categorías: elementos, operadores y procedimientos (figura 1.13).

Si tuviéramos que seleccionar los componentes más representativos de un algoritmo evolutivo, sin duda serían:

- Representación
- Población
- Función de evaluación
- Mecanismo de selección de los padres y de los mejores individuos
- Operadores de recombinación y mutación

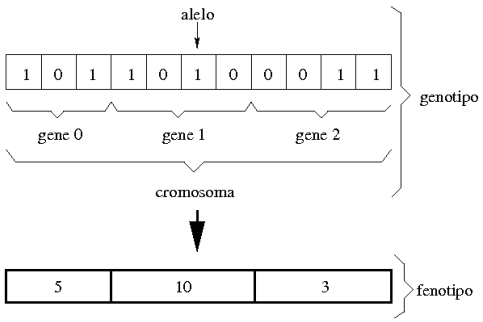



Figura 1.16: Representación del genotipo y fenotipo

#### Representación

Antes de comenzar con un problema, debemos encontrar la forma de ligar dicho problema al contexto de la evolución con el objetivo de resolverlo. Un algoritmo evolutivo consta de una población de individuos o soluciones candidatas al problema; cada individuo se representa por un cromosoma el cual es un esquema que representa un conjunto de genes. Cada uno de estos genes representa, a su vez, una variable del problema planteado. Dicha representación es conocida como fenotipo, mientras que el genotipo es la codificación de cada una de estas variables (figura 1.16).

Un cromosoma generalmente se representa con variables booleanas como se observa en la figura 1.16. Sin embargo, pueden ocuparse variables enteras o reales para lograr el mismo fin. La elección del tipo de representación se realiza en función del problema que se está atacando, ya que es muy probable que si seleccionamos el tipo de representación adecuada, el problema será más sencillo de resolver, al eliminar soluciones no factibles del espacio de búsqueda (figura 1.17).

De igual forma que se selecciona el tipo de representación, la longitud de los cromosomas depende del problema. En la mayoría de los casos se utilizan cadenas de longitud fija para representar las variables del sistema. Sin embargo, existen casos en los cuales es necesario el empleo de cadenas de longitud variable, lo que agrega un grado más de complejidad al problema. En ciertos casos podemos codificar la longitud de la cadena en el mismo cromosoma y sólo tomar la parte que nos interesa. En otros casos se pueden crear estructuras dinámicas, por ejemplo listas, en las que se pueden agregar o eliminar elementos según se requiera.

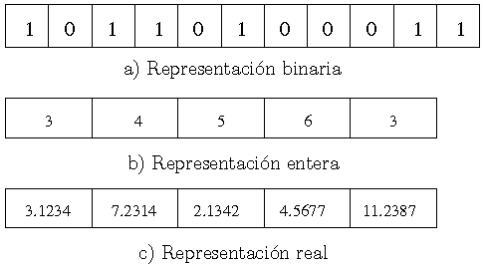



Figura 1.17: Diversos tipos de representación.

**Población**

La población juega un papel sumamente importante, el cual consiste en almacenar las soluciones candidatas al problema que se está resolviendo. La población puede ser vista como el conjunto de genotipos que guarda la información genética que ha sido pasada de generación en generación. Este concepto conforma la unidad básica en la evolución debido a que los individuos son sólo estructuras estáticas que no se adaptan, mientras que la población sí lo hace.

Una vez que hemos seleccionado un tipo de representación para la población, debemos definir el número de individuos que la compondrán. Es común que los algoritmos evolutivos cuenten con un tamaño de población

fijo. Sin embargo, esto no necesariamente tiene que ser así. Al contrario de lo que hacen los operadores de cruza y mutación que actúan sobre individuos, la selección de padres y de los mejores individuos se aplican sobre la población misma. Las decisiones se toman en función de los elementos con los que contamos en ese instante. Por ejemplo, se resguarda al mejor individuo para generar la siguiente población o se selecciona el peor individuo para ser reemplazado por uno nuevo.

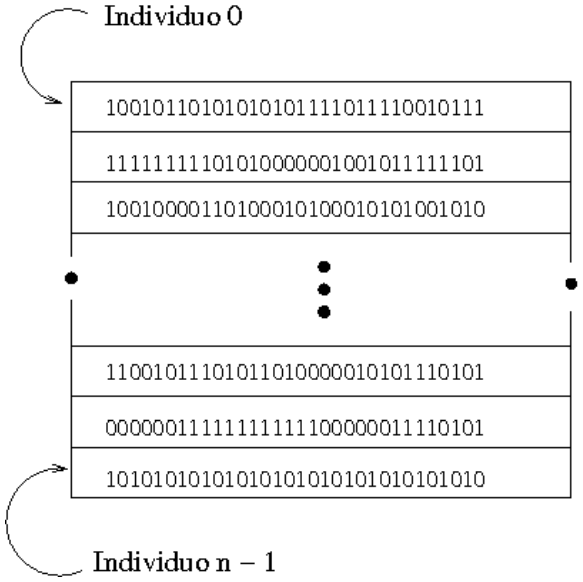



Figura 1.18: Población de $n$ individuos.

**Función de evaluación**

La *función de evaluación* se utiliza para representar los objetivos que deseamos alcanzar. Desde el punto de vista de la selección, la función de evaluación nos dice qué tan bien se ha adaptado el individuo como solución del problema. En realidad, esta función asigna una medida de calidad a las soluciones candidatas.

Es usual llamar a dicha función de evalución como *aptitud*, dentro de la jerga del cómputo evolutivo. Sin embargo, como en la mayoría de las ocasiones, los algoritmos evolutivos tratan de resolver un problema de optimización, como es el caso en esta tesis, es más correcto hablar de una *función objetivo*, mientras que la función de aptitud se podría ver como una simple transformación de dicha función.

### Mecanismo de selección de los padres y de los mejores individuos

La selección de los padres juega un papel preponderante en el esquema general de cualquier algoritmo evolutivo, si tomamos en cuenta que dicho proceso está basado en la calidad de los individuos, y enfatiza la importancia de que los mejores se conviertan en los padres de la siguiente generación. Este mecanismo se encarga de introducir cierta presión para mejorar la calidad de las soluciones candidatas. La selección de padres típicamente involucra un proceso estocástico aunque en ella se le da una mayor probabilidad de convertirse en padres a aquellos individuos que representan soluciones de mayor calidad. Existen diversos esquemas para determinar qué individuos serán los progenitores de la nueva población. En ciertos casos es conveniente dar una probabilidad diferente de 0 a aquellas soluciones con menor calidad, lo cual evitará que el algoritmo se estanque en óptimos locales y nos impida encontrar el óptimo global.

En cuanto a la selección de los mejores individuos, ésta se realiza básicamente en función de su propia aptitud. En ese sentido es muy similar a la selección de padres, aunque cumple con otro objetivo particular. Este proceso se realiza una vez que se ha terminado de crear la nueva generación, como se puede observar en el algoritmo 1. Al contrario de la selección de padres, la selección de los mejores individuos se realiza de manera determinística al asignarle un rango a cada solución en función de su calidad y eligiendo sólo a aquellos individuos que se encuentran en la cúspide, dentro del conjunto formado por la población de los padres e hijos (basada en la aptitud, figura 1.19) o solamente en la población de los hijos (basada en la edad).

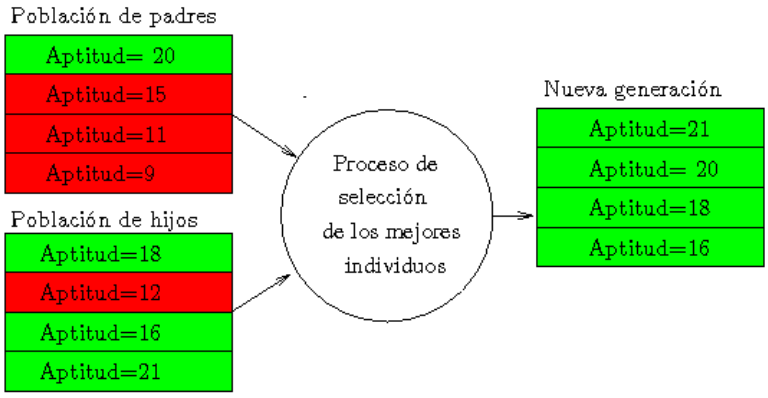



Figura 1.19: Selección basada en la aptitud.

### Operadores de recombinación y mutación

La operación de cruza o recombinación consiste en pasar información genética de dos o más padres a uno o más hijos. Éste es un operador estocástico, en el sentido de la elección de los segmentos de cadena del genotipo de cada padre que serán recombinados y la manera en que éstos se combinarán.

El principio sobre el que se sustenta la recombinación es sencillo: si combinamos las mejores características de dos o más individuos, los cuales tienen propiedades deseables, el resultado será un individuo que cuenta con lo mejor de ambos mundos. Los algoritmos evolutivos crean un número determinado de descendientes a través de la recombinación, esperando que dichos individuos sean mejores que los padres. Usualmente este proceso se aplica con cierta probabilidad por lo que existe la posibilidad de que no se aplique la recombinación (figura 1.20.a).

La mutación es un procedimiento que opera sobre un solo individuo. Es decir, se aplica sobre un genotipo y genera un solo individuo. Este operador es completamente estocástico ya que se aplica con cierta probabilidad, en algún segmento de la cadena el cual es seleccionado de manera aleatoria y la manera en que se modifica dicha sección incluso puede estar relacionada con algún método estocástico. El objetivo principal de este operador es crear un cambio que no siga ningún patrón (figura 1.20.b).

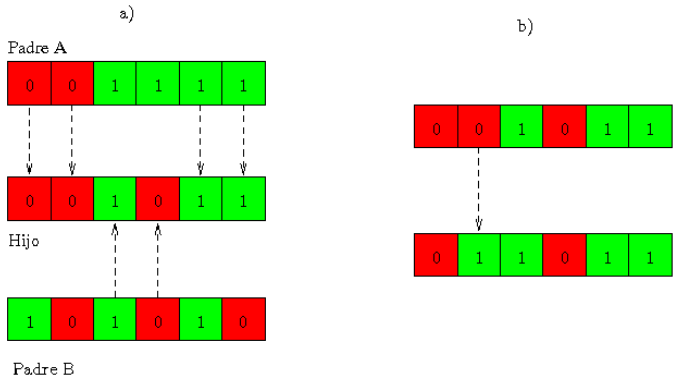



Figura 1.20: Operadores de a)recombinación y b)mutación

## 1.3.3. Principales paradigmas

Como es de esperarse, dentro del campo del cómputo evolutivo, existen diversas corrientes que involucran el manejo de diferentes mecanismos de

selección, recombinación, mutación y representación para lograr los mejores resultados. A continuación se da una somera descripción de las técnicas más representativas del área y la forma en como fueron concebidas por sus creadores.

- Algoritmos genéticos
- Estrategias evolutivas
- Programación evolutiva

### Algoritmos genéticos

Comenzaremos por hablar de los *algoritmos genéticos*, los cuales fueron introducidos en un inicio por John Holland, como un caso de estudio del comportamiento adaptativo.

**Algoritmo 2**: Algoritmo genético simple.

```
1  begin
2      g = 0
3      Inicializar( P )
4      Evaluar( P )
5      while No se satisfaga la CONDICION de paro do
6          P_padres ← SeleccionarPadres( P )
7          P_hijos ← Recombinar( P_padres )
8          Mutar ( P_hijos )
9          Evaluar ( P_hijos )
10         P ← SeleccionarSobrevivientes( P_padres, P_hijos )
11         g = g + 1
12     end
13 end
```

Los algoritmos genéticos utilizan una representación en cadenas de símbolos sobre un alfabeto finito. Por lo regular esta representación es binaria.

Manejan una selección proporcional con base en la aptitud; una alta probabilidad en el operador de cruza, ya que en este tipo de algoritmos tiene mucho mayor importancia la recombinación bioinspirada como una abstracción de lo ocurrido al crear un nuevo ser vivo y una baja probabilidad de mutación [23]. Al esquema mencionado anteriormente se le conoce comúnmente como algoritmo genético simple y se describe en el algoritmo 2.

**Estrategias evolutivas**

Las estrategias evolutivas (EE) fueron inventadas por Ingo Rechenberg y Hans Paul Schwefel [38, 41]. Una de sus principales características, al igual que otras técnicas evolutivas, es la auto adaptación de los parámetros. Con autoadaptación de los parámetros nos referimos al cambio de dichos valores durante la ejecución del algoritmo, los cuales se encuentran codificados, como si se tratase de variables a optimizar dentro del cromosoma y de esta manera evolucionan a la par con las soluciones. Esta peculiaridad es propia de las estrategias evolutivas aunque se describió a mayor detalle a partir de 1977 [40].

**Algoritmo 3**: Estrategia evolutiva simple.

```
1  begin
2      t ← 0
3      Inicializar x⃗ = ⟨x_1^t, x_2^t, ..., x_n^t⟩ ∈ ℝ^n
4      while No se satisfaga la CONDICION de paro do
5          Inicializar z⃗ = ⟨z_1, z_2, ..., z_n⟩, con una distribución normal
6          y_i^t ← x_i^t + z_i para toda i ∈ [1, 2, ..., n]
7          if f(x⃗^t) ≤ f(y⃗^t) then
8              x⃗^{t+1} ← x⃗^t
9          else
10             x⃗^{t+1} ← y⃗^T
11         end
12         t ← t + 1
13     end
14 end
```

Estos algoritmos utilizan una representación de vectores con valores reales, ya que se utilizan típicamente en problemas de optimización en espacios continuos, lo que implica que el genotipo es idéntico al fenotipo. Pueden hacer uso de una selección de los mejores individuos basada en la aptitud $(\mu + \lambda)$ o basada en la generación $(\mu, \lambda)$ , donde $\mu$ representa la población de padres y $\lambda$ representa la población de hijos.

El algoritmo básico de una EE para resolver un problema de optimización de $n$ dimensiones con una sola función objetivo y con sólo dos individuos se presenta en la figura 3. Aunque existen diversas variantes, aquí sólo presentamos la idea principal detrás de las estrategias evolutivas.

**Programación evolutiva**

La programación evolutiva (PE) fue desarrollada originalmente por Lawrence Fogel como una máquina de estados finitos, para simular el proceso de evolución como un mecanismo de aprendizaje continuo, teniendo en mente que esto debería dar cierta inteligencia a los sistemas, es decir, la habilidad para adaptarse, al modificar su comportamiento, y lograr ciertos objetivos en un conjunto de entornos [32, 17]. En el comportamiento adaptativo, al igual que en el caso de las estrategias evolutivas, la clave se encuentra en la predicción del ambiente y de aquí se parte para lograr el comportamiento inteligente.

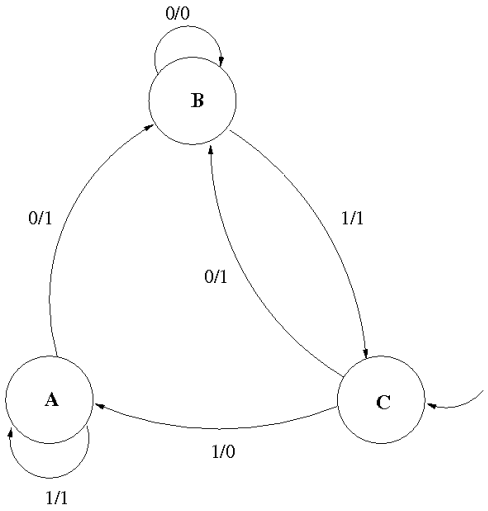



Figura 1.21: Máquina de estados finitos que predice el siguiente símbolo

Como se mencionó anteriormente, en el esquema básico de la PE, los predictores se desarrollaban como una máquina de estados finitos (MEF), la cual se puede modelar por un alfabeto finito de símbolos de entrada y responde con un conjunto finito de símbolos de salida, además de contar con $s$ estados y un número de $n$ transiciones. Estas últimas definen el comportamiento de la MEF ya que en función del estado actual y el símbolo de entrada se obtienen el siguiente estado y el símbolo de salida.

A la PE se le relaciona casi siempre con tareas de predicción y el uso de MEF como representación de las mismas. Sin embargo, éstas se han utilizado para resolver problemas de optimización con valores reales. Actualmente esta rama de la computación evolutiva cuenta con muchas variantes en términos de la representación y los operadores de mutación ya que en

la PE no se acostumbra utilizar el operador de recombinación [15, 16].

Para ejemplificar este tipo de técnica podemos observar la MEF de la figura 1.21, en la que se intenta predecir el siguiente símbolo de entrada sobre un alfabeto binario con tan solo tres estados.

# Capítulo 2

# Conceptos básicos

## 2.1. Teoría básica para problemas con valor inicial

Las ecuaciones diferenciales son usadas para modelar problemas de ingeniería que involucran el cambio de una variable con respecto a otra. En muchas ocasiones, estos problemas requieren de una solución inicial, esto quiere decir, que la solución de una ecuación diferencial satisface ciertas condiciones iniciales.

En la mayoría de los casos del mundo real, las ecuaciones diferenciales modelan problemas tremendamente complejos de resolver de manera exacta, por lo que es necesario intentar aproximarlas de la manera más eficaz posible. Una posible opción consiste en simplificar el problema, de tal forma que éste pueda ser resuelto de manera exacta para después utilizar esta solución de un modelo simplificado en la ecuación original. Existe otra solución que consiste en aproximar la solución por algún método de manera directa. Este último es el mecanismo más comúnmente utilizado debido a su exactitud.

Para comprender de manera más clara los métodos que se utilizan para resolver un sistema de ecuaciones diferenciales, necesitamos primeramente examinar ciertos conceptos de ecuaciones diferenciales y responderemos a ciertas preguntas básicas relacionadas con el tema.

**¿Qué es una ecuación diferencial?**

Una ecuación diferencial es un modelo matemático que involucra derivadas de una función.

**¿Qué es una ecuación diferencial ordinaria?**

Una ecuación diferencial ordinaria es aquella que solamente implica derivadas ordinarias con respecto a una sola variable independiente; un ejemplo de éstas es la ecuación 2.1. En contraste a ellas, nos encontramos con las ecuaciones diferenciales parciales, en las que las derivadas parciales están con respecto a más de una variable independiente y como ejemplo tenemos la ecuación 2.2.

$$\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} + 2y = 0 \tag{2.1}$$

$$\frac{\partial N}{\partial t} = \frac{\partial^2 N}{\partial r^2} + \frac{\partial N}{r\partial r} + kN \tag{2.2}$$

**¿Qué significa que una ecuación diferencial sea de primer orden?**

El orden de una ecuación diferencial, nos dice el orden de la derivada de orden más alto en dicha ecuación. Por ejemplo, la ecuación 2.3 es de primer orden, mientras que la ecuación 2.4 es de segundo orden.

$$\frac{dA}{dt} = -kA \tag{2.3}$$

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = E(t) \tag{2.4}$$

Ahora, nos daremos a la tarea de definir ciertos conceptos.

DEFINICIÓN **1** *Se dice que una ecuación diferencial es lineal si es de la forma*

$$a_n(x)\frac{d^n y}{dx^n} + a_{n-1}(x)\frac{d^{n-1} y}{dx^{n-1}} + ... + a_1(x)\frac{dy}{dx} + a_0(x)y = F(x), \tag{2.5}$$

*donde* $a_n(x), a_{n-1}(x), ..., a_0(x)$ *y* $F(x)$ *dependen única y exclusivamente de la variable* $x$*.*

De la ecuación 2.5 podemos decir que las combinaciones aditivas pueden tener coeficientes ($a_i$ para $i = 1, 2, ..., n$) en función de la variable independiente $x$, y no existen restricciones en cuanto a la naturaleza de dicha dependencia. En caso de que una ecuación diferencial no cumpla con la definición 1, ésta es conocida como **no lineal**. Por ejemplo, las ecuaciones 2.6 y 2.7 son una ecuación lineal y una no lineal respectivamente.

$$3\frac{d^2x}{dt^2} + 4\frac{dx}{dt} + 9x = 2\cos 3t \tag{2.6}$$

$$\frac{dy}{dx} = \frac{y(2-3x)}{x(1-3y)} \tag{2.7}$$

DEFINICIÓN **2** *Se dice que una función* $f(t, y)$ *satisface la condición de Lipschitz en la variable* $y$ *en un conjunto* $\mathcal{D} \subset \mathbb{R}^2$*, si existe una constante* $\mathcal{L} > 0$ *tal que*

$$|f(t, y_1) - f(t, y_2)| \leq \mathcal{L}|y_1 - y_2| \tag{2.8}$$

*para cualquier* $(t, y_1), (t, y_2) \in \mathcal{D}$*. De esta forma, se denota* $\mathcal{L}$ *como la constante de Lipschitz para* $f$*.*

DEFINICIÓN **3** *Se dice que un conjunto* $\mathcal{D} \subset \mathbb{R}^2$ *es convexo si cualquier par de combinaciones* $(t_1, y_1)$ *y* $(t_2, y_2)$ *pertenecen a* $\mathcal{D}$ *y todos los puntos* $((1-\lambda)t_1 + \lambda t_2), (1-\lambda)y_1 + \lambda y_2$ *también pertenecn a* $\mathcal{D} \; \forall \; 0 \leq \lambda \leq 1$*.*

Geométricamente hablando, un conjunto es convexo si todos los puntos del segmento de línea recta que pasa por dos puntos cualesquiera pertenecientes al conjunto, también son elementos del conjunto (ver figura 2.1).

A continuación se muestra una versión de la existencia del teorema de unicidad para ecuaciones diferenciales ordinarias de primer orden.

TEOREMA **1** *Supongamos $f(t, y)$ definida en el conjunto convexo $\mathcal{D} \subset \mathbb{R}^2$. Si existe una constante $\mathcal{L} > 0$ dado que*

$$|\frac{\partial f}{\partial y}| \leq \mathcal{L}, \qquad \forall (t, y) \in \mathcal{D}, \tag{2.9}$$

*entonces $f$ satisface la condición de Lipschitz dentro de $\mathcal{D}$ para la variable $y$ con una constante $\mathcal{L}$ de Lipschitz.*

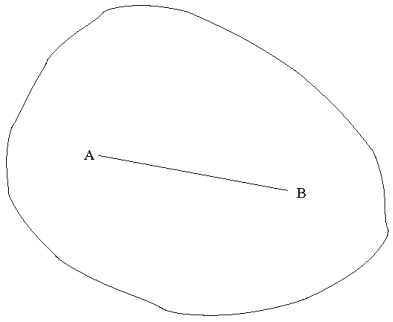



Figura 2.1: Conjunto convexo.

TEOREMA **2** *Supongamos que $\mathcal{D} = \{(t, y)|a \leq t \leq b, -\infty < y < \infty\}$, y $f(t, y)$ es continua en $\mathcal{D}$. Si $f$ satisface la condición de Lipschitz en $\mathcal{D}$ para la variable $y$, entonces el problema de valor inicial*

$$y' = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha \tag{2.10}$$

*tiene una solución única $y(t)$ para $a \leq t \leq b$.*

Del teorema 2 sabemos que el problema de valor inicial tiene una solución única. Ahora, debemos determinar en qué casos particulares cuentan con la propiedad de que los pequeños cambios en el estado de los problemas generan pequeños alteraciones en la solución de los mismos.

DEFINICIÓN **4** *Consideremos el problema de valor inicial*

$$\frac{dy}{dt}' = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha, \tag{2.11}$$

*Se dice que está bien definido si:*

1. *Existe una solución única y(t) del problema.*
2. *Para cualquier $\epsilon > 0$, existe una constante positiva $k(\epsilon)$ con la propiedad que para cualquier $|\epsilon_0| < \epsilon$ y $\delta(t)$ es continua con $|\delta(t)| < \epsilon$ en $[a, b]$, y una solución única $z(t)$ al problema,*

$$\frac{dz}{dt}' = f(t, z) + \delta(t), \qquad a \leq t \leq b, \qquad z(a) = \alpha + \epsilon_0, \tag{2.12}$$

*existe con*

$$|z(t) - y(t)| < k(\epsilon)\epsilon, \qquad \forall a \leq t \leq b \tag{2.13}$$

A los problemas descritos por la ecuación 2.12 se les conoce como problemas perturbados asociados con el problema original caracterizado por la ecuación 2.11. Los problemas perturbados son manejados por medio de métodos numéricos debido al error introducido en el sistema. A menos que el problema original esté bien definido, existen pocas razones para pensar que la solución numérica de un problema perturbado será aproximada de manera exacta.

Para asegurarnos de que un problema de valor inicial está bien definido, éste debe cumplir el teorema 3.

TEOREMA **3** *Supongamos que* $\mathcal{D} = \{(t, y)|a \leq t \leq b, -\infty < y < \infty\}$. *Si* $f(t, y)$ *es continua en* $\mathcal{D}$ *y satisface la condición de Lipschitz en* $\mathcal{D}$ *para la variable* $y$, *entonces el problema de valor inicial*

$$\frac{dy}{dt} = f(t, y), \qquad a \leq t \leq b, \qquad y(a) = \alpha, \tag{2.14}$$

*está bien definido.*

Existe una gran cantidad de métodos a través de los cuales podemos aproximar la solución de un problema de valor incial bien definido como son el método de Euler, el método de Runge-Kutta y las variaciones que de éste se desprenden dependiendo de la complejidad del problema (punto medio, cuarto orden). El objetivo de estos métodos es aproximar de la manera más exacta posible la solución del problema descrito por la ecuación 2.14. En este documento sólo haremos referencia a aquellos métodos que, por su simplicidad o la exactitud de los resultados que arrojan en un tiempo razonable, nos brindan las mayores ventajas.

## 2.2. El método de Euler

El método de Euler no es un método usado frecuentemente en la práctica. Sin embargo, debido a la sencillez del mismo es conveniente revisarlo para entender de manera más clara algunos métodos más complejos que se describen posteriormente.

De la ecuación 2.14, no podemos obtener una aproximación continua de la solución $y(t)$. Sin embargo, en lugar de eso, es posible generar diversas aproximaciones a partir de diferentes valores, los cuales son llamados

**puntos de malla** en el intervalo $[a, b]$. Una vez que se ha obtenido la solución aproximada por los puntos mencionados anteriormente, entonces podemos encontrar otros puntos dentro del intervalo por medio de una interpolación.

En este caso asumimos que los puntos de la malla se encuentran distribuidos de manera uniforme dentro del intervalo $[a, b]$. Para asegurarnos de que esta condición se cumpla, seleccionamos un número entero positivo $\mathcal{N}$ y elegimos dichos puntos $t_0, t_1, t_2, ..., t_{\mathcal{N}}$, donde:

$$t_i = a + ih, \qquad i = 0, 1, 2, ..., \mathcal{N}, \qquad h = \frac{b-a}{\mathcal{N}}.$$

El término $h$, es mejor conocido como **tamaño del paso**.

Este método se deriva de la serie de Taylor. En él suponemos que existen tanto la primera como la segunda derivada de la función $y(t)$ en el intervalo $[a, b]$ y ésta es una solución única para la ecuación 2.14. Entonces, para cada $i = 0, 1, 2, ..., \mathcal{N} - 1$,

$$y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2}y''(\xi_i)$$

Para algún número $\xi_i$ en $(t_i, t_{i+1})$. Si $h = t_{i+1} - t_i$, entonces:

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(\xi_i)$$

y debido a que $y(t)$ satisface la ecuación diferencial 2.14.

$$y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}y''(\xi_i) \tag{2.15}$$

El método de Euler obtiene $w_i$ aproximándola para cada $i = 1, 2, ..., \mathcal{N}$, eliminando el término remanente.

$$w_0 = \alpha, \qquad w_{i+1} = w_i + hf(t_i, w_i), \qquad i = 0, 1, ..., \mathcal{N} - 1. \tag{2.16}$$

La ecuación 2.16 es conocida como ecuación diferencial asociada con el método de Euler.

Para aproximar la solución al problema de valor inicial descrito por la ecuación 2.14, con el método de Euler, podemos usar el algoritmo 4.

**Algoritmo 4**: Pseudocódigo del método de Euler.

**Input:** Puntos extremos $a, b$; entero $\mathcal{N}$; condición inicial $\alpha$.
**Output:** Aproximación $w$ a $y(t)$ por los $\mathcal{N}+1$ valores de $t$
**begin**
  $h \leftarrow \frac{(b-a)}{\mathcal{N}}$
  $t \leftarrow a$
  $w \leftarrow \alpha$
  **for** *i=1* **to** $\mathcal{N}$ **do**
    $w \leftarrow w + hf(t,w)$ //Cálculo de $w_i$
    $t \leftarrow a + ih$ //Cálculo de $t_i$
  **end**
**end**

## 2.3. El método de Runge-Kutta

Los métodos de Taylor tienen la propiedad de eliminar los errores de truncado de alto orden, pero tienen la desventaja de requerir el cómputo y evaluación de las derivadas de $f(t,y)$, los cuales, en la mayoría de los casos, son complejos y consumen mucho tiempo de procesador, por lo que dichos métodos no son muy usados en la práctica.

El método de Runge-Kutta es muy popular para resolver problemas con valor inicial, ya que converge de manera más rápida que los métodos basados en el teorema de Taylor, además de ser fáciles de programar.

TEOREMA **4** *Suponemos que $f(t,y)$ y todas sus derivadas parciales de orden inferior o igual al término $n+1$ son continuas en $\mathcal{D} = \{(t,y)|a \leq t \leq b, c \leq y \leq d\}$. Dado $(t_0, y_0) \in \mathcal{D}$. Y existe para cada $(t,y) \in \mathcal{D}$ un $t < \xi < t_0$ y un $y < \eta < y_0$, entonces:*

$$f(t,y) = P_n(t,y) + R_n(t,y), \tag{2.17}$$

donde:

$$P_n(t,y) = f(t_0,y_0) + \left[(t-t_0)\frac{\partial f}{\partial t}(t_0,y_0) + (y-y_0)\frac{\partial f}{\partial y}(t_0,y_0)\right]$$

$$+ \left[\frac{(t-t_0)^2}{2}\frac{\partial^2 f}{\partial t^2}(t_0,y_0) + (t-t_0)(y-y_0)\frac{\partial^2 f}{\partial t \partial y}(t_0,y_0) + \frac{(y-y_0)^2}{2}\frac{\partial^2 f}{\partial y^2}(t_0,y_0)\right] + \dots$$

$$+\left[\frac{1}{n!}\sum_{j=0}^{n}\left(\begin{array}{c} n \\ j \end{array}\right)(t-t_0)^{n-j}(y-y_0)^j\frac{\partial^n f}{\partial t^{n-j}\partial y^j}(t_0,y_0)\right]$$

$$R_n(t,y)=\frac{1}{(n+1)!}\sum_{j=0}^{n+1}\left(\begin{array}{c} n+1 \\ j \end{array}\right)(t-t_0)^{n+1-j}(y-y_0)^j\frac{\partial^{n+1} f}{\partial t^{n+1-j}\partial y^j}(\xi,\eta)$$

La fución $P_n(t,y)$ es llamada **Polinomio de Taylor de grado $n$ en dos variables** para la función $f(t_0,y_0)$ y $R_n(t,y)$ es el término remanente de $P_n(t,y)$.

Para derivar el método de Runge-Kutta es necesario determinar los valores de $a_1$, $\alpha_1$ y $\beta_1$ dado que

$$a_1 f(t+\alpha_1, y+\beta_1)$$

aproxima el término

$$T^{(2)}(t,y)=f(t,y)+\frac{h}{2}f'(t,y) \tag{2.18}$$

con un error local de truncado para el método de Taylor de orden dos no mayor que $O(h^2)$. Derivando $f(t,y)$, obtenemos

$$f'(t,y)=\frac{df}{dt}(t,y)=\frac{\partial f}{\partial t}(t,y)+\frac{\partial f}{\partial y}(t,y)\cdot y'(t), \qquad y'(t)=f(t,y)$$

y sustituyendo ambas ecuaciones en la fórmula 2.18, tenemos que

$$T^{(2)}(t,y)=f(t,y)+\frac{h}{2}\left[\frac{\partial f}{\partial t}(t,y)+\frac{\partial f}{\partial y}(t,y)\cdot y'(t)\right]$$

$$T^{(2)}(t,y)=f(t,y)+\frac{h}{2}\frac{\partial f}{\partial t}(t,y)+\frac{h}{2}\frac{\partial f}{\partial y}(t,y)\cdot f(t,y) \tag{2.19}$$

Expandiendo $a_1 f(t+\alpha_1, y+\beta_1)$ tenemos que:

$$a_1 f(t+\alpha_1,y+\beta_1)=a_1 f(t,y)+a_1\alpha_1\frac{\partial f}{\partial t}(t,y)+a_1\beta_1\frac{\partial f}{\partial y}(t,y)+a_1 R_1(t+\alpha_1,y+\beta_1) \tag{2.20}$$

$$R_1(t+\alpha_1,y+\beta_1)=\frac{\alpha_1^2}{2}\frac{\partial^2 f}{\partial t^2}(\xi,\eta)+\alpha_1\beta_1\frac{\partial^2 f}{\partial t\partial y}(\xi,\eta)+\frac{\beta_1^2}{2}\frac{\partial^2 f}{\partial y^2}(\xi,\eta)$$

para algún $t < \xi < t + \alpha_1$ y un $y < \eta < y + \eta$.

Comparando los coeficientes de las ecuaciones 2.19 y 2.20 tenemos que

$$\begin{aligned} f(t,y): &\implies a_1 = 1 \\ \frac{\partial f}{\partial t}(t,y): &\implies a_1\alpha_1 = \frac{h}{2} \\ \frac{\partial f}{\partial y}(t,y): &\implies a_1\beta_1 = \frac{h}{2}f(t,y) \end{aligned}$$

de aquí obtenemos que los valores de los parámetros $a_1$, $\alpha_1$ y $\beta_1$ son

$$\begin{aligned} a_1 &= 1 \\ \alpha_1 &= \frac{h}{2} \\ \beta_1 &= \frac{h}{2}f(t,y) \end{aligned}$$

por lo que podemos deducir que

$$T^{(2)}(t,y) = f(t + \frac{h}{2}, y + \frac{h}{2}f(t,y)) + R_1(t + \frac{h}{2}, y + \frac{h}{2}f(t,y))$$

El resultado de reemplazar $T^{(2)}$ en el método de Taylor de orden dos es un método particular de Runge-Kutta conocido como **Método del punto medio**; sin embargo, el método de Runge-Kutta más utilizado es el de orden cuatro.

El esfuerzo computacional del método de Runge-Kutta se encuentra precisamente en la evaluación de la función $f$. En el método de segundo orden, el costo es de dos evaluaciones por cada paso del algoritmo. En el caso de cuarto orden se requieren realizar cuatro evaluaciones en cada paso y el error de truncado local es de $O(h^4)$. Las ecuaciones que describen el método de Runge-Kutta de cuarto orden se tratan a continuación.

$$\begin{aligned} w_0 &= \alpha \\ k_1 &= hf(t_i, w_i) \\ k_2 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_1\right) \\ k_3 &= hf\left(t_i + \frac{h}{2}, w_i + \frac{1}{2}k_2\right) \\ k_4 &= hf(t_{i+1}, w_i + k_3) \\ w_{i+1} &= w_i + \frac{1}{6}(k1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

para cada $i = 0, 1, \ldots, \mathcal{N} - 1$.

El proceso básico para el método de Runge-Kutta de cuarto orden para aproximar la solución de un problema de valor inicial

$$y' = f(t, y) \qquad a \leq t \leq b \qquad y(a) = \alpha$$

con $(\mathcal{N} + 1)$ puntos equidistantes en el intervalo $[a, b]$, se detalla en el algoritmo 5

**Algoritmo 5**: Pseudocódigo del método de Runge-Kutta de cuarto orden.

**Input:** Puntos extremos $a, b$; entero $\mathcal{N}$; condición inicial $\alpha$.
**Output:** Aproximación $w$ a $y(t)$ por los $\mathcal{N} + 1$ valores de $t$
**begin**
  $h \leftarrow \frac{(b-a)}{\mathcal{N}}$
  $t \leftarrow a$
  $w \leftarrow \alpha$
  **for** *i=1* **to** $\mathcal{N}$ **do**
    $K_1 \leftarrow hf(t, w)$
    $K_2 \leftarrow hf\left(t + \frac{h}{2}, w + \frac{K_1}{2}\right)$
    $K_3 \leftarrow hf\left(t + \frac{h}{2}, w + \frac{K_2}{2}\right)$
    $K_4 \leftarrow hf(t + h, w + K_3)$
    $w \leftarrow w + \frac{(K_1+2K_2+2K_3+K_4)}{6}$ $t = a + ih$
  **end**
**end**

La única razón por la que es conveniente utilizar el método con un mayor costo computacional, es que de esta manera podemos obtener una aproximación más exacta a la función real.

## 2.4. Sistemas de ecuaciones diferenciales

Hasta este momento nos encontramos en condiciones para resolver un problema de valor inicial con dos diferentes métodos (Euler y Runge-Kutta de cuarto orden). Sin embargo, el objetivo principal de este capítulo es describir los procedimientos a seguir para resolver un sistema de ecuaciones diferenciales, lo cual es el siguinte punto a tratar.

Un sistema de $m$ ecuaciones diferenciales de primer orden se puede expresar de la siguiente manera

$$\frac{du_1}{dt} = f_1(t, u_1, u_2, \ldots, u_m) \tag{2.21}$$

$$\begin{aligned} \frac{du_2}{dt} &= f_2(t, u_1, u_2, \ldots, u_m) \\ &\vdots \\ \frac{du_m}{dt} &= f_m(t, u_1, u_2, \ldots, u_m) \end{aligned}$$

para $a \leq t \leq b$, sujeto a las condiciones iniciales

$$u_1(a) = \alpha_1, \qquad u_2(a) = \alpha_2, \qquad \ldots, \qquad u_m(a) = \alpha_m \tag{2.22}$$

Lo que se busca son $m$ funciones $u_1, u_2, \ldots, u_m$ que satisfagan el sistema de ecuaciones diferenciales de igual forma como lo hacen las condiciones iniciales. Los métodos para resolver un sistema de ecuaciones diferenciales de primer orden son simples generalizaciones de los métodos para resolver una ecuación diferencial de primer orden. Por ejemplo, el método de Runge-Kutta descrito anteriormente, el cual se usa para resolver un problema de valor inicial de primer orden se puede generalizar de la siguiente forma.

Dado un entero $\mathcal{N} > 0$ y un $h = \frac{b-a}{\mathcal{N}}$, que divide el intervalo $[a, b]$ en $\mathcal{N}$ subintervalos definidos por los $\mathcal{N} + 1$ puntos

$$t_j = a + jh \qquad\qquad j = 0, 1, 2, \ldots, \mathcal{N}$$

Si se define $w_{ij}$ como la aproximación a $u_i(t_j)$ para cada $j = 1, 2, \ldots, \mathcal{N}$, e $i = 1, 2, \ldots, m$; es decir, $w_{ij}$ aproximará la $i$-ésima solución $u_i(t)$ en el $j$-ésimo punto $t_j$. Para el conjunto de condiciones iniciales

$$w_{1,0} = \alpha_1, w_{2,0} = \alpha_2, \ldots, w_{m,0} = \alpha_m \tag{2.23}$$

Si se asume que se han calculado los valores $w_{1,j}, w_{2,j}, \ldots, w_{m,j}$, podremos obtener $w_{1,j+1}, w_{2,j+1}, \ldots, w_{m,j+1}$ calculando primero

$$\begin{aligned}
k_{1,i} &= hf_i(t_j, w_{1,j}, w_{2,j}, \ldots, w_{m,j}) && (2.24)\\
k_{2,i} &= hf_i(t_j + \frac{h}{2}, w_{1,j} + \frac{k_{1,1}}{2}, w_{2,j} + \frac{k_{1,2}}{2}, \ldots, w_{m,j} + \frac{k_{1,m}}{2}) && (2.25)\\
k_{3,i} &= hf_i(t_j + \frac{h}{2}, w_{1,j} + \frac{k_{2,1}}{2}, w_{2,j} + \frac{k_{2,2}}{2}, \ldots, w_{m,j} + \frac{k_{2,m}}{2}) && (2.26)\\
k_{4,i} &= hf_i(t_j + h, w_{1,j} + k_{3,1}, w_{2,j} + k_{3,2}, \ldots, w_{m,j} + k_{3,m}) && (2.27)
\end{aligned}$$

para cada $i = 1, 2, \ldots, m$; ahora tenemos que

$$w_{i,j+1} = w_{i,j} + \frac{1}{6}[k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}] \tag{2.28}$$

para cada $i = 1, 2, \ldots, m$. Debe resaltarse que los valores de $k_{l,1}, k_{l,2}, \ldots, k_{l,m}$ deben ser calculados antes de $k_{l+1,1}, k_{l+1}, 2, \ldots, k_{l+1,m}$.

### 2.4.1. Método de Runge-Kutta para resolver sistemas de ecuaciones diferenciales

Para aproximar la solución de el sistema de primer orden con $m$ ecuaciones diferenciales

$$\begin{aligned}
u_j' = f_j(t, u_1, u_2, \ldots, u_m), &\qquad j = 1, 2, \ldots, m\\
a \leq t \leq b, \quad u_j(a) = \alpha_j &\quad j = 1, 2, \ldots, m
\end{aligned}$$

para $(\mathcal{N}+1)$ puntos equidistantes en el intervalo $[a, b]$, podemos utilizar el algoritmo 6.

**Algoritmo 6**: Pseudocódigo del método de Runge-Kutta de cuarto orden para un sistema de $m$ ecuaciones diferenciales.

**Input:** Puntos extremos $a, b$; número de ecuaciones $m$; entero $\mathcal{N}$; condiciones iniciales $\alpha_1, \ldots, \alpha_m$.
**Output:** Aproximación $w_j$ a $u_j(t)$ por los $\mathcal{N}+1$ valores de $t$
**begin**
  $h \leftarrow \frac{(b-a)}{\mathcal{N}}$
  $t \leftarrow a$
  **for** *j=1* **to** $m$ **do**
    $w_j \leftarrow \alpha_j$
  **end**
  **for** *i=1* **to** $\mathcal{N}$ **do**
    **for** *j=1* **to** $m$ **do**
      $K_{1,j} \leftarrow hf_j(t, w_1, w_2, \ldots, w_m)$
    **end**
    **for** *j=1* **to** $m$ **do**
      $K_{2,j} \leftarrow hf_j\left(t + \frac{h}{2}, w_1 + \frac{K_{1,1}}{2}, \ldots, w_m + \frac{K_{1,m}}{2}\right)$
    **end**
    **for** *j=1* **to** $m$ **do**
      $K_{3,j} \leftarrow hf_j\left(t + \frac{h}{2}, w_1 + \frac{K_{2,1}}{2}, \ldots, w_m + \frac{K_{2,m}}{2}\right)$
    **end**
    **for** *j=1* **to** $m$ **do**
      $K_{4,j} \leftarrow hf_j(t + h, w_1 + K_{3,1}, \ldots, w_m + K_{3,m})$
    **end**
    **for** *j=1* **to** $m$ **do**
      $w_j \leftarrow w_j + \frac{K_{1,j}+2K_{2,j}+2K_{3,j}+K_{4,j}}{6}$
    **end**
    $t = a + ih$
  **end**
**end**

# Capítulo 3

# Optimización multiobjetivo

El mundo de los problemas con múltiples objetivos simultáneos es un área en constante desarrollo. Ésta ha representado un reto, en numerosos y muy diversos sentidos, para los expertos en la materia por un largo tiempo. La posibilidad de explicar el mundo en su totalidad y resolver los problemas que de él emanan es sumamente complicada. Por lo tanto, es necesario recurrir a otra serie de procedimientos alternativos para intentar explicar el mundo que nos rodea y solucionar los inconvenientes que se nos plantean día con día. Este es el caso precisamente de la computación evolutiva, que propone resolver problemas de optimización en una amplia variedad de aplicaciones, que van desde minimizar la cantidad de material utilizado en la fabricación de un producto, hasta el diseño de vehículos espaciales.

En este capítulo se describirán los principales conceptos de la optimización multiobjetivo, sus orígenes y su relación con el cómputo evolutivo como una herramienta en la solución de problemas de esta índole.

## 3.1. Optimización global

Comenzaremos por describir el concepto de **optimización global**. La optimización global es el proceso de encontrar el máximo o mínimo global, según sea el caso, dentro de un espacio $\mathcal{S}$. Formalmente hablando, podemos definirla como [2]:

DEFINICIÓN **5** *Dada una función* $f(\vec{x}) : \Omega \subseteq \mathcal{S} = \mathbb{R}^n \rightarrow \mathbb{R}$, $\Omega \neq \emptyset$, *para* $\vec{x} \in \Omega$ *el valor* $f^* \triangleq f(\vec{x}^*) > -\infty$ *es llamado el mínimo global si y sólo si*

$$\forall \vec{x} \in \Omega : f(\vec{x}^*) \leq f(\vec{x}) \tag{3.1}$$

De esta forma, $\vec{x}$ es el mínimo global, $f$ es la función objetivo y el conjunto $\Omega$ es la región factible dentro del conjunto $\mathcal{S}$. El problema de determinar el mínimo global es llamado **problema de optimización global**.

Al contrario de los problemas de optimización con un solo objetivo, los cuales cuentan con una única solución óptima, los Problemas de Optimización Multiobjetivo (POM) suelen dar pie a un conjunto de soluciones, las cuales al ser evaluadas producen valores en el espacio de los objetivos que representan soluciones compromiso. Es entonces donde el ser humano encargado de tomar las decisiones selecciona uno o más de estos vectores, los cuales representan soluciones aceptables del problema según su propio punto de vista [8].

## 3.2. Problemas de Optimización Multiobjetivo

El **Problema de Optimización Multiobjetivo**, también conocido como Optimización Multicriterio, consiste en encontrar un vector de variables de decisión que satisfaga las restricciones del problema y optimice simultáneamente las funciones objetivo [8].

### 3.2.1. Variables de decisión

Las **variables de decisión** son cantidades numéricas, las cuales deben ser seleccionadas en un problema de optimización. Estas variables se representan por $x_l$ donde $l = 1, 2, \ldots, n$.

El vector de $n$ varibles de decisión $\vec{x}$ es representado por:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3.2}$$

### 3.2.2. Restricciones

Las **restricciones** impuestas por las características particulares del ambiente o los recursos con los que se cuenta, las encontraremos en la mayoría de los problemas de optimización. Dichas condiciones pueden ser limitaciones físicas de espacio o resistencia, obstáculos en el tiempo para la realización de una tarea, entre otras. Para que podamos considerar que cierta

solución es aceptable, al menos debe satisfacer dichas limitaciones. Las restricciones representan dependencias entre los parámetros y las variables de decisión del problema a optimizar. Existen dos tipos diferentes de restricciones, las **restricciones de desigualdad**:

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, \ldots, m$$

y las **restricciones de igualdad**:

$$h_j(\vec{x}) = 0 \qquad j = 1, 2, \ldots, p$$

Cabe destacar que $p$ debe ser menor que $n$, es decir, el número de restricciones de igualdad debe ser menor que el número de variables de decisión, ya que si $p \geq n$ al problema se le conoce como sobrerestringido, lo que significa que habrán más variables desconocidas que ecuaciones. Las restricciones pueden ser explícitas, descritas por una expresión algebraica o implícitas, en cuyo caso, debiéramos contar con un algoritmo o método para calcular dicha restricción para cualquier vector $\vec{v}$.

### 3.2.3. Funciones objetivo

Para saber que tan buena es una solución, es necesario contar con algún criterio para poder evaluarla. Esta pauta debe ser expresada como una función algebraica de las variables de decisión y es conocida como **función objetivo**. En ciertas ocasiones no contamos con este modelo matemático, por lo que al menos debemos tener algún mecanismo para determinar la calidad de las soluciones, las cuales pueden variar dependiendo del problema desde una simulación hasta la prueba real del sistema con las variables de decisión propuestas.

En una gran cantidad de problemas del mundo real, las funciones objetivo se encuentran en conflicto entre sí e incluso en el mismo problema algunas pueden ser funciones que deben ser minimizadas mientras las restantes deben ser de maximización. Las funciones objetivo pueden ser sujetas de ser medidas en las mismas unidades; si éste es el caso se les conoce como conmensurables.

El vector de las funciones objetivo $\vec{f}(\vec{x})$ está definido de la siguiente forma:

$$\vec{f}(\vec{x}) = \begin{bmatrix} \vec{f_1}(\vec{x}) \\ \vec{f_2}(\vec{x}) \\ \vdots \\ \vec{f_k}(\vec{x}) \end{bmatrix} \tag{3.3}$$

El conjunto de tuplas de números reales que se denota por $\mathbb{R}^n$ es llamado espacio Euclidiano de $n$ dimensiones. Para el problema de optimización multiobjetivo se consideran dos espacios Euclidianos, el de las variables de decisión y el de las funciones objetivo. Cada punto en el primer espacio representa una solución y puede ser mapeado uno a uno en el espacio de las funciones objetivo y determina la calidad de dicha solución.

Matemáticamente, el problema de optimización multiobjetivo lo podemos ver como:

DEFINICIÓN **6** *Encontrar un vector* $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ *que optimice el vector de funciones objetivo:*

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]^T \tag{3.4}$$

*sujeto a* $m$ *restricciones de desigualdad:*

$$g_i(\vec{x}) \leq 0 \qquad i = 1, 2, ..., m \tag{3.5}$$

*y a* $p$ *restricciones de igualdad:*

$$h_j(\vec{x}) = 0 \qquad j = 1, 2, ..., p \tag{3.6}$$

Dicho de otra forma, el Problema de Optimización Multiobjetivo consiste en determinar el conjunto de valores de las variables de decisión $x_1^*, x_2^*, \ldots, x_n^*$ que satisfagan las ecuaciones (3.5) y (3.6) y a la vez optimicen (3.4).

Las restricciones dadas por (3.5) y (3.6) definen la **región factible** $\Omega$ y cualquier punto $\vec{x} \in \Omega$ es una **solución factible**. El vector de funciones $\vec{f}(\vec{x})$ mapea el conjunto de soluciones factibles $\Omega$ al conjunto de funciones objetivo factibles $\Lambda$. Las $k$ funciones objetivo que conforman el vector $\vec{f}(\vec{x})$ representan los criterios que pueden estar expresados en diferentes unidades. Las restricciones $g_i(\vec{x})$ y $h_j(\vec{x})$ representan las limitantes impuestas a las variables de decisión. El vector $\vec{x}^*$ representa el conjunto de soluciones óptimas.

### 3.2.4. Tipos de Problemas de Optimización Multiobjetivo

En el ámbito de los problemas multiobjetivo existen tres diferentes variantes; la primera de ellas consiste en minimizar todas las funciones objetivo, la segunda es maximizar todas las funciones objetivo y la tercera es que algunas funciones objetivo sean de minimización y el resto de las funciones objetivo sean de maximización.

Es muy común que si se da el tercer caso, todas las funciones sean transformadas a su versión de minimización o maximización, según se prefiera. Para convertir una función que se desea maximizar a una de minimización podemos utilizar la siguiente ecuación:

$$max(f_i(\vec{x})) = -min(-f_i(\vec{x})) \tag{3.7}$$

De igual forma las restricciones de desigualdad descritas por la ecuación (3.5) pueden ser transformadas multiplicándolas por $-1$ y cambiando el signo de la desigualdad. De esta forma, dicha ecuación es equivalente a:

$$-g_i(\vec{x}) \geq 0 \qquad i = 1, 2, ..., m$$

### 3.2.5. El vector ideal

El vector ideal $\vec{f_i}^{\bigstar}$ está conformado como: $\vec{f_i}^{\bigstar} = [f_1^{\bigstar}, f_2^{\bigstar}, \ldots, f_k^{\bigstar}]^T$, donde $f_i^{\bigstar}$ denota el óptimo para la $i$-ésima función objetivo.

Si hubiera un punto único $\vec{x}^{\bigstar}$ (en el espacio de las variables de decisión), eso implicaría que no existe conflicto alguno entre los objetivos. Esta situación, sin embargo, es muy rara en la práctica.

### 3.2.6. Óptimo de Pareto

Un concepto sumamente importante en el campo de la optimización es el de la **optimalidad**. En los problemas de optimización de una sola función, el objetivo principal se centra en el espacio de las variables de decisión. Al contrario de lo descrito anteriormente, el espacio de las funciones objetivo es mucho más relevante en el contexto de la optimización multiobjetivo, de hecho, los valores de las funciones objetivo se usan para definir el concepto de optimalidad [35].

En los Problemas de Optimización Multiobjetivo es improbable encontrar una solución única que sea óptima para todos los objetivos al mismo

tiempo debido al conflicto que existe entre ellos y el hecho de que no pueden ser medidos por las mismas unidades, en la mayoría de las ocasiones. Dichos POM's no cuentan con un orden natural en el espacio de las funciones objetivo debido a que sólo se encuentran parcialmente ordenados. Por ejemplo, uno podría comparar los vectores $a = [1, 1]^T$ y $b = [4, 4]^T$ y decir que $a < b$. Sin embargo, consideremos los vectores $c = [1, 4]$ y $d = [4, 1]$. En este caso, no podríamos decir que uno es mejor que el otro.

El concepto de optimalidad fue presentado en un principio por Francis Ysidro Edgeworth en 1881. Sin embargo, este es mejor conocido como **optimalidad de Pareto**, en honor al sociólogo y economista italiano Vilfredo Pareto, quien lo desarrolló en 1896. El término de optimalidad nos remite al hecho de que algunos de los vectores en el espacio de las funciones objetivo pueden ser tomados para ser comparados entre sí; los componentes de dichos vectores, es decir, los valores de cada una de las funciones objetivo pueden ser mejorados en mayor o menor medida con el inconveniente de que la calidad de al menos otra de las funciones objetivo se verá degradada [26].

Formalmente podemos dfinir la optimalidad de Pareto como sigue:

DEFINICIÓN 7 *Un vector de variables de decisión $\vec{x}^* \in \mathcal{S}$ es un óptimo de Pareto si no existe ningún otro vector de decisión $\vec{x} \in \mathcal{S}$ tal que $f_i(\vec{x}) \leq f_i(\vec{x}^*)$ para toda $i = 1, 2, \ldots, k$ y $f_j(\vec{x}) < f_j(\vec{x}^*)$ para al menos un índice $j$.*

*Un vector en el espacio de las funciones objetivo $\vec{z}^* \in \mathcal{Z}$ es un óptimo de Pareto si no existe otro vector $\vec{z} \in \mathcal{Z}$ en el mismo espacio, tal que $\vec{z}_i \leq \vec{z}_i^*$ para todo $i = 1, 2, \ldots, k$ y $\vec{z}_j < \vec{z}_j^*$ para al menos un índice $j$. Esto es equivalente a decir que $\vec{z}^*$ es un óptimo de Pareto si el vector de variables de decisión que le corresponde es un óptimo de Pareto.*

En la figura 3.1 podemos observar el espacio de las variables de decisión (en este caso dos) y de las funciones objetivo (también dos, en este caso). En esta figura se hace la distinción entre las soluciones no factibles, las factibles y las no dominadas.

En cualquier problema de múltiples funciones objetivo en el espacio de los reales existen un gran número de soluciones que son óptimos de Pareto, por lo que podemos referirnos a dichas soluciones como el conjunto de óptimos de Pareto. Este conjunto puede ser cóncavo e inclusive desconectado.

El término de óptimo de Pareto también es conocido en muchas ocasiones como solución no dominada y en general se utiliza como un concepto de optimalidad. La definición formal de la **dominancia de Pareto** se explica a continuación:

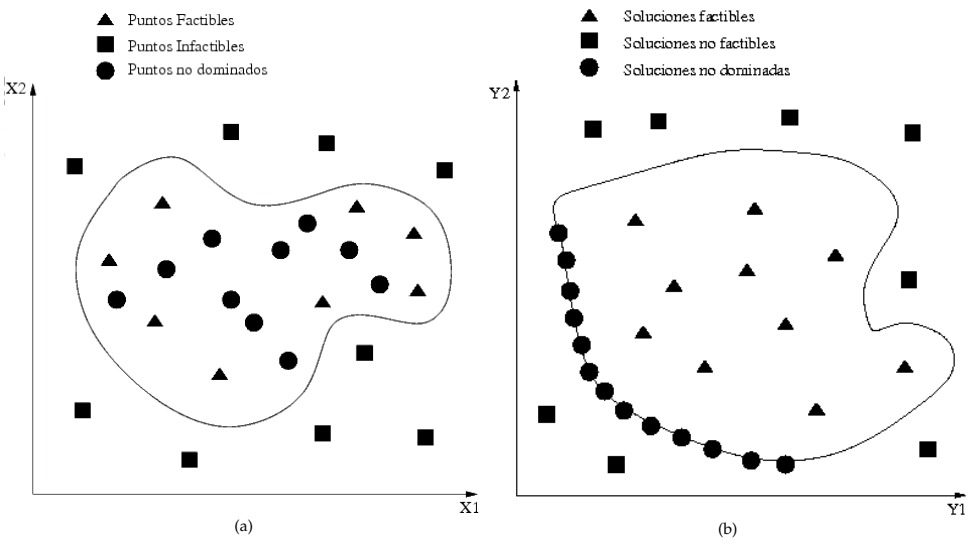



Figura 3.1: (a) Espacio de las variables de decisión y (b) funciones objetivo

DEFINICIÓN **8** *Se dice que un vector* $\vec{u} = [u_1, u_2, \ldots, u_k]^T$ *domina a un vector* $\vec{v} = [v_1, v_2, \ldots, v_k]^T$ *si y sólo si* $\vec{u}$ *es parcialmente menor que* $\vec{v}$*, es decir,*

$$\forall i \in \{1, 2, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < v_i$$

*y se denota por* $\vec{u} \preceq \vec{v}$*.*

DEFINICIÓN **9** *Consideremos un POM* $\vec{f}(\vec{x})$*, entonces el conjunto de óptimos de Pareto* $\mathcal{P}^*$ *está definido como*

$$\mathcal{P}^* = \{\vec{x} \in \Omega \mid \nexists \vec{y} \in \Omega : \vec{f}(\vec{y}) \preceq \vec{f}(\vec{x})\}$$

En la figura 3.2 podemos apreciar el frente de Pareto para las funciones objetivo $f_1$ y $f_2$ resaltado con la línea mas gruesa. El concepto del frente de Pareto se define formalmente de la siguiente forma:

DEFINICIÓN **10** *Considere un POM* $\vec{f}(\vec{x})$ *y un conjunto de óptimos de Pareto* $\mathcal{P}^*$*; el frente de Pareto* $\mathcal{PF}^*$ *se define como*

$$\mathcal{PF}^* = \{\vec{f} = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \mid \vec{x} \in \mathcal{P}^*\}$$

Por lo general, es imposible encontrar una expresión matemática que nos permita determinar los puntos que conforman el $\mathcal{PF}^*$. Para determinar

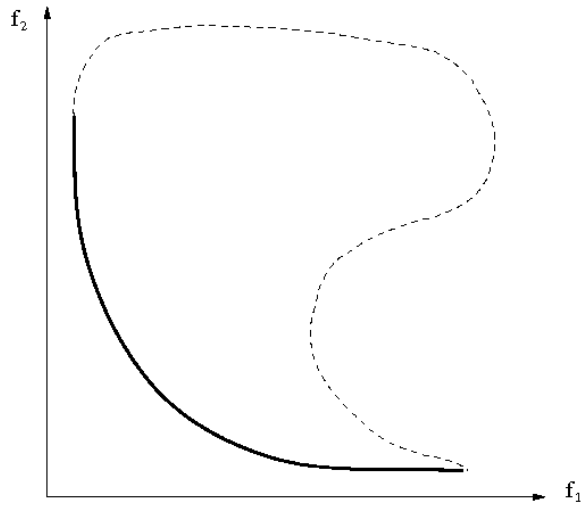



Figura 3.2: Frente de Pareto.

este conjunto, normalmente se calculan las $\vec{f}$ de un número suficiente de puntos que se encuentren en $\Omega$ (región factible). Si el número de puntos calculados es el adecuado, se puede determinar cuáles soluciones son no dominadas y así obtener el frente de Pareto.

Las soluciones no dominadas no tienen ninguna relación entre sí más allá de ser miembros del conjunto de óptimos de Pareto. Este conjunto de soluciones corresponden a los vectores no dominados con respecto al resto de los vectores con los que se compararon basados en la evaluación de sus funciones. Al graficar el conjunto de vectores o soluciones no dominadas en el espacio de las funciones objetivo tenemos como resultado el frente de Pareto.

De acuerdo con la definición de óptimo de Pareto, para ir de una solución de este tipo a otra, es necesario hacer un compromiso entre las funciones, es decir, el mejorar un objetivo se verá reflejado como el deterioro de otro. Este es uno de los conceptos principales en optimización multiobjetivo. La idea del compromiso es sometida a cuestionamientos en algunas ocasiones. Quizá no en todos los casos se debe realizar un compromiso con el objetivo de mejorar los resultados. Podríamos generar mejores resultados, en el sentido de obtener mejor calidad y menor costo, únicamente cambiando la formulación del problema [44].

### 3.2.7. Dominancia débil y fuerte

En conjunto con el concepto de optimalidad de Pareto, existe un par de conceptos relacionados que han sido ampliamente difundidos. A dichos conceptos se les conoce como **dominancia débil de Pareto** y **dominancia fuerte de Pareto**. Un vector es un óptimo de Pareto débil si no existe otro vector para el cual todos sus componentes en el espacio de las funciones objetivo sean mejores. Formalmente hablando, lo podemos definir como sigue:

DEFINICIÓN **11** *Una solución $\vec{x}^* \in \Omega$ es una solución débilmente no dominada si no existe otra solución $\vec{x} \in \Omega$ tal que $f_i(\vec{x}) < f_i(\vec{x}^*)$, para $i = 1, 2, \ldots, k$.*

El concepto de dominancia fuerte se puede resumir de la siguiente forma:

DEFINICIÓN **12** *Una solución $\vec{x}^* \in \Omega$ es una solución fuertemente no dominada si no existe otra solución $\vec{x} \in \Omega$ tal que $f_i(\vec{x}) \leq f_i(\vec{x}^*)$, para $i = 1, 2, \ldots, k$, y existe al menos un valor $j$ para el cual $f_j(\vec{x}) < f_j(\vec{x}^*)$.*

Si un vector es fuertemente no dominado, entonces también es débilmente no dominado, pero lo contrario no necesariamente es cierto en todas las ocasiones.

## 3.3. El tomador de decisiones

Cualquier punto que pertenezca al conjunto de óptimos de Pareto se considera una solución aceptable para un problema de optimización multiobjetivo. De cualquier forma, en la mayoría de las situaciones es deseable tener sólo una solución. Para poder seleccionar un vector del conjunto de óptimos de Pareto debemos utilizar información diferente a la contenida en el vector de funciones objetivo. Una posible solución adoptada recientemente en el campo de la optimización multiobjetivo es la de comparar como si se tratara de un problema mono-objetivo.

Es posible que una persona seleccione la mejor solución con base en su conocimiento del problema, además de poder expresar ciertas preferencias dependiendo de las necesidades actuales, y es esta persona quien tiene la responsabilidad de elegir la solución final. A esta persona se le conoce como **tomador de decisiones**.

Para poder resolver un problema de optimización multiobjetivo es necesario el trabajo conjunto de un tomador de decisiones y de un analista. El

analista no es necesariamente una persona. Este papel lo puede desempeñar un programa especializado, que se encargará de realizar los cálculos y presentar dicha información al tomador de decisiones, el cual elegirá una solución en función de sus preferencias.

Todo lo mencionado anteriormente es verdad si consideramos un único tomador de decisiones, aunque en algunas ocasiones el encargado de tomar la decisión final es un grupo de personas. Este grupo de personas deben negociar y buscar algún mecanismo, como una votación por ejemplo, para converger a una solución única, aún cuando existan diversos intereses dentro del grupo.

Los métodos de solución se pueden clasificar dependiendo del papel que desempeña el tomador de decisiones. Algunos métodos asumen ciertos puntos concernientes con la estructura de las preferencias y el entorno del tomador de decisiones. Cabe recalcar, que aún cuando sólo exista una persona encargada de tomar la decisión final, ésta puede estar influenciada por personas externas con diferentes intereses y preferencias.

En el proceso para resolver un problema, se requiere cierta información del tomador de decisiones. Esta información está relacionada con los niveleas deseables o aceptables en los valores de las funciones objetivo, aunque no tengamos una idea clara de si dichas soluciones son factibles o no. Sin embargo, dichos niveles son de gran importancia para la o las personas encargadas de tomar la decisión.

DEFINICIÓN **13** *Los valores de las funciones objetivo que son satisfactorios o deseables para el tomador de decisiones se les llama* **niveles de aspiración** *y se denotan por* $\vec{z}_i$ *para* $i = 1, 2, \ldots, k$. *Al vector* $\vec{z}$ *conformado por los niveles de aspiración se le llama* **punto de referencia**.

Los expertos en el área de optimización multiobjetivo tienen como objetivo encontrar un vector factible que además sea una solución óptima de Pareto y satisfaga las necesidades y requerimientos del tomador de decisiones. A dicha solución se le conoce como **solución final**. Sin embargo, en ciertas ocasiones es muy complicado para el encargado de tomar la decisión final distinguir entre buenas soluciones y soluciones óptimas en problemas del mundo real. Si éste es el caso, debemos enfocarnos en encontrar buenas soluciones y algunas veces sólo nos concentraremos en encontrar soluciones [46, 45].

El objetivo de esta tesis no se centra en el problema de tomar la decisión final, ya que por sí sola, ésta es una área de estudio. Algunos puntos a remarcar de dicha área son la toma de decisiones con información incom-

pleta, la validez de las soluciones encontradas y la precisión con que se modeló el problema.

Desde el punto de vista de un analista, el tomador de decisiones sólo se interesará por los puntos que son óptimos de Pareto mientras que los demás pueden ser excluidos. Este no es el caso, por ejemplo, cuando el problema no ha sido formulado de la manera más adecuada. Las soluciones que no pertenecen al conjunto de Pareto toman especial relevancia cuando existe algo que no se planteó de manera adecuada o existen funciones objetivo que no se contemplaron o que se mantuvieron al margen por alguna razón en particular. En estos casos , el conjunto de óptimos de Pareto del problema con el que se está trabajando y el problema real que se deseaba resolver, no coinciden. En este punto consideramos que el modelo matemático es correcto de tal modo que nos podemos concentrar en las soluciones óptimas de Pareto.

## 3.4. Algoritmos de optimización

Las técnicas de optimización se pueden clasificar en dos diferentes categorías, los métodos deterministas y estocásticos.

Dentro de los métodos deterministas encontramos los enumerativos, los cuales conforman la estrategia más simple a seguir cuando se trata de un problema definido en un espacio restringido. Estos mecanismos evalúan todas las posibles soluciones. Sin embargo, es fácil ver que esta técnica es muy ineficiente y en algunos casos, virtualmente imposible de llevar a cabo debido al tamaño (potencialmente grande) del espacio de búsqueda. Como los problemas del mundo real son computacionalmente complejos, se prefiere acotar el espacio de búsqueda para encontrar soluciones aceptables en un tiempo razonable [34]. El resto de los métodos determinísticos incorporan cierto conocimiento del problema.

Algunos ejemplos de algoritmos determinísticos son los famosos algoritmos de *Hill-Climbing*, los cuales se utilizan para realizar búsquedas y como optimizadores locales. Éstos no requieren de ningún tipo de estructura de datos especial ya que descartan las soluciones pasadas. En algunas ocasiones funcionan como una búsqueda aleatoria y tienen problemas para salir de un máximo o mínimo local. Sin embargo, este problema se puede resolver usando una reinicialización periódica con un número de iteraciones suficiente. Estos algoritmos tienen un mejor desempeño en funciones unimodales (que tienen un solo óptimo). Sin embargo, la existencia de óptimos locales, aristas en el paisaje de aptitud de las funciones, etc. reducen

la eficiencia del mismo. Los algoritmos *greedy* o glotones siempre toman la mejor opción que tienen en ese momento. Éstos realizan la elección mencionada esperando que dicha opción los lleve a la solución óptima global. A pesar de que estos algoritmos no siempre alcanzan las soluciones óptimas, son métodos muy poderosos que se pueden aplicar en una amplia gama de problemas [9]. La técnica de *branch and bound* requiere de un conocimiento específico del problema que permite limitar el espacio de búsqueda. Éste calcula una cota en un determinado nodo para determinar cuáles de ellos son promisorios, mientras son descartados o, dicho de otra forma, son "podados" por no resultar factibles.

Además de los algoritmos mencionados en el párrafo anterior, tenemos los algoritmos *depth-first* o búsqueda en profundidad, *breadth-first* o búsqueda en anchura, *best-first* o el mejor encontrado, por mencionar algunos, los cuales son métodos determinísticos que se han usado exitosamente en una amplia variedad de problemas multiobjetivo. Sin embargo, existe una gran cantidad de problemas multiobjetivo de alta dimensionalidad, multimodales, con espacios de búsqueda discontinuos, no lineales en los que las técnicas determinísticas resultan no ser efectivas [18]. En estos casos resulta más conveniente utilizar otro tipo de técnicas, como son los algoritmos evolutivos, los cuales caen dentro de la clasificación de los algoritmos estocásticos y fueron descritos con mayor detalle en el capítulo 1.

# Capítulo 4

# Optimización de rutas de vehículos espaciales de bajo impulso

En este capítulo se da una descripción del por qué es interesante el problema de optimización de rutas de vehículos espaciales de bajo impulso, los usos que se le da actualmente y una breve reseña de lo que es un propulsor de iones. Más adelante se plantea el problema de optimización de rutas en vehículos espaciales de bajo impulso y algunos de los métodos que se utilizan en la actualidad para resolver dicho problema. Además, se explican cuáles son los elementos que intervienen en el problema y cada uno de los parámetros que los componen, así como el algoritmo básico para evaluar una trayectoria elíptica.

## 4.1. Introducción

El problema de encontrar la ruta óptima en un vehículo espacial de bajo impulso, consiste en determinar cuál es el camino que requerirá menor tiempo y consumo de combustible para lograr nuestro objetivo, el cual es llegar desde una órbita en particular a otra. La constante necesidad de reducir costos en las misiones interplanetarias, ha conducido a intentar hacer más cortas dichas misiones, crear vehículos más pequeños y sistemas mecánicos, de propulsión y navegación mucho más simples que los utilizados hasta hace algunos años. Dichos requerimientos han volcado el interés de los investigadores en los sistemas de propulsión de bajo impulso debido

a su alto desempeño. Estos novedosos sistemas de propulsión requieren de modelos dinámicos diferentes a los utilizados en los sistemas convencionales de propulsión por medio de reacciones químicas, debido a que la energía liberada por el vehículo espacial se dosifica en periodos finitos de tiempo. Aunque el principio de los propulsores de bajo impulso es precisamente que no se utilicen todo el tiempo, como se mencionó anteriormente, las misiones que incorporan este tipo de tecnología requieren ser impulsadas en un porcentaje importante de la misión en su totalidad, con el objetivo de alcanzar las metas trazadas en un inicio. La optimización de trayectorias de vehículos que cuentan con sistemas de bajo impulso es sumamente complicada, debido al hecho de que la magnitud y dirección de dicha fuerza es una función continua en el tiempo, además de involucrar un gran número de variables.

Hace algunos años, las técnicas de optimización utilizadas sólo modelaban una serie de eventos discretos, como son: el momento del lanzamiento, el momento en el que se acercan más los cuerpos celestes e inclusive ciertas maniobras que era necesario realizar en el espacio. Actualmente, los modelos son mucho más complejos y consideran aspectos como la masa de los cuerpos celestes, la distancia entre los vehículos espaciales y dichos cuerpos, la forma particular de la trayectoria que se debe seguir, etc.,por lo que el problema poco a poco se ha convertido en una inmensa bola de nieve muy difícil de manejar. Dichos métodos están basados en el cálculo de las variaciones de una posición a otra; no obstante, estas técnicas se complican con trayectorias muy grandes o complicadas y pueden requerir un considerable consumo de recursos computacionales en búsquedas exhaustivas e ineficientes. Debido a ésta y algunas otras causas, se ha optado por emplear técnicas estocásticas en el proceso de optimización de rutas, tales como el *Recocido Simulado* y los *Algoritmos Genéticos*, los cuales han sido utilizados con un relativo éxito en la batalla que se libra con el problema planteado.

Los motores de bajo impulso, son máquinas que tienen un empuje menor a $0{,}01$g (g=constante de gravitación de la Tierra, $9{,}81\frac{m}{s^2}$). Estas máquinas han demostrado recientemente su utilidad en misiones espaciales. El lanzamiento exitoso del *Deep Space 1* fue el primer proyecto que usó este tipo de tecnología como medio de propulsión principal el 24 de Octube de 1998. Fue diseñado para probar las nuevas tecnologías, las cuales incluían un sistema de propulsión de bajo impulso de Xenón. Dicho sistema estuvo en uso durante 677 días y fue utilizado de manera continua durante varios meses para que alcanzara la misma aceleración que un motor de propulsión química. Actualmente se trabaja en el lanzamiento de las misiones Dawn y Jimo, las cuales contarán con sistemas de propulsión de bajo impulso. La

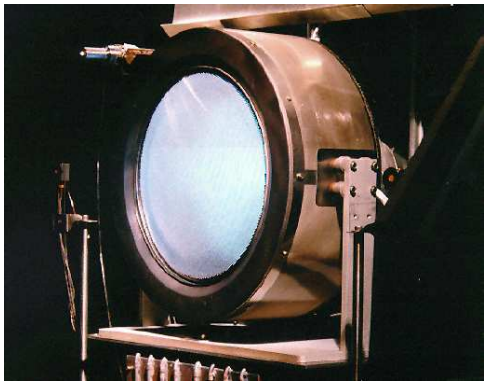


Figura 4.1: Propulsor de iones.

primera de ellas tiene como objetivo caracterizar las condiciones y procesos del sistema solar en sus inicios, investigando en detalle dos de los más grandes protoplanetas que aún se mantienen intactos desde su formación. Estos cuerpos son conocidos como Ceres y Vesta; se encuentran ubicados en el cinturón de asteroides localizado entre los planetas Marte y Júpiter junto con muchos otros cuerpos de menor tamaño. Cada uno de estos cuerpos ha seguido diferentes caminos, limitados por la diversidad de procesos que han operado sobre ellos durante los primeros millones de años en la evolución del sistema solar.

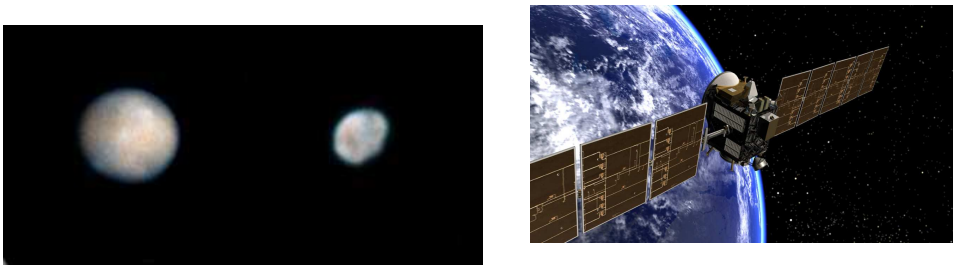


Figura 4.2: (a) Ceres y Vesta (b) Vehículo de la misión Dawn.

La misión espacial JIMO (*Jupiter Icy Moons Orbiter*) fue diseñada con la finalidad de poder explorar las lunas congeladas de Júpiter. En particular se tenía como objetivo Europa, ya que existía la sospecha de que contaba con un océano en donde es probable que hubiera vida. Ahora se consideran también Ganímedes y Calisto ya que se piensa que existen océanos salados debajo de sus superficies congeladas, por lo que también se consideran co-

mo objetivos de interés para la comunidad científica.

Figura 4.3: (a)Júpiter y sus lunas (b)Prometheus.

## 4.2. Encontrando las mejores rutas

El problema de encontrar la ruta óptima entre órbitas con vehículos espaciales de bajo impulso tiene como objetivos principales el encontrar las trayectorias que consuman menor tiempo, y que a la vez consuman menor cantidad de combustible. En este caso, una solución óptima de Pareto (ver definición 7), será aquella en la que no existe otra solución que sea mejor a ésta en términos de ambos objetivos de manera simultánea (tiempo de vuelo y consumo de combustible). El diseño de las primeras misiones consideraba maximizar la cantidad de combustible utilizado. Esto se hacía con la finalidad de minimizar lo más posible el tiempo de vuelo; sin embargo, siempre se tuvo en cuenta la necesidad latente de encontrar un conjunto de soluciones de Pareto que demostraran el compromiso entre ambas funciones objetivo. En general, dicho problema es complicado de resolver no sólo por el tiempo que dura una misión y el hecho de que los vehículos deben dar cierto número de revoluciones para llegar a su cometido, sino que aunado a esto, deben cumplir con la minimización del consumo de combustible.

Se han utilizado algunos métodos para resolver el problema de optimización. La gran mayoría de dichos esfuerzos utilizan métodos directos

o indirectos [43]. Los métodos directos aproximan el problema ajustando las variables de control de manera iterativa para reducir el tiempo de vuelo y la cantidad de combustible [28]. El control continuo y las variables de estado son discretizadas, lo que da como resultado que se tenga que programar un sistema no lineal. Por otro lado, el método indirecto hace uso de una ley de control que se utiliza cuando el problema se formula por medio del cálculo de variaciones [43]. Se introduce un vector de co-estado y el impulso es determinado en su totalidad por los valores iniciales de dicho vector. Estos valores iniciales se convierten en los únicos parámetros a optimizar y la única restricción restante se utiliza para especificar la condición de terminación.

El cálculo de variaciones es una rama de las matemáticas y se puede ver como una generalización del cálculo convencional. Tiene como objetivo encontrar la curva, plano, etc., para el cual una función $f(x)$ dada tiene un valor estacionario (el cual en la mayoría de problemas físicos se trata de un mínimo o un máximo); es decir, se trata de un punto donde la derivada de la función es cero, (ver figura 4.4).


Figura 4.4: Punto estacionario.

Para los problemas de programación no lineal tanto en los métodos directos como en los indirectos, los algoritmos de búsqueda local basados en el gradiente descendiente tales como el método de Newton [6], Fletcher-Reeves y Levenberg-Marquardt, entre otros, son muy populares debido a su facilidad para ser programados y su efectividad en cierto tipo de problemas [39]. De cualquier forma, los algoritmos clásicos sólo son capaces de encontrar óptimos locales en el vecindario cercano a aproximaciones iniciales, además de volverse inestable cuando las funciones objetivo son multimodales y el gradiente es discontinuo. El problema de nuestro interés tiene precisamente la característica de contar con muchos óptimos locales, lo cual hace aún más complicado encontrar el óptimo global. Un inconveniente más que presentan los algoritmos de optimización tradicionales es que no pueden lidiar directamente con problemas multiobjetivo. En lugar de eso, se suelen replantear los problemas multiobjetivo como problemas

con un solo objetivo, el cual se puede ver como una combinación lineal de los objetivos del problema original. Los resultados obtenidos por medio de este tipo de métodos, son muy sensibles a los factores que le dan el peso a cada función objetivo además de dar como resultado una solución única y no un conjunto de óptimos de Pareto. Por las razones mencionadas con anterioridad, es que se buscan algoritmos más robustos, que puedan enfrentarse a las exigencias planteadas.

Cuando se hace uso de un motor de bajo impulso en un vehículo espacial, se introduce un control continuo histórico del ángulo de impulso. Si se utiliza un mecanismo de optimización numérico clásico, sin importar si se trata de métodos directos o indirectos, se requiere de buenas aproximaciones iniciales, para que el algoritmo converja de manera satisfactoria [14]. En contraste, los algoritmos evolutivos y, particularmente, los algorimos genéticos, basados en las ideas evolutivas de la selección, cruza y mutación para generar soluciones a partir de poblaciones completamente aleatorias no requieren de buenas aproximaciones iniciales a las soluciones óptimas, y suelen partir de un conjunto de soluciones aleatorias para realizar la optimización [20].

## 4.3. Metodología

Ahora nos daremos a la tarea de definir el problema de optimización de trayectorias de vehículos de bajo impulso de manera general. Para ello, describiremos los métodos desarrollados hasta el momento para resolver el problema de nuestro interés.

### 4.3.1. Problema de optimización de trayectorias en vehículos de bajo impulso

Los sistemas de propulsión eléctrica son uno de los sistemas de propulsión más eficientes disponibles en la actualidad en cuanto a misiones espaciales se refiere. Si consideramos que estos sistemas, a pesar de gastar mucho menos combustible, tardan más tiempo para llegar a su destino, además de limitar la cantidad de masa que se va a transportar, requiriendo un tamaño más pequeño del vehículo espacial, podríamos pensar que no son la mejor opción para una misión espacial. No obstante, este tipo de propulsión es muy eficiente, lo que los hace ser muy atractivos sobre todo cuando los recursos de los programas espaciales se encuentran limitados, como ocurre en la actualidad. El impulso provisto por estos motores

es relativamente pequeño, y típicamente es del orden de fracciones de un Newton. Como resultado, para poder realizar cualquier tipo de maniobra se requiere un impulso continuo durante periodos de tiempo prolongados. Este último punto hace más desafiante el problema de optimización de rutas de vehículos espaciales de bajo impulso que la optimización de trayectorias en vehículos de propulsión química donde sólo se deben optimizar un número reducido de maniobras.

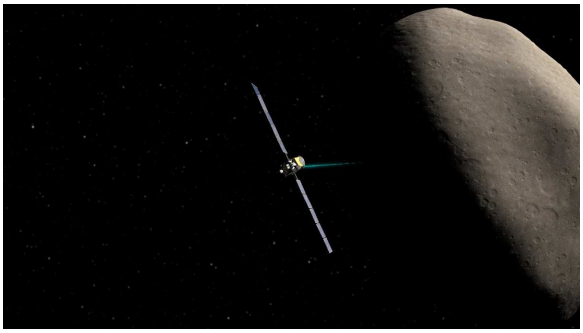


Figura 4.5: Vehículo impulsado por un propulsor de iones.

Las ecuaciones de movimiento de un vehículo espacial están dadas por un conjunto de ecuaciones diferenciales ordinarias, las cuales incluyen tanto el efecto del impulso del motor, como las fuentes de gravedad externas, así como las fuerzas inerciales, cuando no se consideran marcos de referencia no inerciales.

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \tag{4.1}$$

La ecuación 4.1 describe el sistema que tratamos de optimizar, donde $\mathbf{x}(t)$ es conocido como vector de estado, y está constituido por la posición, velocidad y masa del vehículo espacial; $\mathbf{u}(t)$ es el vector de control y representa el vector de aceleración, $\mathbf{p}$ son los parámetros del sistema que no dependen del tiempo y finalmente $t$ es el tiempo. El problema de optimización de trayectorias de vehículos de bajo impulso consiste en minimizar el índice de desempeño $J$, sujeto a las ecuaciones de movimiento y a las condiciones inicial $\psi_0$ y final $\psi_f$ del vehículo.

$$J = \int_{t_0}^{t_f} \mathbf{q}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t]dt \tag{4.2}$$

sujeto a

$$\psi[\mathbf{x}(t_0), \mathbf{u}(t_0), \mathbf{p}, t_0] = \psi_0 \quad (4.3)$$
$$\psi[\mathbf{x}(t_f), \mathbf{u}(t_f), \mathbf{p}, t_f] = \psi_f \quad (4.4)$$

Típicamente, el índice de desempeño $J$ está dado por la masa final del vehículo espacial, el tiempo de vuelo o una combinación lineal de ambos. Un mecanismo alternativo puede ser utilizar el concepto de dominancia de Pareto, donde se optimizan de manera simultánea ambos objetivos, tanto la cantidad de combustible como el tiempo de vuelo. El conjunto de soluciones son soluciones óptimas de Pareto, y son consideradas igual de buenas todas ellas en términos del vector de objetivos dado por la masa final y el tiempo de vuelo. Ambas aproximaciones, tanto el modelo escalar como el vectorial se utilizaron en diferentes pruebas de esta tesis con distintos algorimos de optimización.

Formalmente, el problema de optimización de trayectorias o problema de control óptimo se puede definir como un conjunto de $N$ fases. En general, la variable independiente $t$ (tiempo) para la fase $k$ se define sólo en la región $t_0^{(k)} \leq t \leq t_f^{(k)}$. En muchas aplicaciones, tanto el tiempo como la fase son secuenciales, esto quiere decir que $t_0^{(k+1)} = t_f^{(k)}$. Dentro de la fase $k$ la dinámica del sistema se describe como el vector

$$\mathbf{z} = [\mathbf{x}^{(k)}(t), \mathbf{u}^{(k)}(t), \mathbf{p}, t]^T \quad (4.5)$$

formado por $n_x$ variables de estado y $n_u$ variables de control, respectivamente. Adicionalmene, se pueden incorporar $n_p$ parametros $\mathbf{p}^{(k)}$, los cuales no dependen de $t$.

Como se mencionó anteriormente, la dinámica del sistema se define por un conjunto de ecuaciones diferenciales ordinarias escritas de forma explícita, el cual se conoce como ecuaciones de estado del sistema 4.1. donde $\mathbf{x}$ es el vector de estado de $n_x$ componentes. Las condiciones iniciales descritas en las ecuaciones 4.3 se deben definir dentro de un intervalo que sea válido, como se muestra en la ecuación 4.6.

$$\psi_{0l} \leq \psi_0 \leq \psi_{0u} \quad (4.6)$$
$$\psi_{fl} \leq \psi_f \leq \psi_{fu} \quad (4.7)$$

Además, la solución debe satisfacer las restricciones algebraicas de desigualdad de la forma

$$\mathbf{g}_l \leq \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \leq \mathbf{g}_u \tag{4.8}$$

donde $\mathbf{g}$ es un vector de tamaño $n_g$. También se imponen límites en las variables de estado

$$\mathbf{x}_l \leq \mathbf{x}(t) \leq \mathbf{x}_u \tag{4.9}$$

y en las variables de control

$$\mathbf{u}_l \leq \mathbf{u}(t) \leq \mathbf{u}_u \tag{4.10}$$

Cabe destacar que las restricciones de igualdad pueden ser definidas si el límite superior y el inferior son iguales, es decir, $\mathbf{g}_l = \mathbf{g}_u$.

La ecuación 4.2 envuelve las funciones de cuadratura $q$. De manera conjunta nos podemos referir a las funciones evaluadas durante una fase como el vector de funciones continuas $\mathbf{F}(t)$, el cual está dado por

$$\mathbf{F}(t) = \begin{bmatrix} \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \\ \mathbf{g}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \\ \mathbf{q}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}, t] \end{bmatrix} \tag{4.11}$$

De manera similar, las funciones evaluadas en un punto en particular, como es el caso de las condiciones inicial ($\psi_0$) y final ($\psi_f$), se les conoce como puntos de las funciones.

El problema de control óptimo básico consiste en determinar los $n_u$ vectores de control $u^{(k)}(t)$ para las $k$ fases que minimicen el índice de desempeño

$$\begin{aligned} J = \phi[&\mathbf{x}(t_0^{(1)}), t_0^{(1)}, \mathbf{u}(t_0^{(1)}), \mathbf{p}^{(1)}, \mathbf{x}(t_f^{(1)}), t_f^{(1)}, \mathbf{u}(t_f^{(1)}), \ldots, \\ &\mathbf{x}(t_0^{(N)}), t_0^{(N)}, \mathbf{u}(t_0^{(N)}), \mathbf{p}^{(N)}, \mathbf{x}(t_f^{(N)}), t_f^{(N)}, \mathbf{u}(t_f^{(N)})] \end{aligned} \tag{4.12}$$

De la ecuación 4.12 podemos notar que el valor de las funciones objetivo depende de los valores obtenidos en cada una de las $N$ fases del proceso.

Al concepto de fase, también se le conoce como arco, y se introduce para segmentar el dominio del tiempo. De esta forma, el conjunto de ecuaciones diferenciales no puede cambiar en una misma fase; sin embargo, puede ser modificado de una fase a otra. La razón por la que se introducen fases en el diseño del problema, es para cambiar la dinámica del sistema; por ejemplo, cuando nos referimos a un cohete de múltiples etapas. El límite de una fase es conocida usualmente como punto de unión. Una condición en el límite de una fase que define únicamente el final de ésta es llamada criterio para

un acontecimiento (un problema bien planteado de la definición 4, para el cual sólo se puede tener un criterio por cada evento). Normalmente, la simulación de una trayectoria liga las fases contiguas, forzando a que los estados sean continuos; por ejemplo $\mathbf{x}(t_f^{(1)}) = \mathbf{x}(t_0^{(2)})$. El conjunto de ecuaciones diferenciales 4.1 están escritas como un sistema explícito de ecuaciones diferenciales de primer orden, es decir, que la primera derivada aparece en la parte izquierda de cada una de las ecuaciones. Al caso en que la función objetivo, descrita por la ecuación 4.12, está escrita en términos de cantidades evaluadas al final de cada fase, se le conoce como problema de Mayer. Si la función objetivo sólo involucra una integral (ecuación 4.2), entonces se trata de un problema de Lagrange. Si ambos términos se encuentran presentes, entonces estaremos hablando del problema de Bolza.

Todos los métodos numéricos para resolver el problema de optimización de trayectorias incorporan algún tipo de iteración con un número indefinido de incógnitas. La solución del problema de control óptimo se fue desarrollando paralelamente con los métodos de programación no lineal (PNL). El esquema principal de los métodos iterativos para resolver problemas de optimización fue desarrollado por Newton hace más de 3 siglos. Éste y otros métodos son muy utilizados en el campo de la optimización; sin embargo, se requiere que tanto la $\mathbf{F}(x)$, $\nabla\mathbf{F}(x)$ y $\nabla^2\mathbf{F}(x)$, existan y sean continuas, por lo que dichos métodos son insuficientes para poder resolver el problema de optimización de trayectorias.

### 4.3.2. Método directo

El método directo optimiza directamente las variables de control de impulso $\mathbf{u}(t)$, para minimizar o maximizar el índice de desempeño $J$. En este método, las variables de control contínuo son discretizadas, es decir, se dividen en fases o arcos en el dominio del tiempo y de la longitud. Las variales de control discretas se ajustan directamente y se optimizan para satisfacer tanto las condiciones en los límites como las restricciones. Existen diversos medios para discretizar las variables de control; dependiendo del mecanismo adoptado, se tendrá un mejor desempeño en el proceso de optimización. Si se realiza una discretización demasiado apegada a la realidad, se requerirá mucho mayor tiempo de cómputo mientras que una discretización muy somera, tardará mucho menos tiempo; sin embargo, no estaremos resolviendo el problema planteado originalmente. Por lo tanto, debemos hacer un compromiso entre la fidelidad y la eficiencia del sistema.

### 4.3.3. Método indirecto

El método indirecto introduce un vector de estado externo y optimiza de manera indirecta las variables de control ajustando los valores iniciales del nuevo vector de estado. Este método utiliza las condiciones de optimalidad para transformar el vector de control a una función de valor inicial del nuevo vector de estado. Esta transformación combinada con las condiciones de Euler-Lagrange nos permiten trabajar con un problema cuya solución satisface las condiciones de optimalidad local. Este método es un híbrido ya que el vector de estado introducido es en realidad el conjunto de parámetros de control en el propio vector de impulso. Esta aproximación planteada es conveniente integrarla dentro de cualquier algoritmo de optimización con restricciones, tales como los algoritmos evolutivos multiobjetivo (AEMO). Además de los buenos resultados que suelen arrojar los algoritmos evolutivos, plantean otra bondad más en este problema en particular, ya que no es necesario que las funciones objetivo sean continuas ni que satisfagan las condiciones que se requieren para calcular el gradiente, lo cual los hace relativamente más sencillos y generales de usar que los métodos tradicionales.

Los algoritmos genéticos se han utilizado para optimizar los objetivos del problema de trayectorias interplanetarias con motores de bajo impulso en el pasado. Algunos de los autores más importantes en el área son la Dra. Seungwon Lee y su equipo de colaboradores del Jet Propulsion Laboratory (JPL), los cuales modelaron el sistema como un problema de optimización multiobjetivo, y utilizando el *Nondominated Sorting Genetic Algorithm* [11] (NSGAII) y métodos indirectos en 5 diferentes instancias del problema original han llegado a soluciones de referencia en tiempos aceptables, [28, 30, 29]. Otro pionero en el campo fue John William Hartmann quien desarrolló en los 1990's un algoritmo basado en el NSGA-II, [22].

## 4.4. Órbitas y trayectorias

A continuación se describe lo que es una órbita, los elementos que la conforman y los tipos de órbitas planteadas en el desarrollo del problema, así como la trayectoria para la que debemos encontrar la secuencia de arcos o fases.

### 4.4.1. Elementos que componen una órbita

Una órbita es la trayectoria que realiza un objeto alrededor de otro mientras está bajo la influencia de una fuerza, como es la gravedad. Las órbitas elípticas están definidas por seis parámetros, éstos son: el semieje mayor($a$), la excentricidad($e$), la inclinación($i$), la periapsis($\omega$), la longitud del nodo ascendente($\Omega$) y la anomalía($f$).

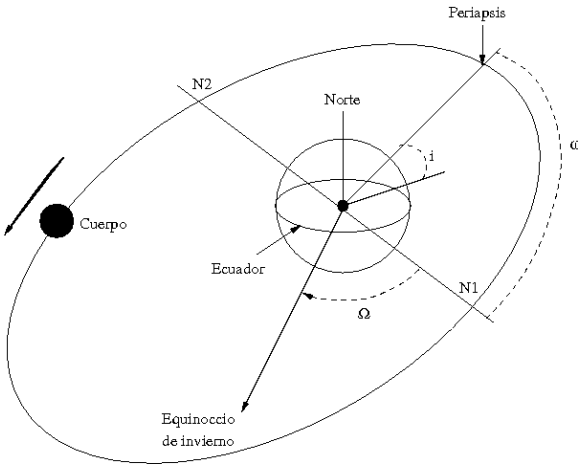



Figura 4.6: Parámetros que definen una órbita.

Una órbita satelital sigue una forma ovalada conocida como elipse con el cuerpo circundado en uno de los focos de la misma. Una elipse se define como la curva que tiene la propiedad de que para cada punto que la forma, la suma de la distancia de los focos al punto se mantiene constante. A la línea recta más larga que se puede dibujar dentro de la elipse y que pase por el centro de la misma se le conoce como eje mayor y a la mitad de la longitud de dicha recta se le conoce como semieje mayor, mientras que a la recta más corta que pasa por el centro se le conoce como eje menor.

La excentricidad está definida como la distancia entre los focos dividida por la longitud del eje mayor. La excentricidad puede tomar valores entre $0 \leq e \leq 1$. Cuando $e = 0$, se trata de una parábola; si $e = 1$, entonces estamos hablando de una hipérbola y si el valor es $0 < e < 1$, entonces se trata de una elipse.

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{4.13}$$

La inclinación es la distancia angualr entre el plano de la órbita del saté-

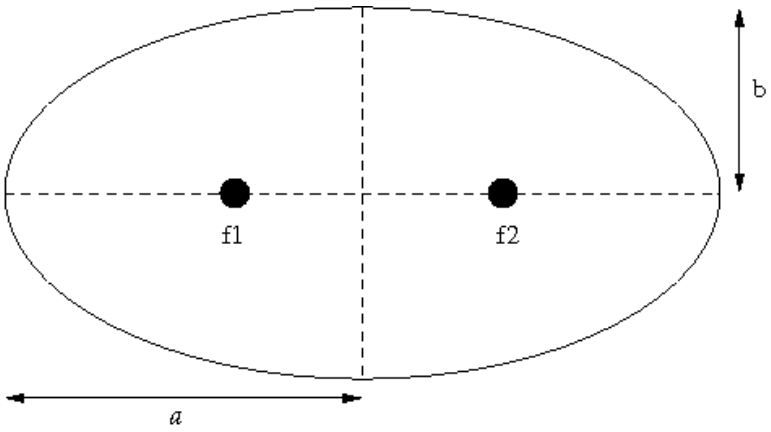



Figura 4.7: Elipse con semieje mayor $a$ y semieje menor $b$.

lite y el ecuador del cuerpo primario o central. Una inclinación de $0$ grados indica que la órbita alredor del ecuador del cuerpo primario está en la misma dirección que el movimiento de rotación y se le conoce como directa. Una inclinación de 90 grados indica un órbita polar. Mientras que una inclinación de 180 grados es conocida como una órbita ecuatorial invertida.

La periapsis es el punto en una órbita del satélite más cercano al cuerpo primario. De manera contraria, el punto más alejado desde la órbita al cuerpo central es conocido como apoapsis. Suelen tomar diferentes nombres dependiendo del cuerpo que se circunde, por ejemplo, si el cuerpo central se trata del sol se le conoce como perihelio y afelio, respectivamente, mientras que si se trata de la Tierra se le conoce como perigeo y apogeo.

Los nodos ascendente (N1) y descendente (N2), son los puntos donde la órbita del satélite cruza un plano imaginario sobre el que se encuentra precisamente el ecuador del cuerpo central. Si el satélite cruza el plano de sur a norte del cuerpo central, entonces nos referimos al nodo ascendente, y si lo cruza de norte a sur nos estaremos refiriendo al nodo descendente. La longitud del nodo ascendente es la longitud celestial de dicho nodo. La longitud celestial es análoga a la longitud medida en la Tierra y está dada en grados en dirección de las manecilla del reloj desde el eje formado por ambos nodos y en dirección del punto del equinoccio de invierno.

### 4.4.2. Trayectoria geoestacionaria

El caso planteado que se pretende resolver consiste esencialmente en una trayectoria coplanar, de una trayectoria circular a otra, desde una órbita

baja de la Tierra a una órbita geoestacionaria. No existen restricciones de Periapsis durante la trayectoria. El tiempo de vuelo máximo permisible es de 500 días.

Una órbita geoestacionaria, consiste en un tipo particular de ruta que siguen los cuerpos alrededor de la Tierra a la misma velocidad que su propio movimiento de rotación, lo cual se puede traducir como si el objeto que circunda a la tierra no se moviera en el horizonte, y se mantuviera en el mismo punto aunque en realidad sí lo está haciendo. Este tipo de órbitas son las que siguen típicamente los satélites artificiales.

La trayectoria se puede ver como un conjunto de arcos, en donde cada uno es un segmento de ella misma y donde cada arco está definido por los ángulos $\alpha$, $\beta$ (ángulo azimutal y de elevación) y una bandera que nos dice si en ese momento se aplica o no un impulso. En la figura 4.8 se pueden apreciar las fases de una trayectoria para ir de una órbita a otra.

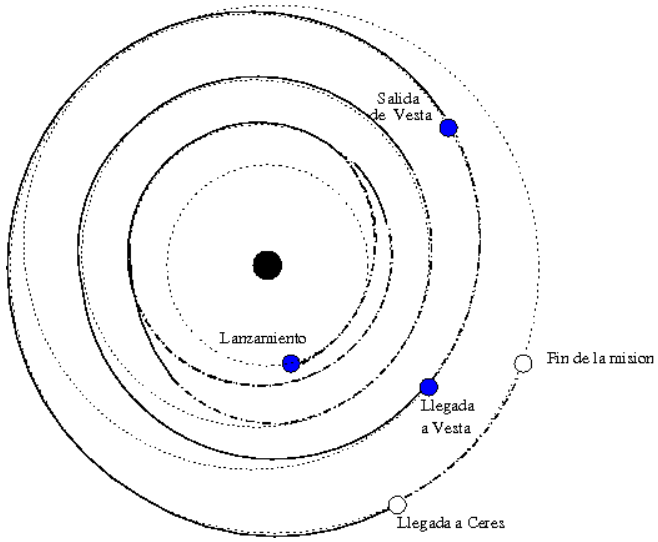



Figura 4.8: Trayectoria de una órbita inicial a una órbita final de la misión Dawn.

## 4.5. Análisis numérico

Las soluciones numéricas para los problemas de valor inicial, descritos a mayor detalle en el capítulo 2, en sistemas de ecuaciones diferenciales ordinarias, es fundamental en casi todos los métodos de optimización de

trayectorias. Existen una gran variedad de estos métodos como son el de Euler, el del punto medio, Runge-Kutta, etc, cada uno con sus debilidades y fortalezas. A continuación presentamos el algorimo utilizado para determinar la ruta del vehículo espacial por medio del método de Runge-Kutta de cuarto orden.

**Algoritmo 7**: Pseudocódigo de evaluación de las funciones objetivo.

```
1  Input: A //Arcos de la trayectoria
2  Output: F //Funciones objetivo
3  begin
4      O_Inicial ← FijarOrbitaInicial()
5      O_Final ← FijarOrbitaFinal()
6      V ← FijarParametrosVehiculo()
7      C ← FijarParametrosCuerpoCentral()
8      OP ← InicializarPropagadorOrbitas(O_inicial, V, C)
9      P ← FijarParametrosArcosImpulso()
10     S ← CrearTrayectoriaConArcoImpulsoUniforme(OP, A, P)
11     F ← CaulcularAptitud(S);
12 end
```

El algoritmo 7 describe el procedimiento para hacer el cáculo numérico de una sola evaluación de las funciones objetivo ya planteadas. El primer paso de este algoritmo es fijar los valores de la órbita inicial y final, es decir debemos tener una descripción completa del punto de partida y llegada del vehículo espacial. En la tabla 4.1 se muestran los valores de la órbita inicial para el caso de estudio, es decir, los valores propuestos describen una órbita baja de la Tierra. De igual forma se presentan los valores de la órbita final y las tolerancias en cada uno de dichos parámetros. El valor numérico del semieje mayor de la órbita final es de 42000 km., el cual describe una órbita geoestacionaria, y es más o menos la distancia a la que se colocan los satélites artificiales.

### 4.5.1. Inicialización de parámetros

Ya hemos descrito los elementos que conforman una órbita. Ahora toca el turno de describir las variables involucradas en un vehículo espacial. En

general son sólo tres parámetros a considerar cuando nos referimos a la nave. Estos son la masa incial del vehículo, la cual está dada en kilogramos; el impulso específico del motor dado en segundos y la magnitud del impulso, el cual se expresa en Newtons. En la tabla 4.2 se listan los valores de los parámetros antes descritos.

El caso de estudio presenta un órbita concéntrica alrededor de la Tierra o geocéntrica, por lo que los parámetros del cuerpo central son tomados precisamente de ésta. El primero de ellos es la constante de gravitación y la segunda es el radio de la Tierra. La tabla 4.3 muestra los parámetros a considerar para este caso particular, cuando la nave espacial gira alrededor de la Tierra.

Los arcos o fases de las trayectorias están definidos por dos ángulos y un valor escalar, los cuales nos indican la dirección del impulso en coorde-

Tabla 4.1: Elementos de la órbita inicial

| **Parámetro** | **Variable** | $\mathbf{O_{Inicial}}$ | $\mathbf{O_{Final}}$ | $\epsilon$ |
|---|---|---|---|---|
| Semieje mayor($km$) | $a$ | 7000,0 | 42000 | 100 |
| Excentricidad | $e$ | 0,01 | 0,01 | 0,001 |
| Inclinación($rad$) | $i$ | $\frac{,05\pi}{180}$ | $\frac{,05\pi}{180}$ | 0 |
| Periapsis ($rad$) | $\omega$ | 0 | 0 | 0 |
| Longitud del nodo ascendente ($rad$) | $\Omega$ | 0 | 0 | 0 |
| Anomalía ($rad$) | $f$ | 0 | 0 | 0 |

Tabla 4.2: Elementos de la nave espacial

| **Parámetro** | **Variable** | **Valor numérico** |
|---|---|---|
| Masa inicial ($kg$) | $m_{inicial}$ | 300,0 |
| Impulso específico ($s$) | $isp$ | 3100,0 |
| Magnitud del impulso ($N$) | $ts$ | 1 |

Tabla 4.3: Constantes de la Tierra

| **Parámetro** | **Variable** | **Valor numérico** |
|---|---|---|
| Constante gravitacional ($\frac{km^3}{s^2}$) | $\mu$ | 398600,485042960 |
| Radio ($km$) | $r$ | 6578,00 |

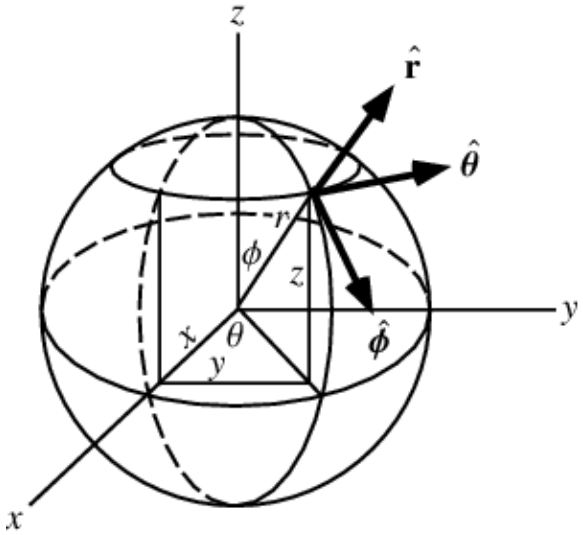



Figura 4.9: Coordenadas esféricas.

nadas esféricas y la magnitud del vector. Éstas también son conocidas como coordenadas polares. Con base en este sistema podemos describir la posición en una esfera por medio de estos 3 parámetros. El ángulo $\alpha$ o ángulo azimutal se mide con respecto al eje de las $X$ en direción del eje $Y$ y puede tomar valores en el intervalo $0 \leq \alpha \leq 2\pi$. Al ángulo azimutal también se le conoce como longitud ($\theta$). El ángulo polar o de elevación se mide desde el eje $Z$ en dirección al plano $XY$ y puede tomar valores en el intervalo $0 \leq \beta \leq \pi$, y también se le conoce como colatitud o complemento de la latitud ($\phi$). La magnitud del vector en este caso es tomada como el impulso del motor en un instante $t_0$ y lo denotamos por la letra $r$.

### 4.5.2. Creación de la trayectoria

El algoritmo 8 es utilizado para calcular el punto al cual se llega a partir de los parámetros iniciales y la configuración del sistema. Este algoritmo recibe los valores iniciales tanto de la órbita, la nave espacial y el cuerpo central, los arcos de la ruta trazada y la longitud máxima del arco, para devolver los parámetros de la órbita calculada y la masa final de la nave, a partir de la cual podemos calcular la cantidad de combustible utilizado por medio de una simple resta y el tiempo de vuelo requerido.

**Algoritmo 8**: Pseudocódigo del mecanismo propagador de órbitas.

```
1  Input: OP, A, P
2  Output: S
3  begin
4      tiempo_vuelo ← 0
5      for i=0 to 𝒩 do
6          tiempo_gastado ← 0
7          while tiempo_gastado < tiempo_max do
8              f ← (ts/m_inicial)
9              f_r ← (f cos(β[i]) sin(α[i]))
10             f_t ← (f cos(β[i]) cos(α[i]))
11             f_h ← (f sin(β[i]))
12             Runge-Kutta45()
13             m_initial ← (m_initial − combustible_gastado)
14             tiempo_gastado ← (tiempo_gastado + tiempo)
15             tiempo_vuelo ← (tiempo_vuelo + tiempo)
16             if VerificarOrbita() != OK then
17                 Error()
18                 Regresar S
19         end
20     end
21     Regresar S
22 end
```

# Capítulo 5

# El Algoritmo Genético Multiobjetivo de Múltiples Resoluciones II

En este capítulo se presenta la descripción del algoritmo utilizado para resolver el problema de optimización de rutas de vehículos espaciales. Además se describen los parámetros relacionados con el proceso de optimización y los valores que se asignaron a cada uno de ellos.

El algoritmo utilizado es conocido como MRMOGA o Algoritmo Genético Multiobjetivo de Múltiples Resoluciones. Este algoritmo ha demostrado ser un digno contendiente en el campo de los Algorimos Evolutivos Multiobjetivo Paralelos. Sin embargo, hubo de sufrir modificaciones, ya que el problema en particular que aquí tratamos de resolver así lo exigía.

## 5.1. Introducción a los algoritmos genéticos paralelos

El diseño e implementación de algoritmos paralelos ha sido motivado en gran medida con el objetivo de reducir el tiempo que se tarda en ejecutar un proceso, distribuyendo el trabajo entre las diversas entidades que conforman la red, de tal forma que se procesen de manera paralela.

Existen diversos modelos para paralelizar un Algoritmo Genético los cuales se han clasificado en cuatro diferentes tipos [37]: los modelos maestro-esclavo, de islas, celulares y los AGP jerárquicos.

- Los algoritmos construidos bajo el esquema maestro-esclavo cuentan

con una sola población y la evaluación de la aptitud se distribuye entre varios procesadores. Los mecanismos de selección, cruza y mutación consideran a la población en su totalidad.

- El modelo de isla o algoritmo genético de grano grueso consiste en un conjunto de subpoblaciones que evolucionan de manera independiente entre ellas pero que realizan migraciones eventuales de un número determinado de individuos.
- El modelo celular o algoritmo genético de grano fino consiste en una sola población global la cual es estructurada espacialmente. Es muy común ver que sólo se encuentra un individuo por cada procesador en este modelo, y los operadores de selección, cruza y mutación se encuentran restringidos a un pequeño vecindario.
- El modelo jerárquico combina el modelo de islas ya sea con el modelo maestro-esclavo o celular.

El modelo maestro-esclavo no modifica significativamente lo que plantea originalmente un AG convencional, así que puede verse simplemente como un mecanismo para disminuir los tiempos de procesamiento. En contraste, los tres modelos restantes cuentan con diferencias de fondo con respecto a los AG comunes. Dichos cambios se ven reflejados tanto en el propio algoritmo, como en la forma en que van evolucionando las poblaciones. El número de generaciones, el tamaño de las subpoblaciones, el porcentaje de individuos que migran, la frecuencia de migración, etc. son parámetros fundamentales que impactan directamente con el desempeño del modelo de islas.

Un punto más que puede ser considerado en los AG al momento de paralelizarlos es la implementación de las operaciones de cruza y mutación. Encontrar un buen balance entre estos operadores es sumamente importante para mejorar el desempeño. Una forma de lograr esto es modificando los porcentajes de cruza y mutación conforme se avanza en cada generación. Los AG convencionales intentan aplicar la cruza y un porcentaje de mutación variable junto con una presión de selección muy elevada, la cual además también se incrementa gradualmente conforme van pasando las generaciones. Algunos de los AG convencionales que implementan este tipo de mecanismos, han demostrado ser mucho más efectivos que aquellos que utilizan una presión de selección baja, además de mantenerla constante durante todo el proceso de evolución, al igual que el porcentaje de cruza y mutación. Es por esto último que el primer modelo descrito es mucho más atractivo para ser implementado en un AG paralelo.

El modelo que se utilizó para el desarrollo de esta tesis es el modelo de islas, el cual fue descrito en términos generales con anterioridad. Específicamente se utilizó el MRMOGA (Algoritmo Genético Multiobjetivo de Múltiples Resoluciones)[25, 31], desarrollado por el grupo EVOCINV [1], al cual se le realizaron diversas modificaciones en los operadores de cruza y mutación, además de modificar la forma como se manejan las poblaciones de cada isla y el mecanismo de migración. Esto, debido a que era necesario adaptarlo para que pudiera manejar individuos de longitud variable, lo cual es un requerimiento indispensable para atacar el problema de optimización de rutas de vehículos espaciales. Otro punto importante a destacar es el manejo de las restricciones, el cual también fue modificado, ya que originalmente sólo se contaba con un mecanismo muy simple, basado en el número de restricciones que eran violadas. Sin embargo, en nuestro problema era necesario saber cuáles eran aquellos individuos que violaban en menor medida las restricciones.

A continuación se describe de manera general el MRMOGA así como las modificaciones que se realizaron sobre cada una de los etapas del algoritmo a fin de que pudiera ser utilizado en este problema en particular.

## 5.2. El Algoritmo Genético Multiobjetivo de Múltiples Resoluciones

El MRMOGA es un algoritmo basado en el modelo de islas, en el que cada isla podría ser diferente de las demás. Es decir, cada subpoblación es totalmente independiente una de otra y puede manejar sus propios parámetros, por lo que se le considerará un modelo heterogéneo.

Su principal característica es que cada población codifica las soluciones con una resolución diferente, es decir, se utiliza un número diferente de dígitos de precisión después del punto decimal. Otro punto a destacar es que el mecanismo de migración entre las subpoblaciones es asíncrono y en él se ven inmiscuídas tanto la población principal como la población secundaria, lo cual evidentemente se realiza con el objetivo de intercambiar sólo a los mejores individuos. Cuenta con un esquema que divide el espacio de búsqueda en el dominio de las variables de decisión, por lo que cada isla realiza la búsqueda sobre diferentes espacios, los cuales a pesar de no ser mutuamente excluyentes, sí varían en cuanto a su tamaño. Y, finalmente, incorpora una técnica que determina cuando una de las islas converge a

[1] Ver http://www.cs.cinvestav.mx/~EVOCINV

una solución con el objetivo de variar la resolución actual de las soluciones candidatas.

Este algoritmo se utiliza única y exclusivamente en problemas de optimización numérica y no combinatoria, por lo que también cuanta con un mecanismo para adaptar una cadena binaria de cierta resolución a otra, cuando esto sea necesario.

### 5.2.1. ¿Por qué múltiples resoluciones?

Cualquier persona que haya tenido cierto acercamiento con los algoritmos evolutivos multiobjetivo, sabe que los problemas con espacios de búsqueda más pequeños siempre serán más fáciles de resolver, que aquellos que cuenten con espacios de búsqueda muy grandes. Esta última aseveración suena lógica, inclusive no sólo en el campo de la optimización multiobjetivo. Imaginemos que se ha perdido una aguja en un pajar; no se requerirá el mismo esfuerzo para encontrar dicha aguja en un pajar que ocupa un espacio de $1m^3$ que si se tratara de uno que abarque varias hectáreas. Como hemos visto en el ejemplo anterior, el tamaño sí importa. Si nos basamos en el principio descrito, y lo extrapolamos al campo de la optimización multiobjetivo, podríamos decir que un algoritmo podría acercarse de manera más rápida, es decir en un menor número de iteraciones, utilizando una resolución baja para la representación de las soluciones del problema que si se utilizara una resolución mayor, esto debido a que también el espacio de búsqueda sería mucho más grande. Aunque las soluciones encontradas con una resolución baja, no son parte necesariamente del frente de Pareto, sí son buenas aproximaciones que ayudarán en gran medida a guiar la búsqueda si se pudiera de alguna forma modificar dicha resolución para las generaciones subsecuentes. Este es precisamente uno de los principios sobre los cuales se sustenta el MRMOGA.

El MRMOGA, como se precisó con anterioridad, está basado en el modelo de islas, en el cual cada una de dichas islas puede tener una resolución diferente e, inclusive, esta resolución puede variar en el tiempo. Inicialmente habrá islas con una resolución baja, las cuales tendrán el objetivo de acercarse de manera acelerada al frente de Pareto, al encontrar buenas soluciones sin necesidad de que sean las mejores. Las mejores soluciones de cada isla serán migradas por medio de algún mecanismo a otra isla con mayor resolución para realizar una búsqueda más minuciosa en el vecindario de dichas soluciones que, de antemano, se sabe son buenas.

Si analizamos un poco más a detalle la estrategia planteada anteriormente, nos daremos cuenta que el espacio de búsqueda de las poblaciones

con menor resolución, además de ser más pequeño y más sencillo de manejar, también está contenido dentro de los espacios de mayor resolución ya que todos los puntos del espacio con la resolución menor estarán contemplados en el espacio más grande.

### 5.2.2. Descripción del MRMOGA

El MRMOGA se puede ver como un conjunto de AG secuenciales, que comparten individuos por medio de un mecanismo de migración y que pueden cambiar de resolución debido a una convergencia prematura en las soluciones encontradas en cada isla. Es decir, el algoritmo se compone de diversas subpoblaciones las cuales son manipuladas, cada una de manera independiente, como si se tratara de diversas instancias de algoritmos genéticos multiobjetivo secuenciales, con la peculiaridad de que cada proceso puede compartir las mejores soluciones que haya encontrado con los demás. Esto busca lograr un mejor desempeño de nuestro algoritmo de manera global, además de que cada instancia es capaz de modificar, de manera autónoma, el número de bits para codificar una solución, con lo que podremos estar seguros que nos acercaremos de manera progresiva al frente de Pareto.

Los puntos importantes a tratar del MRMOGA son los siguientes:

- El Algoritmo Genético Multiobjetivo Paralelo.
- El Algoritmo Genético Multiobjetivo Secuencial.
- Los mecanismos de selección, cruza y mutación.
- Representación de los individuos.
- Mecanismos de migración.
- Conversión de individuos de una resolución a otra.
- El manejo de restricciones.

#### El Algoritmo Genético Multiobjetivo Paralelo

Para hacer más eficiente el proceso de búsqueda a lo largo del espacio de las variables, se decidió utilizar un algorimo genético paralelo, el cual distribuye de manera equitativa el trabajo a los procesos que se crean en cada uno de los nodos de la red. Cada nodo de la red de datos se considera

como una de las islas, ya que la ejecución del algoritmo se planea con la intención de tener subpoblaciones del mismo tamaño en cada nodo. Además, en cada isla se realizará el mismo proceso de búsqueda, con la única diferencia de que cada una contará con su propia resolución. El algoritmo 9 describe a mayor detalle el procedimiento global del MRMOGA.

La función *Evolucionar($\mathcal{P}_i$)* descrita en el algoritmo 9, se refiere precisamente al proceso ejecutado por el algoritmo genético secuencial que se describe más adelante, y es un componente esencial junto con el cambio de resolución de la isla y la migración de los individuos más aptos.

### Algoritmo Genético Multiobjetivo Secuencial

En el capítulo 1, se describió de manera general la forma en que opera un AG secuencial; ahora vamos a describir de manera particular la forma en que opera el AG secuencial implementado por el MRMOGA.

Como se mencionó en la sección anterior, cada nodo representa una isla, y en cada isla se procesa una subpoblación. Precisamente el algoritmo que procesa las subpoblaciones en cada isla es un AGMO secuencial. El procedimiento secuencial que se lleva a cabo en cada isla se describe en el algoritmo 10.

El primer paso que realiza el AGMO secuencial es calcular el fenotipo de los individuos de la subpoblación que le ha sido asignada. Este procedimiento consiste básicamente en decodificar el genotipo y expresarlo en base 10, ya que las variables del problema se encuentran codificadas en binario. Para obtener el fenotipo, se requiere la ecuación 5.2.

$$I = \sum_{j=n_{genes}*bits_{gen},k=1}^{j=(n_{genes}+1)*bits_{gen}-1} cromosoma[j] * base^k \tag{5.1}$$

$$fenotipo = limite_{inf} + \frac{(limite_{sup} - limite_{inf}) * I}{(base^{bits_{gen}} - 1)} \tag{5.2}$$

El procedimiento que evalúa las funciones objetivo consiste básicamente en llamar al propagador de órbitas descrito en el capítulo anterior. Éste realiza una simulación de lo que ocurriría en el modelo con los parámetros de la órbita en cuestión y a partir de ellos obtiene los valores de la funciones objetivo (tiempo de vuelo y cantidad de combustible utilizado). El tiempo de vuelo, se va calculando conforme va avanzando el algoritmo y es devuelto como un resultado más, mientras que la cantidad de combustible,

**Algoritmo 9**: Pseudocódigo del algoritmo del MRMOGA.

```
1  Input:
2  E : número máximo de épocas
3  G : número de generaciones por época
4  N_isl : número de isla
5  N_ind : número de individuos por isla
6  N_mig: número de individuos que serán migrados
7  P_mig: porcentaje de migración
8  P_cru: porcentaje de cruza
9  P_mut: porcentaje de mutación
10 A_prec[N_isl] : precisiones
11 Output:
12 F : archivo de salida con los mejores individuos de todas las islas
13 begin
14     InicializarParametrosIsla(i, N_ind, P_cru, P_mut, E, G)
       P_i ←InicializarPoblación(i)
15     for e=1 to E do
16         for g=1 to G do
17             Evolucionar( P_i )
18             AlmacenarMejoresSoluciones( P_i )
19             if converge( P_i ) then
20                 cambiarResolucion( A_prec[i] )
21             if volado( P_mig ) then
22                 migrar( P_i, N_mig )
23         end
24     end
25     if soyNodoPrincipal( i ) then
26         F ←ObtenenMejoresIndividuos();
27 end
```

**Algoritmo 10**: Pseudocódigo del AGMO secuencial.

```
1  Input:
2  P_i : población a evolucionar
3  begin
4      CalcularFenotipo( P_i )
5      EvaluarFuncionesObjetivo( P_i )
6      AsignarJerarquia( P_i )
7      AsignarAptitudLineal( P_i )
8      EscalarAptitud( P_i )
9      for j=1 to N_ind do
10         InsertarIndividuoEnFrenteLocal( P_i[j] )
11     end
12     Seleccion( P_i )
13     Cruza( P_i )
14     Mutacion( P_i )
15 end
```

se debe calcular al obtener la diferencia entre la masa actual del vehículo espacial y su masa inicial.

Una vez que se ha obtenido tanto el fenotipo como el valor de las funciones objetivo de cada individuo de la población, se le debe asignar a cada uno de ellos un identificador para jerarquizarlos en función de su aptitud. Este procedimiento consiste básicamente en aplicar la dominancia de Pareto al conjunto de individuos de cada isla. El algoritmo 11, describe el procedimiento para asignar la jerarquía a cada individuo.

Con el algoritmo 12 se determina si un individuo domina a otro. El primer aspecto a considerar es si alguno satisface de mejor manera las restricciones, si es así, el mejor en este rubro dominará al otro. Si ambos satisfacen las restricciones o las rompen en la misma proporción, entonces se debe verificar la siguiente instancia. Esta se refiere a la comparación de las funciones objetivo de ambos individuos, para de esta forma determinar quién es el mejor. Existen tres posibilidades; si el primer individuo es al menos igual de bueno en todos los objetivos y estrictamente mejor en uno de ellos entonces el primer individuo domina al segundo. Si el segundo individuo es al menos igual de bueno en todos los objetivos y es estrictamente mejor en al menos uno de ellos, entonces el segundo individuo dominará al primero. Si el primer individuo es estrictamente mejor al segundo individuo en al menos un objetivo y el segundo individuo es estrictamente mejor al

**Algoritmo 11**: Pseudocódigo para asignar jerarquías

```
1  Input:
2  P_i : población i-ésima a evolucionar
3  begin
4      for j=1 to N_ind do
5          P_i[j].FijarJerarquia(1)
6      end
7      for j=1 to N_ind do
8          for k=1 to N_ind do
9              if domina( P_i[k], P_i[j] ) then
10                 P_i[j].FijarJerarquia( P_i[j]+1)
11         end
12     end
13 end
```

primero en al menos alguno de los objetivos, entonces son individuos no dominados entre sí.

El cálculo de la aptitud lineal, no está relacionada con el valor de las funciones objetivo de cada individuo, sino que se relaciona de manera inversamente proporcional con su jerarquía. De esta forma, los individuos con el índice de jerarquía más baja, tendrán la aptitud lineal más alta.

Después de calcular la aptitud lineal, se escala la aptitud real de cada individuo. Esto se realiza, encontrando la aptitud más alta dentro de la población actual. Una vez que se ha encontrado dicha aptitud, se procede a dividir la aptitud de todos los individuos entre dicho valor.

Una vez calculada la aptitud de cada individuo, intentamos insertar a cada uno de éstos dentro del archivo secundario. El algoritmo 13 describe el procedimiento para insertar un individuo dentro del archivo secundario.

#### Los mecanismos de selección, cruza y mutación

El MRMOGA utiliza una selección universal estocástica, la cual se mantuvo intacta en esta nueva versión salvo por el mecanismo descrito en la siguiente sección, debido al tamaño variable de las soluciones candidatas. Esta técnica fue propuesta originalmente en [21]. Dicho procedimiento se describe a mayor detalle en el algoritmo 14.

**Algoritmo 12**: Pseudocódigo para obtener la dominancia

**Input:**
$\mathcal{I}_1$ : Primer individuo para comparar
$\mathcal{I}_2$ : Segundo individuo para comparar
**begin**
  **if** *Restricciones($\mathcal{I}_1$) <Restricciones($\mathcal{I}_2$)* **then**
    $\mathcal{I}_1$ domina a $\mathcal{I}_2$
  **else if** *Restricciones($\mathcal{I}_2$) <Restricciones($\mathcal{I}_1$)* **then**
    $\mathcal{I}_2$ domina a $\mathcal{I}_1$
  **else**
    **for** *j=1* **to** $\mathcal{N}_{OBJS}$ **do**
      **if** $\forall\, j\, \mathcal{I}_1.funcion(j) \leq \mathcal{I}_2.funcion(j)$ **then**
        **if** $\exists e_1 \therefore \mathcal{I}_1.funcion(e_1) < \mathcal{I}_2.funcion(e_1)$ **then**
          $\mathcal{I}_1$ domina a $\mathcal{I}_2$
      **if** $\forall j \mathcal{I}_2.funcion(j) \leq \mathcal{I}_1.funcion(j)$ **then**
        **if** $\exists e_2 \therefore \mathcal{I}_2.funcion(e_1) < \mathcal{I}_1.funcion(e_2)$ **then**
          $\mathcal{I}_2$ domina a $\mathcal{I}_1$
      **if** $\exists e_1$ *y* $\exists e_2$ **then**
        $\mathcal{I}_2$ y $\mathcal{I}_1$ son no dominados
    **end**
**end**

**Algoritmo 13**: Pseudocódigo para insertar un individuo en el archivo secundario

**Input:**
$\mathcal{I}$ : Individuo a insertar
**begin**
  **if** *$\mathcal{I}$ es el primer individuo a insertar* **then**
    Insertar($\mathcal{I}$)
  **if** $\nexists \mathcal{I}_x \in \mathcal{P}_i \therefore \mathcal{I}_x \preceq \mathcal{I}$ **then**
    **if** *hay espacio en archivo secundario* **then**
      Insertar($\mathcal{I}$)
    **if** *esta lleno el archivo* **then**
      $j \leftarrow$IndividuoRegionMasPoblada()
      ReemplazarIndividuo($\mathcal{P}_i[j]$, $\mathcal{I}$)
**end**

**Algoritmo 14**: Pseudocódigo del algoritmo de selección

```
1  Input:
2  I : Individuo a insertar
3  begin
4      sum ← CalcularSumaAptitudes()
5      med ← CalcularMediana()
6      CalcularValorEsperado(P_i, sum, med)
7      ale ← GenerarNumeroAleatorio(0.0, 1.0)
8      for j=0, i=0 to N_ind, i++ do
9          for tot=ValorEsperado(i); tot > ale; j++ do
10             Seleccionar(i)
11             ale leftarrow ale+1
12         end
13     end
14 end
```

**Representación de los individuos**

Uno de los principales aspectos a atacar en el proceso de optimización de rutas de vehículos espaciales es precisamente que no se sabe de antemano el número de arcos que definen la ruta óptima. Dicho problema plantea una interrogante que nos obliga a manejar de manera especial a los individuos de una población, es decir, éstos deberían ser manejados como cadenas de longitud variable.

Manejar cadenas de longitud variable hace del algoritmo general un proceso más complicado aún. Imaginemos que necesitamos migrar a un individuo de una isla a otra, pero no sabemos de antemano la longitud del mismo. El problema plantea la posibilidad de asignar memoria de manera dinámica a los cromosomas que sean migrados a cada uno de los nodos en particular. Ahora, pensemos en cruzar dos individuos de longitudes diferentes, el resultado sería catastrófico, ya que existen regiones de memoria en las que el individuo más pequeño no existe, por lo que debemos fijar límites dentro de los cuales se puede realizar tanto el proceso de cruza como de mutación.

La manera de resolver dicho problema fue definir un tamaño máximo para el cromosoma. El tamaño máximo que se manejó, depende del grado de exactitud con el que se deseaba manejar el problema, teniendo siempre en consideración, que a un número mayor de arcos, el proceso se vuelve

mucho más lento por el número de variables involucradas.

El número de variables a optimizar está dado por:

$$N_{variables} \leftarrow 2 * N_{MaxArcos} + 2 + 2 * N_{Restricciones} \tag{5.3}$$

donde $N_{MaxArcos}$, representa el número máximo de arcos de los que podría constar una ruta y $N_{restricciones}$ es el número de restricciones que se deben satisfacer del problema planteado. Dependiendo de la instancia del problema, el número de restricciones aumenta.

El caso particular que se trata y que fue descrito en el capítulo anterior fue fijado con un máximo de $8000$ arcos por ruta, es decir que se intentaba encontrar la mejor ruta posible, que no sobrepasara los $8000$ cambios de dirección del vehículo espacial. Este caso en particular cuenta con dos restricciones, una relacionada con el semieje mayor de la ruta y la restante con la excentricidad de la misma. De lo anterior podemos concluir que el número de variables que se requiere optimizar es de:

$$N_{variables} \leftarrow 16006$$

La repartición de las variables está dada de la siguiente manera: $8000$ variables para los ángulos de elevación ($\beta$), $8000$ variables para los ángulos azimutales ($\alpha$) de cada arco, 4 variables para el manejo de restricciones, una varible más que determina la longitud de los arcos y otra que determina el número de arcos, la cual solamente puede tomar valores dentro del intervalo mostrado a continuación:

$$100 \leq N_{arcos} \leq 8000$$

De lo anterior podemos notar que se descartan de antemano todas aquellas soluciones con un número menor a los $100$ arcos y mayor a los $8000$. Los intervalos válidos de cada una de las variables se describieron en el capítulo anterior, excepto para el número de arcos y la longitud de ellos mismos. La longitud de cada arco puede estar dentro de $5 \leq Arc_{length} \leq 30$ grados pra evitar los cambios bruscos de dirección que se ven reflejados como una potencial pérdida de energía, es decir, el cambio de dirección debe ser realizado de manera gradual para obtener el resultado esperado.

Una restricción más que se planteó durante el desarrollo del algoritmo fue evitar cruzar individuos cuya diferencia del número de arcos fuera mayor al $20\,\%$ del número máximo de arcos. Por ejemplo, no se podrían cruzar dos individuos que contaran con $8000$ y $6399$ arcos respectivamente, ya que la diferencia entre ambas soluciones era demasiado grande. Para implementar esta técnica, se verifica en el proceso de selección si efectivamente

ambas soluciones cumplen con dicha restricción impuesta de manera arbitraria, pero que al final se considera fue una buena estrategia para evitar estar realizando una búsqueda completamente aleatoria en el espacio de las variables.

El número máximo de arcos ($8000$) puede sonar hasta cierto punto arbitrario. Sin embargo, está relacionado con las pruebas iniciales que se realizaron al trabajar con este problema, en donde se notó que las mejores soluciones se encontraban muy cerca de los $8000$ arcos. Dichas pruebas fueron realizadas con un algoritmo representativo del estado del arte en su versión secuencial y paralela, y por esa razón es que se adoptó este valor. Se dejó que la variable del número de arcos pudiera oscilar entre los $0$ y los $32000$ arcos, teniendo muy buenos resultados con valores cercanos a $8000$ en tiempos razonablemente cortos, ya que la evaluación de las funciones objetivo con $16000$ variables era terriblemente costoso, computacionalmente hablando.

**Mecanismo de migración**

Recordando lo descrito previamente en la sección 5.2.1 se puede enfatizar el hecho de que en el modelo de islas con múltiples resoluciones, las islas con mayor resolución contemplan el espacio de búsqueda de aquellas islas que cuenten con menor resolución. De tal modo que si fuera necesario intercambiar individuos entre dichas islas tendríamos que convertir el individuo de menor resolución a la resolución adecuada. Sin embargo, el caso contrario no es posible realizarlo, ya que el pasar de una resolución mayor a una menor implicaría la pérdida de información valiosa. Por lo descrito anterirmente, sólo es posible migrar individuos que pertenezcan a islas con menor resolución a aquellas islas que cuenten con una resolución mayor.

Se sigue el esquema de la topología totalmente conectada, la cual permite migrar individuos a las islas con resolución mayor a la propia. El grado de dicha topología es $\delta(i) = i$. En esta disposición lógica, el nodo con mayor resolución recibe individuos de todas las islas con resoluciones menores, lo que lo convierte básicamente en una isla que se encarga de mejorar las soluciones encontradas por sus predecesores.

Un punto relevante a destacar es que se puede dar al caso en el que una isla esté lista para migrar $N_{mig}$ candidatos a las islas con mayor resolución, pero que algunas de ellas no estén listas para recibirlos, debido a que el proceso evolutivo típicamente tomará mayor tiempo si las cadenas binarias son más largas. Es por eso que el mecanismo de comunicación seleccionado para intercambiar soluciones utiliza primitivas no bloqueantes para evitar

que las islas con mayor resolución retrasen a las de menor resolución.

El esquema para migrar individuos consiste en seleccionar $N_{mig}$ individuos del frente de Pareto local de cada isla de manera estocástica, los cuales serán migrados a las islas correspondientes siguiendo la topología descrita anteriormente. Por lo general, el frente de Pareto que se encuentra en cada isla no tiene más de un individuo, para el caso de estudio, es decir, sólo se migra al mejor individuo de cada isla. Esto se debe a que después de realizar muchas y diferentes pruebas se llegó a la conclusión de que era mejor tener poblaciones pequeñas que evolucionen por un número elevado de generaciones a tener poblaciones muy grandes las cuales tienden a decrementar el desempeño del algoritmo en general. El algoritmo 15 describe el procedimiento realizado para distribuir los individuos a través de la red de nodos.

Una época es un número definido de generaciones al final de las cuales se realiza un proceso de migración. La isla con identificador $0$ sólo envía pero no recibe ningún individuo. La $i$-ésima isla (donde $0 < i < n - 1$) envía un individuo a $n - i - 1$ islas y recibe $i$ individuos, uno de cada una de las islas restantes. El proceso con identificador $n - 1$ recibe $i$ individuos pero no envía ninguno, por tratarse del proceso con la resolución más alta.

**Conversión de individuos de una resolución a otra**

Ahora que sabemos qué individuos se migran y a dónde serán enviados, centramos nuestra atención en el proceso para reinsertarlos en la población actual de la isla a la que arribaron. Para cambiar la resolución de un individuo que ha sido migrado de una isla a otra debemos obtener primero que nada los valores reales de los genes que lo conforman. Es decir, a partir de la cadena en binario, obtendremos el valor real de cada una de los genes. Para hacer esto aplicamos la expresión 5.2 a cada una de las variables codificadas en el cromosoma, donde $base = 2$.

Para obtener el valor codificado con la nueva resolución debemos obtener el valor de $I$ de la ecuación 5.2. De tal forma que el nuevo valor calculado será el número real más próximo al valor original pero codificado con la nueva precisión. La ecuación 5.4 se utiliza para realizar el cálculo antes mencionado.

$$I' = \frac{fenotipo - limite_{inf}}{limite_{sup} - limite_{inf}}(base^{bits_{gen}} - 1) \tag{5.4}$$

Finalmente sólo resta convertir dicho entero en base $10$ a su valor correspondiente en binario y concatenarlo junto con las demás variables del nue-

**Algoritmo 15**: Pseudocódigo del algoritmo de selección de individuos para ser migrados

```
1  Input:
2  N_mig : Número de individuos migrados
3  P_i: Población de la i-ésima isla
4  FP: Frente de Pareto actual
5  begin
6      t_FP ←Tamaño(FP)
7      if t_FP ≥ N_mig then
8          M ←Selecciona(FP, N_mig)
9      else
10         M_FP ←Selecciona(FP, N_mig)
11         t_M_FP ← N_mig-Tamaño(M_FP)
12         M_P_i ←Selecciona(P_i, t_M_FP)
13         M ← M_FP + M_P_i
14     if i <> n − 1 then
15         for j = i + 1 to n − 1 do
16             Enviar(j, M)
17         end
18     if i <> 0 then
19         for j = 0 to i − 1 do
20             Recibir(j, R)
21         end
22 end
```

vo individuo.

Hasta el momento nos hemos encargado de migrar los individuos y codificarlos con la precisión de la nueva isla. Sin embargo, aún no sabemos cómo serán reinsertados en la nueva isla. Para hacer esto es necesario jerarquizar los nuevos individuos dentro del nuevo nodo. Una vez realizada esta acción se eliminan de la población aquellos individuos que se encuentren en el último frente y éstos son sustituidos por los nuevos elementos que acaban de arribar.

Las precisiones con las que inician cada una de las islas en el algoritmo están en función del número de variables y el número de bits por variable. Por ejemplo, la isla con identificador $0$ utiliza $10$ bits para codificar cada una de sus variables. Si contamos con $16006$ por individuo, eso nos da como resultado una resolución de $160060$ bits.

**Manejo de restricciones**

Originalmente, el MRMOGA manejaba las restricciones contando únicamente el número de ellas que habían sido violadas por cada individuo. Sin embargo, para nuestro caso de estudio dicho mecanismo resultaba insuficiente, por lo que se modificó para que determinara un valor único que representaba cuánto se habían violado dichas restricciones y no sólo se contara el número de ellas que habían sido quebrantadas.

Las restricciones planteadas para el caso de estudio son dos básicamente, la primera está íntimamente relacionada con el semiejemayor de la órbita final y la segunda con la excentricidad de dicha órbita elíptica.

Consideremos que deseamos llegar de una órbita elíptica a otra del mismo tipo pero cuyas dimensiones difieren a la inicial. Debemos acercarnos dando cierto margen de tolerancia a dicha órbita. El algoritmo propagador de órbitas, recibe los parámetros iniciales y la descripción de la ruta a seguir y nos genera los parámetros descriptores de la órbita alcanzada, en función de los cuales podemos determinar si se cumplen las restricciones planteadas inicialmente. De igual forma se obtienen los valores de las funciones objetivo.

Originalmente se plantea un esquema que nos ayuda a maximizar la diferencia entre el parámetro objetivo y el alcanzado por medio de la ecuación 5.5, la cual daba resultados aceptables pero no los mejores, ya que era muy complicado obtener soluciones factibles, por lo que se modificó por otro esquema planteado por el Dr. Oliver Schütze:

$$Restriccion_i = \frac{(Parametro_{alcanzado} - Parametroobjetivo)^2}{Parametro_{tolerancia}} \tag{5.5}$$

El mecanismo utilizado en la práctica no sólo enfatiza la diferencia entre el valor alcanzado y el valor objetivo de los parámetros, sino que además los intenta fijar en el mismo orden de magnitud, de tal forma que las restricciones pudieran sumarse y obtener un único valor que nos indicara qué tanto se han violado las restricciones.

Primero calculamos la diferencia entre los parámetros obtenidos y los alcanzados

$$t_{i+} = \sqrt{Parametro_{alcanzado} - Parametro_{objetivo} + Parametro_{tolerancia}} \tag{5.6}$$
$$t_{i-} = \sqrt{Parametro_{alcanzado} - Parametro_{objetivo} - Parametro_{tolerancia}} \tag{5.7}$$

$$\mathcal{H}_{i+} = Parametro_{alcanzado} - Parametro_{objetivo} + Parametro_{tolerancia} - t_{i+}^2 \tag{5.8}$$
$$\mathcal{H}_{i-} = Parametro_{alcanzado} - Parametro_{objetivo} - Parametro_{tolerancia} + t_{i-}^2 \tag{5.9}$$

Sólo resta obtener la penalización por romper dichas restricciones, la cual se expresa en la ecuación 5.10.

$$\mathcal{P}enalizacion = \sum_{i=1}^{N_{rest}} w_i(\mathcal{H}_{i+} + \mathcal{H}_{i-}) \tag{5.10}$$

El caso de estudio planteaba dos restricciones, por lo que debemos calcular cuatro valores de $t_{ix}$. Aquí presentamos el modelo matemático que considerá todas las restricciones posibles.

$$t_{a+} = \sqrt{a_{alcanzado} - a_{objetivo} + a_{tolerancia}}$$
$$t_{a-} = \sqrt{a_{alcanzado} - a_{objetivo} - a_{tolerancia}}$$
$$t_{e+} = \sqrt{e_{alcanzado} - e_{objetivo} + e_{tolerancia}}$$
$$t_{e-} = \sqrt{e_{alcanzado} - e_{objetivo} - e_{tolerancia}}$$

$$\mathcal{H}_{a+} = a_{alcanzado} - a_{objetivo} + a_{tolerancia} - t_{a+}^2$$
$$\mathcal{H}_{a-} = a_{alcanzado} - a_{objetivo} - a_{tolerancia} + t_{a-}^2$$

$$\mathcal{H}_{e+} = e_{alcanzado} - e_{objetivo} + e_{tolerancia} - t_{e+}^2$$
$$\mathcal{H}_{e-} = e_{alcanzado} - e_{objetivo} - e_{tolerancia} + t_{e-}^2$$

$$\mathcal{P}enalizacion = w_a(\mathcal{H}_{a+} + \mathcal{H}_{a-}) + w_e(\mathcal{H}_{e+} + \mathcal{H}_{e-}) \tag{5.11}$$

### 5.2.3. Búsqueda local

El proceso de optimización de rutas de vehículos espaciales con el MR-MOGA parece ser insuficiente para obtener resultados satisfactorios, por lo que se agregó un mecanismo de búsqueda local, el cual se aplica cada cierto número de generaciones. El procedimiento utilizado como optimizador local se presenta en el algoritmo 16, el cual utiliza estrategias de interpolación.

**Algoritmo 16**: Pseudocódigo del optimizador local

**Input:**
$x_k$ : Fenotipo del individuo
$\mathcal{N}_{vars}$: Número de variables
$eps$: Tolerancia
**Output:**
$x_{best}$ : Mejor individuo encontrado
**begin**
  $\mathcal{N}_{MaxIter} \leftarrow 10$
  $x_{best} \leftarrow x_k$
  Evaluar($x_{best}$)
  **while** $eps \leq fx_{best}$ & $iter \leq \mathcal{N}_{MaxIter}$ **do**
    $x_{aux} \leftarrow x_{best}$
    $N_{ale} \leftarrow$Aleatorio() $\%\mathcal{N}_{vars}$
    **for** $i = 1$ **to** $N_{ale}$ **do**
      $var_{rand} \leftarrow$Aleatorio()
      ModificarVariable($var_{rand}$, $x_{aux}$)
    **end**
    Evaluar($x_{aux}$)
    **if** $fx_{aux} \preceq fx_{best}$ **then**
      $x_{tmp} \leftarrow x_{best}$
      $x_{best} \leftarrow x_{aux}$
      $x_{aux} \leftarrow x_{tmp}$
    $e \leftarrow$Aleatorio(1,05, 2,0)
    **for** $j = 1$ **to** $N_{vars}$ **do**
      $\delta x \leftarrow x_{best} - x_{aux}$
      $x_{aux2} \leftarrow x_{aux} + (e * \delta x)$
    **end**
    Evaluar($x_{aux2}$)
    **if** $fx_{aux2} \preceq fx_{best}$ **then**
      $x_best \leftarrow x_{aux2}$
    **else**
      $a \leftarrow fx_{aux2} - fx_{aux} - \frac{e*fx_{best}-fx_{aux}}{e^2-e}$
      $b \leftarrow fx_{best} - fx_{aux} - a$
      $t_{best} \leftarrow \frac{-b}{2a}$
      $x_{aux} \leftarrow x_{aux} + t_{best}\delta x$
      Evaluar($x_{aux}$); **if** $fx_{aux} \preceq fx_{best}$ **then**
        $x_{best} \leftarrow x_{aux}$
  **end**
**end**

# Capítulo 6

# Resultados

En este capítulo se presentan los resultados obtenidos con el MRMOGA, los cuales corresponden al caso descrito en la sección 4.4.2. Se realizaron diferentes pruebas con diferente número de procesos y procesadores dentro de un cluster de 32 procesadores.

El MRMOGA ha demostrado ser un algoritmo competitivo en problemas con y sin restricciones y con un número de variables considerable [24]. Sin embargo, nunca había sido puesto a prueba con un problema que además de presentar la dificultad de tener una muy alta dimensionalidad, además que la cantidad de variables de decisión es variable. Las pruebas a las que se ha sometido el MRMOGA frente a la versión paralela del NSGA-II (conocida como PNSGA-II), demostraron la amplia ventaja que tiene el NSGA-II sobre el MRMOGA cuando se utilizan pocos procesadores. Esta condición se revierte conforme el número de procesadores aumenta, ya que el desempeño del MRMOGA es considerablemente mejor con respecto al PNSGA-II si incrementamos el número de procesos.

Otro problema, es que no se cuenta con una solución de referencia para comparar los resultados obtenidos. Aquí presentamos los resultados obtenidos entre el NSGA-II paralelo con un tamaño de 4 y 8 procesadores y las correspondientes obtenidas con el MRMOGA, donde se aprecia claramente cómo el MRMOGA se muestra superior a su contrincante, pero sólo después de haber sido modificado en el manejo de las restricciones planteado en la sección 5.2.2 y de agregarse el mecanismo de búsqueda local descrito en la sección 5.2.3.

También se muestran las órbitas generadas por cada una de las pruebas realizadas en tres y dos dimensiones para tener una idea de lo que se encontró físicamente como una solución factible.

## 6.1. Ajustes preliminares

En un inicio, no se contaba con información respecto del número de variables que se deberían considerar, por lo que se empezaron a realizar pruebas totalmente arbitrarias, sin ningún tipo de conocimiento del dominio, buscando acotar el espacio de búsqueda. Dicho espacio al ser totalmente desconocido, tuvo que ser explorado de manera sistemática a fin de poder detectar regiones prometedoras de búsqueda. Los resultados de las primeras pruebas fueron desalentadores, ya que no se encontraban soluciones que por lo menos pudieran satisfacer las restricciones planteadas desde un inicio. Se optó por realizar una búsqueda más especializada, es decir, se definieron diferentes intervalos para el número de arcos.

Esta prueba consistiría en realizar diferentes corridas que buscaran en diversos intervalos de arcos para determinar aquellos que resultaran ser los mejores, y de esta forma enfocarnos en esa región específica del espacio de búsqueda.

De dichas pruebas se obtuvo que los mejores resultados, es decir, los que al menos resultaban ser factibles, estaban compuestos por menos de 8000 arcos, por lo que se limitó el número de arcos, de 32000 a tan sólo 8000 como máximo. Con esto, el problema se acotó de manera significativa debido al número de variables que ya no serían tomadas en cuenta.

Al tomar tan sólo 8000 arcos, esto nos dio un total de a lo más 16000 variables, ya que se debe multiplicar por 2 el número de arcos, debido al ángulo azimutal y de elevación.

El tiempo de ejecución para realizar una evaluación de las funciones objetivo está ligado al número de arcos inmiscuidos en dicha solución propuesta. Al realizar una evaluación de la función objetivo el tiempo varía. Sin embargo, es más o menos constante para el mismo número de arcos. Se realizó una prueba de 30 corridas para evaluar la función objetivo, la cual arrojó los resultados mostrados en la tabla 6.1.

Tabla 6.1: Tiempos requeridos para evaluar las funciones objetivo

| Número de arcos | Tiempo promedio(s) | Desviación |
|---|---|---|
| 1000 | 1.2 | 0.00127 |
| 2000 | 1.9834 | 0.00556 |
| 3000 | 2.563 | 0.00127 |
| 4000 | 3,004 | 0.003323 |
| 5000 | 3.46 | 0.02934 |
| 6000 | 4.0021 | 0.003356 |
| 7000 | 4.763 | 0.001953 |
| 8000 | 5.3 | 0.000845 |

Obviamente, si el número de arcos aumentaba, el tiempo de ejecución sería mayor, hasta llegar a un valor tope de 14 segundos aproximadamente cuando se realizan evaluaciones de órbitas con 16000 arcos.

## 6.2. Pruebas realizadas con el PNSGA-II

El PNSGA-II se probó con 4 y 8 procesadores. El número de evaluaciones fue exactamente el mismo es decir 1600000. Sin embargo, cambiaron ciertos parámetros, como son el número de procesadores, lo que nos condujo directamente a disminuir el número de generaciones para hacer exactamente el mismo número de evaluaciones.

Tabla 6.2: Parámetros del PNSGA-II

| Número Procesadores | Individuos | Generaciones |
|---|---|---|
| 4 | 200 | 2000 |
| 8 | 200 | 1000 |

Con los resultados obtenidos de las pruebas con el PNSGA-II, se obtuvieron no sólo órbitas factibles, sino que resultaban ser muy buenas. El único inconveniente era el tiempo de procesamiento el cual era realmente elevado.

Las pruebas con un mayor número de procesadores fueron mucho más rápidas, ya que el trabajo fue repartido entre los procesadores del cluster. Sin embargo, solamente se pudieron realizar pruebas con 4 y 8 procesadores debido a las limitantes impuestas por el hardware disponible. La tabla

Tabla 6.3: Tiempos PNSGA-II con 1600000 evaluaciones

| Número Procesadores | Tiempo(s) | Tiempo(días) |
|---|---|---|
| 4 | 5745723 | 66,501 |
| 8 | 3576877 | 41,399 |

6.3 muestra los tiempos de ejecución para 4 y 8 procesadores con el algoritmo PNSGA-II.

### 6.2.1. Descripción del clúster utilizado para realizar las pruebas

Para realizar estas pruebas se contó con hardware un tanto limitado, sobre todo si consideramos el número de evaluaciones de la función objetivo y el tiempo que tardaba cada una de ellas. Se utilzaron 4 máquinas con procesadores Opteron de 64 bits duales y 4GB de Memoria en RAM. El número máximo de procesos que se podían ejecutar realmente en paralelo era de 8. Se utilizó la versión 7.1.4 de LAM-MPI sobre sistemas Red Hat Linux de 64 bits.

### 6.2.2. Pruebas realizadas con 4 y 8 procesadores

A continuación se presentan gráficamente los resultados obtenidos con el PNSGA-II.

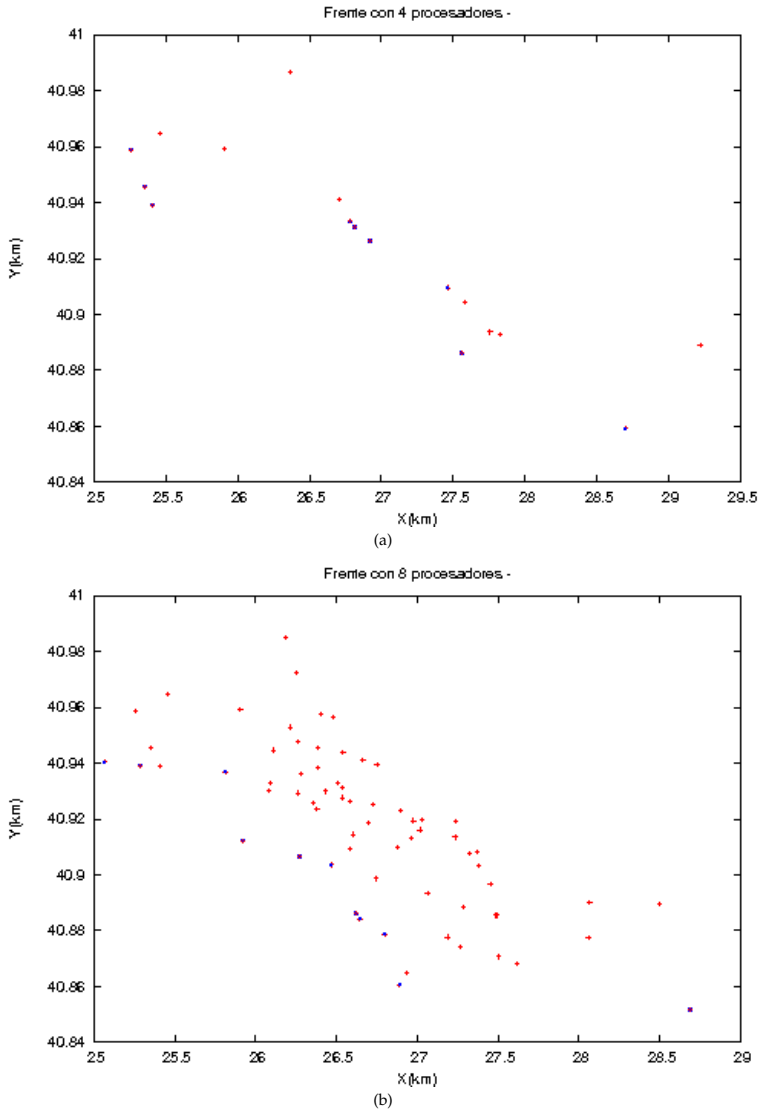



Figura 6.1: Frentes encontrados con el PNSGA-II con (a) 4 procesadores y (b) 8 procesadores

En la figura 6.1 podemos ver cómo las pruebas realizadas con 8 procesadores fueron más eficaces para encontrar mejores soluciones, las cuales fueran factibles. Sin embargo, cabe recalcar el tiempo invertido para poder generar estos resultados. La tabla 6.2 muestra los tiempos empleados para realizar las pruebas con 4 y 8 procesadores en promedio, ya que se realizaron 4 corridas con 4 procesadores. Mientas que con 8 procesadores solamente se realizaron 2 corridas por el tiempo que éstas duraban. Para generar los frentes mostrados en la figura 6.1, se mezclaron las soluciones de las corridas generadas con el número correspondiente de procesadores.

La figura 6.1.a, representa todas las soluciones factibles encontradas por el PNSGA-II, con las pruebas combinadas para 4 procesadores. En esta figura destacan las soluciones que se encuentran en el frente de Pareto a fin de que se pueda tener una idea de las soluciones que fueron encontradas durante el proceso. La figura 6.1.b, muestra exactamente el mismo resultado, pero en este caso se pueden apreciar las soluciones factibles encontradas con 8 procesadores.

La figura 6.2, muestra claramente cómo el frente generado con las pruebas realizadas con 8 procesadores, superaron a las pruebas con 4 procesadores.

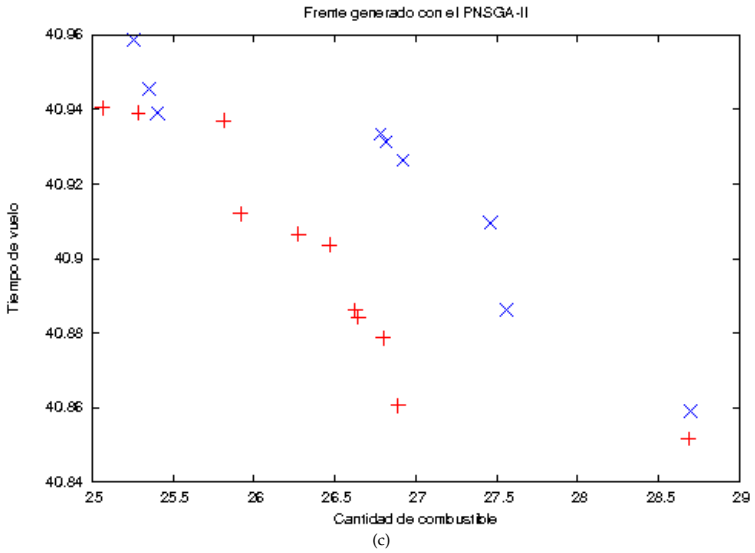



Figura 6.2: Frentes encontrados con el PNSGA-II con 4 y 8 procesadores combinados

Las figuras 6.3 y 6.4 representan las órbitas elípticas encontradas con

el PNSGA-II, en las corridas realizadas con 4 y 8 procesadores. En ellas se puede apreciar de manera gráfica la ruta que debe seguir el vehículo espacial para llegar desde una órbita elíptica geoestacionaria a otra del mismo tipo, concéntrica pero de un radio mayor.

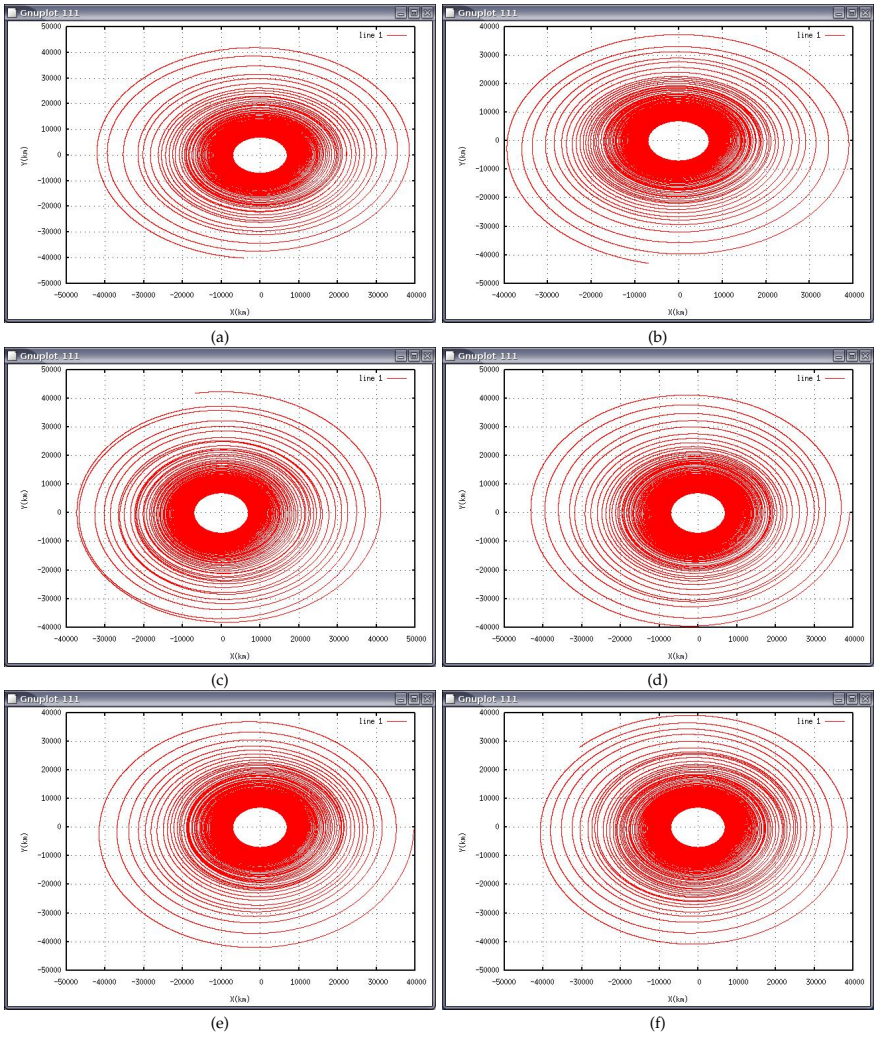



Figura 6.3: Órbitas generadas con 8 procesadores.

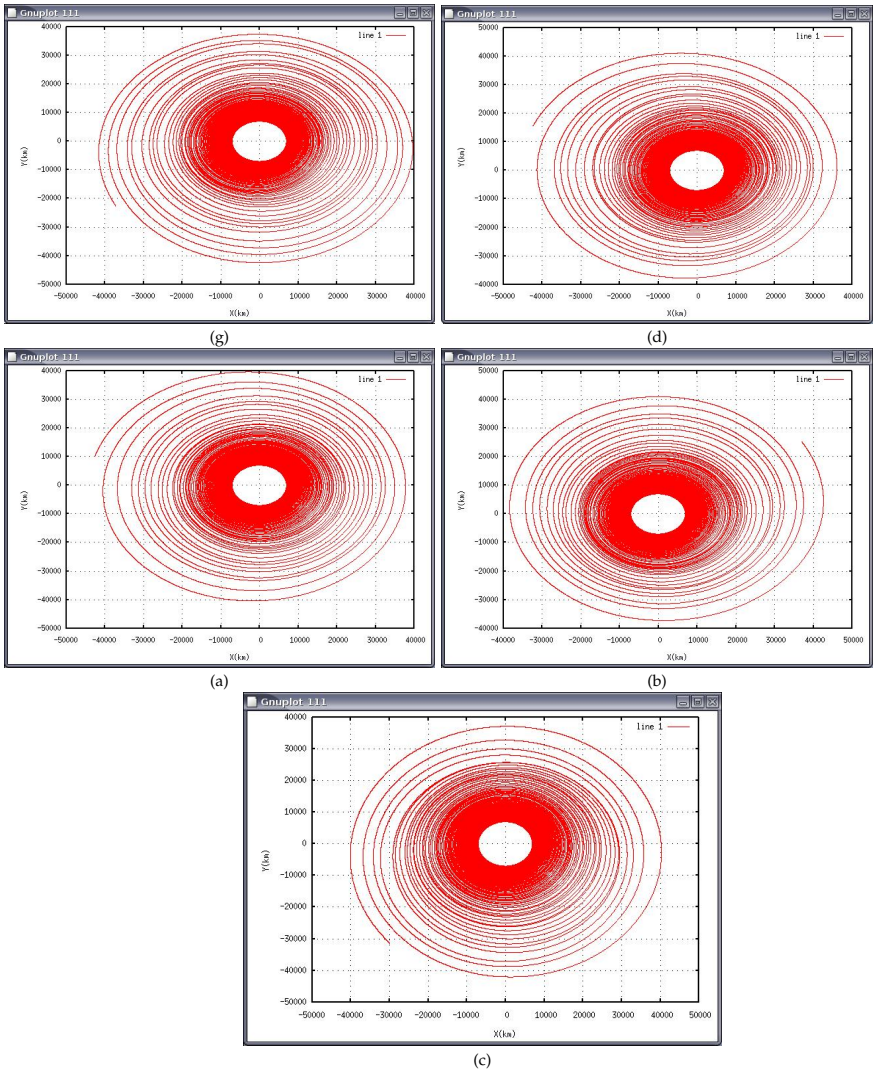



Figura 6.4: Órbitas generadas con 8 procesadores.

Todas las órbitas presentadas representan soluciones factibles. Para lograr encontrar resultados factibles se delimitó el espacio de búsqueda, al fijar el número máximo de arcos en 8000 y utilizar una precisión muy burda en todas las islas al inicio de cada corrida. En cada prueba se utilizaron diferentes porcentajes de cruza y mutación. Dichos porcentajes se describen a mayor detalle en las tablas 6.4 y 6.5.

Tabla 6.4: Parámetros de cruza y mutación para 4 procesadores

| Número de la corrida | % de cruza | % de mutación |
|---|---|---|
| 1 | 0,95 | 1/16000 |
| 2 | 0,90 | 0,1 |
| 3 | 0,85 | 0,125 |
| 4 | 0,80 | 0,15 |

Tabla 6.5: Parámetros de cruza y mutación para 8 procesadores

| Número Procesadores | % de cruza | % de mutación |
|---|---|---|
| 1 | 0,95 | 1/16000 |
| 2 | 0,90 | 0,1 |

## 6.3. Pruebas realizadas con el MRMOGA

Como se mencionó en el capítulo 5, el algoritmo del MRMOGA [24] fue modificado de su versión original para adecuarlo a las características propias del problema de optimización de rutas de vehículos espaciales. Se agregó el manejo para soportar un número variable de incógnitas, el mecanismo para manejar la restricciones, un buscador local para mejorar las soluciones y, finalmente, se debieron realizar varios ajustes empíricos de los parámetros del algoritmo, a fin de obtener los mejores resultados posibles.

Se hicieron pruebas con 4, 8, 16 y 32 procesadores. Al ser muy elevado el tiempo de procesamiento requerido, lo cual está intimamente ligado con el número de evaluaciones que se deben realizar de las funciones objetivo, sólo se pudieron realizar un número reducido de corridas. A cambio de eso se realizaron pruebas con un número de evaluaciones que, aunque puede parecer pequeño, es suficientemente grande para un clúster.

### 6.3.1. Descripción del cluster utilizado para realizar las pruebas

El clúster consta de 64 nodos homogéneos de servidores Apple G5 Xserve Duales. Cada uno de dichos nodos cuenta con dos procesadores Power PC G5 de 64 bits que corren a 2.3 Ghz. Cada uno cuenta con 1 GB de RAM y un disco duro de 80 GB. El sistema operativo instalado en dichas máquinas es Mac OS X Server 10.4.6. La versión de MPI con la que se cuenta instalada es Open MPI 1.1.1. El compilador instalado es *gcc* de Apple versión 4.0.

El nodo principal cuenta con 2 procesadores G5 de 64 bits a 2.3 Ghz, 8 GB de RAM y un disco duro de 80 GB.

La red de interconexión es una red Ethernet Gigabit a través de Switches Foundry Networks de 48 puertos.

### 6.3.2. Pruebas con 4 procesadores

Se realizaron 5 pruebas con 4 procesadores. La tabla 6.6 muestra los tiempos requeridos por cada corrida.

Tabla 6.6: Tiempos obtenidos con 4 procesadores

| Número de la corrida | Tiempo(s) | Tiempo(días) |
|---|---|---|
| 1 | 5993342 | 69.367 |
| 2 | 5809014 | 67.234 |
| 3 | 5952097 | 68.889 |
| 4 | 6006944 | 69.525 |
| 5 | 6384528 | 73.895 |
| Promedio | 6029185 | 69,782 |

El frente de la figura 6.5, muestra los individuos factibles encontrados con el algoritmo planteado con 4 procesadores.

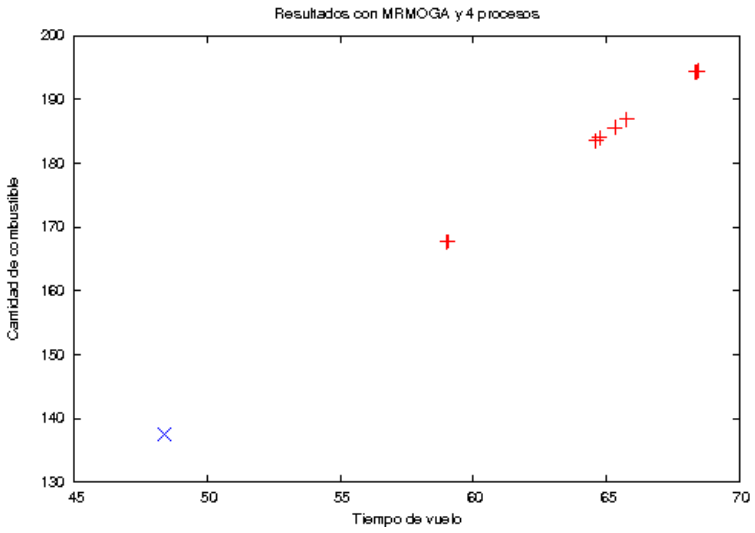



Figura 6.5: Soluciones factibles encontradas con 4 procesadores

A continuación se presentan las órbitas factibles encontradas con las pruebas realizadas con 4 procesadores. Las figuras 6.6, 6.7, 6.8, 6.9 y 6.10 presentan dichas órbitas en dos y tres dimensiones para visualizar de manera más clara dichas trayectorias. Los diferentes colores de las figuras en tres dimensiones, representan la diferencia generada por la inclinación de la órbita destino con respecto a la órbita tomada como origen.

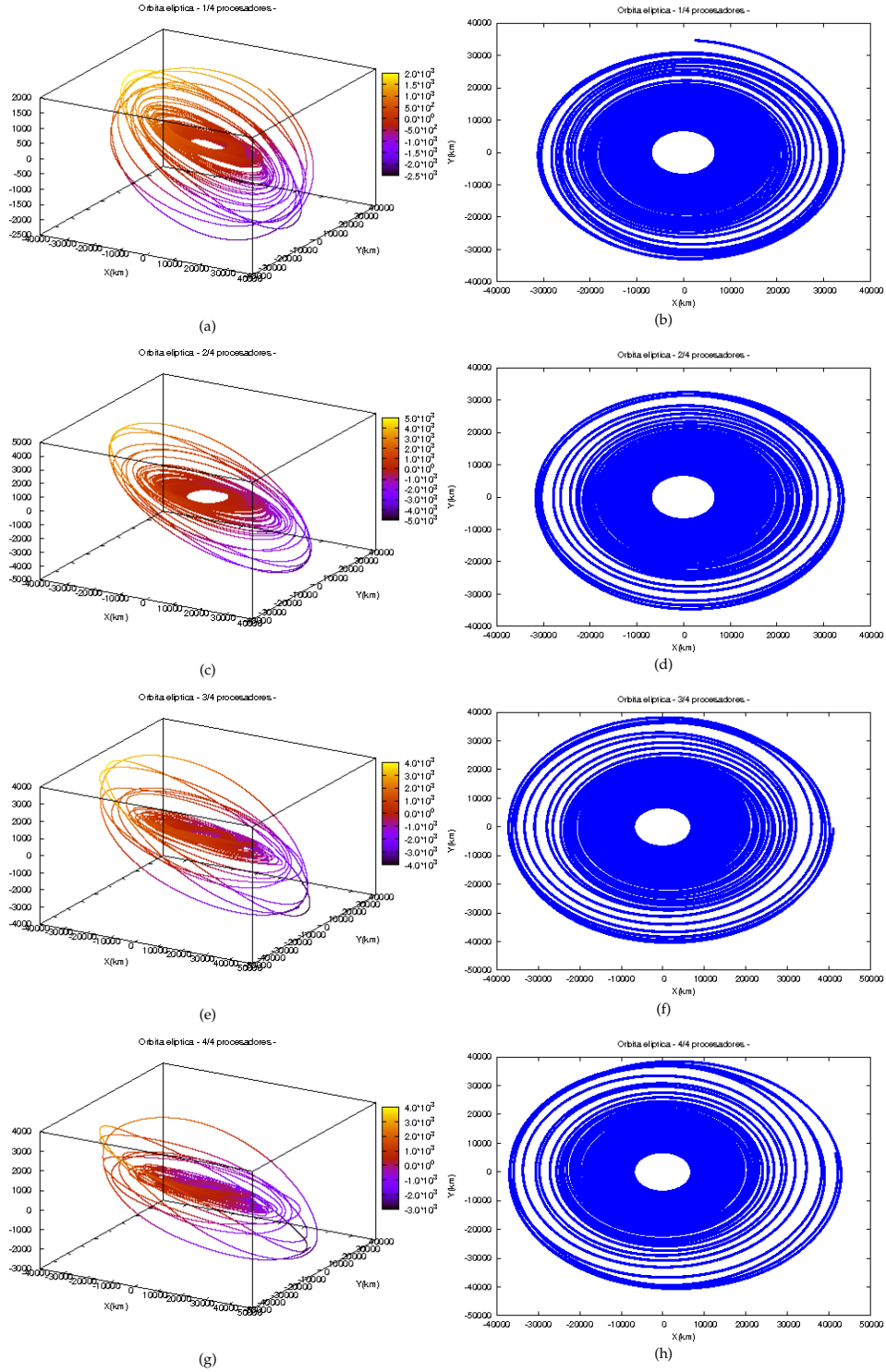



Figura 6.6: Órbitas factibles encontradas con 4 procesadores

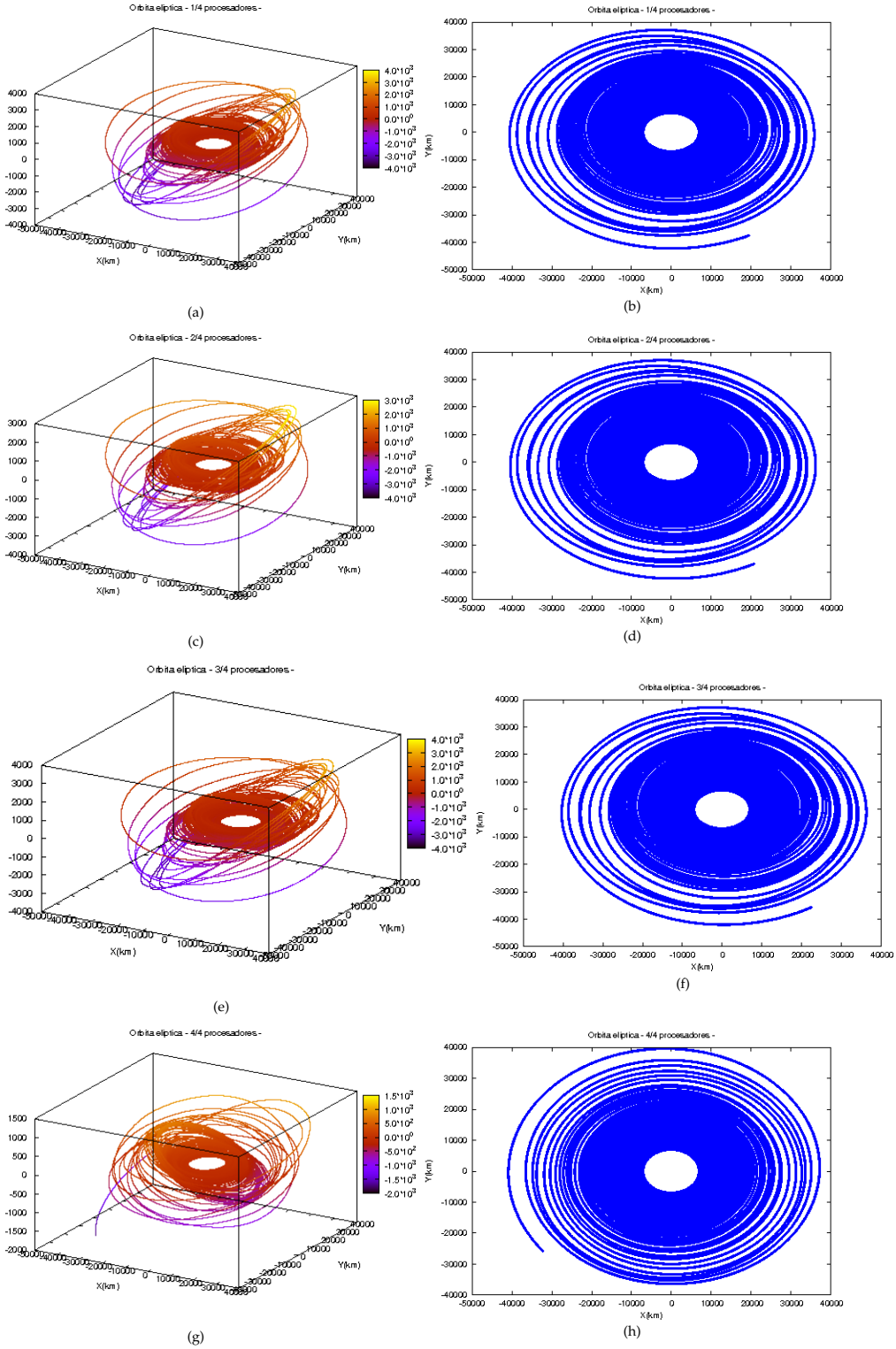



Figura 6.7: Órbitas factibles encontradas con 4 procesadores

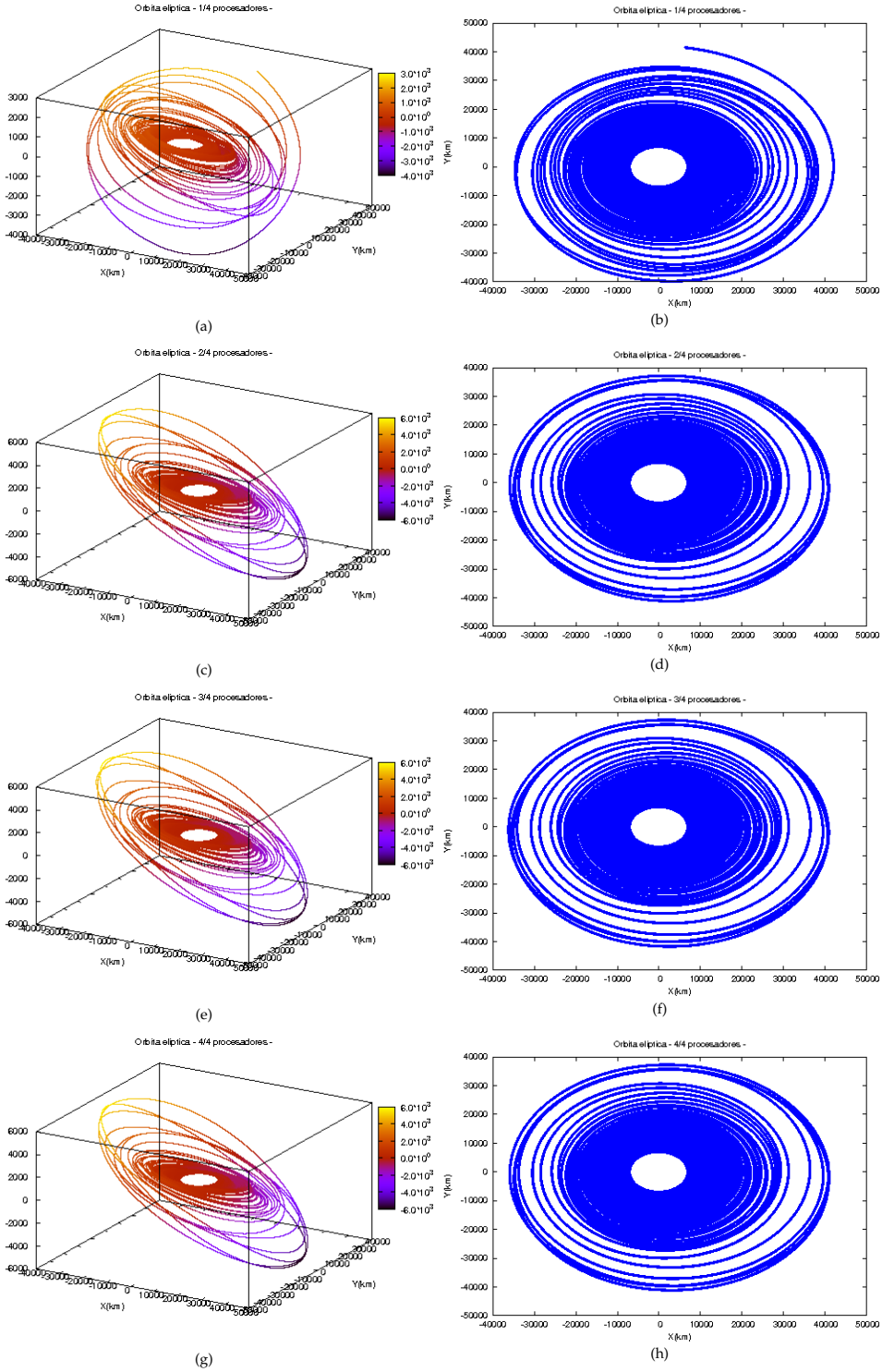



Figura 6.8: Órbitas factibles encontradas con 4 procesadores

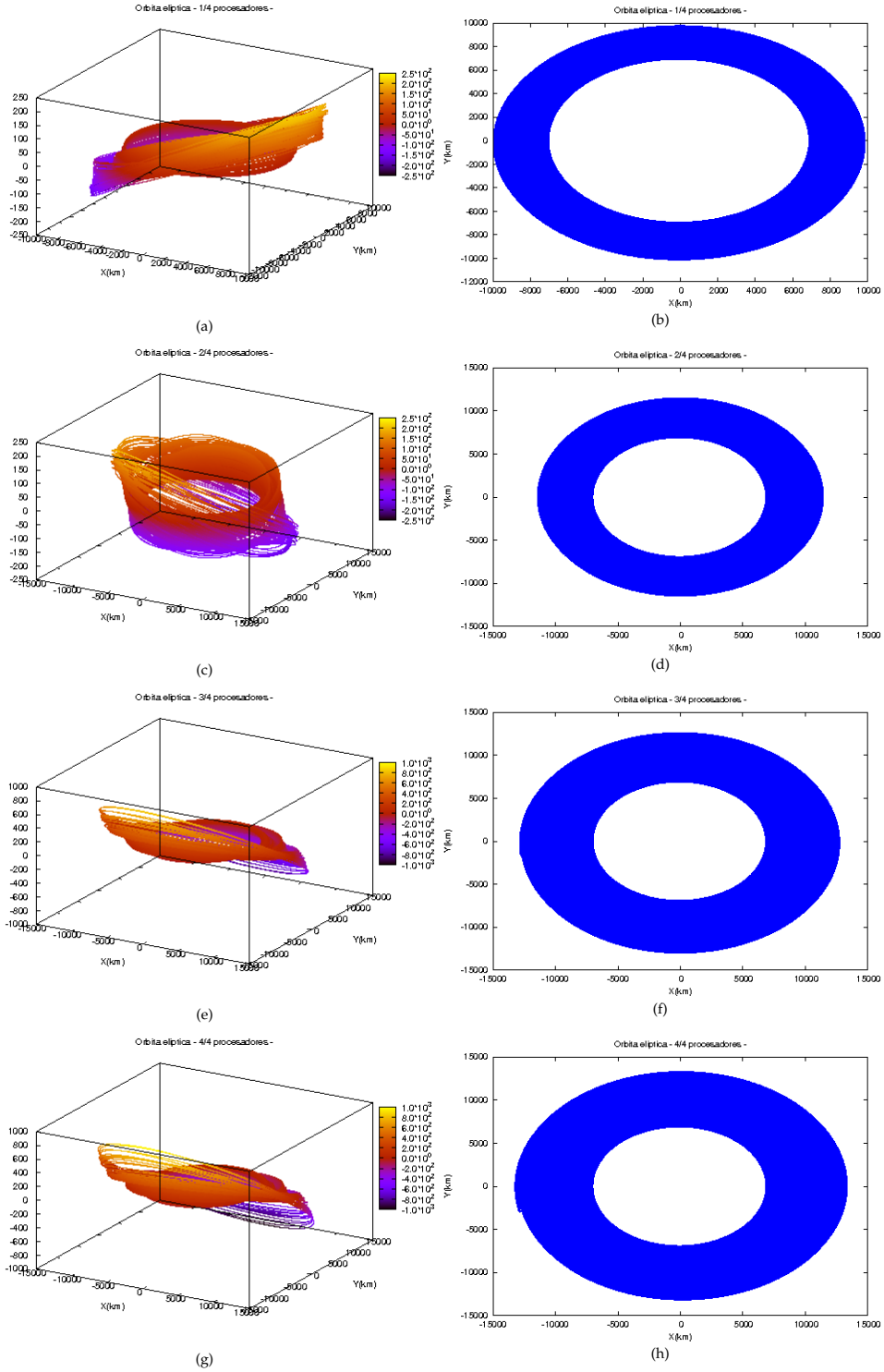



Figura 6.9: Órbitas factibles encontradas con 4 procesadores

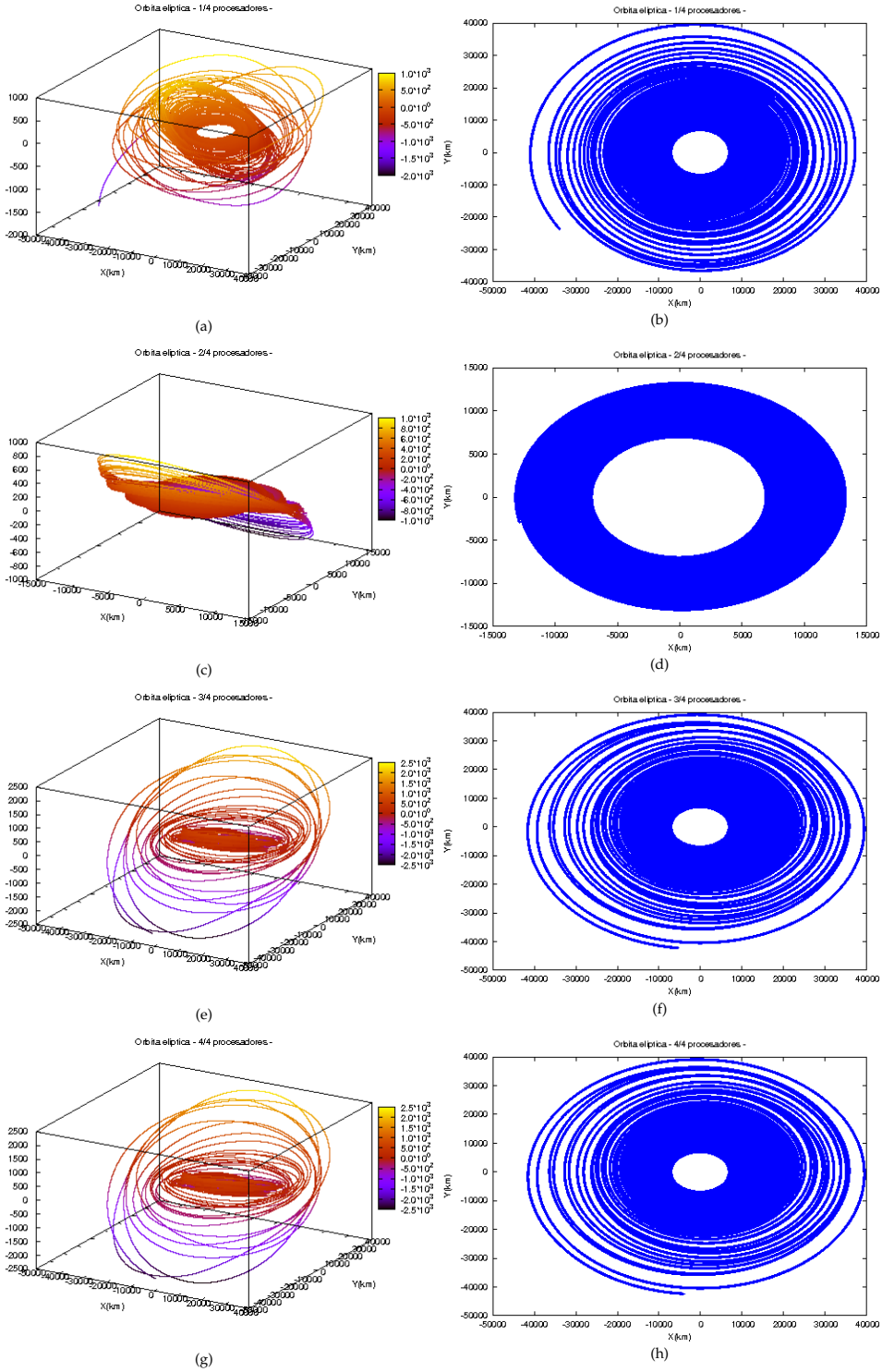



Figura 6.10: Órbitas factibles encontradas con 4 procesadores

### 6.3.3. Pruebas con 8 procesadores

Se realizaron 4 pruebas con 8 procesadores. La tabla 6.7, muestra los tiempos de ejecución de dichas pruebas así como el tiempo promedio invertido para que fueran ejecutadas.

Tabla 6.7: Tiempos obtenidos con 8 procesadores

| Número de la corrida | Tiempo(s) | Tiempo(días) |
|---|---|---|
| 1 | 5732234 | 66.345 |
| 2 | 5534377 | 64.055 |
| 3 | 5952489 | 68.895 |
| 4 | 5734875 | 66.376 |
| Promedio | 5738547 | 66,418 |

El frente de la figura 6.11, muestra los individuos factibles encontrados con el algoritmo planteado con 8 procesadores.

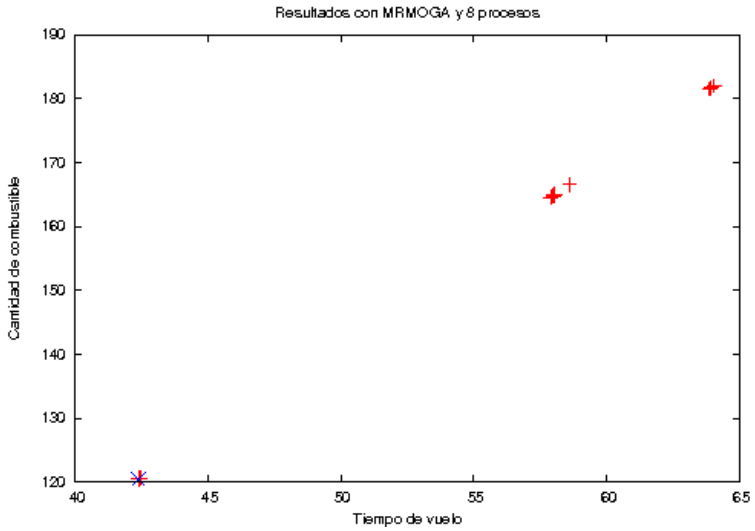



Figura 6.11: Soluciones factibles encontradas con 8 procesadores

A continuación se presentan las órbitas factibles encontradas con las pruebas realizadas con 8 procesadores.

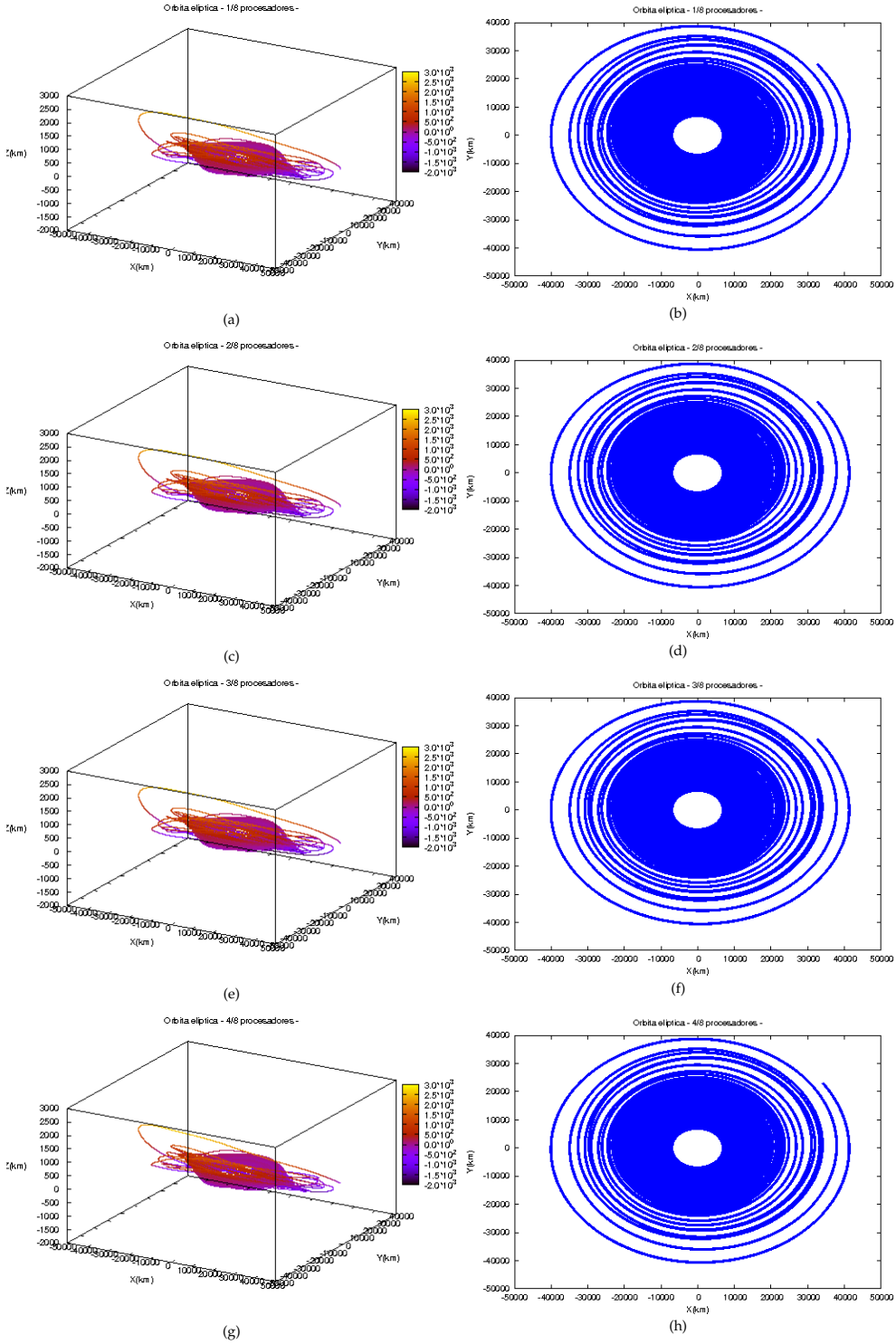



Figura 6.12: Órbitas factibles encontradas con 8 procesadores

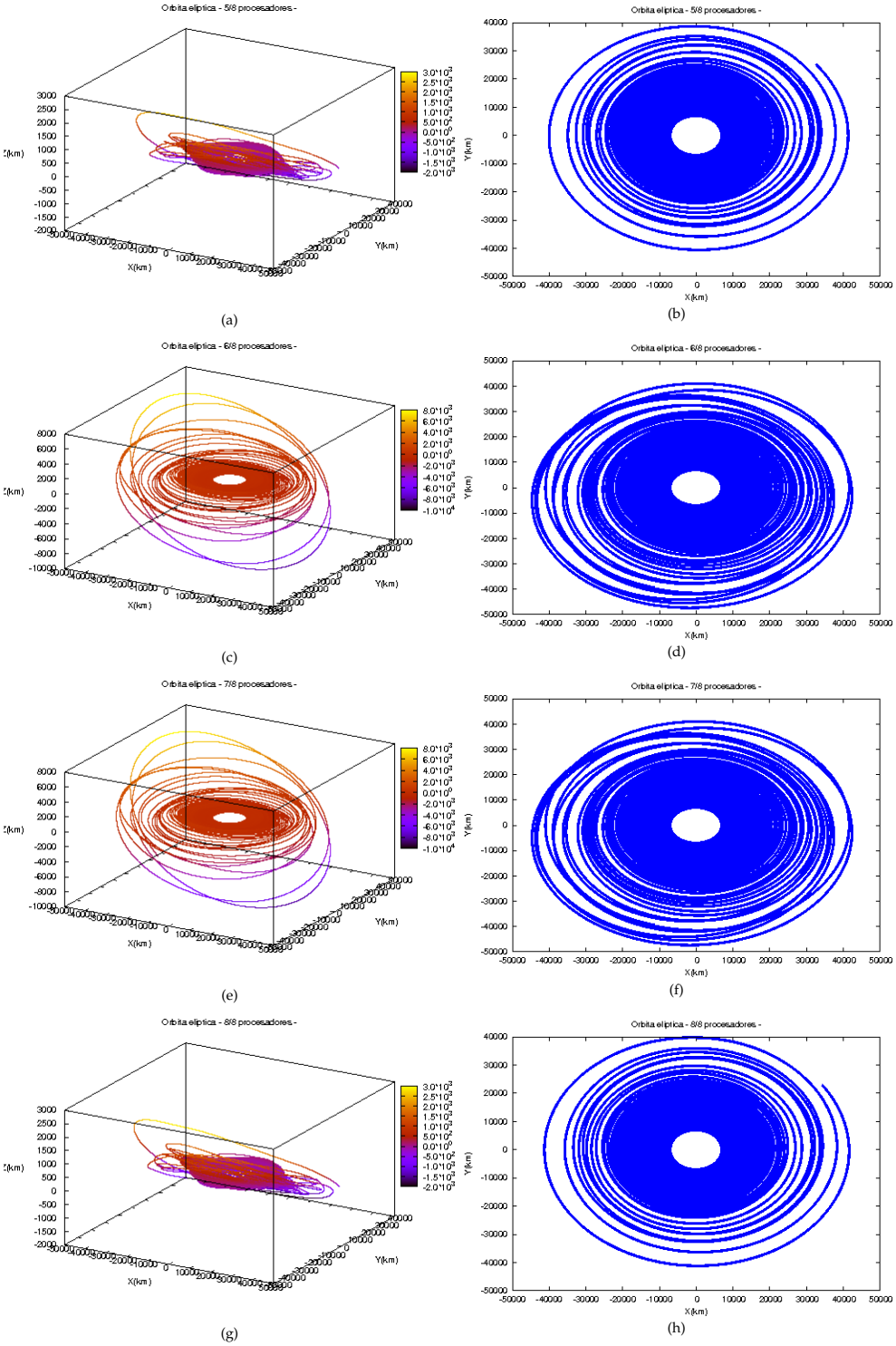



Figura 6.13: Órbitas factibles encontradas con 8 procesadores

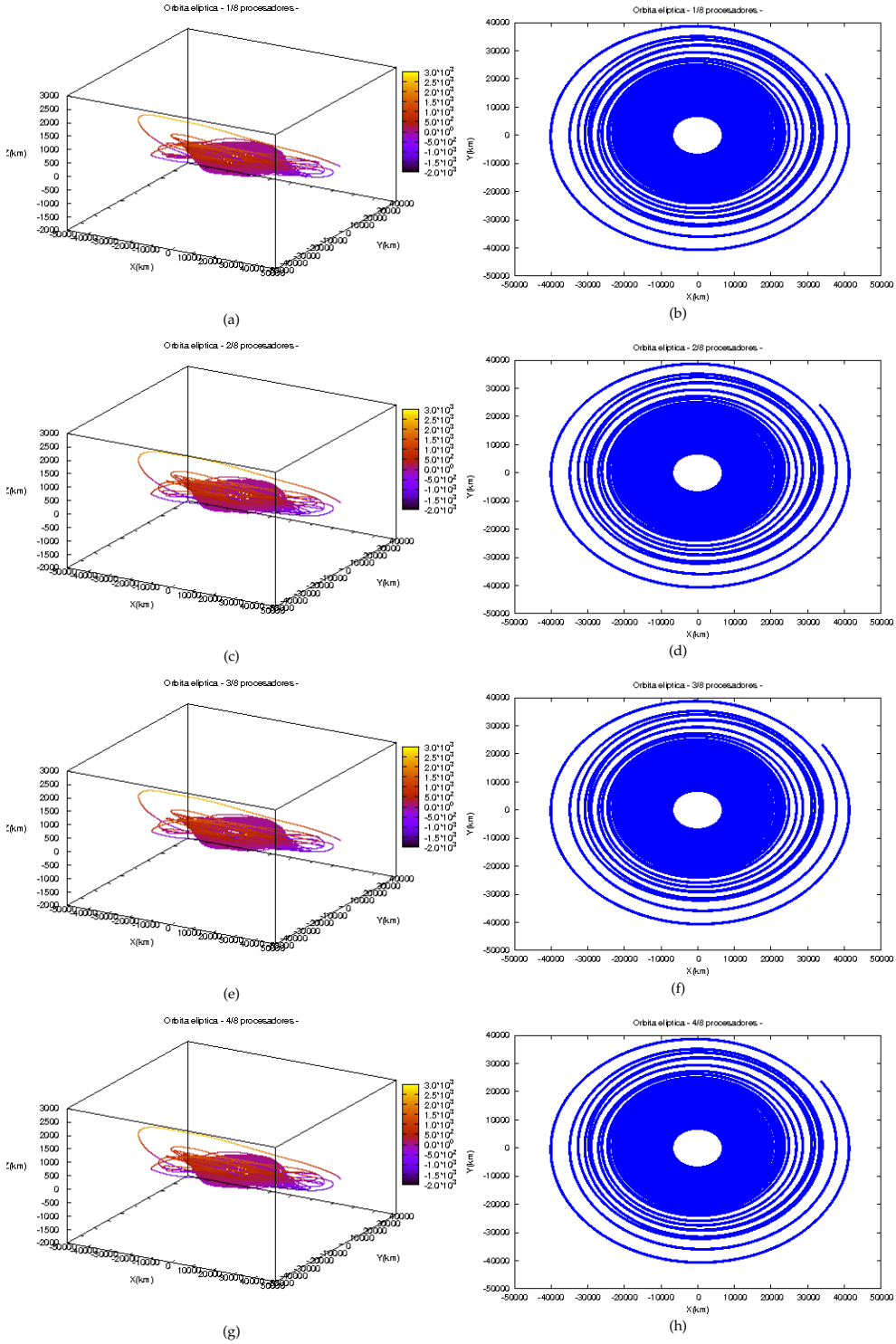



Figura 6.14: Órbitas factibles encontradas con 8 procesadores

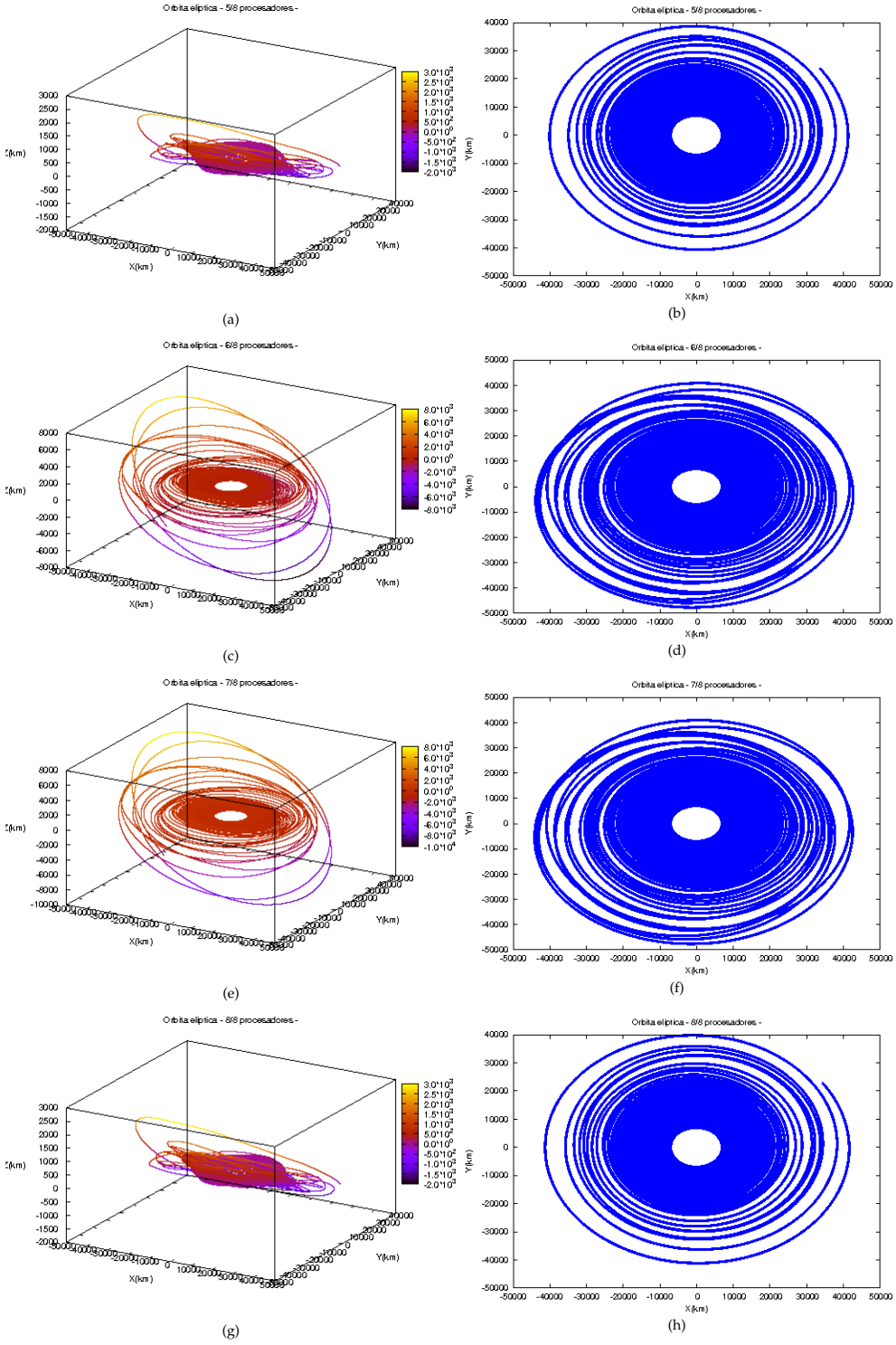



Figura 6.15: Órbitas factibles encontradas con 8 procesadores

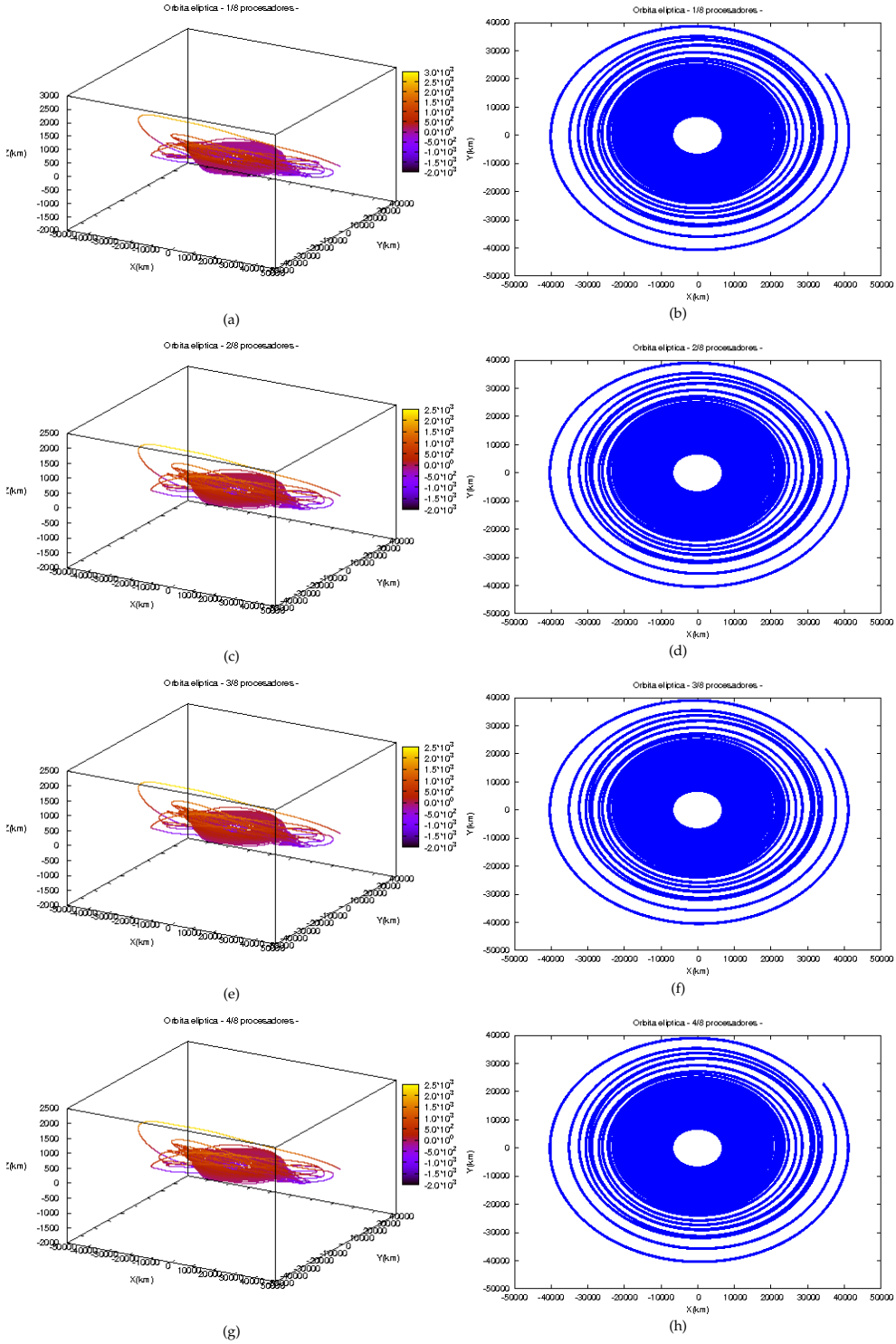



Figura 6.16: Órbitas factibles encontradas con 8 procesadores

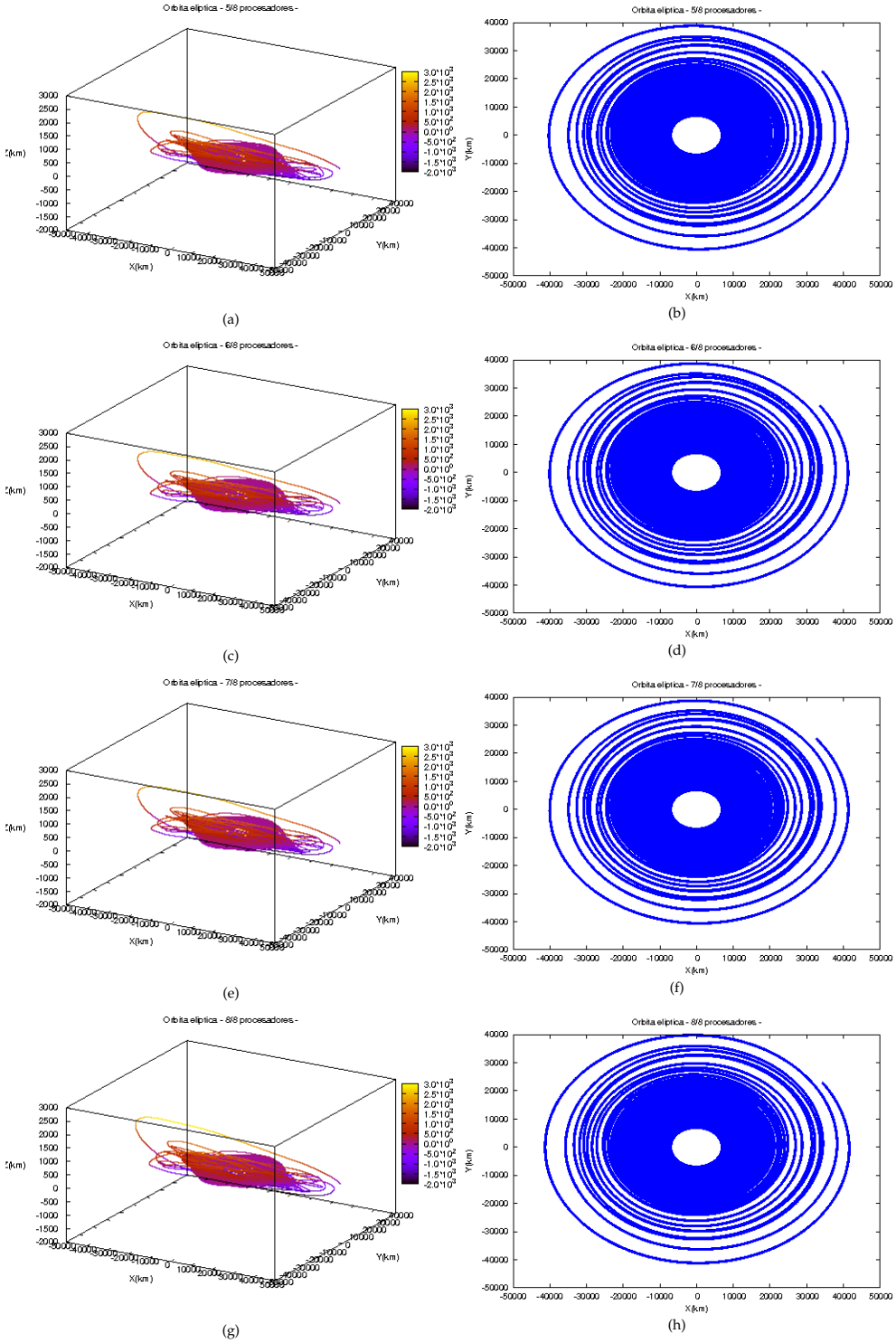



Figura 6.17: Órbitas factibles encontradas con 8 procesadores

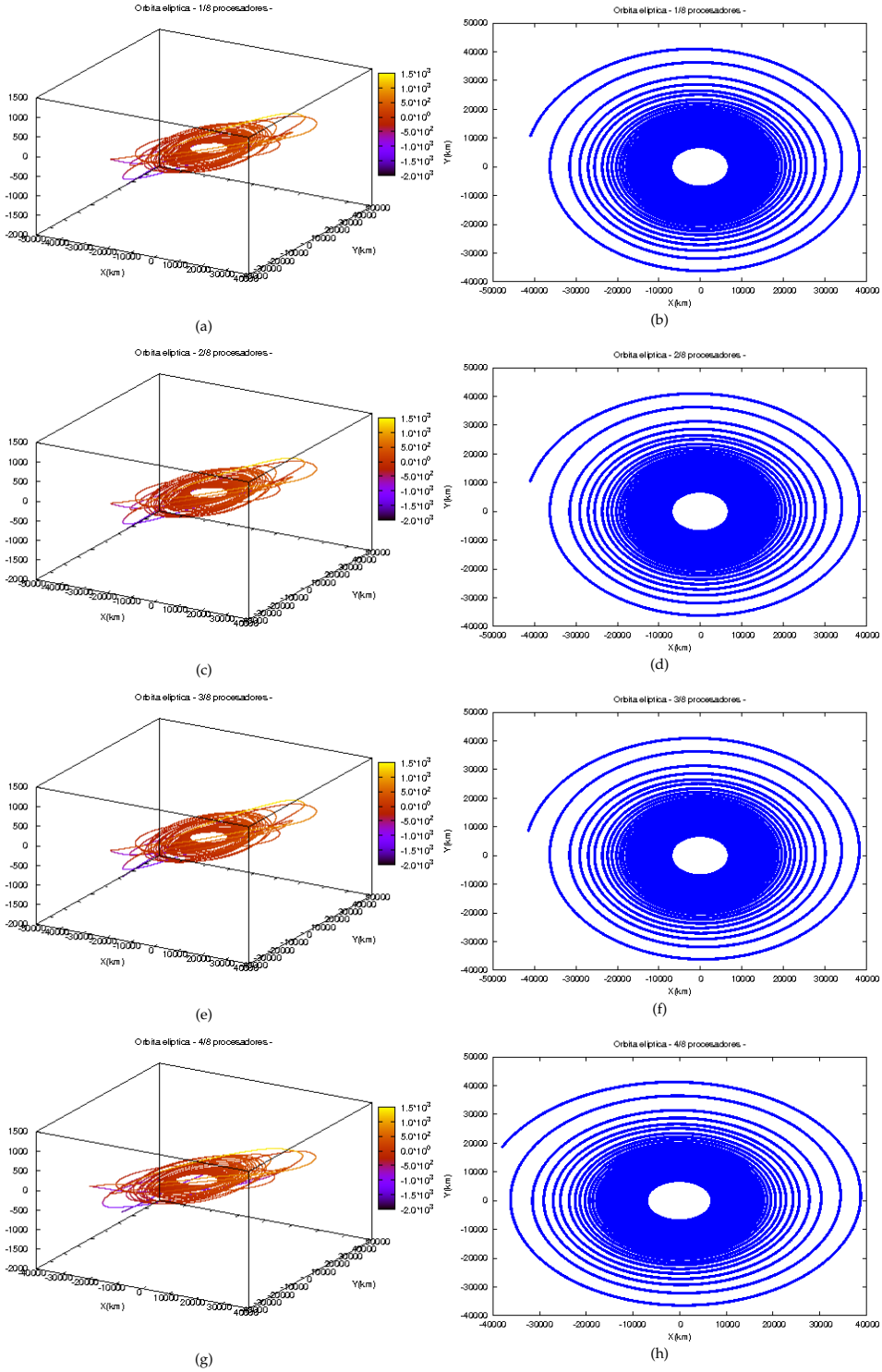



Figura 6.18: Órbitas factibles encontradas con 8 procesadores

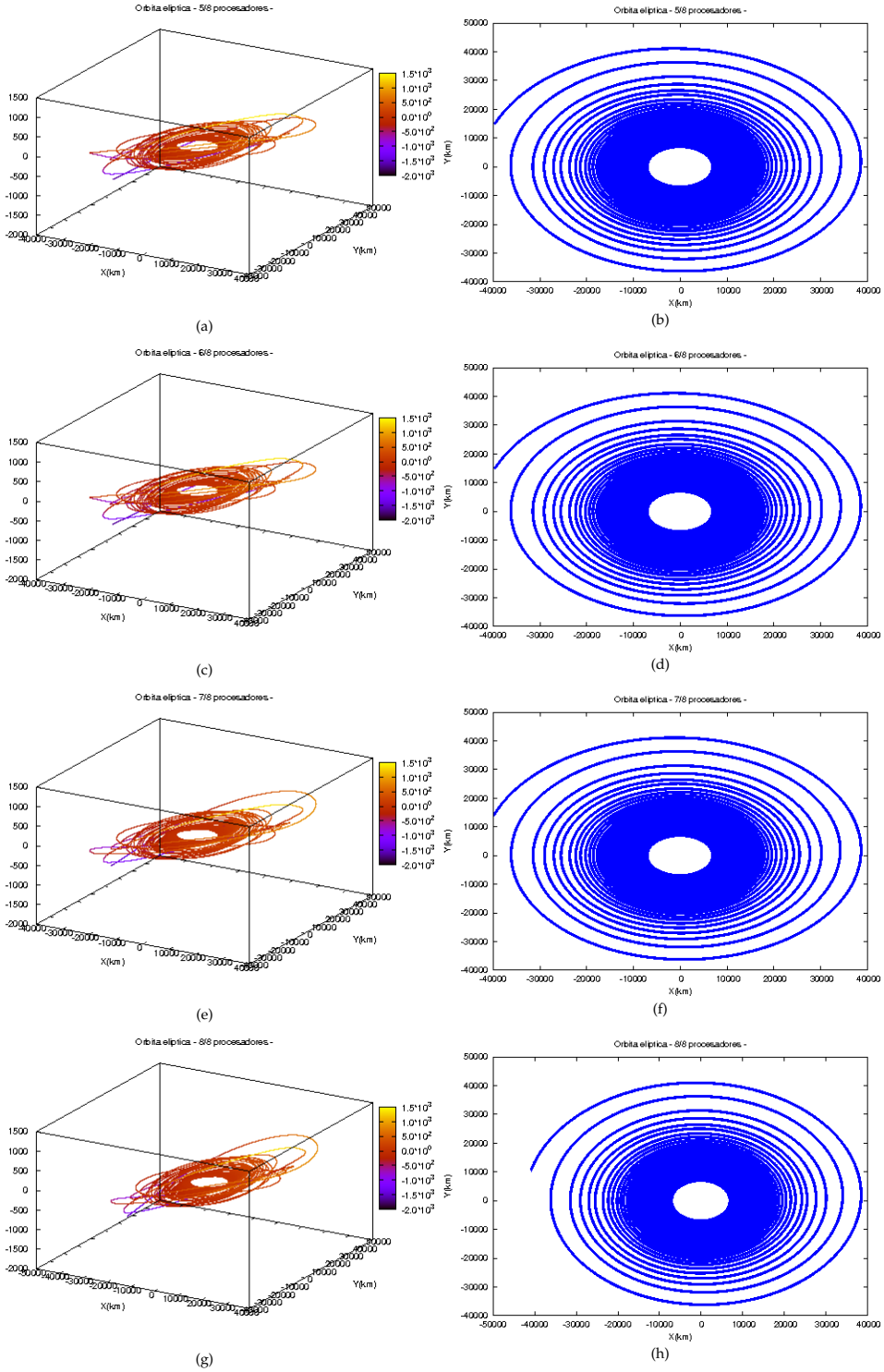



Figura 6.19: Órbitas factibles encontradas con 8 procesadores

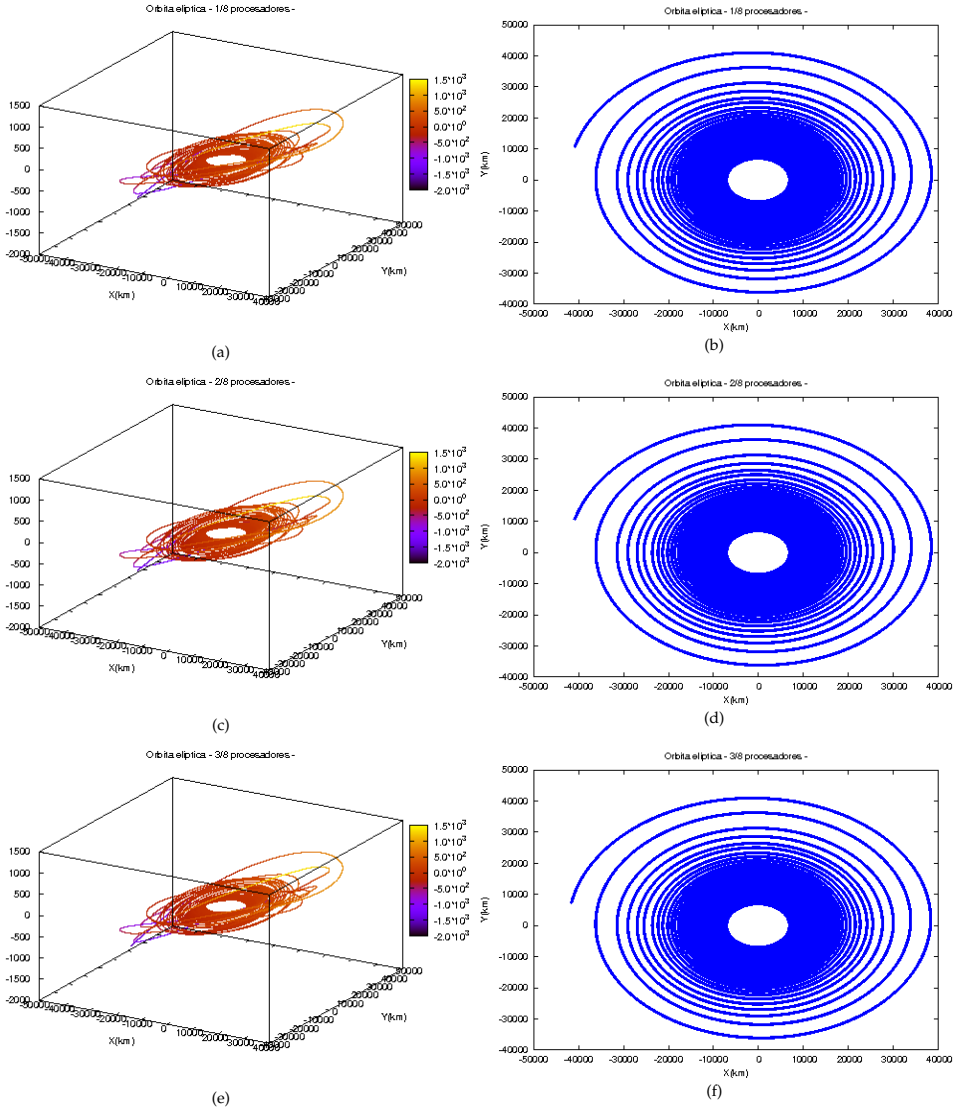



Figura 6.20: Órbitas factibles encontradas con 8 procesadores

### 6.3.4. Pruebas con 16 procesadores

Se probó el algoritmo con 16 procesadores, aunque sólo fue posible realizar una prueba con este número de procesos. No obstante, se pudieron encontrar soluciones factibles que mejoran las encontradas con 4 y 8 procesadores.

El tiempo de ejecución de esta prueba fue de 4,832,343 segundos, lo cual equivale a aproximadamente 55.93 días.

El frente de la figura 6.21 muestra los individuos factibles encontrados con el algoritmo.

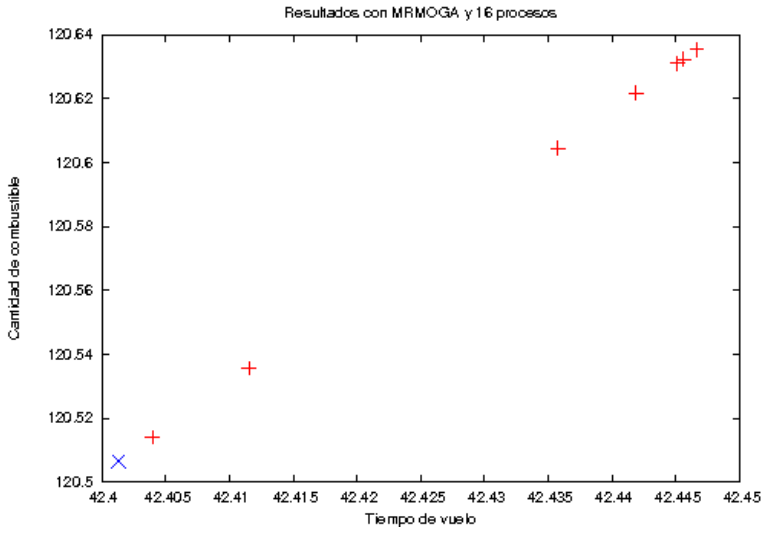



Figura 6.21: Soluciones factibles encontradas con 16 procesadores

Las soluciones factibles encontradas con 16 procesadores fueron graficadas y se presentan en las figuras 6.22, 6.23 y 6.24.

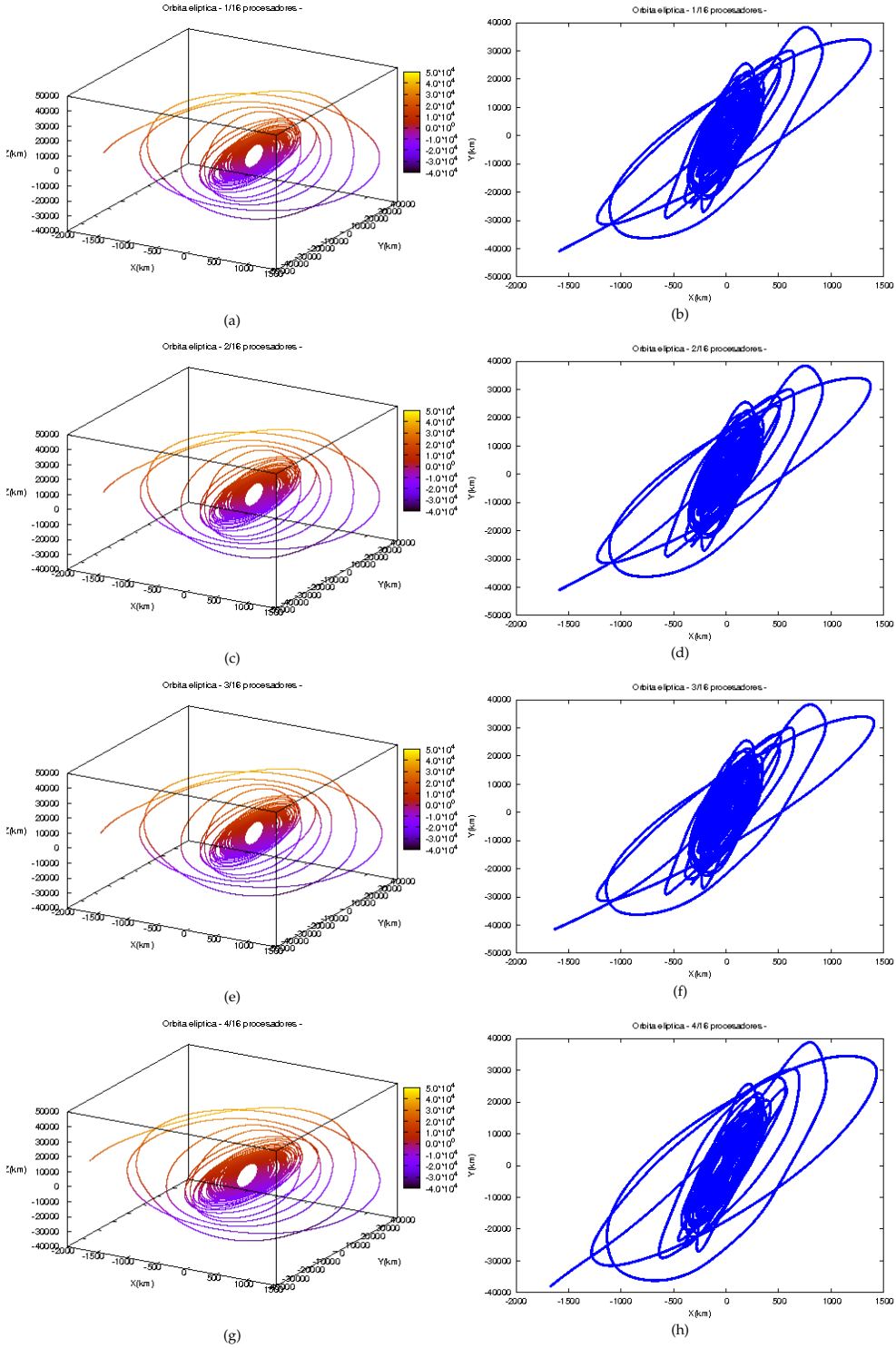



Figura 6.22: Órbitas factibles encontradas con 16 procesadores

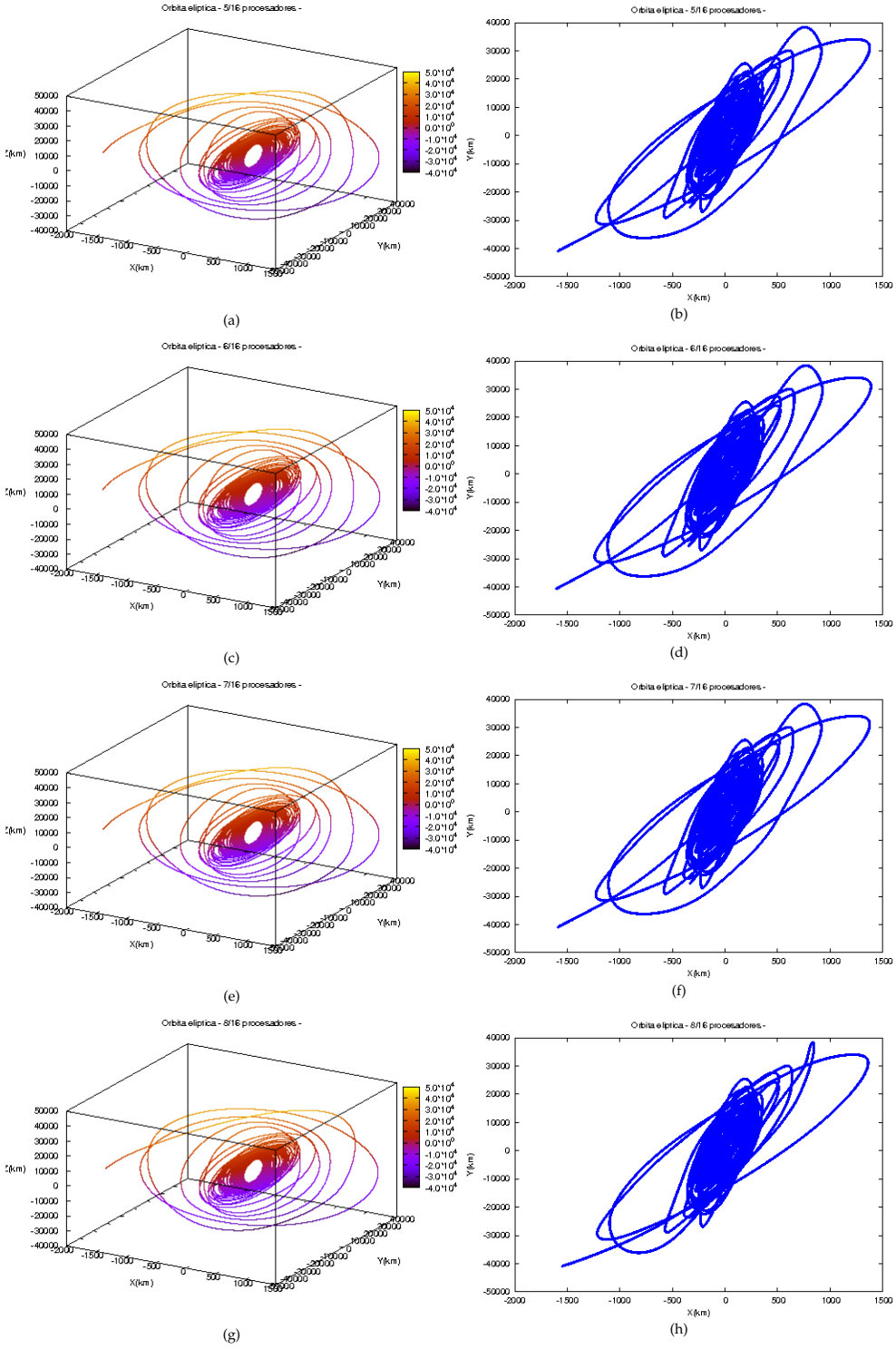
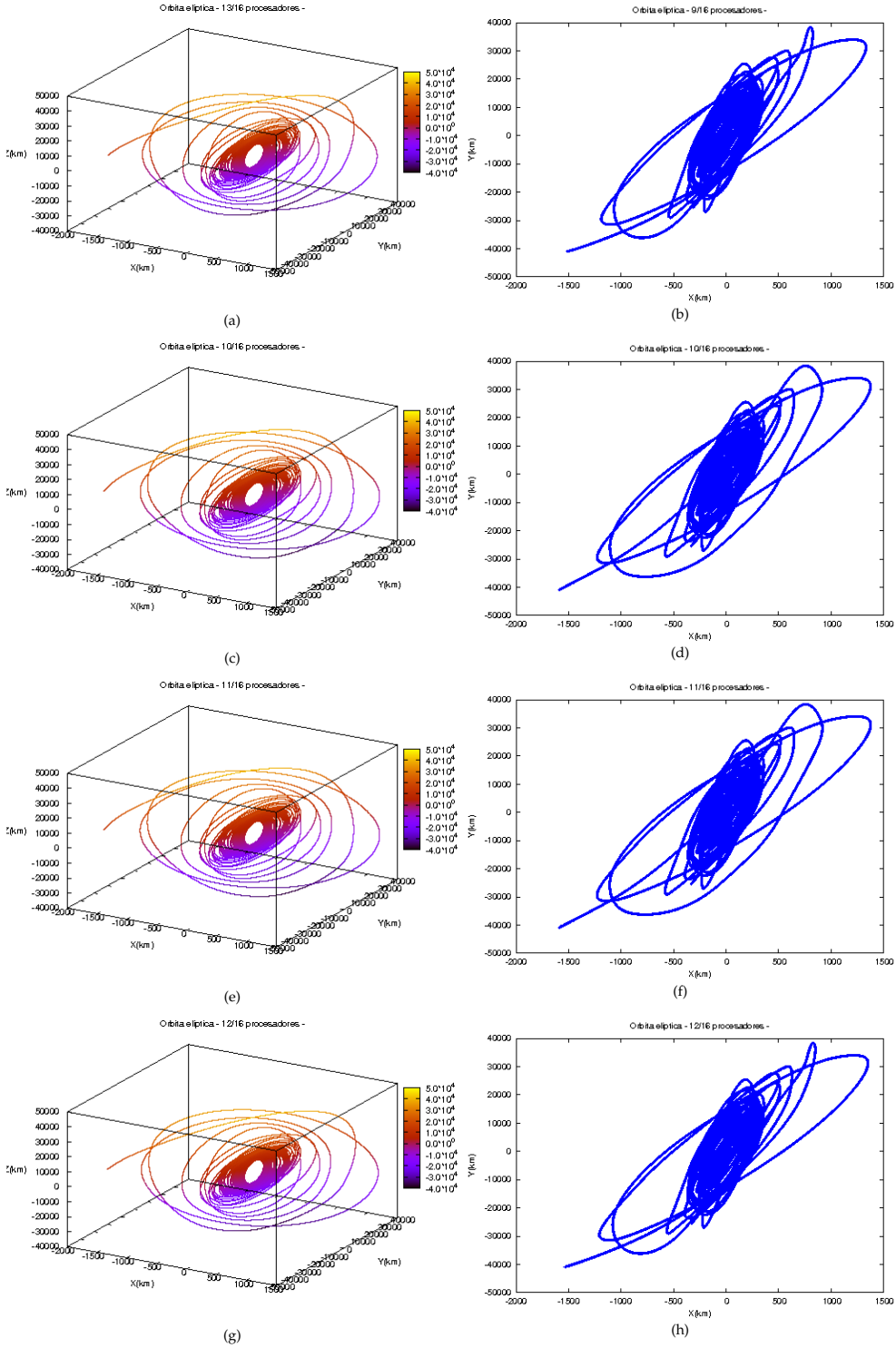



Figura 6.23: Órbitas factibles encontradas con 16 procesadores




(a) (b) (c) (d) (e) (f) (g) (h)

Figura 6.24: Órbitas factibles encontradas con 16 procesadores

### 6.3.5. Pruebas con 32 procesadores

Se probó el algoritmo con 32 procesadores, aunque sólo fue posible realizar una prueba con este número de procesos. No obstante, se pudieron encontrar soluciones factibles que mejoran las encontradas con 4, 8 y 16 procesadores.

El tiempo de ejecución de esta prueba fue de 4,356,785 segundos, lo cual equivale a aproximadamente 50.426 días. Lo cual reduce considerablemente el tiempo invertido cuando se usaban 4 procesadores.

El frente mostrado en la figura 6.25 muestra los individuos factibles encontrados con el algoritmo al crear 32 procesos.

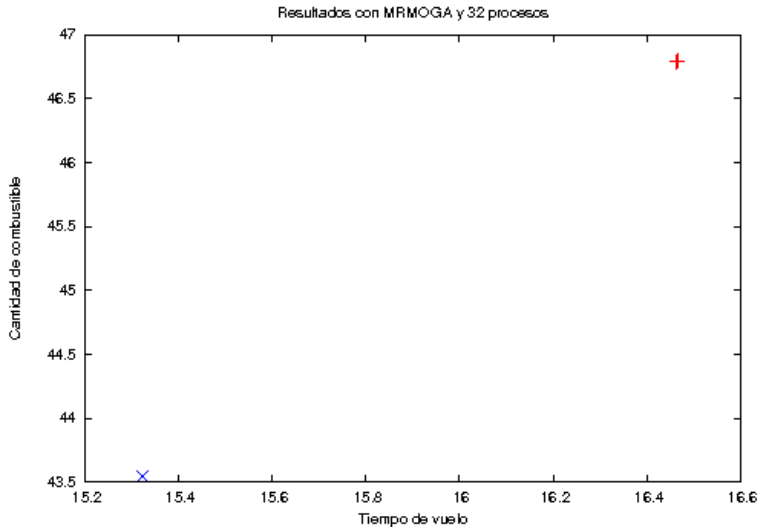



Figura 6.25: Soluciones factibles encontradas con 32 procesadores

Las soluciones factibles encontradas con 32 procesadores fueron graficadas y son las mejores encontradas con el MRMOGA.

La Dra. Seungwon Lee y su equipo de trabajo han realizado investigación en este campo con métodos indirectos, por lo que no se cuenta con soluciones de referencia contra las cuales se podría comparar. Además, en este documento de tesis se presenta una propuesta de solución para el problema de tamaño variable, en el que no se sabe de antemano el número de arcos de las mejores soluciones.

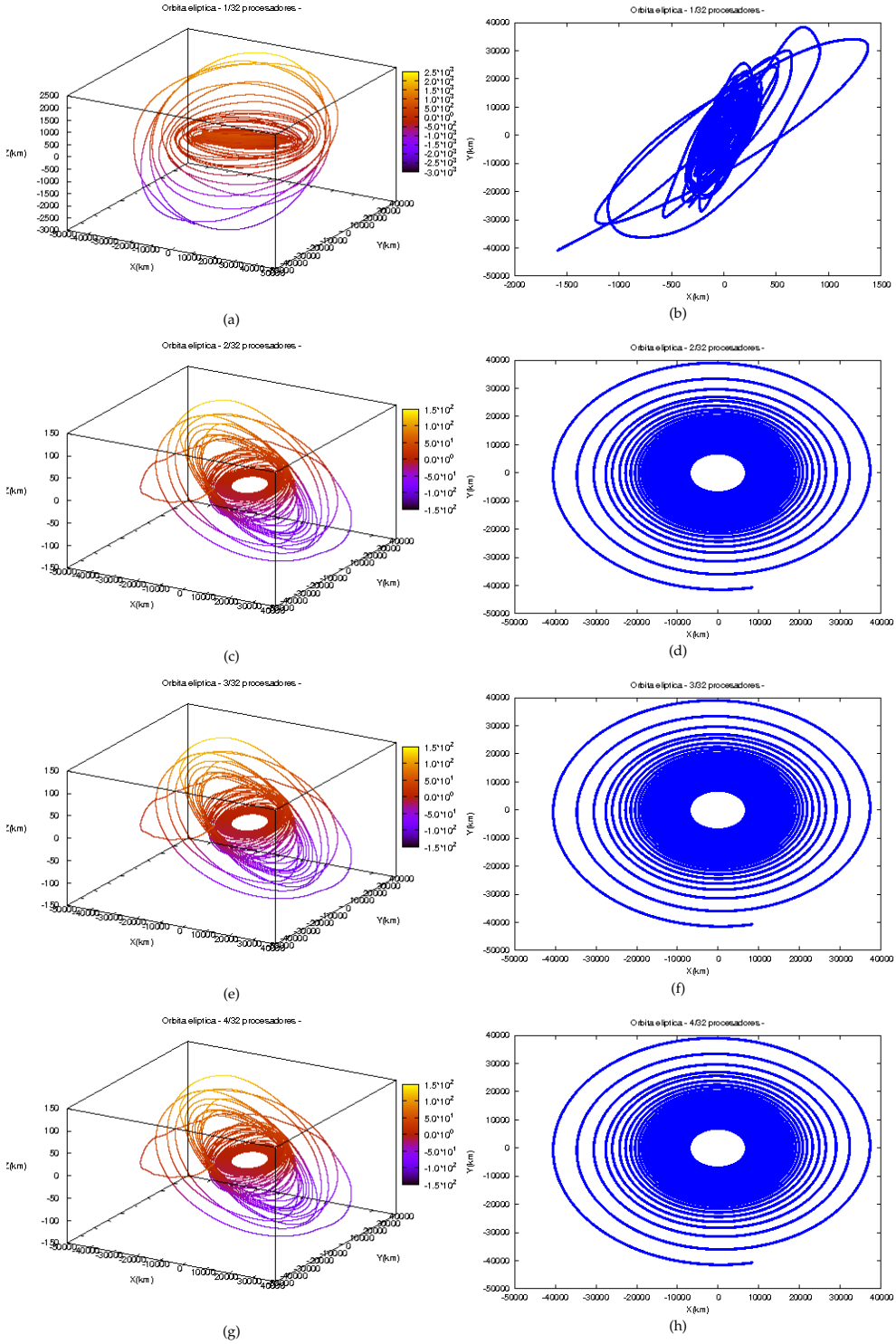



Figura 6.26: Órbitas factibles encontradas con 32 procesadores

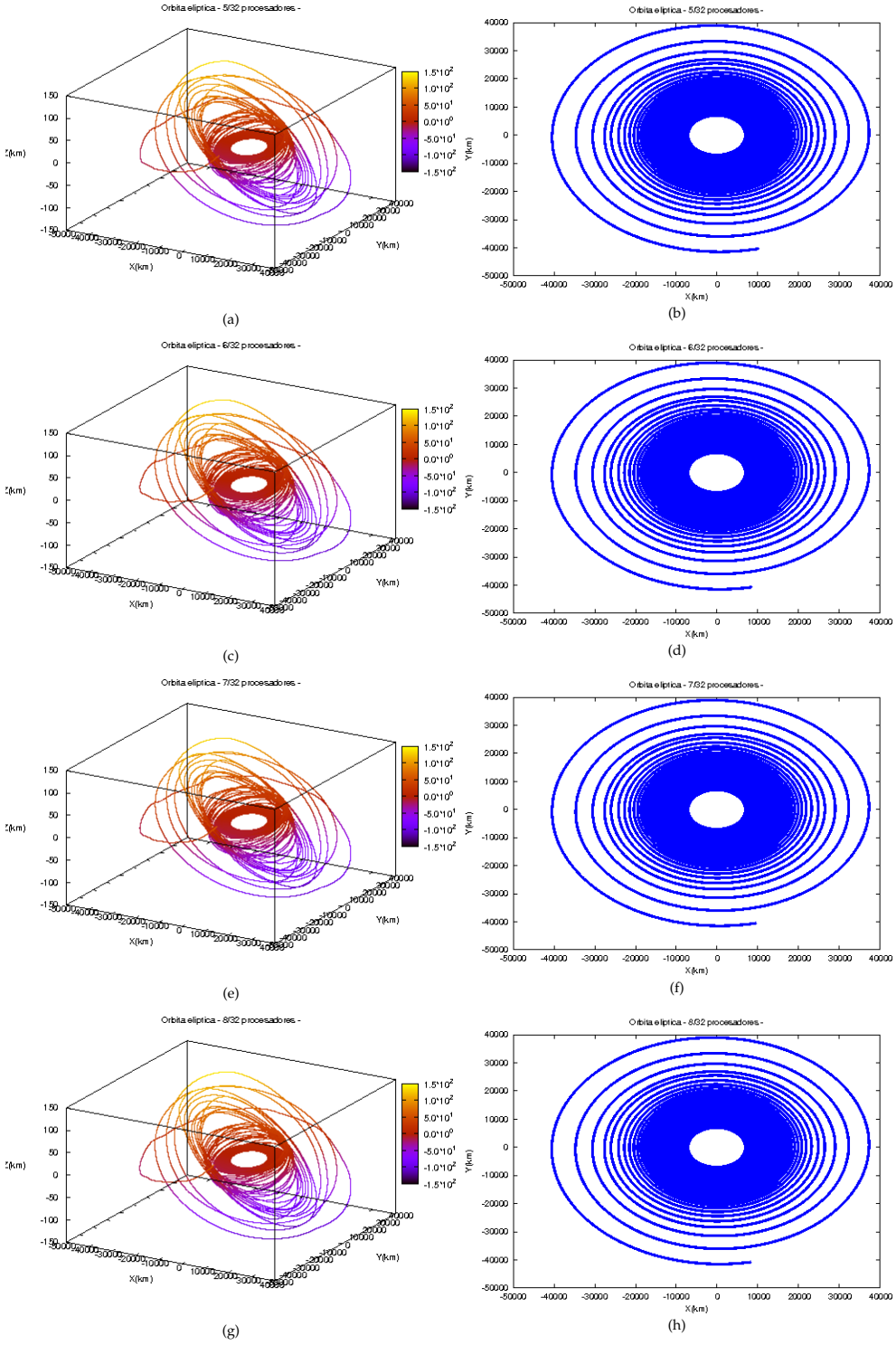



Figura 6.27: Órbitas factibles encontradas con 32 procesadores

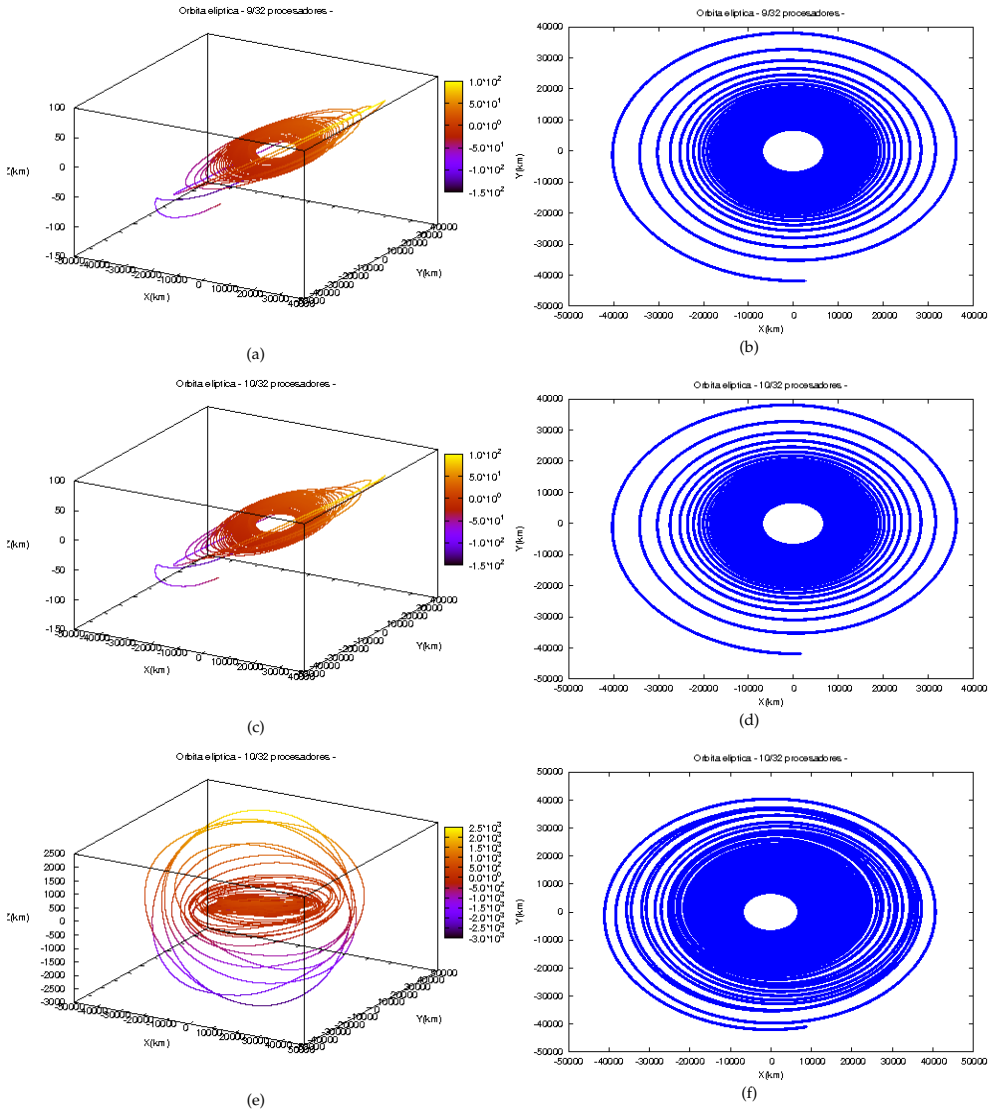



Figura 6.28: Órbitas factibles encontradas con 32 procesadores

# Conclusiones

A lo largo de este documento se detallaron los pormenores para resolver el problema de optimización de rutas de vehículos espaciales de bajo impulso. Se planteó una solución y se implementó con un algoritmo genético multiobjetivo paralelo.

La solución consistió en modificar el Algoritmo Genético Paralelo de Múltiples Resoluciones para que pudiera lidiar con problemas de longitud variable, modificar el mecanismo para manejo de restricciones e integra de un algoritmo de búsqueda local para mejorar el desempeño del algoritmo genético paralelo sin realizar más evaluaciones de la función objetivo, la cual es bastante costosa. Se realizaron diferentes corridas con el mismo número de evaluaciones y diferentes valores en los parámetros del algoritmo, como son el procentaje de cruza y mutación. Se plantea la modificación de dichos parámetros conforme se avanza en el proceso evolutivo.

El problema planteado es muy complicado debido a la alta dimensionalidad del mismo, aunado al hecho de no saber cuál es el número exacto de variables de la solución óptima. Se tuvo que acotar el espacio de búsqueda realizando pruebas de manera sistemática para encontrar el espacio donde se localizaban las mejores soluciones. De las pruebas realizadas se determinó que las mejores soluciones se encuentraban cerca de los 8000 arcos.

Otro problema planteado es el tiempo de evaluación de las funciones objetivo el cual es alto y está relacionado precisamente con el número de variables de la solución propuesta. Mientras mayor sea el número de arcos, el tiempo de evaluación aumenta. Esto se debe a que prácticamente se hace una simulación de la ruta trazada segmento por segmento, lo cual es terriblemente costoso, sobe todo si consideramos los 8000 arcos planteados en el párrafo anterior.

Los algoritmos genéticos son una herramienta poderosa en el desarrollo de nuevas soluciones a problemas sumamente complicados del mundo real. En particular, los problemas de alta dimensionalidad plantean una doble dificultad, ya que no pueden ser evaluados de manera rápida con un

algoritmo genético convencional. Es ahí donde entran las herramientas de alto desempeño, para realizar el mayor número de evaluaciones en el menor tiempo posible. MPI resulta ser una herramienta muy poderosa para el desarrollo de sistemas paralelos, ya que brinda una interfaz sencilla para el paso de mensajes y la asignación de recursos del sistema.

Las soluciones encontradas y las órbitas que se presentan son buenas soluciones que minimizan el consumo de combustible y el tiempo de vuelo del vehículo espacial. Es posible encontrar mejores soluciones con un número mayor de generaciones y manteniendo un tamaño de población pequeño en cada isla del algoritmo. Se encontró que el algoritmo funciona mucho mejor para poblaciones reducidas y un número de generaciones alto, siempre y cuando se utilice el método de búsqueda local en cada isla cada cierto número de generaciones

## Trabajo futuro

El problema de optimización de rutas es tremendamente complicado y existe un gran número de vertientes que atacar. Nosotros sólo atacamos un caso en particular, el cual resultó ser muy difícil, pero se pueden estudiar éste y otros casos con otro tipo de técnicas heurísticas que puedan manejar problemas con funciones muy costosas, computacionalmente hablando, para reducir el número de evaluaciones.

Otro aspecto a considerar pueden ser los métodos para resolver sistemas de ecuaciones diferenciales de primer orden, ya que los métodos planteados en este trabajo resultan ser muy costosos.

Por las razones planteadas anteriormente, no se pudieron realizar todas las pruebas que se hubieran deseado. Sin embargo, podrían realzarse más pruebas para ajustar los parámetros de entrada del MRMOGA y obtener un mejor desempeño.

Se pudo observar en los frentes obtenidos que no se tenía una buena cobertura del frente. Se puede trabajar en este aspecto para diversificar las soluciones y poblar los frentes encontrados, ya que en cada corrida sólo se encontraban uno o dos soluciones que pertenecían al frente de Pareto

# Bibliografía


[1] Aristóteles. *Metafísica*. Espasa-Calpe, México, 1989. ISBN 968-413-110-0.

[2] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[3] Jeffrey L. Bada and Antonio Lazcano. Origin of life: Some like it hot, but not the first biomolecules. *Science*, 296:1982–1983, June 2002.

[4] John E. Barker. *Patrick Matthew - Forest Geneticist(1790-1874)*. Forest History Today, 2001.

[5] William Bateson. *Mendel's Principles of Heredity, a Defense*. Cambridge University Press, London, 1902.

[6] Richard L. Burden and J. D. Faires. *Numerical Analysis*. Brooks/Cole Publishing Company, 1997.

[7] Carlos A. Coello and Gary B. Lamont. *Applications of Multi-Objective Evolutionary Algorithms*, volume 1. World Scientific, New Jersey, 2004. ISBN 981-256-106-4.

[8] Carlos A. Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 1990. ISBN 0-262-03293-7.

[10] Charles Robert Darwin. *On the Origin of Species by Means of Natural Selection Or the Preservation of Favoured Races in the Struggle for Life*. Cambridge University Press, Cambridge, UK, 1964. Originally published in 1859.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

[12] Theodosius Dobzhansky. Genetics and the Origin of Species. *Science*, 364:430–431, April 1952.

[13] Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[14] P. J. Enright and B. A. Conway. Discrete approximations to optimal trajectories using direct transcription and nonlinear programming. *Guidance, Control and Dynamics*, 15(4):994–1002, 1992.

[15] David B. Fogel, editor. *Evolutionary Computation. Toward a New Philosophy of Machine Intelligence*, New York, 1995. The Institute of Electrical and Electronic Engineers.

[16] David B. Fogel, editor. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms*, New York, 1998. The Institute of Electrical and Electronic Engineers.

[17] Lawrence J. Fogel. Artificial intelligence through simulated evolution. *John Wiley*, 1966.

[18] Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution*. John Wiley and Sons, New York, 1999.

[19] Edward Francis. Antony van leeuwenhoek and his "little animals". *Science*, 76:597, December 1932.

[20] David Goldberg. *Genetic Algorithms in Search Optimization, and Machine Learning*. Addison Wesley Longman, Reading, MA, 1989. Chap. 1.

[21] John J. Grefenstette, editor. *Reducing Bias and Inefficiency in te Selection Algorithm.*, Nueva Jersey, 1987. Genetic Algorithms and their applications: Proccedings of de second International Conference on Genetic Algorithms.

[22] John W. Hartmann. Low-thrust trajectory optimization using stochastic optimization methods. Master's thesis, Graduate College of the University of Illinois, Urbana-Champaign, Illinois, USA, 1999.

[23] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[24] Antonio López Jaimes. Diseño de un Algoritmo Genético Paralelo. Master's thesis, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, D.F., Febrero 2005. (in Spanish).

[25] Antonio López Jaimes and Carlos Coello Coello. MRMOGA: Parallel Evolutionary Multiobjective Optimization using Multiple Resolutions. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 3, pages 2294–2301, Edingurgh, Scotland, September 2005. IEEE Service Center.

[26] T. C. Koopmans, editor. *Analysis and production as an Efficient Combination of Activities*. Yale University Press, New Haven, London, 1971.

[27] Antonio Lazcano-Araujo, editor. *El origen de la vida: evolución química y evolución biológica*. Trillas, México, 2006. ISBN 968-24-3371-1.

[28] Seungwon Lee, Wolfang Fink, Ryan P. Russell, Paul von Allmen, Anastassios E. Petropoulos, and Richard J. Terrile. Evolutionary computing for low-thrust navigation. *AIAA Space Conference*, August 2005.

[29] Seungwon Lee, Paul von Allmen, Wolfang Fink, Anastassios E. Petropoulos, and Richard J. Terrile. Comparison of multi-objective genetic algorithms in optimizing q-law low-thrust orbit transfers. *Genetic and Evolutionary Computation Conference*, pages 221–226, 2005.

[30] Seungwon Lee, Paul von Allmen, Wolfang Fink, Anastassios E. Petropoulos, and Richard J. Terrile. Design and optimization of low-thrust orbit transfers. *AeroSpace Conference*, March 2005.

[31] Antonio López Jaimes and Carlos A. Coello Coello. MRMOGA: A New Parallel Multi-Objective Evolutionary Algorithm Based on the Use of Multiple Resolutions. *Concurrency and Computation: Practice and Experience*, 19(4):397–441, March 2007.

[32] A. Callahan M. Maxfield and L.J. Fogel, editors. *Artificial Intelligence through a Simulation of Evolution*, Washington D.C., 1965. Biophysics and Cybernetics Systems: Proceedings of the Second Cybernetics Sciences, Spartan Books.

[33] Michael T. Madigan, John M. Martinko, and Jack Parker. *Brock. Biología de los microorganismos*. Pearson. Prentice Hall, Madrid, 2004.

[34] Zbigniew Michalewicz and David B. Fogel. *How to solve it: Modern Heuristics*. Springer, Berlin, 2000.

[35] Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publisher, Massachusetts, 1999. ISBN 0-7923-8278-1.

[36] Salvador Moyà-Solà, Meike Köhler, David M. Alba, Isaac Casanovas-Vilar, and Jordi Galindo. Pierolapithecus catalanicus, a New middle Miocene Great Ape from Spain. *Science*, 306:1339–1344, November 2004.

[37] Erick Cantú Paz. *Efficient and Accurate Parallel Genetic Algorithms*. Springer, México, 2000. ISBN 978-07923-7221-9.

[38] Ingo Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, Alemania, 1973.

[39] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization*. John Wiley and sons, USA, 1983.

[40] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, Alemania, 1977.

[41] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[42] William M. Spears. *Evolutionary algorithms: the role of mutation and recombination*. Springer-Verlag, 2000.

[43] John T.Betts. Survey of numerical methos for trajectory optimization. *Journal of Guidance, Control and Dynamics*, 21(2):193–207, 1998.

[44] M. Zeleny. Towards the Tradeoffs-Free optimality in mcdm. *Multicriteria Analysis*, 1:596–601, 1997.

[45] S. Zionts. Decision making: Some experiences, myths ans observations. *Multiple Criteria Decision Making: Proceedings of the Twelfth International Conference*, 1:233–241, 1997.

[46] S. Zionts. Some Thoughts on MCDM: Myths and Ideas. *Multicriteria Analysis*, 1:602–607, 1997.