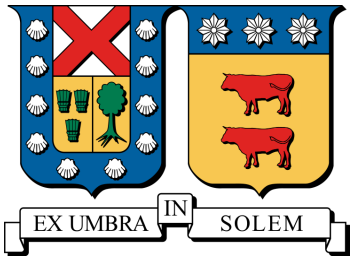
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE INFORMÁTICA
VALPARAíSO – CHILE



"IDIOTYPIC IMMUNE NETWORKS FOR ALMA ARRAY SCHEDULING PROBLEM"

RODRIGO ANTONIO GREGORIO LOVERA

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE:
INGENIERO CIVIL EN INFORMÁTICA

PROFESOR GUÍA : **Dr. Mauricio Solar**
PROFESOR CORREFERENTE : **Dra. Lorna Figueroa**

ENERO 2014

Agradecimientos

Quiero darle las gracias a mis padres por todo su apoyo en este largo proceso. Además, agradecer la confianza que ha depositado en mi el profesor M. Solar.

arigatô gozai-mashita.

Rodrigo A. Gregorio Lovera

Resumen

ALMA es un proyecto de cooperación multinacional, el cual estará compuesto por 66 antenas radio telescópicas. Las observaciones son solicitadas a través de proyectos de observación que deben ser planificados dentro de una temporada y se apunta al uso eficiente de tiempo disponible de observación y la importancia científica.

Hay varios factores que afectan la planificación de observaciones que incluyen condiciones climáticas, la posición de los arreglos, el grado científico, el estado de los instrumentos, etc.

En esta memoria se plantea el uso de una red inmune artificial para resolver este problema de planificación.

Abstract

ALMA is a multinational cooperation project, which will consist of 66 radiotelescopic antennas. The observations are submitted through observation projects, this projects should be scheduled within a season and should search the efficient use of observation available time and scientific importance.

There are several factors that affect the scheduling of observations including weather conditions, the position of arrays, the scientific degree, the state of the instruments, etc..

It is proposed using artificial immune network for solve this scheduling problem.

Índice de Contenidos

Agradecimientos **iii**

Resumen **iv**

Abstract **v**

Índice de Contenidos **vi**

Índice de Figuras **ix**

1 Introducción **1**

2 Problemas de Scheduling **5**

2.1 Introducción 5

2.2 Scheduling en Astronomía 6

2.3 Estado Actual en Observatorios Astronómicos 9

2.4 ALMA Array Scheduling Problem 14

2.4.1 Scheduling en ALMA 14

2.4.2 Subsistema de Scheduling de ALMA 15

2.4.3 Scheduling Dinámico en ALMA 18

3 Sistema Inmune Artificial **20**

3.1 Introducción 20
3.2 Sistema Inmune 20
3.2.1 Principio de Selección Clonal 24
3.2.2 Teoría de la Red Inmune 26
3.3 Sistema Inmune Artificial 27
3.3.1 Red Imune Artificial 28

4 Algoritmo Inmune para ALMA Array Scheluding Problem 33
4.1 Descripción 33
4.2 Condiciones Generales y Supuestos 33
4.3 Modelo Simplificado 36
4.4 Algoritmo $aiNET_{asp}$ 37
4.4.1 Representación 37
4.4.2 Función de Evaluación 38
4.4.3 Algoritmo 38

5 Validación de la Propuesta 43
5.1 Configuración 43
5.1.1 Sintonización 43
5.2 Diseño del Algoritmo 46
5.3 Resultados 50
5.4 Discusión 51
5.4.1 Análisis de Resultados 53

6 Conclusiones 57
6.1 Resultados y Contribuciones 57
6.2 Trabajo Futuro 58

Bibliografía **61**

A Diagrama de Clases de $aiNet_{asp}$ **65**

B Resultados del Algoritmo **67**

Índice de figuras

1.1 Esquema que sigue una señal astronómica captada por ALMA. ALMA es controlada desde el Centro de Operaciones (OSF), donde se recibe , procesa y almacena la información. 2

3.1 Diagrama de las barreras que componen el sistema inmunológico 22

3.2 Respuesta inmune adaptativa 25

3.3 Red inmune artificial 27

3.4 Red idiotípica. (A) Relación anticuerpo-antígeno (B) Configuración de una red inmune en cascada. Fuente:Preeti Gokal Kochar 29

5.1 Sintonización cantidad de iteraciones 44

5.2 Sintonización tamaño de la población 45

5.3 Sintonización cantidad de clones por individuo 45

5.4 Sintonización porcentaje de reemplazo 45

5.5 Diseño simplificado de clases de la implementación 46

A.1 Diseño de clases de la implementación 66

Capítulo 1

Introducción

Atacama Large Millimeter/submillimeter Array (ALMA) es un observatorio que está actualmente en construcción el Llano de Chajnantor en el desierto de Atacama, a 5000 metros sobre el nivel del mar. Estará compuesto por 66 radio-antenas las que operarán en conjunto para realizar interferometría en rangos milimétricos y sub-milimétricos de longitudes de onda.

La idea básica de la interferometría consiste simplemente en la recolección de una señal proveniente del cielo usando como medio de captura dos o más antenas y combinarlas para analizar la señal, con lo cual se obtiene información sobre la fuente de emisión sea una estrella, planeta o galaxia.

Combinando las ondas de radio capturadas es posible obtener imágenes con una altísima precisión. Las imágenes obtenidas pueden ser comparadas a las que se obtendrían con un telescopio o antena gigante de unos 14 kilómetros de diámetro. Sin embargo, la construcción y operación de una antena de ese tamaño es tecnológicamente imposible actualmente, por lo que la construcción de varias antenas pequeñas y usarlas de forma combinada resulta más viable.

ALMA es un arreglo o conjunto de antenas. El desafío es poder apuntar simultáneamente las antenas a una misma región en el cielo, con cada una capturar la señal astronómica, luego la señal convertirla a un formato digital, para luego transmitirla a un edificio central, donde

el correlador combinará las señales recibidas con el objetivo de crear datos a partir de los cuales se podrá realizar un análisis científico (ver figura 1.1)

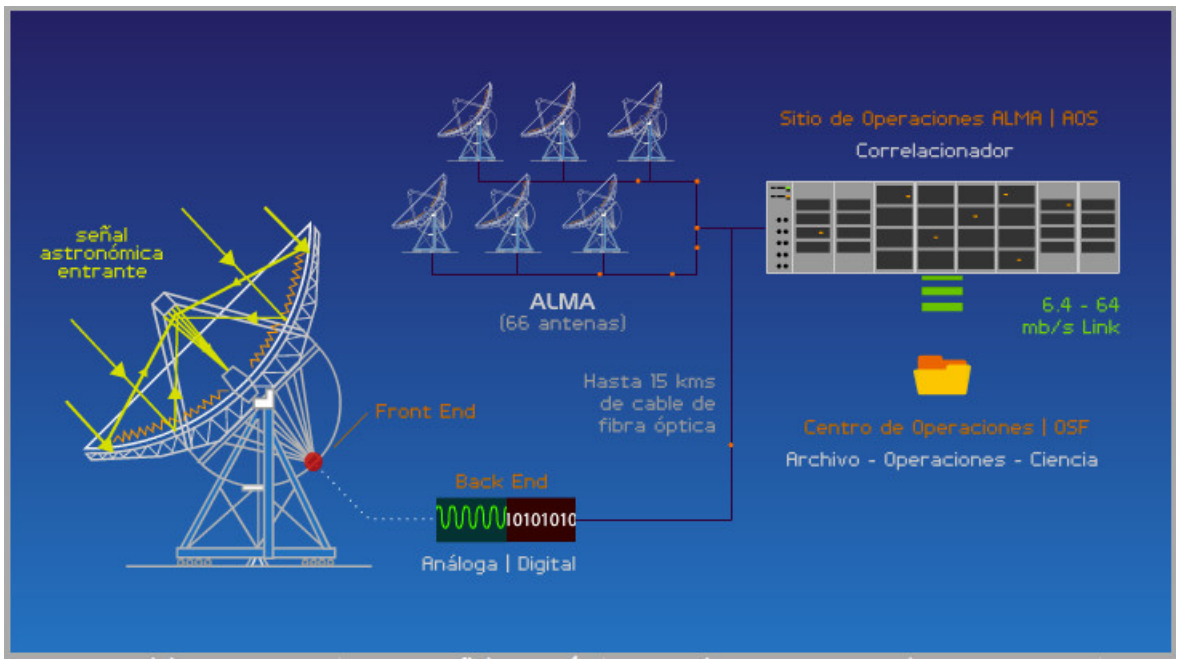


Figura 1.1: Esquema que sigue una señal astronómica captada por ALMA. ALMA es controlada desde el Centro de Operaciones (OSF), donde se recibe , procesa y almacena la información.

El 10 % del tiempo de observación está asignado para el uso por parte de Chile, el restante porcentaje se reparte entre los asociados de ALMA según el nivel de contribución financiera que han realizado hacia el proyecto. Aunque se aceptarán los proyectos de observación de científicos de cualquier parte del mundo, los provenientes de países que financian el proyecto tendrán una consideración especial.

Las propuestas presentadas por los científicos compiten por el tiempo de observación, las cuales serán consideradas en base a su importancia científica. Los usuarios no viajan al Llano de Chajnantor para realizar las observaciones. En cambio, éstas serán programadas dinámicamente, dependiendo de las restricciones meteorológicas y de la configuración. Las observaciones serán realizadas durante las 24 horas del día.

Los datos producidos durante las observaciones de ALMA se almacenarán en el Archivo de

ALMA. Todos los datos científicos de ALMA estarán sujetos a un período propietario de un año a partir de la fecha en que fueron distribuidos al Investigador Principal. Cumplido del período propietario, los datos serán públicos y cualquier investigador podrá acceder a ellos por medio de una solicitud.

ALMA proporcionara una nueva visión del universo. Explorará lugares desconocidas que hasta ahora no se han podido observar, con lo cual se podrá obtener datos sobre el origen del universo. ALMA tendrá la capacidad de observar astros más lejanos de los que se pueden observar actualmente con los observatorios instalados en en planeta.

Dentro de los variados problemas que genera ALMA se encuentra La planificación de observaciones. Este es un nuevo problema de Scheduling que resulta bastante interesante de tratar y resolver, ya que consiste en la planificación de observaciones al lo largo de una temporada, sujeta a una gran cantidad de restricciones. De lo anterior, se desprende la necesidad de implementar un algoritmo que facilite la planificación y cumpla con todas las restricciones asociadas.

Las observaciones astronómicas requieren de pre-condiciones para ser ejecutadas, tales como, el tipo específico de instrumento a ser usado, condiciones nocturnas específicas, datos climáticos, etc. En conjunto con los objetivos científicos de observación son presentados como una propuesta de observación. Un comité de científicos determina si la propuesta es aceptada como válida, rechazada o debe ser modificada. Con esto se puede determinar el tiempo de observación de un telescopio.

Es este trabajo se planeta como como objetivo general: proponer y aplicar un Sistema Inmune Artificial para ALMA Array Scheduling Problem en ALMA y como objetivos específicos:

- Diseñar un modelo de Red Idiotípica Inmune.
- Implementar un algoritmo de Red Inmune para un ALMA Array Scheduling Problem simplificado.
- Probar el algoritmo y obtener una mecanismo que permita la comparación con otros modelos implementados.

La metodología de trabajo consiste en investigar sobre el algoritmo y el problema, entender los alcances del problema y establecer un modelo simplificado el se busca ser resuelto mediante una heurística basada en una red inmune artificial. A continuación se resumen los principales hitos que se desarrollaron:

- Investigar y redactar el estado del arte.
- Simplificar el ALMA Array Scheduling Problem.
- Diseñar un algoritmo basado en una red inmune artificial.
- Obtener datos de entrada.
- Probar las instancias.
- Validar resultados.

La estructura del documento se divide en secciones. En la sección 2 se realiza un revisión del problema de planificación en astronomía y se describe el Array Scheduling Problem en ALMA. La sección 3 consiste en una descripción general de los sistemas inmunes y en que consiste un sistema inmune artificial basado en la teoría de red inmune. El modelado del problema y las características del algoritmo se describen en la sección 4. La sección 5 consiste en la validación de la solución propuesta, acompañada de un discusión sobre lo datos obtenidos. Finalmente, en la última sección se dan las conclusiones y el trabajo futuro. En el apéndice se entrega descripciones de la implementación.

Capítulo 2

Problemas de Scheduling

2.1 Introducción

El Scheduling Problem consiste en un problema de asignación de tareas o trabajos, los cuales pueden estar sujetos a restricciones y/o criterios de desempeño. La teoría de scheduling ha sido desarrollada para solucionar problemas de fábricas de producción. Se puede describir como la búsqueda de las tareas (trabajos) e intervalos de tiempo en los cuales una máquina puede ejecutar dicha tarea. Cumpliendo todas las restricciones asociadas. Se busca encontrar una planificación como solución óptima, minimizando la función objetivo. Dado N trabajos, éstos son planificados sobre un conjunto que están disponibles a partir del tiempo inicial, cada máquina puede manejar una solo trabajo a la vez.

Dependiendo de la configuración de las máquinas se distinguen entre problemas de scheduling de una sola máquina, máquinas paralelas y modelos job-shop [13].

En el campo astronómico los proyectos de observación deber ser planificados a lo largo de una temporada. Los proyectos están divididos en unidades atómicas o “scheduling blocks” (SB) . La ejecución de los proyectos esta sujeta a varios parámetros tales como prioridad científica, condiciones instrumentales, condiciones climáticas, etc.

Un telescopio debe manejar una gran cantidad de proyectos para cada temporada de observación, entonces, es necesario reducir los costos y optimizar el uso del observatorio.

2.2 Scheduling en Astronomía

Las observaciones astronómicas requieren de condiciones específicas para ser ejecutadas. Estas condiciones, en conjunto con los objetivos científicos, son presentadas por los astrónomos a través de una propuesta de observación. La solicitud es por tiempo de observación y el formato depende de cada Observatorio.

Las propuestas deben ser enviadas a los correspondientes Comités de Asignación de Tiempo de cada telescopio, los cuales evalúan y asignan un prioridad científica a cada solicitud, y se encarga de la aprobación o rechazo de las solicitudes de tiempo de observación. Una observación puede ser ejecutada en modo visita o modo servicio. El modo visita requiere la presencia física del investigador principal en el sitio del observatorio para adquirir los datos, en cambio, el modo servicio puede ser ejecutado por operarios o miembros del personal científico del observatorio.

Las propuestas aceptadas son proyectos divididos en bloques de observación, estos son los que finalmente son planificados en una temporada de observación. La ejecución de las observaciones dependen de factores externos a la propuesta, por lo tanto, el sistema automático o el operador del telescopio deben decidir la mejor asignación de tiempo para cada caso. Este problema considera una gran cantidad de proyectos a ejecutar en una temporada, esta cifra sobrepasa los miles. Se debe tener presente que los objetivos no están visibles durante toda la temporada, existe un plan a largo plazo que considera factores como la visibilidad sobre el horizonte y el brillo de la Luna, mientras que un plan a corto plazo considera los SB específicos para la siguiente noche basado en factores más inmediatos. La probabilidad de ejecución de un proyecto depende de su prioridad científica y de las condiciones necesarias para su observación. Un proyecto se considera completo cuando todos sus bloques de observación has sido ejecutados y completados exitosamente.

El scheduling en observaciones astronómicas es una variación del problema de scheduling

dinámico y es del tipo NP-duro. El problema es modelado en [11], en donde se realiza una formulación matemática básica del problema de scheduling.

El problema general de scheduling consiste en determinar el más eficiente procedimiento para la observación de un conjunto de objetivos astronómicos con un telescopio dado. cada objetivo es calificado mediante cinco parámetros:

- Coordenadas astronómicas (p. ej. ascensión recta y declinación)
- Tiempo de exposición
- Configuración instrumental
- Calidad científica del proyecto
- Restricciones temporales definidas por el observatorio (p. ej. visibilidad del objetivo) o por el observador (p. ej. observaciones simultáneas con otros observatorios)

Se debe tener en cuenta que la valoración científica involucra la calificación del proyecto por su importancia científica, entonces, se debe agregar una restricción que asegure que todos los objetivos del proyecto sean tratados de forma igualitaria independiente de lus restricciones temporales.

La planificación puede ser dividido en componentes a largo plazo y a corto plazo.

Componente a largo plazo Se refiere a una planificación de las observaciones en un lapso de tiempo de aproximadamente un año. Las restricciones de observación deben ser tratadas. En consecuencia, el problema de scheduling se convierte en una búsqueda sofisticada de la mejor solución factible, de la manera más eficiente en costo y tiempo. La solución óptima del problema de planificación depende fuertemente de las instalaciones astronómicas, por esta razón en difícil establecer un espacio común donde diferentes algoritmos y técnicas puedan ser probadas y comparadas. Una propiedad adicional es que no se lleva en tiempo real y por lo tanto el uso de algoritmos muy eficientes en tiempo no son tan críticos como en la planificación a corto plazo.

Componente a corto plazo Trata la planificación en un período de uno pocos días. Algunas noches de observación planificadas por el componente a largo plazo pueden interrumpirse debido a malas condiciones climáticas, fallas en el telescopio o en algún instrumento, o por pérdida del objetivo de observación. Lo anterior requiere de una replanificación de las observaciones, ahora con factores que necesitan ser absorbidos tan pronto como sea posible debido a las restricciones. El problema puede ser simplificado a realizar un planificación de una lista de observaciones con muchas posibles configuraciones instrumentales y sin restricciones de observación. Por lo tanto, el componente a corto plazo es básicamente independiente de las instalaciones astronómicas y constituye un buen área para crear un modelo matemático y realizar experimentos.

El Basic Scheduliung Problem [11] esta planteado como una planificación a corto plazo y considera una cantidad de objetivos de observación:

1. Una prioridad que refleja la relevancia científica de la observación.
2. La ascensión recta y declinación del objetivo.
3. El tiempo estimado para llevar a cabo la observación, el cual incluye los requerimientos de tiempo para la configuración de los instrumentos y la calibración, así como, el tiempo de exposición.

Un problema de planificación básico es declarado cuando el objetivo es maximizar el total de pesos de los objetivos a lo largo del tiempo de observación disponible. Este problema puede ser visto como un problema de permutación en el sentido que la solución óptima puede ser caracterizada como una permutación de objetivos. En este sentido, la variable de decisión únicamente considerara el orden en el cual los objetivos son observados.

Este problema es clasificado como NP-Duro. El tiempo computacional crece exponencialmente cuando la cantidad de objetivos aumenta. Algoritmos de aproximación o heurísticas pueden ser usados para resolver problemas de gran tamaño.

Un vector de objetivos será considerado una solución válida si cumple los requerimientos de tiempo.

Con el tiempo de observación posible y la disposición de los objetivos se puede construir una solución válida para el Basic Scheduling Problem usando un algoritmo greedy.

2.3 Estado Actual en Observatorios Astronómicos

Para conocer el problema de scheduling en observatorios se presenta el estado actual del problema. En [20], M. Mora realiza un estudio sobre el problema de Scheduling dinámico en los Observatorios Astronómicos.

Hubble Space Telescope

Hubble Space Telescope (HST) es un observatorio satélite de grandes dimensiones que se encuentra orbitando el planeta, consiste en un telescopio de reflexión con un espejo primario de 2.4 metros de diámetro el cual enfoca la luz sobre un arreglo de cinco instrumentos científicos. La resolución, sensitividad, cobertura UV del HST es mayor en comparación con lo que se puede obtener con los telescopios terrestres que sufren de la influencia de la atmósfera.

El proyecto SPIKE fue iniciado por el Space Telescope Science Institute. SPIKE trata la construcción de la planificación como problema de optimización de restricciones y usa una heurística de búsqueda basada en reparar planificaciones llamada “Multistart Stochastic Repair” [17]. Esta técnica consiste el los siguientes pasos:

1. Asignación de Pruebas: Crea una asignación de prueba de actividades a tiempos, basado en heurísticas. Como una planificación generalmente tendrán violaciones de restricciones, así como, sobrecarga de recursos.

2. Reparación: Aplicar técnicas heurísticas de reparación para eliminar violaciones de restricciones, hasta que otro preestablecido nivel de esfuerzo haya sido gastado o no queden conflictos.

3. Evitar Conflictos: Eliminar conflictos mediante la remoción de cualquier actividad con violación de restricciones, o por relajación de restricciones, hasta que una planificación factible quede.

Las heurísticas usadas en SPIKE son estocásticas. La estrategia es seleccionar al mejor de varias ejecuciones, posiblemente probando distintas configuraciones iniciales y heurísticas de reparación.

Las propuestas de observación pueden tener una amplia gama de restricciones con el fin de asegurar el logro de los objetivos científicos buscados (los más comunes son requerimientos de tiempo). Además, se consideran los requerimientos operacionales de la nave espacial y los instrumentos, y de los recursos y políticas de restricciones.

Restricciones críticas tienen efectos que varían drásticamente sobre las escala de tiempo (de minutos a meses). El objetivo del scheduling es maximizar la utilización eficiente del telescopio en algún periodo de tiempo.

Si se ignora el rol de las preferencias del problema de planificación, se puede considerar como un problema de satisfacción de restricciones. Dado un conjunto de variables representando actividades a ser planificadas sobre un intervalo de tiempo y un conjunto de restricciones, se busca una asignación de tiempo cumpliendo todas las restricciones [16].

Very Large Telescope

En el Very Large Telescope (VLT) [23], es operado por la ESO y es uno de los más grandes telescopios ópticos del mundo. Esta ubicado en el norte de Chile, en la cima del cerro Paranal. Esta compuesto de 4 telescopios de 8.2 metros de apertura y un interferometro (VLT interferometer) para obsrvaciones con mayor resolución.

El sistema automatizado de scheduling es pensado solo como una ayuda para las decisiones humanas. Las decisiones mezclan los modos de observacion para visitante o de servicio.

Generalmente los usuarios envían dos propuestas de observación al año para que sean revisadas por la ESO Time Allocation Commitee, conocido como el Observing Programme

Commitee (OPC). Las propuestas son enviadas para un periodo específico de seis meses (los periodos comienzan en Abril y Octubre).

Antes de cada reunión del OPC, ESO determina el tiempo total disponible. En un periodo normal, cada telescopio del VLT puede tener 140 noches disponibles. Las otras 42 noches son usadas para el ESO Calibration Plan, Director's Discretionary Time, y mantenciones técnicas de instrumentos y telescopios.

La principal función del OPC es generar una lista priorizada científicamente de ejecuciones y ordenar el tiempo total disponible. Una vez completada la revisión por parte del OPC, es responsabilidad de ESO generar el Long-Term Schedule (LTS). El objetivo es planificar y ejecutar todas los propuestas por encima del limite de corte de la OPC, por ejemplo la linea definida por el tiempo disponible en cada telescopio e instrumento .

Green Bank Telescope

El Robert C. Byrd Green Bank Telescope (GBT) es un radio telescopio de antena movible, que se extiende en un gran rango de frecuencias que otros telescopios centimétricos/milimétricos de un solo plato. Está ubicado en una región donde el clima esta dominado por el vapor de agua y efectos de menos escala.

El GBT ha empleado una forma simple de planificación dinámico dividido en dos proyectos, uno de alta frecuencia y otro de baja frecuencia. Ambos son planificados juntos en dos temporadas. Este esquema genera que a menudo el observador de alta frecuencia recibe poco o ningún tiempo si ellos esperan por un clima de alta frecuencia, lo cual compromete la capacidad de descargar este proyecto, y podrían retardar la ejecución de los proyectos de baja frecuencia. No todos los programas de alta frecuencia requieren de los mismos climas.

El Dynamic Scheduling System (DDS) trabaja rompiendo todas las propuestas de observación en pequeñas temporadas, donde cada temporada contiene un único conjunto de datos que puede ser planificados sobre el telescopio, en uno o más periodos de telescopio.

Una vez obtenida toda la información sobre las propuestas de observación aceptadas son

introducidas y validadas, el DSS puede proveer los potenciales observadores con probabilidades que indiquen cuando el proyecto esta listo para ser planificado en el telescopio en un periodo de cuatro meses. Usando el actual pronóstico de clima, una planificación adelantada de 24 horas es creada. Esta planificación es revisada y modificada por un planificador humano y finalmente aprobado [21].

El DSS usa una ecuación que asigna un puntaje según un ranking de posibles sesiones de observación para un tiempo dado sobre una variedad de factores agrupados en: clima, factores de presión de planificación , limites de desempeño, y otros factores. Los parámetros relacionados al clima son eficiencia de observación y robustez, la presión de planificación en una medición de demanda insatisfecha. Un candidato a sesión de observación debe satisfacer todos los limites relevantes de desempeño, en otro caso será cero o un valor muy reducido. Los límites son asociados con eficiencia de observación, ángulo hora absolutos, ángulo del zenit, error de seguimiento y estabilidad atmosférica. los otros factores son necesarios para implementar decisiones administrativas [1].

Stratospheric Observatory for Infrared Astronomy

Stratospheric Observatory for Infrared Astronomy (SOFIA) [9] consiste en un avión Boing 747-SP modificado sobre el cual ha sido instalado un telescopio de 2.5 metros que entró en operación en 2005. Se espera volar alrededor de 140 vuelos científicos por año durante durante 20 años de tiempo de vida. Dependiendo de los instrumentos usados se pueden obtener entre 5 y 15 observaciones por vuelo. Un problema importante en la operación de SOFIA es la planificación de vuelos.

La planificación de vuelos de la generación de observatorios aéreos, el Kuiper Airborne Observatory (KAO), fue hecho manualmente.

Aproximadamente la mitad de los vuelos de SOFIA son manejados por los Investigadores Generales, quienes proponen una cantidad pequeña de observaciones que deben ser unidas a vuelos por el staff de operación de SOFIA. Los planificadores de vuelos reciben una lista de propuestas para realizar observaciones con un determinado instrumento. A cada propuesta

se le asigna un puntaje según importancia por el Time Allocations Commitee (TAC), usando estas propuestas se genera una planificación semestral o trimestral. Un instrumento puede permanecer sobre el avión durante un periodo de semanas (son cambiados los fin de semana). Se espera tener de dos a tres vuelos por semana. El conjunto de objetivos y duración de las observaciones debe ser determinado en su totalidad, proporcionando menos latitud al planificador de vuelos. Debido al número esperado de vuelos y la complejidad creciente del problema, la planificación manual es muy costosa.

Se plantea el problema como un planificador de vuelos, que se presenta con una lista de unos diez a cien peticiones de observación, y un periodo de semanas en los cuales deben ser planificados. Los días de vuelos son diseñados por adelantado, las observaciones son configurada con un instrumento particular, y la planificación de los instrumentos es planificada trimestralmente o semestralmente. Observaciones son agrupadas dentro de propuestas, a cada cual se le entrega un TAC ranking de 1 a 5, el ranking 5 es el más importante. La tarea del planificador de vuelos es seleccionar que observaciones serán planificadas, agrupándolos en días de vuelos. Los planes de vuelos deben satisfacer todas las restricciones descritas. Las observaciones deben tener restricciones sobre line-of-sight water vapor (LOS WV) permitidos en una planificación, o se debe minimizar la LOS WV.

Herramientas de planificación en la ESO

El Long Term Scheduling (LTS) y el Short Tern scheduling (STS) forman parte del ESO Observatoring Handling Subsystem (OHS, un conjunto de herramientas que administras las propuestas de observación, planificación de observaciones y secuencias de observaciones en forma de Bloques de Observación (BO). El LTS es usado dos veces al año para analizar las características de un conjunto de propuestas de observación aceptadas y genera una planificación de seis meses. Por otro lado, el STS es usado cada noche para producir secuencias de ejecución de BO que minimicen el desborde frente a un conjunto seleccionable de condiciones climáticas, configuraciones de instrumentos y prioridades científicas.

Durante el año 2000 el staff de operaciones científicas del VLT estaba usando el STS como

herramienta de soporte, pero no planificaba BOs en una base regular. STS reporta y el programa generador de restricciones es usado para organizar la información de los BO y decide cual se debe ejecutar, pero la secuencia de ejecución de BO es construida manualmente [10].

2.4 ALMA Array Scheduling Problem

El ordenamiento de propuesta para realizar observaciones astronómicas es un problema complejo, el cual se ha tratado en varios caminos en las últimas décadas. Ademas muchos de los modernos observatorios, que actualmente están en funcionamiento, usan algún grado de automatización en el proceso de scheduling; hay todavía una gran intervención humana en la construcción de la planificación diaria y tomar la última decisión dado las restricciones y cambios en los parámetros externos.

Los parámetros externos pueden variar durante cualquier instante de tiempo mientras una observación se esta ejecutando, por lo tanto se necesita realizar una re-planificación dinámica. Por otro lado las propuestas de observación dependen de la visibilidad de las fuentes astronómicas a ser observadas, estas fuentes podrían ser visibles durante cierto lapso de tiempo durante el día o el año . A consecuencia de lo anterior, una propuesta con alta prioridad podría ser menos considerada para su ejecución o definitivamente no realizarse en algún periodo de la temporada actual.

2.4.1 Scheduling en ALMA

ALMA, debido a los rangos de bandas de frecuencia, vapor de agua y otras restricciones pueden posibilitar o no la obtención de una datos buenos. Para hacer más eficiente el uso de las capacidades del radiotelescopio bajo las condiciones de ambiente variable, una planificación dinámica es implementada como primer modo de scheduling.

ALMA operará exclusivamente en el modo de servicio. Por lo tanto, se espera que el software del subsistema de scheduling provea una plataforma de planificación dinámica completamente automático de quasi tiempo real. Tendrá de uno a seis arreglos de antenas, asignando

bloques de observación correspondientes a propuestas de observación que son previamente enviadas por el investigador principal, y son evaluadas por un Comité Científico (APRC: ALMA Programm Review Committee). El conjunto de arreglos contara de tres tipos de antenas:

- 12 antenas de 7 metros: Son las antenas de forman parte del ALMA compact Array (ACA), el cual funcionará como único arreglo aparte, pero parte del arreglo principal. Ellas son costruidas por MEICo [1]

- 4 antenas de poder total de 12 metros: Estas son antenas especiales, parte de ACA. Ellas también son construidas por MEICo.

- 50 antenas de 12 metros: Son las que pertenecen al arreglo principal y son las que se van operar principalmente, también son las que se dividirán en varios arreglos. Ellas son construidas por Vertex[2] y AEM [3]

Gran parte de la operación del telescopio se manejará a través del software de ALMA, el cual esta dividido en varios subsistemas como Control, Correlador, Archivo, etc. El subsistema de scheduling en uno de los encargados de manejar el arreglo de antenas y ejecutar dinámicamente los proyectos de observación.

2.4.2 Subsistema de Scheduling de ALMA

El subsistema de scheduling es el único a cargo del manejo del arreglo de antenas y la ejecución dinámica de bloques de observación. El diseño de este subsistema fue primeramente discutido en [27] , donde una lista priorizada de observación es planificada. Se describe un aplicación para planificación llamada “taco” y se recomienda la implementación de un scheduling automático.

[1] Mitsubishi Electric Corporation
[2] Vertex Antennatechnik GmbH
[3] Alcatel Alenia Space France, Alcatel Alenia Space Italy, European Industrial Engineering S.r.L., MT Aerospace

Algunos agentes externos que interactúan de alguna manera con el subsistema de scheduling son:

Operador del Telescopio : Es quien puede cambiar o reemplazar las decisiones de scheduling que el subsistema toma.

APRC : El comité se encarga de graduar científicamente cada uno de los proyectos de observación que han aceptado.

Investigador Principal : Usualmente un astrónomo, en conjunto con otros colegas, crea un proyecto de observación. Ellos presentan su proyecto al ARPC para que lo revise. Una vez aprobado y graduado el proyecto entra al Archivo para su eventual ejecución. Es importante mencionar que no se garantiza la ejecución de un proyecto.

Un proyecto de observación es dividido en unidades pequeñas denominadas bloques de observación (scheduling blocks). Estos bloques son unidades atómicas en sentido de ejecución por el sistema de control. Cada uno de ellos contiene la información necesaria para saber que recursos usar, los instrumentos necesarios, la sensibilidad necesitada y las un conjunto de restricciones (como tiempo y restricciones de clima)[12].

El propósito de subsistema de scheduling es manejar y programar la ejecución de lo proyectos de observación aprobados y sus bloques de observación. El diseño de los bloques del subsistema considera varios modos de operación.

Fully Automatic Mode : Permite a ALMA programar los bloques de observación de acuerdo al Dynamic Scheduling Algorithm (DSA)

Interactive Mode : Permite al operador del telescopio cambiar el próximo bloque de observación a ser ejecutado de una lista preparada por el DSA

Queue Mode : El operador del telescopio prepara una lista de bloques de observación para ser ejecutados. El DSA puede ser o no considerado como una referencia.

El subsistema de scheduling de ALMA evaluará los bloques de observación en dos niveles.

La primera evaluación será antes que se cree el planificador dinámico. El planificador master, el cual maneja todos los subsecuentes planificadores y tiene una cola master de bloques de observaciones, se agruparán los similares en la sub-cola. Durante esta evaluación el planificador master asegura el objetivo sea óptimamente visible para la duración total de los bloques de observación. También un bloque de observación debe poderse ejecutar durante el tiempo disponible. Desde luego los requerimientos de antenas y de hardware deben estar disponible. Esta sub-cola es enviada a un nuevo planificador dinámico con una política de scheduling dada.

El segundo nivel de evaluación sucede sobre la sub-cola. El planificador dinámico usa el cálculo de las políticas de scheduling para decidir cual bloque de observación será observado basado en las condiciones actuales. Esta evaluación sucede alrededor del tiempo real y no se enfoca sobre las condiciones a largo plazo del sistema.

Un proyecto de observación estará compuesto por uno o más bloques de observación, los cuales deben ser ejecutados en un único arreglo. Éstos contienen varias configuraciones que especifican las condiciones climáticas, tipo de instrumentos, banda de frecuencia, visibilidad, tiempo de ejecución,etc.

Algunos factores, que podrían ser usados para el cálculo de las políticas de scheduling, son:

- Humedad
- Temperatura
- Velocidad del Viento
- Cantidad de Antenas Disponibles
- Configuración de los Arreglos
- Visibilidad del Objetivo
- Tiempo de Observación Asignado al País
- Calibraciones Necesarias

- Prioridad Científica del Proyecto
- Banda de Frecuencia

Una función de ranking es usada para ordenar los bloques de observación en la sub-cola de planificación dinámica de forma descendente. También se usará esta fórmula para el subconjunto de factores disponibles en ALMA. Un ejemplo de formula es ordenar cada proyecto por la prioridad científica.

La puntuación final será usualmente una simple formula. El bloque de observación con más alto puntaje será el mejor candidato para la próxima observación [19].

2.4.3 Scheduling Dinámico en ALMA

El planificador dinámico en ALMA operará sobre un secuencia de bloques de observación que posee exclusivamente. Ningún otro planificador podrá tener acceso a éstos. El proceso de selección consta de cuatro etapas.

Primero, todos los datos relevantes concernientes a las condiciones de entorno y de estado del sistema de telescopio son reunidas, incluyendo los resultados de las calibraciones.

Segundo, una lista de bloques de observación es seleccionada. Los objetivos deber ser óptimamente visibles durante toda la duración del bloque. Todo bloque debe proporcionar la cantidad de antenas requeridas.

Tercero, cada bloque de la lista candidata, se le asigna un número. Esta asignación toma las condiciones de entorno actuales y las calibraciones recientes. Los factores varios empleados para realizar esta determinación son entregados como pesos y parámetros a una formula que puede ser fácilmente modificada.

Cuarto, un número es computado para cada bloque, que ordena a todos los bloques bajo ciertas condiciones deseables. Factores involucrados en el cálculo: prioridad científica, tiempo de configuración, recursos usados, costo de inicio de un nuevo proyecto, número de los bloques que faltan ejecutar en el presente proyecto, el tiempo de ejecución de bloque del

proyecto actual y su “stringency”. Cada factor tiene un valor y un peso, estos factores son parametrizados dentro de una fórmula.

El puntaje final es calculado para cada bloque de observación en la lista por medio de fórmulas. A menos que sea anulado por el operador del telescopio, el bloque con mejor ranking es ejecutado [8].

Capítulo 3

Sistema Inmune Artificial

En este capítulo se presenta una descripción y el estado del arte de los sistemas inmunes artificiales. Se detallan las principales implementaciones de SIA, definiendo sus características y aplicaciones principales. Finalmente, se realiza una revisión de la literatura más importante en el ámbito de la aplicación de los SIA en la resolución de problemas de optimización. Una revisión completa del sistema inmune se encuentra en [22].

3.1 Introducción

En los últimos años, el interés por estudiar los sistemas artificiales inspirados en la biología se han incrementado significativamente basado en la idea de que la naturaleza es sabia y podemos tomar algunos de sus modelos para resolver problemas. De los sistemas bioinspirados que se estudian se considera el **Sistema Inmune Artificial**.

3.2 Sistema Inmune

Los organismos vivos poseen un sistema de protección, el cual cumple la función de reconocer y eliminar cualquier patógeno (tanto propio como externo) o cuerpo extraño que pueda

poner en peligro el funcionamiento normal del sistema. La característica más importante de este sistema, llamado sistema inmunológico, es la capacidad de distinguir lo propio de lo ajeno y de esta manera detectar posibles amenazas y así proteger al organismo.

Los mecanismos inmunológicos más sofisticados en la naturaleza se pueden encontrar en los vertebrados, estos incluyen componentes que actúan en forma conjunta para proporcionar una respuesta inmune. La propiedad que hace a estos sistemas los mejores de la naturaleza es la capacidad de adaptarse y de generar memoria al encuentro con un elemento patógeno (antígeno), lo que produce mejores respuestas en encuentros futuros. Su complejidad es comparable a la complejidad del cerebro y su efectividad es muy alta, pudiendo proteger al cuerpo de un gran número de patógenos. A partir de esta extraordinaria capacidad deriva el hecho que la comunidad científica se interesara por comprender e imitar su estructura y procesos.

Se pueden distinguir tres niveles de defensa en la respuesta generada por el sistema inmunológico:

Barreras físicas y químicas : Impiden y neutralizan la entrada de patógenos al cuerpo, la más conocida es la piel.

Respuesta Inmune Innata : Controlada principalmente por granulocitos y macrófagos, puede combatir un amplio rango de bacterias sin requerir previa exposición a los patógenos, por lo tanto su respuesta no varía en el tiempo ni de individuo en individuo.

Respuesta Inmune Adaptativa : Controlada principalmente por los linfocitos, consiste en la producción de anticuerpos en respuesta a un determinado antígeno. Está en constante aprendizaje y especialización en concordancia a los antígenos a los cuales ha sido es expuesto el organismo.

Estas tres barreras conforman un sistema muy efectivo, las barreras físicas y químicas impiden la entrada de los patógenos, si ésta respuesta es inefectiva la respuesta inmune adaptativa genera inmunidad contra las re-infecciones y la respuesta innata provee una defensa de amplio espectro no específica (ver figura 3.1). Esto le permite al organismo vivir en un entorno rodeado de agentes potencialmente patógenos y desarrollar enfermedades en pocas ocasiones.

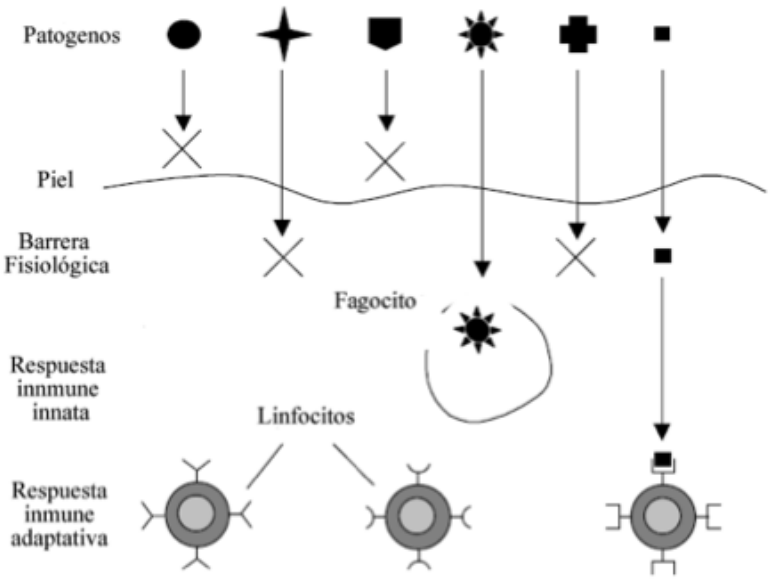


Figura 3.1: Diagrama de las barreras que componen el sistema inmunológico

Las responsables directas de la defensa de nuestro organismo frente a elementos patógenos son las secreciones y células, entre las células más importantes se destacan los fagocitos, encargados de la respuesta innata y los linfocitos B y T encargados principalmente de la respuesta adaptativa.

Fagocitos

Los fagocitos (macrófagos y neutrófilos) son células blancas capaces de ingerir y digerir microorganismos y partículas antigénicas, son los principales actores en la respuesta innata del sistema inmune. Algunos fagocitos tienen la habilidad de presentar antígenos a los linfocitos, por lo que pueden ser clasificadas como células presentadoras de antígenos (CPA). Entre ellos podemos encontrar a los monocitos, quienes circulan en la sangre y migran hacia los tejidos donde se convierten en macrófagos, estas células juegan un rol muy importante en la respuesta inmune, debido a que son capaces de presentar antígenos en su superficie luego de digerirlos, lo que se puede traducir como una señal de alerta al sistema.

Anticuerpos

Los anticuerpos son moléculas de proteínas altamente especializadas, cumplen el rol de reconocer un tipo específico de antígeno, su nombre químico es inmunoglobulinas o gammaglobulinas y son producidos a partir de la activación de los linfocitos B, quienes también los utilizan en su superficie celular como detectores de antígenos. Todas las moléculas de anticuerpos tienen una estructura básica en forma de Y o combinaciones de ésta, su composición es en base a dos cadenas livianas y dos pesadas unidas por puentes de disulfuro, cada una de estas cadenas posee una región constante y una variable. La región constante define a que tipo de anticuerpo pertenece. La región variable es responsable de la afinidad específica de anticuerpo frente a los antígenos.

No es necesario un reconocimiento completo, los anticuerpos tienen cierta afinidad con los antígenos que reconocen y estos pueden reconocer antígenos que estén estructuralmente relacionados con diferentes niveles de afinidad. Cuando un anticuerpo reconoce un antígeno se une a éste, marcándolo para su eliminación y junto con otros componentes del sistema desencadena una serie de reacciones que llevan a la destrucción de la amenaza.

Linfocitos T

Se desarrollan en la médula ósea a partir de células madres para luego madurar en el timo, desde donde migran hacia el bazo, médula ósea, nodos linfáticos y la sangre. La respuesta inmune generada por estas células recibe el nombre de respuesta celular, ellas atacan directamente a los invasores y además actúan como reguladoras del sistema inmunológico. Cada linfocito T reconoce un antígeno en específico y normalmente ligado a una molécula propia llamada complejo mayor de histocompatibilidad (CMH).

Linfocitos B

Se encuentran principalmente en la médula ósea, bazo y algunos sectores del intestino, tienen como función principal producir anticuerpos. A diferencia de los linfocitos T pueden

reconocer antígenos sin necesidad que estén acoplados a una molécula CMH. Cuando un linfocito B se encuentra con un antígeno reconocido, éste pasa a su interior y es procesado para finalmente exponer segmentos del antígeno unido a una molécula CMH clase II en su superficie. Esto atrae a las células T colaboradoras complementarias lo cual desencadena en la activación del linfocito B. Un vez activado comienza a dividirse madurando en un tipo de células llamadas células plasmáticas. Cada anticuerpo puede reconocer un antígeno en especial, la cantidad de anticuerpos posibles es tan grande que el sistema inmunológico puede proteger al cuerpo virtualmente de todo. Esta respuesta inmune recibe el nombre de respuesta humoral.

3.2.1 Principio de Selección Clonal

Burnet propuso en 1959 [2] el principio de selección clonal para explicar como funciona el sistema inmune adaptativo. Actualmente es ampliamente aceptado como un modelo para la respuesta inmune. Esta teoría explica como al ser activado un linfocito este se prolifera en un proceso llamado expansión clonal (el proceso es diferente entre células B y T). En las células B se produce mutación durante la reproducción y se generan células excretoras de anticuerpos, en las células T no se produce mutación y se generan células excretoras de linfoquines. Al ser activado un linfocito B por el reconocimiento de su antígeno específico, éste prolifera en clones, los cuales se diferencian en células plasmáticas que producen anticuerpos o en células de memoria quienes circulan por el torrente sanguíneo y que al encontrarse con el antígeno para el cual han sido diseñadas se reproducen en células plasmáticas que producen anticuerpos de alta afinidad (ver figura 3.2).

Al proliferar una célula B en clones estos son sometidos a un mecanismo de mutación, el cual consiste en modificar los genes que dan origen a las regiones variables de las cadenas livianas y pesadas. Este proceso se llama hipermutación somática y origina que los clones posean anticuerpos con una ligera diferencia de afinidad. Estas pequeñas variaciones aumentan la diversidad en el población de anticuerpo. En consecuencia, la mutación de los clones juega un rol primordial en la maduración de la respuesta inmune, entregando la diversidad

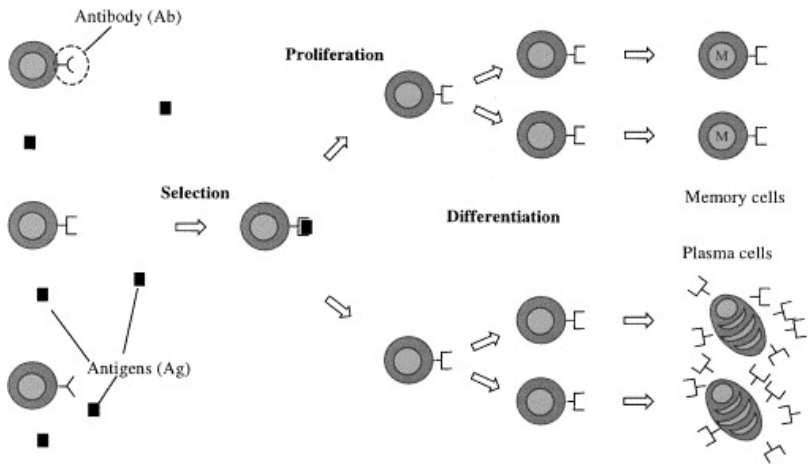


Figura 3.2: Respuesta inmune adaptativa

necesaria para encontrar mejores especificidades. Debido a la naturaleza de estas mutaciones aleatorias, se debe regular que simplemente no se generen clones peores que la célula actual, el mecanismo por lo tanto genera mutaciones de acuerdo a la afinidad presentada por la célula, de manera de conservar los buenos patrones de receptores y mutar mayormente los que han sido menos exitosos.

La expansión clonal, memoria y edición de receptores permiten que el sistema inmunológico esté en constante proceso de adaptación y aprendizaje. El cuerpo al encontrarse con un antígeno por primera vez, solo dispondrá de un par de anticuerpos específicos, luego de un cierto tiempo de espera, estos anticuerpos se encontrarán en mayor cantidad y afinidad, ésta es la llamada respuesta primaria. Luego de eliminar las amenazas,la cantidad de anticuerpos disminuye. En una segunda exposición al antígeno se provoca una respuesta más veloz y específica (memoria), debido a que ya existen registros del invasor. Esto sucede no solo para el un antígeno específico, sino que para cualquiera estructuralmente similar a él, esto se conoce como reacción inmune cruzada (memoria asociativa). La expansión clonal y la edición de receptores cumplen un papel fundamental en la maduración de afinidad de los receptores, la primera permite la "explotación" del espacio de búsqueda generando pequeñas diferencias en los clones, que podrán llevar a un mejor receptor (óptimo local) y la segunda permitirá realizar grandes saltos en este sentido y generar receptores completamente diferentes explorando otras áreas del espacio de búsqueda.

3.2.2 Teoría de la Red Inmune

La teoría de la red inmune fue postulada por Jerne [15] en 1974, la cual introduce una nueva visión sobre el comportamiento del sistema inmunológico basada en los descubrimientos realizados hasta esa fecha, hoy en día esta teoría es poseedora de buena cantidad de evidencia científica que la respalda. La propuesta de Jerne se basa en que el sistema inmune está compuesto por una red regulada de moléculas y células que se reconocen las unas a las otras, incluso en la ausencia de antígenos. El fragmento de unión al antígeno en el anticuerpo se denomina parátopo, a su vez el determinante antigénico se denomina epítopo. Debido a que constantemente se crean nuevos anticuerpos Jerne supone que es lógico pensar que estos desconocidos anticuerpos pueden ser reconocidos al igual que los antígenos (ver figura 3.4).

Se demostró experimentalmente que los anticuerpos tienen epítopos, determinados por la misma región variable que determina a los parátopos. Se define por lo tanto como idiótopo la parte de un anticuerpo que puede ser reconocida por otros elementos del sistema inmune, de esta manera los anticuerpos y células no solo podrán reconocer antígenos, sino que también podrán ser reconocidos por otros anticuerpos o células. Esta idea implica que el sistema inmunológico se define como una compleja red de parátopos que reconocen un set de idiótopos, de manera que cada elemento pueda reconocer y ser reconocido. En este esquema los linfocitos podrían responder de forma positiva o negativa al reconocimiento de un antígeno o detector, resultando en la activación y proliferación de la célula o en tolerancia y supresión de ella.

Este modelo sugiere que el sistema inmune está en continua adaptación, sometido a estímulos que pueden modificar su estructura y que sin embargo logra mantener estados estables dentro de esta oscilación constante. Este comportamiento dinámico hace emerger propiedades como el aprendizaje, memoria, tolerancia a lo propio, diversidad y tamaño de población, propiedades que no pueden ser entendidas desde el punto de vista de componentes aislados. Su estructura que describe los tipos de interacción entre sus componentes sin hacer referencia a las consecuencias funcionales que éstas ocasionarán, la dinámica explica las variaciones en el tiempo de la concentración y afinidad de las células y moléculas de la red, explicando como la red se adapta a si misma y al entorno y finalmente la metadinámica, que caracteriza

la continua producción de nuevos anticuerpos, los cuales pueden interactuar con el sistema, causando una renovación constante de la red, en resumen la metadinámica representa el mecanismo inmune para el aprendizaje [25].

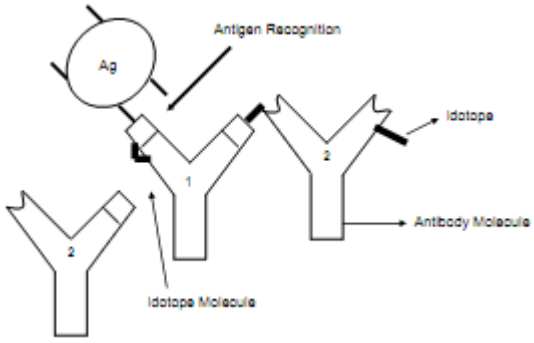


Figura 3.3: Red inmune artificial

3.3 Sistema Inmune Artificial

El Sistema Inmune Artificial (SIA) es un modelo computacional de nuestro sistema inmune biológico, el cual tiene la capacidad de realizar algunas tareas como reconocimiento de patrones, aprendizaje, adquisición de memoria, generación de diversidad, tolerancia al ruido, generalización, detección distribuida y optimización. basados en los principios inmunológicos, nuevas técnicas computacionales se ha desarrollado enfocándose no solo en una mejor comprensión del sistema mismo, sino que también a la solución de problemas de la ingeniería.

Varias características del sistema inmune son interesantes:

- **Singularidad:** Cada elemento posee su propio sistema inmune, con sus propias capacidades y puntos vulnerables.
- **Reconocimiento de Cuerpos Extraños:** Las moléculas (antígenos) que no son propias del cuerpo son reconocidas y eliminadas por el sistema inmune.
- **Detección de Anomalías:** El sistema inmune puede detectar y reaccionar a los patógenos que el cuerpo no había reconocido antes.

- **Detección Distribuida:** Las células del sistema inmune están distribuidas por todo el cuerpo, y más importante, es que ninguna esta regida por un sistema de control centralizado.

- **Detección Imperfecta:** No es necesario que se haga una detección exacta de los cuerpos extraños dado que el sistema inmune es flexible.

- **Reforzamiento del aprendizaje y Memoria:** El sistema puede "aprender" las estructuras de los patógenos, de manera que las respuestas futuras a los mismo patógenos son más rápidas y fuertes.

El concepto de sistemas inmunes artificiales (SIA) abarca variadas disciplinas, se puede definir un sistema inmune artificial como [6]:

"Sistemas adaptivos inspirados en la inmunología teórica y las funciones, principios y modelos inmunes observados, los cuales son aplicados a la resolución de problemas."

Según esta definición un sistema puede ser clasificado como un SIA si posee al menos un modelo básico de un componente inmune, está diseñado incorporando ideas de la inmunología teórica o experimental y resuelve un problema.

3.3.1 Red Imune Artificial

El la teoría de red idiotípica de Jerne se propone que los anticuerpos poseen un conjunto de epítopos y pueden ser reconocidos por otros anticuerpos(ver figura 3.4). Cuando un idiótopo de un anticuerpo es reconocidos el parátopo de otro anticuerpo es suprimido y su concentración reducida. Sin embargo, cuando un anticuerpo reconoce los idiótopos de otro anticuerpo o los epítopos de un antígeno es estimulado y su concentración aumenta. La teoría de Jerne ve al sistema inmune como una red compleja de parátopos que reconocen idiótopos, e idiótopos que son reconocidos por parátopos [26].

La red inmune actúa como un sistema autónomo que regula el rango específico de actividad de la red. Se logra modificando la población de anticuerpos y las afinidades con los antígenos.

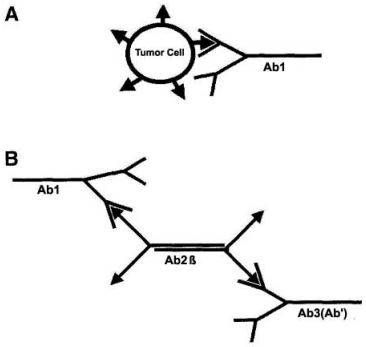


Figura 3.4: Red idiotípica. (A) Relación anticuerpo-antígeno (B) Configuración de una red inmune en cascada. Fuente:Preeti Gokal Kochar

Estado del Arte

Una revisión completa sobre la aplicación de sistemas inmunes artificiales se encuentra en [22]

Los algoritmos basados en redes inmunes tienen su origen en el trabajo de Farmer (1986) [7], el primer algoritmo basado en redes inmunes fue propuesto por en 1990 por Ishida [14] y finalmente en el año 2000 en [24, 18] refinan y reimplementan el algoritmo en lo que fue llamado AINE. Un algoritmo general para una red inmune es propuesto en [4] (ver algoritmo 1).

Al ser propuesta la teoría de las redes inmunes se generó mucho interés en crear modelos de esta red, de manera de explicar como funcionaba el sistema inmunológico natural. Una vez que estos trabajos se llevaron a cabo los investigadores en inteligencia computacional se interesaron en la utilización de estos modelos para la solución de problemas computacionales. Los primeros trabajos se basaron en un set de ecuaciones diferenciales que dictaban las variaciones en el tamaño de la población de anticuerpos, estos se pueden clasificar como modelos continuos de red inmune. Con posterioridad se propusieron modelos que no consideraban ecuaciones diferenciales, sino procesos de adaptación no iterativos o ecuaciones de diferencia, estos fueron clasificados como Modelos discretos de red inmune.

Son variados los algoritmos propuestos que están basados en las redes inmunes y las teorías acerca de su funcionamiento. El modelo de redes idiotípicas fue el primero en surgir, se basa

Algorithm 1 Algoritmo Génerico para una red inmune

1. Inicialización: inicializar un red de células inmunes
2. Bucle de población: Para cada antígeno
   (a) Reconocimiento de antígeno: calcular la estimulación de las células en la red con el antígeno
   (b) Interacciones de red: calcular estimulación de las células de la red sobre si mismas (otras células en la red)
   (c) Metadinámica: introducir nuevas células en la red y eliminar células inútiles en la red (bajo cierto criterio)
   (d) Nivel de estimulación: evaluar el nivel de estimulación de cada células de la red utilizado $S = N_{st} - N_{sup} + A_s$, donde N_{st} es la estimulación de la red sobre la célula, N_{sup} la supresión de la red sobre al célula y A_s corresponde a la estimulación por parte del antígeno
   (e) Dinámica de red: realizar actualización de la estructura de la red y parámetros de acuerdo a los niveles de estimulación
3. Ciclo: volver a paso 2 si no se ha cumplido el criterio de convergencia

en la teoría original de Jerne, suponiendo una red idiotípica global, en la cual las células pueden estimularse y suprimirse, afectando esto a la reproducción de las células B. Un algoritmo de este tipo es AINE el cual se basa en el comportamiento de las células B, incorpora ciertas ideas básicas de la teoría de selección clonal por lo que estas células son clonadas, mutadas y seleccionadas al ser simuladas en la red. Relacionado al anterior es AiNet propuesto en [3], el cual está diseñado para el aprendizaje. Otros modelos de red inmune son los de reconocimiento dinámico; basados en la interacción entre anticuerpos y antígenos, los algoritmos de modelo de memoria inmunológica asociativa; los cuales se basan en la idea que la memoria generada por el sistema inmune luego de controlar a los antígenos es asociativa.

Constraint Directed - Networked Artificial Immune System (CD-NAIS) es un algoritmo basado en Sistemas Inmunes Artificiales para la resolución de CSP binarios, fue propuesto por Zunñiga en [28]. El aporte es la definición de la creación de red inmune la cual a partir de un conjunto ordenado según su afinidad genera las células de memoria. Se selecciona secuencialmente los clones con mejor afinidad y que no han sido marcados para ser suprimidos. La eliminación de un clon se determina según la diferencia entre el clon y el clon seleccionado para ser parte de la memoria, su usa la distancia de Hamming para esto. Si el clon se parece mucho a la célula de memoria este es marcado para ser suprimido. Si faltan clones para el conjunto de memoria, no quedando clones sin marca de supresión, se completa con los de menor grado de supresión.

Consiste en:

- Presentación Antigénica: Determina la afinidad entre anticuerpos y antígenos
- Repetir hasta cumplir el criterio de termino
  - Selección Clonal: Se selecciona elementos del conjunto N, mediante el método de la ruleta. Donde N son los mejores anticuerpos
  - Expansión Clonal: Se generan clones de los elementos seleccionados proporcionales a su afinidad
  - Maduración de la Afinidad: Se mutan los clones inversamente proporcional a su afinidad

- Construcción de la Red Inmune: Se incorporan los nuevos individuos al conjunto de memoria evaluando los clones entre si para mantener diversidad
- Meta-dinámica: Reemplazar cierto número de individuos con baja afinidad.
- Mejor Solución: Actualizar el mejor individuo

Capítulo 4

Algoritmo Inmune para ALMA Array Scheluding Problem

4.1 Descripción

En este capítulo se presenta el algoritmo $aiNET_{asp}$, está basado en la teoría de red inmune, y busca resolver una simplificación del problema de Array Scheduling en ALMA . Este algoritmo sigue la estructura propuesta por De Castro en el algoritmo 1 y el modelo Optimization Artificial Immune Network (opt-aiNet). Este algoritmo es una implementación basada en las especificaciones realizadas por De Castro [5].

El modelo simplificado está basado en las especificaciones realizadas en [20].

4.2 Condiciones Generales y Supuestos

Dentro del ALMA Array Scheduling Problem solo se consideran arreglos que tengan una de las siguientes características:

Arreglo de Todas las Antenas: Este es el caso más simple, en donde hay un solo arreglo

con todas las antenas disponibles.

Arreglo Simple: En este caso un arreglo simple es usado, pero con un número variable de antenas.

Arreglos Múltiples: Cada nuevo arreglo puede ser considerado como un nuevo telescopio, además cada nueva combinación en una nueva instancia que no necesariamente sea equivalente a la anterior.

La instancia que se usa en este trabajo es el arreglo de todas las antenas. Por lo tanto, el scheduler global es el mismo que el scheduler del arreglo.

En orden a mantener la integridad del problema analizado, se asumen las siguientes consideraciones:

- Todas las antenas funcionan como un solo instrumento que no varian en toda la temporada.
- Los trabajos que van a ser ejecutados son llamados scheduling blocks (SB), los cuales pertenecen a un proyecto de observación. Pueden existir restricciones de precedencia entre scheduling blocks dentro de un mismo proyecto. Un proyecto adquiere el estado de completado solo si todos sus scheduling blocks han sido correctamente ejecutados, además de debe asegurar que éstos sean ejecutados una única vez.
- El tiempo total para un SB puede ser calculado como: calibraciones + on-source pointing + exposición de adquisición de datos.
- Tiempo de exposición de adquisición de datos depende, en realidad, del logro del objetivo de sensibilidad, el cual puede variar dependiendo de las condiciones atmosféricas. En este trabajo el tiempo de exposición se considera como un parámetro estático el cual es conocido al inicio del periodo de observación.
- Todos los proyectos aprobados por el comité de asignación de tiempo de observación son conocidos al inicio de periodo de observación, además a ellos se les asigna un grado científico (es un parámetro que mide la importancia del proyecto).

Existen varios parámetros a considerar en el modelo de scheduling, los cuales se dividen en estáticos y dinámicos.

Parámetros estáticos:

- Grado científico, determinado por el comité se asignación de tiempo.
- Visibilidad del objetivo en el cielo, incluye zonas excluyentes por efecto de la posición del Sol o la Luna, la cual depende de las coordenadas terrestres desde donde las observaciones son realizadas. Esto determina en que periodos de tiempo los SBs son factibles de ser ejecutados inicialmente.
- Tiempo de observación disponible durante el periodo factible, incluyendo bloques de mantención y bloques reservados.
- Tiempo total de ejecución.

Parámetros dinámicos:

- Condiciones de clima on-line, incluyen: temperatura, humedad, opacidad y velocidad y dirección del viento.
- Datos históricos de clima y pronósticos climáticos. Esto determina un patrón de clima esperado y deriva en una probabilidad de observación.
- Porcentaje de completitud de un proyecto y dependencias de secuencia de scheduling blocks.
- Tiempos de caída no planificados, no considerados en la entrada estática.

Proyectos y Scheduling Blocks

Un proyecto puede contener uno o más scheduling blocks, los cuales son la unidad atómica de observación. Un proyecto es independiente de los otros proyectos y sus scheduling blocks.

Generalmente, los scheduling blocks de un mismo proyecto son independientes entre sí, pero pueden existir dependencias opcionales llamadas sequence blocks. En el caso de existir dependencias entre los scheduling blocks ésta en solo dentro del mismo proyecto de observación. Si no existe dependencias los scheduling blocks se ejecutan en cualquier orden. Lo principal es que todos los scheduling blocks de un proyecto sean ejecutados correctamente porque se busca tener la mayor cantidad de proyectos de observación completados.

4.3 Modelo Simplificado

Proyectos y Scheduling Blocks

- P_i : Proyecto de observación i.
- $SciGr_i$: Grado científico para un proyecto de observación P_i.
- SB_{ij} : Scheduling blocks j perteneciente al proyecto P_i.
- $ExecT_{ij}$: Tiempo de ejecución para SB_{ij}.

Variables Dinámicas

- $S(SB_{ij})$: Estado del SB_{ij} (no ejecutado = 0, ejecutado = 1).
- $S(P_i)$: Estado de completación del proyecto P_i, basado sobre el estado individual de cada SB, $S(P_i) \; = \; 1 \; \Leftrightarrow \; S(SB_{ij}) \, \forall j$.

Función Objetivo

Se busca maximizar el valor científico de los proyectos completados (ecuación 4.1).

$$MAX \sum_{i=1}^{NumP} SciGr(P_i) * S(P_i) \tag{4.1}$$

Además, se tiene como objetivo alterno maximizar la proporción de tiempo usado, es decir, minimizar el tiempo ocioso (ecuación 4.2).

$$MAX \frac{\sum_{i=1}^{NumP} \sum_{j=1}^{NumSB} ExecT_{ij} * S(SB_{ij})}{T} \tag{4.2}$$

Restricciones

- Un arreglo puede ejecutar un solo scheduling blocks a las vez.
- La duración de un scheduling block no puede superar los 30 [min.]:

$$ExecT_{ij} \leq 30[min.]$$

- Los espacios de tiempo (TimeSlot) tienen una duración de 30 [min].
- Un scheduling block no puede ser ejecutado si no cumple las condiciones de disponibilidad para un TimeSlot.
- El criterio de término es alcanzar el total de SB ejecutados o alcanzar el fin de la temporada.
- Para este modelo no existen dependencias de secuencia.
- El algoritmo propuesto se ejecuta en ventanas de tiempo de un dia (48 Timeslot).

4.4 Algoritmo $aiNET_{asp}$

4.4.1 Representación

La representación se basa en claves aleatorias. Consiste en un vector con valores de dos decimales entre 0 y 1 que representan a cada SB, los cuales están almacenados en un arreglo del mismo tamaño. Cada valor en el vector está asociado al SB en el arreglo en el mismo orden. El vector es la secuencia de ejecución de los SB. Para establecer la secuencia se ordenan los valores del vector de forma ascendente (Tabla 4.1).

Tabla 4.1: Ejemplo de representación de una secuencia de ejecución

Arreglo_SB	1	2	3	4	5
Vector	0.49	0.91	0.33	0.75	0.51
Sequencia_SB	3	1	5	4	2
Vector Ordenado	0.33	0.49	0.51	0.75	0.91

4.4.2 Función de Evaluación

El procedimiento consiste ordenar las soluciones según su clave y ejecutar los SB según ese orden. Luego, se verifica que el SB esté disponible en el TimeSlot actual en caso contrario se continua con el siguiente TimeSlot, es decir, se avanza hasta que el SB esté disponible para ser ejecutado. Si el SB es ejecutado se continua con el siguiente SB en la lista. Para determinar la calidad de la solución se utiliza la ecuación 4.3, consiste en sumar los aportes en grado científico de cada SB de un proyecto i; la idea es agregar una porción de grado científico por cada SB ejecutado, esta porción se calcula dividiendo el grado científico del proyecto i correspondiente por la cantidad de SB del mismo.

$$Fitness = \sum_{i=1}^{NumP} \sum_{j=1}^{NumSB} \frac{SciGr(P_i)}{numSB(P_i)} * S(SB_{ij}) \tag{4.3}$$

4.4.3 Algoritmo

$aiNET_{asp}$ en un algoritmo, basado en la teoría de red inmune, implementado para resolver una simplificación del Array Scheduling Problem en ALMA, aplicado en ventanas de tiempo. En el algoritmo 2 se describen los principales componentes basado en el trabajo de De Castro [5]. Este algoritmo es un híbrido que mezcla un algoritmo CONALG y un algoritmo de red inmune.

Características generales de la heurística:

Algorithm 2 Algoritmo $aiNET_{asp}$

Require: $Population_{size}$, N_{clones}, N_{random}, AffinityThreshold

Ensure: S_{best}

```
Population ← InitializePopulation(Population_size)
while ¬StopCondition() do
    EvaluatePopulation(Population)
    S_best ← GetBestSolution(Population)
    Progeny ← ∅
    for Cell_i ∈ Population do
        Clones ← CreateClones(Cell_i, N_clones)
        for Clone_i ∈ Clones do
            Clone_i ← MutateRelativeToFitnessOfParent(Clone_i, Cell_i)
        end for
        EvaluatePopulation(Clones)
        Progeny ← GetBestSolution(Clones)
    end for
    Progeny ← CreateRandomCells(N_random)
    Population ← Progeny
end while
return (S_best)
```

- La cantidad de Mutaciones de los clones es proporcional al fitness de los padres (mayor fitness, menor cantidad de mutación).

- La adición de soluciones aleatorias le da capacidad de diversidad al algoritmo.

- El mecanismo elitista de selección del mejor clon es un mecanismo de reducción de la redundancia.

- El tamaño de la población es estático, además es elitista al conservar a la mejor solución del población anterior.

- La población final en cada iteración consiste en la mezcla del mejor de la población anterior, los mejores clones generados en la mutación y el reemplazo de las N peores soluciones.

Inicialización

La creación de la población inicial es un proceso importante ya que ellas constituyen el punto de partida de la búsqueda. Es deseable obtener soluciones de buena calidad sin un alto costo computacional de manera de centrar la búsqueda en el algoritmo inmune. En base a las características de la representación se optó por crear una población inicial de forma aleatoria.

Selección Clonal

El modelo planteado por de Castro consiste en una mezcla de un algoritmo CLONALG y la teoría de red inmune, en el cual se crea la misma cantidad de clones por cada individuo de la población, además, se considera a toda la población para ser clonada.

Por cada población de clones se elige al mejor individuo, el cual pasa a la nueva población. Con esta selección elitista se reduce la redundancia de soluciones debido al proceso de clonación.

Hipermutación

La hipermutación consiste aplicarle una ligera modificación en la codificación de cada clon generado. Se recorre todas las claves de los individuos, luego, mediante un valor aleatorio se determina si la clave actual es candidata a ser mutada según una probabilidad de mutación que es inversamente proporcional al fitness de padre. Si el número aleatorio es inferior a la probabilidad de mutación la clave se reemplaza por un valor aleatorio. Un ejemplo con una probabilidad de mutación de 0.125 se muestra en el Tabla 4.2.

Tabla 4.2: Ejemplo de Hipermutación con probabilidad de mutación de 0.125

Probabilidad	0.14	0.20	0.60	0.05	0.90
Padre	0.57	0.93	0.36	0.12	0.78
Hijo	0.57	0.93	0.36	0.82	0.78

Para la determinación de la probabilidad de mutación se establece una función dada por la ecuación 4.4. El parámetro f_n corresponde al fitness normalizado de toda la población actual, este valor se calcula en cada iteración. Si el fitness del padre es alto el ratio de mutación es bajo, manteniendo las condiciones de un sistema inmune artificial. El valor constante fue determinado de forma experimental.

$$mutation\_rate = 2{,}5 * e^{-f_n} \tag{4.4}$$

Supresión

Esto es el proceso que caracteriza a una red inmune artificial. Primero, se busca determinar la afinidad que existe entre los individuos posterior al proceso de hipermutación, luego se pasa a un proceso de eliminación de aquellas soluciones que son muy similares entre si, con lo cual se elimina redundancia y proporciona al algoritmo un mecanismo de diversidad.

Se implementa la elección del mejor clon de cada individuo como un mecanismo de supresión. Como los clones son ligeramente parecidos luego de la hipermutación, la elección del mejor reduce la posibilidad de tener soluciones parecidas.

Selección del Repertorio

Al finalizar la hipermutación se inicia un proceso de selección, el cual tiene como objetivo definir el repertorio que pasará a la siguiente iteración. Esta selección se realiza en base a la calidad de las soluciones, manteniendo en el repertorio los mejores individuos a los que puede acceder en la iteración actual, ya sean clones o miembros actuales del repertorio. Este operador ejerce una gran presión de selección, favoreciendo la convergencia del algoritmo y por lo tanto intensificando la búsqueda en una zona en particular. A medida que se avanza en la búsqueda, los mejores individuos generarán clones con configuraciones y funciones objetivos más cercanas a un óptimo local global.

El operador de selección causará que el algoritmo converja rápidamente a un repertorio poco diverso y no será capaz de salir de esta situación, experimentando el fenómeno de convergencia prematura. Entonces, se provee de nuevos individuos al repertorio utilizando el reemplazo de individuos, este procedimiento es importante debido a que favorece la exploración, insertando nuevos puntos de partida para la búsqueda. El porcentaje de remplazo es un parámetro de entrada del algoritmo.

Capítulo 5

Validación de la Propuesta

5.1 Configuración

La configuración inicial de la solución del Array Scheduling Problem está compuesta por dos tipos: la configuración del problema y la configuración del $aiNet_{asp}$. Las configuraciones se dan en las tabla 5.1 y tabla 5.2 respectivamente.

Tabla 5.1: Configuración del Problema

Parámetros	Valor
Duración TimeSlot	1800 s
Ventana de Tiempo	48 timeslots
Máxima cantidad SBs por Ventana de Tiempo	48 SBs

5.1.1 Sintonización

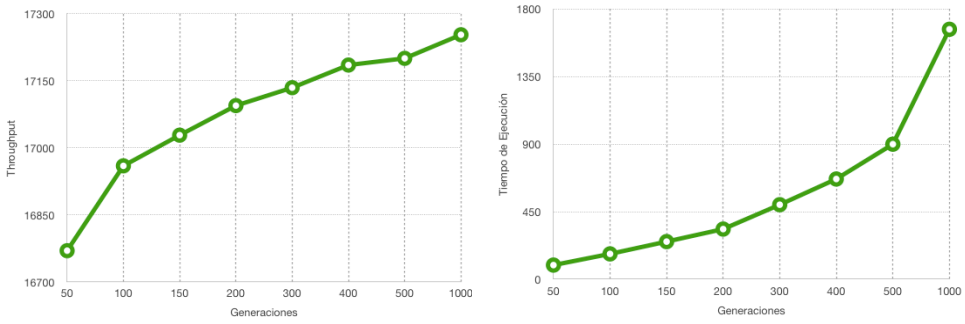
Para establecer los parámetros de la Tabla 5.2 se realizó una sintonización manual, la cual consistió en probar valores de forma independiente por cada parámetro. La configuración final se estableció en base a una mezcla entre los valores encontrados sujetos a una restricción

Tabla 5.2: Configuración de $aiNet_{asp}$

Parámetros	Valor
Generaciones	300
Tamaño de Población	8
Número de Clones	30
% Reemplazo	0.2

dura de tiempo de ejecución, éste establece que la duración máxima del algoritmo no debe superar los 15 minutos.

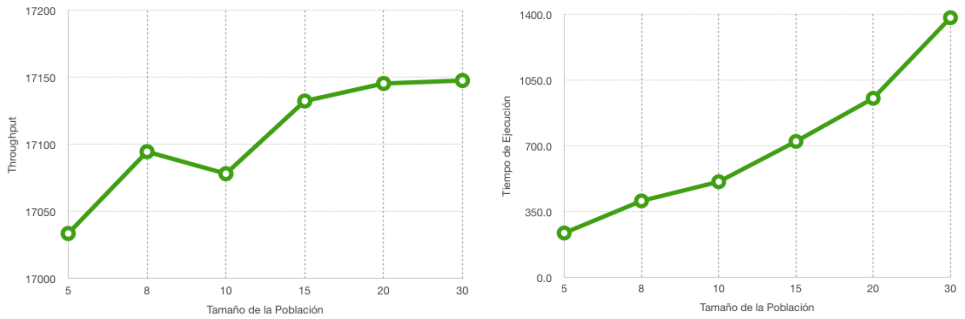
Los siguientes gráficos muestran las pruebas para cada parámetro del $aiNet_{asp}$: cantidad de iteraciones (figura 5.1), tamaño de la población (figura 5.2), cantidad de clones por individuo (figura 5.3) y porcentaje de reemplazo (figura 5.4).



(a) Throughput (b) Tiempo de Ejecución

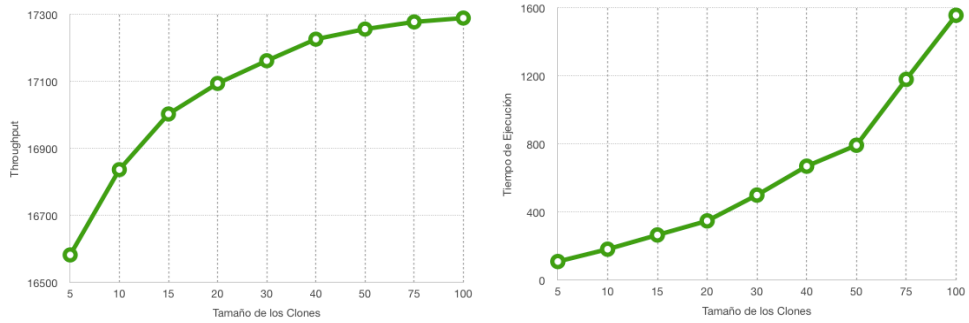
Figura 5.1: Sintonización cantidad de iteraciones

Se puede mencionar que debido a la restricción de los 15 minutos hay una baja en el desempeño del algoritmo. Si se tuviera más tiempo se puede mejorar el desempeño, eso se ve en los gráficos en donde las soluciones siguen mejorando pero el tiempo se vuelve exponencial excepto en el caso del porcentaje de reemplazo.



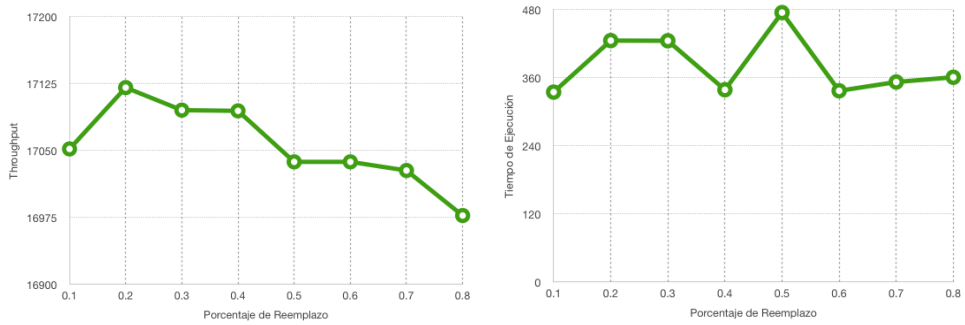
(a) Throughput (b) Tiempo de Ejecución

Figura 5.2: Sintonización tamaño de la población



(a) Throughput (b) Tiempo de Ejecución

Figura 5.3: Sintonización cantidad de clones por individuo



(a) Throughput (b) Tiempo de Ejecución

Figura 5.4: Sintonización porcentaje de reemplazo

5.2 Diseño del Algoritmo

La representación de la solución permite un bajo uso de memoria, al solo considerar un vector con datos de posición y solo se accede a los datos mediante punteros. Por lo tanto, el vector de SBs y proyectos no necesitan ser ordenados y solo se ordenan sus identificadores como una secuencia de ejecución.

En la figura 5.5 se muestra un diagrama de clases simplificado de modelo de clases del algoritmo $aiNet_{asp}$.

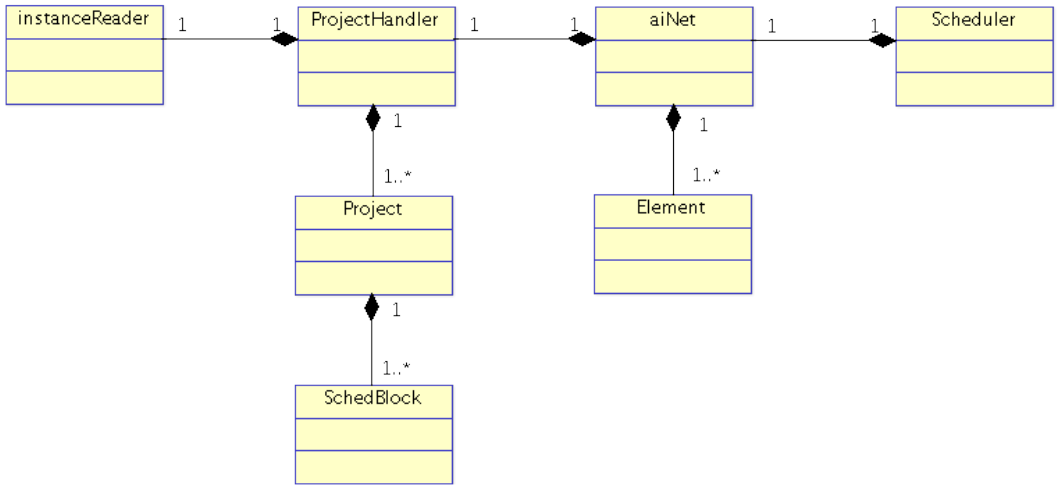


Figura 5.5: Diseño simplificado de clases de la implementación

A continuación se dan las descripciones de cada clase:

instanceReader : Clase que lee los datos de los Proyectos y SchedBlocks.

ProjectHandler : Usa la clase instanceReader para leer los datos de entrada que están en formato de texto plano y crea las instancias de las clases SchedBlock y Project. Provee acceso a atributos de proyectos y datos generales.

Project : Representa a un proyecto de observación. Permite el acceso a los datos de los schedblocks del proyecto.

SchedBlock : Representa a un SchedBlock, asociado una instancia de proyecto de observación padre. Contiene atributos relevantes para su ejecución.

aiNet : Implementación del algoritmo de red inmune. Se le asocia una instancia de ProjectHandler para obtener los datos para la planificación. Retorna un mapa con la planificación de los proyectos de observación.

Element : Implementación de un individuo para el aiNet. Contiene la representación de la secuencia de ejecución de los SchedBlocks en una ventana de tiempo y el fitness asociado.

Scheduler : Esta es la clase principal. Se comunica únicamente con el algoritmo. Proporciona los mecanismos para la obtención y representación de las soluciones.

El periodo de inicialización se ejecuta una única vez al comienzo de la ejecución, y considera la creación y la inicialización de los objetos necesarios para la ejecución:

1. $Scheduler$ crea una instancia de $aiNet$ y todos sus métodos de inicialización.
2. $aiNet$ crea e inicializa $ProjectHandler$.
3. $ProjectHandler$ accede a los datos de entrada de los proyectos de observación y crea las instancias de $Project$ y sus instancias asociadas de $SchedBlock$.
4. $aiNet$ se inicializa y configura los sus parámetros de ejecución.
5. $aiNet$ crea las instancias de la población de $Element$.
6. Finalement, $aiNet$ inicia la búsqueda y entrega los resultados a $Scheduler$.

El algoritmo propuesto va creando la solución de forma iterativa. La temporada se divide en trozos de tiempo denominados ventanas de tiempo, cada ventana tiene una duración de 1 día o 24 horas. Las ventanas de tiempo están divididas en TimeSlots, los cuales tienen una duración de media hora. Por lo tanto, cada ventana de tiempo está compuesta pos 48 TimeSlot. para el problema se asume que cada SB tiene un tiempo de ejecución igual a un TimeSlot.

El algoritmo $aiNet_{asp}$ se encarga de determinar la mejor secuencia de ejecución para cada ventana de tiempo basado en la disponibilidad de SB. La mejor configuración, para la ventana de tiempo, se determina en base al fitness obtenido mediante la maximización del aporte de grado científico de los SB ejecutados (ecuación 5.1).

$$Fitness = \sum_{j} \sum_{i} \frac{SciGr(P_i)}{numSB(P_i)} * S(SB_{ij}) \tag{5.1}$$

En cada ventana de tiempo $aiNet_{asp}$ proporciona una secuencia de ejecución con un máximo tamaño que es igual a la cantidad de TimeSlot disponibles. Para determinar la disponibilidad de los SB se consulta a la lista de TimeSlots asociada, ésta contiene los identificadores de los TimeSlot en los que el SB puede ser ejecutado.

Al momento de la evaluación de la secuencia se presume que el orden de ejecución es inalterable, es decir, se avanza entre los TimeSlots hasta que el SB sea ejecutado y no sobrepase el tamaño máximo de la ventana de tiempo. Una vez que un SB es correctamente ejecutado se pasa al siguiente SB en la lista y se actualiza el tiempo. Para la siguiente ventana de tiempo los SBs ejecutados son marcados y a los restantes SB sin ejecutar se le quitan los TimeSlot ya pasados, de esta forma se reduce el tamaño de las estructuras que están en la memoria y se acelera la ejecución. El criterio de término del $aiNet_{asp}$ es alcanzar el máximo de generaciones, en cambio, el algoritmo de construcción avanza por las ventanas de tiempo hasta completar la cantidad total de SBs ejecutados o alcanzar el fin de la temporada.

Datos de Entrada

En un comienzo para probar el algoritmo se utilizó los datos de entrada proporcionados por el ALMA Planning Model Simulator [12], y que fue utilizado por M. Mora en su tesis [20]. Principalmente considera: datos de proyectos de observación creados para 2010/2011 ALMA preliminary end-to-end tests.

En la tabla 5.3 se muestran las características generales de estos datos de entrada.

La cantidad de SchedBlocks es alrededor de un 10 % del total de tiempo disponible en un

Tabla 5.3: Configuración de los datos de entrada usados en [20]

Cantidad de Proyectos	463
Cantidad de Scheduling Blocks	950
Promedio de Duración de Scheduling Blocks	181.832 [seg.]
Fecha de Inicio/Término	01.09.2011/31.05.2012 (273 días)
Coordenadas del Centro del Arreglo	-23.0229° Lat., -67.7549° Long.

periodo estándar de 6 meses, y se espera tener un tiempo promedio de ejecución de 30 [min.]. En consecuencia, esta instancia desde el punto de vista de las observaciones no representa exactamente las condiciones de complejidad de una planificación real.

Por lo anterior de analizó el contenido de los datos de las propuestas de proyectos del el ciclo 0 y ciclo 1, y se realizaron modificaciones para poder crear un conjunto de pruebas que fuese más representativo del problema real en el futuro.

Se crearon dos configuraciones una de 5.000 SBs en una temporada de 3 meses y otra de 10.000 SBs y una temporada de 6 meses. La cantidad de proyectos es la mitad de la cantidad de SB y por lo tanto cada proyecto tiene 2 SBs. En la situación real pueden existir dependencias entre los SBs, pero para esta versión no se considera que existan dependencias.

Para establecer la lista de TimeSlot en los que el SB puede ejecutarse se establece dos criterios:

1. ciclo 0: cantidad de TimeSlot determinada por la ecuación 5.2.

$$f(n) = \begin{cases} random(1; 0{,}2 * totalSlot) & \text{if } n \leq 0{,}2 \\ random(0{,}8 * totalSlot; totalSlot) & \text{if } n \geq 0{,}8 \\ 0{,}7 * timeSlot & \text{e.c.o.c.} \end{cases} \tag{5.2}$$

2. ciclo 1: cantidad de TimeSlot aleatoria entre 1 y el total de TimeSlot de la temporada.

Además por cada configuración se utilizaron 5 semillas distintas para los números aleatorios. Finalmente se establecieron 20 infancias para realizar las pruebas.

Los datos de entrada están en texto plano y tienen la siguiente estructura:

```
1 ---- General data ----
  numProject TotalTime
3 ---- Project 0 ----
  numSB SciGrade
5 -- SchedBlock 0 --
  SBTimeExecution numTimeSlot
7 - time slots -
  TimeSlot_0
9 ...
  - predecessors -
11 0
  -- SchedBlock 1 --
13 ...
  ---- Project 1 ----
15 ...
```

En las línea 2 se entrega los valores generales de la instancia , los cuales son el número total de proyectos para la temporada y el total de tiempo disponible (en segundos). A continuación, se especifican los características de cada proyecto, que son la cantidad total de SB que tiene el proyecto y el grado científico del mismo (línea 4); luego se entrega la lista con las características de los SB del proyecto, compuesto por el tiempo máximo de ejecución (en segundos) y cantidad de TimeSlot en los cuales esta disponible (línea 6). Luego, se prosigue con la lista de los identificadores de los TimeSlot (línea 8 y 9). Finalmente, se entrega la lista de SB predecesores, que este modelo no se considera (línea 11). La línea 12 marca el comienzo del siguiente SB del proyecto y la línea 14 marca el comienzo del siguiente proyecto.

5.3 Resultados

El algoritmos implementado fue programado en C/C++.

Se realizaron experimentos en un computador con procesador Intel(R) Core(TM)2 Duo CPU E7400 2.80 GHz con 3.0 GB de memoria RAM, corriendo bajo linux 2.6.32-5-686 (Debian Squeeze). Estas pruebas se realizaron para probar la entrada que usó M. Mora [20].

Por las características y volumen de los datos de entrada se logra el 100 % de ejecución de los proyectos. Una comparación de tiempo con el algoritmo de M. Mora se muestra en la tabla 5.4.

Tabla 5.4: Comparación entre $aiNet_{asp}$ y Prioridades Dinámicas

	$aiNet_{asp}$	Dynamic Priorities
Periodo de Inicialización	8.365 [s]	11.939 [s]
Tiempo de Ejecución	60.344	24.831 [s]

Las pruebas realizadas con las instancias generadas se ven en las tablas 5.5, 5.6, 5.7, 5.8. Se dividen por su configuración y por el tipo de ciclo con que se creó la lista de TimeSlot. Para éstas pruebas se usó un MacBook Pro 13-inch (Early 2011) con un procesador Intel Core i5 de 2.3 GHz de 4 GB de memoria ram. Para mayor detalle ir a la sección Apéndice B.

Tabla 5.5: Pruebas para 2500 proyectos y 5000 SBs en ciclo 0

Instancia	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_1	8504.325	0.9721552	0.9644179	4164.5	0.83292	6	188.2 [s]
test_2	8530.629	0.9680412	0.9604862	4149.3	0.82986	6.1	181.7 [s]
test_3	8397.464	0.9726025	0.9641666	4165.2	0.83304	4.2	186.2 [s]
test_4	8632.425	0.9681154	0.96533034	4168.5	0.8337	7.9	181.2 [s]
test_5	8451.49	0.9712056	0.96689394	4176.1	0.83522	5.9	185.4 [s]

Tabla 5.6: Pruebas para 2500 proyectos y 5000 SBs en ciclo 1

Instancia	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_1	8310.134	0.9639523	0.9349997	4039.2	0.80784	10.6	189.1 [s]
test_2	8400.391	0.9603186	0.9353009	4130.5	0.8081	9.3	187.5 [s]
test_3	8328.384	0.9609834	0.9321758	4027	0.81008	11.6	190.1 [s]
test_4	8372.607	0.96397011	0.937361	4049.4	0.80988	8.6	188.6 [s]
test_5	8446.824	0.9608971	0.9357607	4038.6	0.80772	9.8	188.8 [s]

5.4 Discusión

Se probó el algoritmo con lo propuesto por M. Mora [20]. Para la medida de desempeño se obtuvo que se cumplió con un 100 % de ejecución de los proyectos, con lo cual se obtiene un

Tabla 5.7: Pruebas para 5000 proyectos y 10000 SBs en ciclo 0

Instancia	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_1	17233.85	0.9715977	0.9683218	8366.3	0.83663	6.1	773.6 [s]
test_2	17021.94	0.9731023	0.9654167	8341.2	0.83412	7.8	735.3 [s]
test_3	17136.69	0.97226617	0.9640046	8329	0.8329	6.4	707.1 [s]
test_4	17137.38	0.9701427	0.9646296	8334.4	0.83434	5.6	700.3 [s]
test_5	16951.73	0.9708343	0.9624653	8315.7	0.83157	8.3	701.6 [s]

Tabla 5.8: Pruebas para 5000 proyectos y 10000 SBs en ciclo 1

Instancia	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_1	16917.27	0.9623612	0.9346644	8165.5	0.80755	9.7	723.8 [s]
test_2	16926.54	0.96193	0.93336	8064.2	0.80642	7.8	795.1 [s]
test_3	17036.61	0.9615326	0.9302662	8037.5	0.80375	9.1	688.7 [s]
test_4	16807.01	0.9623373	0.9354833	8074.8	0.80748	9.6	714.8 [s]
test_5	16654.89	0.9636835	0.935972	8086.8	0.80868	11	718.6 [s]

throughput máximo para cualquier planificación generada. Por lo tanto, esta instancia no es real en términos de cantidad de proyectos y SBs, pero sirve para probar que se está realizando un algoritmo que soluciona el problema. Aunque el $aiNet_{asp}$ demora el doble esto no es tan significativo.

Para solucionar lo anterior se crearon pruebas con datos simulados que son una representación del problema de array scheduling más real. En esta implementación solo consideran la variable de visibilidad. La idea es resumir todas las restricciones (clima, instrumentos, etc) en un solo valor de disponibilidad que indica cuando un SB puede ser ejecutado y así facilitar la búsqueda de una planificación adecuada. Esta aproximación es útil para generar una planificación a largo plazo, en el caso de corto plazo (un día) y se tenga que realizar una reparación a la planificación se debiese volver a calcular las restricciones y generar una nueva lista de disponibilidad por cada SB no ejecutado.

La propuesta de solución es escalable mediante la aplicación de ventanas de tiempo variables. Además, se puede variar la heurística que se aplique. Se puede implementar paralelismo al algoritmo $aiNet_{asp}$ y así mejorar el rendimiento al aumentar los valores de los parámetros de entrada.

El modelo $aiNet_{asp}$ es una derivación del modelo de prioridades dinámicas, el cual incluye el uso de algoritmos inmunes. La mejora que se implementa consiste en seleccionar la mejor configuración de scheduling blocks entre los disponibles para una ventana de tiempo, este mecanismo es elitista favoreciendo los SBs con alto grado científico para que ayude a maximizar la solución. En cambio, el modelo dinámico es una lista de ejecución ordenada por prioridades dinámicas. El modelo propuesto ayuda a concentrar la ejecución de SB al comienzo de la temporada .

Esta aproximación es buena para probar la simplificación del problema de array scheduling en ALMA, con lo cual se puede estimar si un algoritmo está bien encaminado para encontrar una solución para el problema general.

5.4.1 Análisis de Resultados

Los resultados entregados son valores promedios correspondientes a 10 ejecuciones para cada instancia de cada configuración.

La mayor diferencia entre las instancias es que en ciclo 0 hay un 60 % de instancias con un 70 % de disponibilidad, en cambio, en ciclo 1 la disponibilidad es lineal.

Para la configuración de 3 meses (tabla 5.5, tabla 5.6) se obtienen las siguientes características:

- Para la configuración con ciclo 0 se obtiene un rendimiento aproximado de 97 % y con ciclo 1 un rendimiento aproximado de 96 %, lo que indica que la configuración con ciclo 1 tiene más restricciones de disponibilidad.

- En relación al tiempo de ejecución se obtiene en promedio 184.54 segundos para ciclo 0 y 188.82 segundos para ciclo 1, se observa que en ciclo 1 se demora más. Esto se debe a que las instancias con ciclo 1 tienen menor disponibilidad de TImeSlots.

- Los mejores valores para los parámetros mostrados indican que no hay una relación entre ellos.

- Se observa que la proporción de uso de los SB no tiene una incidencia directa sobre los SB descartados.
- Por el tamaño de las instancias se tiene un periodo de lectura bajo (tabla 5.9).

Tabla 5.9: Tiempos promedios de lectura y ejecución para configuración de 3 meses

Ciclo 0	Tiempo de lectura	Tiempo de Ejecución	Ciclo 1	Tiempo de lectura	Tiempo de Ejecución
test_1	9.5 s	179.0 s	test_1	7.6 s	181.4 s
test_2	8.9 s	172.8 s	test_2	7.4 s	180.1 s
test_3	9.2 s	177.0 s	test_3	7.2 s	182.8 s
test_4	9.0 s	172.2 s	test_4	7.2 s	181.4 s
test_5	9.0 s	176.5 s	test_5	7.4 s	181.3 s

Para la configuración de 6 meses (tabla 5.7, tabla 5.8) se obtienen las siguientes características:

- Para la configuración con ciclo 0 se obtiene un rendimiento aproximado de 97 % y con ciclo 1 un rendimiento aproximado de 96 %, se mantiene las mismas proporciones que en ciclo 0.
- En relación al tiempo de ejecución se obtiene en promedio 723.58 segundos para ciclo 0 y 728.2 segundos para ciclo 1, se observa que en ciclo 1 se demora más. Los tiempos se cuadruplican a los de ciclo 0, pero la cantidad de proyectos SB solo se duplican.
- Los mejores valores para los parámetros mostrados nuevamente indican que no hay una relación entre ellos.
- Por el tamaño de las instancias se tiene un periodo de lectura alto por la densidad de los datos (tabla 5.10).

Se observa que entre las distintas variantes de las instancias para ciclo 0 y ciclo 1 los valores encontrados no varian mucho. Por otro lado, una mayor cantidad de SB ejecutados no indican que esto de un throughput mejor.

Tabla 5.10: Tiempos promedios de lectura y ejecución para configuración de 6 meses

Ciclo 0	Tiempo de lectura	Tiempo de Ejecución	Ciclo 1	Tiempo de lectura	Tiempo de Ejecución
test_1	38.4 s	735.2 s	test_1	29.4 s	694.5 s
test_2	38.0 s	697.3 s	test_2	32.2 s	762.9 s
test_3	37.2 s	669.9 s	test_3	29.0 s	660.0 s
test_4	35.8 s	664.5 s	test_4	28.8 s	686.0 s
test_5	36.1 s	665.5 s	test_5	28.8 s	690.0 s

Se puede tomar los resultados y establecer criterios que relacionen los distintos parámetros según las necesidades. Esto se puede realizar asignándole peso a los porcentajes de rendimientos de throughput, tiempo usado y SB ejecutados (ecuación 5.3).

$$results = A * \frac{Throughput}{TotalSciGrade} \; + \; B * \frac{TiempoUsado}{TiempoTotal} \; + \; C * \frac{SBsEjecutados}{TotalSBs} \quad (5.3)$$

Donde A, B, C son valores entre 0 y 1.

En el Apéndice B se encuentran los mejores valores encontrados para cada instancia.

Capítulo 6

Conclusiones

6.1 Resultados y Contribuciones

El sistema inmune natural es esencial para los organismos vivos, desempeña una función sumamente crítica y compleja que mantiene a los individuos protegidos del peligro exterior. Actualmente se dispone de un conjunto de teorías y modelos del funcionamiento del sistema inmune, pero quedan muchas interrogantes para la investigación.

Los sistemas inmunes artificiales son aplicados en diversas áreas como la detección de errores, seguridad informática, reconocimiento de patrones, optimización, entre otras. La aplicación de los sistemas inmunes artificiales a problemas de optimización combinatorial es bastante nueva, sin embargo los estudios realizados han demostrado la capacidad de estos sistemas para resolver este tipo de problemas. Este trabajo se enfoca en la resolución del problema de array scheduling en ALMA, usando sistemas inmunes artificiales.

Los algoritmos diseñados en este trabajo se inspiran en la teoría de red inmune, proceso mediante el cual el sistema inmune promueve la reproducción de anticuerpos de alta afinidad y supresión, debido a sus características, este tipo de algoritmos pueden ser clasificados como algoritmos evolutivos.

A través del desarrollo de este trabajo se aborda un nuevo problema con nuevos modelos de

algoritmos y se ve su utilidad para la solución de problemas de optimización. La implementación de red inmune analiza el problema simplificado y la solución mediante su aplicación en ventanas de tiempo de un día o 48 TimeSlot. Esta propuesta es una mezcla de algoritmo de búsqueda completa y una heurística del tipo evolutiva.

Se rescata que actualmente no hay soporte completo para scheduling dinámico en los observatorios. Los cuales en su mayoría realzan una planificación de forma manual. El modelo presentado es una aproximación, pero necesita una acuciosa revisión y modificación para llegar a ser implementado en el sub-sistema de scheduling.

El array scheduling problem se puede tratar como una derivación del job shop scheduling, y que considera nuevas restricciones.

La aplicación de lista de TimeSlot disponibles ayuda significativamente al desempeño de la solución, al entregar los SB que se pueden programa en un determinado tiempo.

No se usan prioridades dinámicas, pero el $aiNet_{asp}$ proporciona un configuración que maximiza el grado científico al considerar en primer lugar a los SB con mayor grado en cada ventana de tiempo. Esto ayuda a la concentración y ejecución de los SB con mayor grado científico de forma temprana.

Las instancias creadas nos permiten evaluar el desempeño del algoritmo. Por medio de las pruebas realizadas se observa que el $aiNet_{asp}$ tiene un desempeño aceptable porque esta limitado a la restricción de 15 minutos, si se pudiera contar con un mayor tiempo se podría obtener un resultado mejor.

6.2 Trabajo Futuro

El algoritmo implementado es eficaz para solucionar la instancia del problema, pero se ve que es necesario algunas modificaciones para mejorar su eficiencia y acercarse al problema real:

- Incluir un mecanismo que traduzca las restricciones de [20] en el formato que se utilizó.
- Se pude agregar las condiciones climáticas, pero éstas son muy variable y hacen muy difícil su pronóstico. Actualmente, en ALMA se considera que las condiciones son fijas a lo más por un día. Esto dificulta la planificación de una temporada. En consecuencia se tienen que realizar nuevas planificaciones a muy corto plazo.
- Los parámetros del algoritmo fueron configurados de forma manual. Aquí se pueden aplicar algoritmos de sintonización, como REVAC. Esto puede ayudar a mejorar el desempeño de algoritmo.
- Se puede realizar una estudio sobre las ventanas de tiempo para analizar cual sería una duración más apropiada para el problema.

Bibliografía

[1] D. S. Balser, C. Bignell, J. Braatz, M. Clark, J. Condon, J. Harnett, K. O'Neil, R. Maddalena, P. Marganian, M. McCarty, E. Sessoms, and A. Shelton. GBT Dynamic Scheduling System: Algorithms, Metrics, and Simulations. In D. A. Bohlender, D. Durand, & P. Dowler, editor, *Astronomical Data Analysis Software and Systems XVIII*, volume 411 of *Astronomical Society of the Pacific Conference Series*, pages 330–+, September 2009.

[2] F. M. Burnet. *The clonal selection theory of acquired immunity.* Nashville,Vanderbilt University Press,, 1959.

[3] L. N. de Casto and F. J. Von Zuben. An evolutionary immune network for data clustering. In *Neural Networks, 2000. Proceedings. Sixth Brazilian Symposium on*, pages 84–89, 2000.

[4] L. N. de Castro. Immune, swarm, and evolutionary algorithms. part i: basic models. In *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*, volume 3, pages 1464–1468 vol.3, 2002.

[5] L. N. de Castro and J. Timmis. An artificial immune network for multimodal function optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 699–704, 2002.

[6] L. N. de Castro and J. I. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, Berlin, Heidelberg, New York, 2002.

[7] J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation, and machine learning. *Physica D: Nonlinear Phenomena*, 22(1–3):187 – 204, 1986. Proceedings of the Fifth Annual International Conference.

[8] A. Farris and S. Roberts. Dynamic Scheduling in ALMA. In F. Ochsenbein, M. G. Allen, & D. Egret, editor, *Astronomical Data Analysis Software and Systems (ADASS) XIII*, volume 314 of *Astronomical Society of the Pacific Conference Series*, pages 1001–1004, July 2004.

[9] J. Frank. Sofia's challenge: automated scheduling of airborne astronomy observations. In *Space Mission Challenges for Information Technology, 2006. SMC-IT 2006. Second IEEE International Conference on*, pages 464–472, 2006.

[10] G. Giannone, A. M. Chavan, D. R. Silva, A. P. Krueger, and G. E. Miller. Long and Short Term Scheduling Tools in ESO. In N. Manset, C. Veillet, & D. Crabtree, editor, *Astronomical Data Analysis Software and Systems IX*, volume 216 of *Astronomical Society of the Pacific Conference Series*, pages 111–+, 2000.

[11] A. I. Gómez de Castro and J. Yáñez. Optimization of telescope scheduling. Algorithmic research and scientific policy. *aap*, 403:357–367, May 2003.

[12] A. Hoffstadt. Planning mode simulator: A simulation tool for studying alma's scheduling behaviour. Master's thesis, Universidad Técnica Federico Santa María, Departamento de Informática, Valparaíso, Chile, 2010.

[13] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, pages 592–623, 2005.

[14] Y. Ishida. Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 777–782 vol.1, 1990.

[15] N. K. Jerne. Towards a network theory of the immune system. *Annales d'immunologie*, 125C(1-2):373–389, January 1974.

[16] M. D. Johnston. Spike: Ai scheduling for nasa's hubble space telescope. In *Proceedings of the sixth conference on Artificial intelligence applications*, pages 184–190, Piscataway, NJ, USA, 1990. IEEE Press.

[17] M. D. Johnston and G. E. Miller. Spike: Intelligent scheduling of hubble space telescope observations. In *Intelligent Scheduling*, pages 391–422. Morgan Kaufmann Publishers, 1994.

[18] T. Knight and J. Timmis. Aine: an immunological approach to data mining. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 297–304, 2001.

[19] S. Lucero and A. Farris. Dynamic Scheduling in ALMA. In R. A. Shaw, F. Hill, & D. J. Bell, editor, *Astronomical Data Analysis Software and Systems XVI*, volume 376 of *Astronomical Society of the Pacific Conference Series*, pages 673–+, October 2007.

[20] M. Mora. A scheduling algorithm with dynamic priorities. Master's thesis, Universidad Técnica Federico Santa María, Departamento de Informática, Valparaíso, Chile, 2011.

[21] K. O'Neil, D. Balser, C. Bignell, M. Clark, J. Condon, M. McCarty, P. Marganian, A. Shelton, J. Braatz, J. Harnett, R. Maddalena, M. Mello, and E. Sessoms. The GBT Dynamic Scheduling System: A New Scheduling Paradigm. In D. A. Bohlender, D. Durand, & P. Dowler, editor, *Astronomical Data Analysis Software and Systems XVIII*, volume 411 of *Astronomical Society of the Pacific Conference Series*, pages 147–+, September 2009.

[22] L. Pérez. Diseño de un algoritmo inmune artificial para resolver traveling tournament problem. Master's thesis, Universidad Técnica Federico Santa María, Departamento de Informática, Valparaíso, Chile, 2011.

[23] D. R. Silva. Service mode scheduling at the eso very large telescope. In *Proceedings SPIE 4844*, page 94, 2002.

[24] J. Timmis, M. Neal, and J. Hunt. An artificial immune system for data analysis. *Biosystems*, 55(1–3):143 – 150, 2000.

[25] F. Varela, A. Coutinho, B. Dupire, and N. N. Vaz. Cognitive networks: immune, neural and otherwise. *Theoretical immunology*, 2:359–375, 1988.

[26] A.M. Whitbrook, U. Aickelin, and J.M. Garibaldi. Idiotypic immune networks in mobile-robot control. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 37(6):1581 –1598, dec. 2007.

[27] M. C. H. Wright. Dynamic scheduling. Technical report, Radio Astronomy Laboratory, University of California, Berkeley, 1999.

[28] M. Zuñiga. Un sistema inmune artificial para la resolución de problemas de satisfacción de restricciones. Master's thesis, Universidad Técnica Federico Santa María, Departamento de Informática, Valparaíso, Chile, 2004.

Apéndice A

Diagrama de Clases de $aiNet_{asp}$

Diagrama de clases del código del algoritmo A.1. Es un forma más detallada que la mostrada en la figura 5.5.

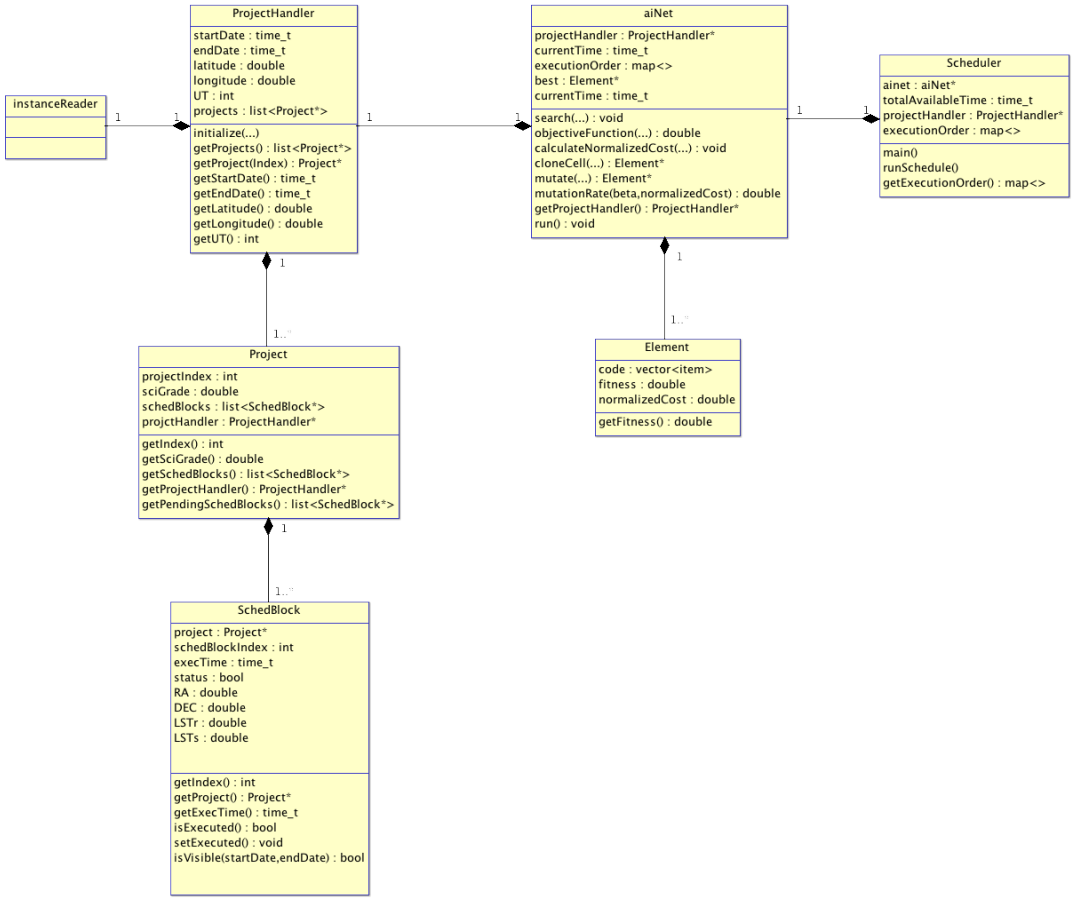


Figura A.1: Diseño de clases de la implementación

Apéndice B

Resultados del Algoritmo

Tablas con todas las pruebas realizadas.

Tabla B.1: 2500 proyectos con 5000 SB en ciclo 0 en semilla 1

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8506.13	0.972361	0.964583	4167	0.8334	7	189.329
test_1	8499.28	0.971578	0.9625	4158	0.8316	8	190.174
test_2	8507.96	0.972571	0.966667	4176	0.8352	6	186.585
test_3	8500.98	0.971773	0.962963	4160	0.832	8	188.545
test_4	8511.46	0.972971	0.965471	4171	0.8344	4	188.725
test_5	8498.62	0.971503	0.961343	4153	0.8306	7	187.596
test_6	8513.09	0.973158	0.9713	4178	0.8356	4	187.563
test_7	8501.71	0.971856	0.964815	4168	0.8336	8	186.764
test_8	8493.13	0.970875	0.958565	4141	0.8282	3	188.132
test_9	8510.89	0.972906	0.965972	4173	0.8346	5	188.593
promedio	8504.325	0.9721552	0.9644179	4164.5	0.83292	6	188.2006
sd	6.57147	0.00075	0.00343	11.52051	0.00232	1.88562	1.11385

Tabla B.2: 2500 proyectos con 5000 SB en ciclo 0 en semilla 2

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8529.02	0.967858	0.959491	4145	0.829	5	181.965
test_1	8533.41	0.968357	0.961574	4154	0.8308	6	182.604
test_2	8525.12	0.967416	0.958333	4140	0.828	6	181.19
test_3	8518.66	0.966683	0.957176	4135	0.827	9	184.727
test_4	8525.45	0.967454	0.959028	4143	0.8286	7	183.997
test_5	8532.91	0.9683	0.96088	4151	0.8302	5	180.193
test_6	8536.89	0.968751	0.962269	4157	0.8314	5	180.167
test_7	8543.8	0.969536	0.964815	4168	0.8336	4	180.572
test_8	8525.07	0.967411	0.958796	4142	0.8284	8	179.873
test_9	8535.96	0.968646	0.9625	4158	0.8316	6	182.015
promedio	8530.629	0.9680412	0.9604862	4149.3	0.82986	6.1	181.7303
sd	7.36321	0.00084	0.00233	10.06700	0.00201	1.52388	1.66360

Tabla B.3: 2500 proyectos con 5000 SB en ciclo 0 en semilla 3

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8394.43	0.972252	0.962731	4159	0.8318	5	185.543
test_1	8391.06	0.971861	0.96088	4151	0.8302	3	185.115
test_2	8397	0.972548	0.965741	4172	0.8344	6	183.829
test_3	8397.85	0.972647	0.965046	4169	0.8338	3	187.736
test_4	8401.36	0.973054	0.965509	4171	0.8342	3	191.518
test_5	8397.8	0.972642	0.963889	4164	0.8328	4	189.134
test_6	8397.97	0.972661	0.963889	4164	0.8328	4	183.765
test_7	8400.05	0.972902	0.965509	4171	0.8342	5	183.148
test_8	8396.53	0.972494	0.963657	4163	0.8326	5	184.446
test_9	8400.59	0.972964	0.964815	4168	0.8336	4	188.202
promedio	8397.464	0.9726025	0.9641666	4165.2	0.83304	4.2	186.2436
sd	3.04522	0.00035	0.00151	6.52857	0.00131	1.03280	2.76463

Tabla B.4: 2500 proyectos con 5000 SB en ciclo 0 en semilla 4

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8632.42	0.968124	0.9688124	4168	0.8336	8	182.097
test_1	8631.18	0.967986	0.964583	4167	0.8334	9	182.904
test_2	8640.53	0.969035	0.96713	4178	0.8356	6	180.218
test_3	8633.75	0.968274	0.965278	4170	0.834	8	183.995
test_4	8630.09	0.967864	0.963657	4163	0.8326	7	183.076
test_5	8627.49	0.967472	0.96412	4165	0.833	9	181.979
test_6	8641.87	0.969185	0.96875	4185	0.837	7	181.611
test_7	8632.46	0.968129	0.964815	4168	0.8336	8	180.962
test_8	8624.61	0.967249	0.962269	4157	0.8314	9	178.309
test_9	8629.85	0.967836	0.963889	4164	0.8328	8	177
promedio	8632.425	0.9681154	0.96533034	4168.5	0.8337	7.9	181.2151
sd	5.33039	0.00061	0.00220	7.90569	0.00158	0.99443	2.18154

Tabla B.5: 2500 proyectos con 5000 SB en ciclo 0 en semilla 5

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8455.74	0.971694	0.9677824	4181	0.8362	5	181.867
test_1	8456.95	0.971833	0.969676	4189	0.8378	5	181.284
test_2	8446.25	0.970603	0.965741	4172	0.8344	6	199.764
test_3	8452.25	0.971293	0.966667	4176	0.8352	6	202.27
test_4	8449.52	0.970979	0.966204	4174	0.8348	8	186.1
test_5	8454.21	0.971518	0.968287	4183	0.8366	9	179.546
test_6	8451.94	0.971258	0.96898	4177	0.8354	7	182.652
test_7	8438.07	0.969663	0.960648	4150	0.83	4	180.025
test_8	8453.41	0.971427	0.966435	4175	0.835	3	179.977
test_9	8456.56	0.971788	0.968519	4184	0.8368	6	180.84
promedio	8451.49	0.9712056	0.96689394	4176.1	0.83522	5.9	185.4325
sd	5.74339	0.00066	0.00255	10.56672	0.00211	1.79196	8.44419

Tabla B.6: 2500 proyectos con 5000 SB en ciclo 1 en semilla 1

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8315.49	0.964574	0.9375	4050	0.81	14	209.958
test_1	8296.49	0.962369	0.931944	4026	0.8052	10	189.888
test_2	8312.71	0.964251	0.936574	4046	0.8092	10	193.238
test_3	8311.35	0.964093	0.934491	4037	0.8074	11	188.596
test_4	8303.99	0.963239	0.933102	4031	0.8062	9	186.846
test_5	8303.65	0.9632	0.93333	4032	0.8064	10	183.459
test_6	8314.84	0.964498	0.936806	4047	0.8094	13	185.873
test_7	8307.68	0.96367	0.934028	4035	0.807	11	182.399
test_8	8311.84	0.96415	0.934259	4036	0.8072	10	184.103
test_9	8323.3	0.965479	0.937963	4052	0.8104	8	186.169
promedio	8310.134	0.9639523	0.9349997	4039.2	0.80784	10.6	189.0529
sd	7.50684	0.00087	0.00206	8.90443	0.00178	1.77639	8.01950

Tabla B.7: 2500 proyectos con 5000 SB en ciclo 1 en semilla 2

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8411.52	0.961591	0.937731	4051	0.8102	7	190.859
test_1	8391.83	0.95934	0.932639	4029	0.8058	9	186.443
test_2	8395.72	0.959785	0.93287	4030	0.806	6	188.688
test_3	8403.32	0.960654	0.937269	4049	0.8098	11	185.361
test_4	8406	0.96096	0.936806	4047	0.8094	9	189.151
test_5	8405.48	0.9609	0.936574	4046	0.8092	8	184.929
test_6	8397.15	0.959948	0.934722	4938	0.8076	12	186.213
test_7	8391.06	0.959252	0.934259	4036	0.8072	12	189.789
test_8	8400.59	0.960341	0.934491	4037	0.8074	9	186.464
test_9	8401.24	0.960415	0.935648	4042	0.8084	10	187.33
promedio	8400.391	0.9603186	0.9353009	4130.5	0.8081	9.3	187.5227
sd	6.52545	0.00075	0.00179	283.83064	0.00155	2.00278	1.99208

Tabla B.8: 2500 proyectos con 5000 SB en ciclo 1 en semilla 3

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8329.09	0.961065	0.931944	4026	0.852	12	188.102
test_1	8321.41	0.960179	0.929861	4017	0.8034	13	192.919
test_2	8340.86	0.962423	0.93588	4043	0.8086	9	189.721
test_3	8323.63	0.960436	0.931481	4024	0.8048	12	189.649
test_4	8326.46	0.960761	0.931713	4025	0.805	11	191.862
test_5	8336.8	0.961954	0.934491	4037	0.8074	11	193.465
test_6	8325.2	0.960617	0.931481	4024	0.8048	12	196.123
test_7	8315.77	0.959528	0.928935	4013	0.8026	15	186.557
test_8	8339.98	0.962321	0.934491	4037	0.8074	7	186.554
test_9	8324.64	0.96055	0.931481	4024	0.8048	14	186.141
promedio	8328.384	0.9609834	0.9321758	4027	0.81008	11.6	190.1093
sd	8.29761	0.00096	0.00216	9.33333	0.01485	2.31900	3.39591

Tabla B.9: 2500 proyectos con 5000 SB en ciclo 1 en semilla 4

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8376.55	0.964408	0.937963	4052	0.8104	8	188.382
test_1	8363.67	0.962925	0.934259	4036	0.8072	8	187.737
test_2	8367.04	0.963313	0.936111	4044	0.8088	8	189.356
test_3	8376.1	0.964355	0.939815	4060	0.812	12	188.779
test_4	8370.14	0.96367	0.935185	4040	0.808	6	187.137
test_5	8378.54	0.964637	0.939583	4059	0.8118	9	186.631
test_6	8375.28	0.9644261	0.9375	4050	0.81	8	187.787
test_7	8367.46	0.963361	0.934259	4036	0.8072	6	190.66
test_8	8375.54	0.964291	0.940509	4063	0.8126	11	191.328
test_9	8375.75	0.964315	0.938426	4054	0.8108	10	188.501
promedio	8372.607	0.96397011	0.937361	4049.4	0.80988	8.6	188.6298
sd	5.07733	0.00060	0.00231	9.96884	0.00199	1.95505	1.48177

Tabla B.10: 2500 proyectos con 5000 SB en ciclo 1 en semilla 5

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	8441.08	0.960243	0.933796	4034	0.8068	12	182.446
test_1	8444.01	0.960577	0.93333	4032	0.8064	10	178.931
test_2	8434.33	0.959476	0.931944	4026	0.8052	12	190.562
test_3	8464.52	0.962911	0.939583	4059	0.8118	7	210.447
test_4	8448.98	0.961142	0.935185	4040	0.808	10	188.756
test_5	8461.24	0.962537	0.939352	4058	0.8116	10	195.874
test_6	8443.37	0.960504	0.933565	4033	0.8066	9	186.493
test_7	8449.37	0.961187	0.935648	4042	0.8084	8	185.501
test_8	8441.17	0.960254	0.932176	4027	0.8054	9	185.243
test_9	8440.17	0.96014	0.943028	4035	0.807	11	183.229
promedio	8446.824	0.9608971	0.9357607	4038.6	0.80772	9.8	188.7482
sd	9.52940	0.00108	0.00369	11.58735	0.00232	1.61933	8.94041

Tabla B.11: 5000 proyectos con 10000 SB en ciclo 0 en semilla 1

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	17234.4	0.971626	0.968634	8369	0.8369	7	745.278
test_1	17225.4	0.971117	0.966782	8353	0.8353	7	835.656
test_2	17247.4	0.972358	0.970718	8387	0.8387	5	821.683
test_3	17247.2	0.972346	0.970718	8387	0.8387	5	790.489
test_4	17227.9	0.971259	0.96713	8356	0.8356	6	783.854
test_5	17232.8	0.971532	0.968056	8364	0.8364	6	778.956
test_6	17229.1	0.971327	0.967361	8358	0.8358	6	775.763
test_7	17237.6	0.971804	0.968866	8371	0.8371	5	738.44
test_8	17224	0.97108	0.966782	8353	0.8353	7	730.024
test_9	17232.7	0.971528	0.968171	8365	0.8365	7	735.382
promedio	17233.85	0.9715977	0.9683218	8366.3	0.83663	6.1	773.5525
sd	8.19895	0.00046	0.00146	12.57025	0.00126	0.87560	36.49124

Tabla B.12: 5000 proyectos con 10000 SB en ciclo 0 en semilla 2

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	17028.3	0.973466	0.966551	8351	0.8351	7	696.85
test_1	17013.6	0.972625	0.964236	8331	0.8331	9	712.855
test_2	17029.9	0.973556	0.967014	8355	0.8355	9	702.949
test_3	17015.4	0.972729	0.964005	8329	0.8329	7	702.731
test_4	17011.2	0.972489	0.962731	8318	0.8318	6	694.289
test_5	17015.8	0.972749	0.964699	8335	0.8335	9	696.0145
test_6	17012.2	0.972548	0.963426	8324	0.8324	8	769.447
test_7	17032.7	0.97372	0.96713	8356	0.8356	6	797.579
test_8	17023.3	0.973179	0.965509	8342	0.8342	8	807.285
test_9	17037	0.973962	0.968866	8371	0.8371	9	773.019
promedio	17021.94	0.9731023	0.9654167	8341.2	0.83412	7.8	735.30185
sd	9.47795	0.00054	0.00194	16.73187	0.00167	1.22927	45.90330

Tabla B.13: 5000 proyectos con 10000 SB en ciclo 0 en semilla 3

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	17136.3	0.97227	0.96412	8330	0.833	8	720.323
test_1	17138.4	0.97239	0.964005	8329	0.8329	5	698.108
test_2	17140.6	0.972515	0.964583	8334	0.8334	6	708.587
test_3	17140.7	0.9722517	0.964468	8333	0.8333	5	701.26
test_4	17139.5	0.972453	0.964236	8331	0.8331	5	702.559
test_5	17129.2	0.971866	0.962269	8314	0.8314	6	691.489
test_6	17130.3	0.97193	0.962847	8319	0.8319	7	698.585
test_7	17146.2	0.972834	0.965856	8345	0.8345	7	708.73
test_8	17132.8	0.972073	0.964352	8332	0.8332	8	725.478
test_9	17132.9	0.972079	0.96331	8323	0.8323	7	716.341
promedio	17136.69	0.97226617	0.9640046	8329	0.8329	6.4	707.146
sd	5.36666	0.00029	0.00100	8.64099	0.00086	1.17379	10.82504

Tabla B.14: 5000 proyectos con 10000 SB en ciclo 0 en semilla 4

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	17131.2	0.969795	0.963426	8324	0.8324	6	709.603
test_1	17147.6	0.970721	0.966204	8348	0.8438	4	703.787
test_2	17129.6	0.969702	0.963657	8326	0.8326	8	701.866
test_3	17134.1	0.969956	0.963773	8327	0.8327	5	683.371
test_4	17133.1	0.9699	0.964005	8329	0.8329	7	695.016
test_5	17134.9	0.97	0.964005	8329	0.8329	5	701.052
test_6	17143.4	0.970485	0.966319	8349	0.8349	5	704.267
test_7	17139	0.970237	0.964699	8335	0.8335	5	700.874
test_8	17142.1	0.970408	0.965509	8342	0.8342	6	697.176
test_9	17138.8	0.970223	0.964699	8335	0.8335	5	705.882
promedio	17137.38	0.9701427	0.9646296	8334.4	0.83434	5.6	700.2894
sd	5.78846	0.00033	0.00106	9.11897	0.00341	1.17379	7.25119

Tabla B.15: 5000 proyectos con 10000 SB en ciclo 0 en semilla 5

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	16949.5	0.970708	0.962616	8317	0.8317	9	693.289
test_1	16958.3	0.971211	0.963657	8326	0.8326	10	693.23
test_2	16942.4	0.970297	0.959954	8294	0.8294	6	703.275
test_3	16957.4	0.97116	0.963194	8322	0.8322	8	702.345
test_4	16953.9	0.970958	0.962963	8320	0.8320	8	707.605
test_5	16966.2	0.971663	0.964583	8334	0.8334	6	696.593
test_6	16946.9	0.970557	0.962153	8313	0.8313	9	707.071
test_7	16941.6	0.970254	0.960301	8297	0.8297	7	707.14
test_8	16942.3	0.970294	0.96169	8309	0.8309	11	710.608
test_9	16958.8	0.971241	0.963542	8325	0.8325	9	694.992
promedio	16951.73	0.9708343	0.9624653	8315.7	0.83157	8.3	701.6148
sd	8.47402	0.00049	0.00147	12.73708	0.00127	1.63639	6.57479

Tabla B.16: 5000 proyectos con 10000 SB en ciclo 1 en semilla 1

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	16939	0.963599	0.939236	8115	0.8115	11	724.309
test_1	16907.8	0.96182	0.933218	8063	0.8063	9	715.616
test_2	16932.3	0.963213	0.936921	8095	0.8095	11	714.361
test_3	16944.3	0.963896	0.939468	8117	0.8117	9	723.828
test_4	16912.1	0.962068	0.933565	8066	0.8066	8	722.476
test_5	16900.8	0.961427	0.931597	8049	0.8049	11	753.939
test_6	16914.4	0.9622	0.934838	8977	0.8077	13	723.555
test_7	16897.6	0.961242	0.93125	8046	0.8046	8	718.71
test_8	16909.4	0.961913	0.933102	8062	0.8062	10	725.665
test_9	16915	0.962234	0.933449	8065	0.8065	7	715.724
promedio	16917.27	0.9623612	0.9346644	8165.5	0.80755	9.7	723.8183
sd	15.91163	0.00090	0.00293	286.25484	0.00253	1.82878	11.35835

Tabla B.17: 5000 proyectos con 10000 SB en ciclo 1 en semilla 2

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	16928.7	0.96205	0.934028	8070	0.8070	10	783.157
test_1	16925.4	0.961864	0.932986	8061	0.8061	7	800.449
test_2	16936.2	0.962476	0.935185	8080	0.808	8	786.277
test_3	16930.2	0.962131	0.933912	8069	0.8069	9	774.133
test_4	16927.6	0.961988	0.93287	8060	0.806	4	808.837
test_5	16912.1	0.961106	0.93109	8044	0.8044	8	814.757
test_6	16931.2	0.962191	0.934259	8072	0.8072	8	819.246
test_7	16929	0.962065	0.93333	8064	0.8064	6	797.331
test_8	16927	0.961953	0.93333	8064	0.8064	6	781.59
test_9	16918	0.96144	0.932639	8058	0.8058	12	785.394
promedio	16926.54	0.96193	0.93336	8064.2	0.80642	7.80000	795.11710
sd	6.85747	0.00039	0.00110	9.67011	0.00097	2.25093	15.37870

Tabla B.18: 5000 proyectos con 10000 SB en ciclo 1 en semilla 3

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	17015.6	0.960358	0.926389	8004	0.8004	8	682.966
test_1	17038.8	0.961667	0.930208	8037	0.8037	7	678.103
test_2	17065.2	0.963155	0.935417	8082	0.8082	10	679.175
test_3	17028.7	0.961099	0.928819	8025	0.8025	9	679.688
test_4	17019.2	0.960558	0.928125	8019	0.8019	13	688.18
test_5	17035.9	0.961503	0.930093	8036	0.8036	8	679.282
test_6	17050.7	0.962234	0.932176	8054	0.8054	8	692.002
test_7	17046.2	0.962083	0.932292	8055	0.8055	11	674.362
test_8	17026.5	0.960972	0.928935	8026	0.8026	10	691.76
test_9	17039.3	0.961697	0.930208	8037	0.8037	7	741.627
promedio	17036.61	0.9615326	0.9302662	8037.5	0.80375	9.1	688.7145
sd	15.01750	0.00084	0.00253	21.89495	0.00219	1.91195	19.53116

Tabla B.19: 5000 proyectos con 10000 SB en ciclo 1 en semilla 4

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	16806.8	0.962326	0.933796	8068	0.8068	6	721.274
test_1	16811.9	0.962617	0.934838	8077	0.8077	7	717.915
test_2	16814.7	0.962778	0.935532	8083	0.8083	9	705.802
test_3	16795.1	0.961654	0.933102	8062	0.8062	10	721.484
test_4	16801.7	0.962031	0.933796	8068	0.8068	10	707.335
test_5	16824.3	0.96333	0.937616	8101	0.8101	9	707.282
test_6	16805.7	0.962261	0.943722	8076	0.8076	12	708.827
test_7	16805.2	0.962234	0.934144	8071	0.8071	11	721.395
test_8	16805.5	0.96225	0.934375	8073	0.8073	11	715.716
test_9	16799.2	0.961892	0.933912	8069	0.8069	11	720.883
promedio	16807.01	0.9623373	0.9354833	8074.8	0.80748	9.6	714.7913
sd	8.30227	0.00048	0.00316	10.89138	0.00109	1.89737	6.72073

Tabla B.20: 5000 proyectos con 10000 SB en ciclo 1 en semilla 5

	Throughput	$\frac{Throughput}{TotalSciGrade}$	$\frac{TiempoUsado}{TiempoTotal}$	SBs Ejecutados	$\frac{SBsEjecutados}{TotalSBs}$	SBs Descartados	Tiempo de Ejecución
test_0	16663.4	0.964175	0.937963	8104	0.8104	10	719.777
test_1	16662	0.964095	0.936921	8095	0.8095	11	723.627
test_2	16657.1	0.963812	0.935995	8087	0.8087	11	712.626
test_3	16667.4	0.964406	0.937847	8103	0.8103	11	724.25
test_4	16649.6	0.963374	0.934606	8075	0.8075	13	713.483
test_5	16661.5	0.964068	0.936458	8091	0.8091	9	715.877
test_6	16647.6	0.963262	0.935648	8084	0.8084	14	721.109
test_7	16637.2	0.962661	0.933796	8068	0.8068	10	720.796
test_8	16641.6	0.962916	0.934259	8072	0.8072	12	715.785
test_9	16661.5	0.964066	0.936227	8089	0.8089	9	718.8
promedio	16654.89	0.9636835	0.935972	8086.8	0.80868	11	718.613
sd	10.23300	0.00059	0.00143	12.32703	0.00123	1.63299	4.04071