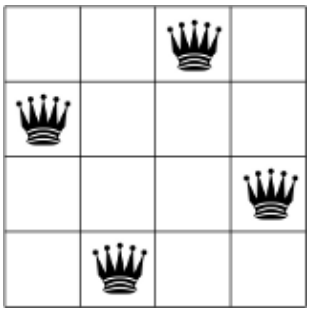
UNIVERSIDAD NACIONAL COSTA RICA


Facultad de Ciencias Exactas y Naturales

Curso:

Paradigmas de Programación

# Proyecto 1: NQueens



Profesor:

Eddy Miguel Ramírez

Estudiantes:

Mario Arguello
Josué Céspedes Miranda

II CICLO

2020

# Índice

# 1. Resumen ejecutivo

El presente trabajo esta realizado con el objetivo de hacer un uso práctico de distintos modelos con el fin de brindar una solución al popular problema NReinas bajo el uso de un paradigma funcional, que permita conocer nuevos conceptos y permita establecer diferentes perspectivas a la hora de brindar soluciones a distintos problemas mediante programación.

El problema de NReinas es un acertijo con bastante popularidad, propuesto por el ajedrecista alemán Maxx Bezzel en 1848, consiste en colocar N reinas en un tablero con dimensiones NxN, su principal desafío radica en colocar las N reinas, mencionadas anteriormente, de tal manera que ninguna reina quede atacando otra. Dado que en el ajedrez esta pieza puede atacar a las piezas que se encuentren en su misma fila, misma columna, o en sus respectivas diagonales.

En este documento se detallan dos posibles métodos para dar solución al problema explicado previamente. La primera, y más común a la hora de resolver este algoritmo, es una solución implementando backtracking, la cual se ha diseñado y escrito en Scheme, el cual como se mencionó al inicio tiene características de programación funcional y es un subconjunto del lenguaje de programación LISP; fue desarrollado en el MIT (Massachusetts Institute of Technology). La segunda solución ha sido implementada sobre algoritmos genéticos, se ha trabajado con elitismo y con un porcentaje de mutación de un 5 %, la solución para este caso esta escrita en Erlang, el cual es también es un lenguaje de programación funcional de alto nivel, diseñado en los años 80 en los laboratorios de Ciencias y Computación de la compañía de telefonía sueca Ericsson, y fue diseñado para escribir aplicaciones concurrentes y distribuidas de funcionamiento ininterrupido.

Además, este trabajo pretende plasmar las distintas diferencias presentes en los métodos utilizados para dar solución al problema, establecer una comparación basada en el tiempo de ejecución de cada una de las soluciones y que permita obtener como resultado cual presenta mejores características de desempeño. Por otra parte, ambas soluciones representan una oportunidad para los estudiantes participantes en conocer nuevos paradigmas, poder incursionar en temas de mucho interés como en el caso de algoritmos genéticos que son temas muy amplios y que lamentablemente es díficil encontrar en un curso dentro del programa de la carrera de Ingeniería en Sistemas de Información en la Universidad Nacional de Costa Rica que abarque la investigación e implementación de este tipo de algoritmos, que se encuentran relacionados a temas de inteligencia artificial y que son utilizados para resolver problemas de búsqueda y optimización, que concuerdan con los parámetros necesarios para dar una solución al problema de NReinas. Por lo que este proyecto representa: 1) un reto, por la parte de investigación y la utilización de un paradigma distinto a la programación orientada a objetos(POO), que incluye un cambio en la forma de pensar, y 2) la oportunidad de aprender temas nuevos y relevantes que pueden llegar a tener un impacto positivo en un futuro profesional.

Aquí se documentan la descripción de las soluciones, los principales problemas encontrados durante su desarrollo, aprendizajes obtenidos, se adjuntan los resultados de las pruebas efectuadas y demás aspectos concernientes para el proyecto.

# 2. Introducción

# 3. Marco teórico

Como se ha mencionado previamente, para dar solución al problema de NReinas se han utilizado dos lenguajes de programación funcionales. A continuación de profundiza un poco más en la historia, características de Scheme y Erlang y sus principales usos.

1. **Scheme**
   Es un lenguaje de programación de alto nivel, se le considera un lenguaje altamente expresivo. Su sintaxis es fácil de leer pero no lo es tanto comparándolo con Python. Opera sobre estructuras como listas, cadenas, vectores, tuplas y posee conjuntos de datos que lo hacen muy versátil.

   - Historia: surgió en los laboratorios del MIT en 1975, cuando Guy L. Steele y Gerald J. Sussman buscaban un lenguaje con un semántica clara y sencilla. Originalmente se llamaba "Schemer", se le cambió el nombre ya que sus creadores usaban el sistema operativo ITS, que limitaba la longitud del nombre de los ficheros a 6 caracteres.
   - Antecedentes: altamente influenciado por el cálculo lambda. Su desarrollo ha sido lento, ya que las personas encargadas de su estandarización es muy conservadora y ha sido difícil añadirle nuevas características. Se le considera una de los mejores lenguajes de propósito general. Fue el primer dialecto de Lisp que usó ámbito estático o léxico en lugar de dinámico de forma exclusiva.
   - Características: filosofía minimalista, su mecanismo principal para el control de flujo son las llamadas recursivas finales. Además, Scheme ofrece gestión automática de memoria, popularmente conocido como recolector de basura. Las listas son la estructura de datos básica, no hay sintaxis explícita para crear registros o estructuras o para programación orientada a objetos. La programación funcional pura no necesita de variables ni manejo de memoria compartida, por lo que hacen a Scheme una buena opción en presencia de procesos concurrentes. Una de las desventajas es que existen múltiples implementaciones diferentes, cada una utiliza extensiones y bibliotecas propias que se tornan incompatible entre sí.
   - Usos: la mayoría de uso que se le da a Scheme se refiere a la parte educativa. Profesores de universidad, como es el caso del profesor R. Kent Dybvig que trabaja en la Universidad de Indiana, se concentra en el prototipado de nuevas semánticas para Scheme, incluso tiene a la venta un compilador de Scheme con su colega Daniel P. Friedman. También, hay muchos otros profesores de distintas universidades que lo usan como lenguaje de enseñanza.

2. **Erlang**
   Es un lenguaje de programación funcional de alto nivel, que se ubica también dentro del paradigma de Programación Declarativa, diseñado para escribir aplicaciones concurrentes y distribuidas de funcionamiento ininterrumpido. Erlang cuenta con tipado dinámico y asignación única, esto último consiste en que una vez asignado un valor a una variable, este valor no puede cambiar.

   - Historia: los inicios de Erlang toman lugar en los años 80, en los laboratorios de Ciencias de Computación, de compañía de telefonía sueca Ericsson, como un intento de desarrollar un lenguaje de programación de alto nivel y con capacidad para

afrontar proyectos relacionados a temas de telecomunicaciones, en las que Ericsson participaba

- Antecedentes: para su desarrollo se analizaron alrededor de 300 lenguajes de programación existentes a la fecha, de todos los analizados se seleccionaron: Lisp, Haskell y Prolog en conjunto con algunas características resaltantes de otros lenguajes.
- Características: lenguaje de alto nivel basado en procesos, orientado a concurrencia, ideal para proyectos que necesiten tráfico de información de respuesta en tiempo real. Utiliza pattern matching, la cual es una técnica básica de Erlang en la cual es posible hacer un balance entre variables y valores a los dos lados del signo de igualdad o fallar si no es posible encontrar coincidencia. Al igual que Scheme, posee un método interno para la organización de memoria automática, que ofrece una limpieza de memoria que permite al programador olvidarse de hacer la respectiva liberación de los recursos utilizados por el programa. Entre sus grandes cualidades destacan las listas por comprensión, el cual es un mecanismo matemático que permite generar valores a partir de una sintaxis simple. Otro aspecto a destacar es que el código de Erlang permite comunicarse con librerías y otros códigos, que esten escritos en otros lenguajes , utilizando el mismo mecanismo que utiliza internamente para comunicarse entre procesos.
- Usos: dentro de algunos de los usos que se han encontrado durante la investigación para este proyecto se encuentra **Wings3d**, este proyecto es un modelador 3D que esta implementado en Erlang , es sencillo de usar y ofrece un amplia gama de herramientas de modelado, una interfaz personalizable, soporte para luces y materiales, sin embargo Wings no permite animación de ningún estilo, se puede encontrar más información de Wings3d en `http://www.wings3d.com/`. Otro usos que se han podido encontrar se encuentran enfocados en microelectrónica y microinformática y en la parte de enseñanza en algunas universidades.
- Bibliotecas utilizadas: para la solución de algoritmos genéticos se ha hecho uso de la librería **lists**, que contiene las principales funcionalidades para el procesamiento de listas, las principales funciones utilizadas han sido: revertir una lista, generar una lista con una secuencia de números, obtener el termino en determinada posición, ordenar una lista, y poder hacer split de una estructura en determinada posición. Todas las funciones son de vital importancia al trabajar con algoritmos genéticos orientado a generar una solución para el problema de NReinas, ya que se ha representado un individuo mediante una lista de genes, y todos los individuos se encuentran en una lista que representa la población. Además, se ha utilizado la librería **rand**, para obtener números aleatorios y que son utilizados para generar individuos que forman parte de la población, determinar si debe aplicarse mutación a determinado individuo y similares funciones.

Dentro de los IDE's utilizados para realizar la escritura del código que produjo la solución al problema, para la parte de Scheme se ha utilizado Dr.Racket, el cual facilita la ejecución y prueba de código Scheme. Para la solución de algoritmos genéticos se ha escrito utilizando Visual Studio Code, y para la compilación de código se ha utilizado el prompt interactivo de erl dentro de la terminal ofrecida por Visual Studio Code. Ambas funciones fueron escritas y probadas sobre distribuciones de Linux Mint y Ubuntu 18.04. Se ha utilizado `https://www.erlang.org/` y `https://schemers.org/` como principales fuentes para consulta de sintaxis determinada.

# 4. Descripción de la solución

Para describir la solución de algoritmos genéticos, primero es importante definir algunos conceptos básicos en términos del problema para que esta explicación sea más entendible.

- Algoritmo genético: es hacer uso de mecanismos que simulan la evolución de las especies de la biología para formular los pasos que describen la búsqueda de una solución a un problema concreto, es una técnica de inteligencia artificial inspirada en la idea de que sobrevive el mejor adaptado a las condiciones.
- N: número de reinas sobre las que el algoritmo genético va a buscar una solución, su dominio son los números naturales mayores a 3.
- Cromosomas: un número que abarca de 1 a N, representa una fila donde se encuentra la reina en relación al tablero.
- Individuo: es una estructura de datos que representa la posición de las N reinas en el tablero, esta compuesto por N cromosomas.
- Población: estructura de datos que representa la cantidad de individuos sobre los cuales se realiza el algoritmo génetico, para la solución esta dada una población de 4 * N.
- Elitismo: técnica utilizada, cuando se resuelve un problema con la metodología generacional, en la cual se asegura la sobrevivencia del mejor individuo haciendo que esté presente en la siguiente generación para así no perder la buenas caracteristicas del mismo.
- Cruces: es el proceso en el cuál los individuos de una población se cruzan para generar una nueva generación, este cruce está dado de forma aleatoria entre los miembros de la población de la población eso sí con una técnica de selección la cual favorece a los mejor calificación en la función fitness.
- Función fitness: es un función que "*evalua*" a los individuos de la población y conforme a sus características les otorga mejor o peor calificación, en términos del problema, dicha calificación esta dada de acuerdo a cuantas veces hay colisiones entre cromosomas, entre menores colisiones, mejores probalidades de perdurar.
- Generaciones: son el producto de los cruces de las población, entre más generaciones haya se espera que sea más precisa la aproximación a la respuesta correcta.
- Mutación: proceso por el cual un crosoma es alterado, se ha trabajado con un 5 % de probalidad de mutación y la estrategia utilizada se basa en escoger dos posiciones de los cromosomas del individuo e intercambiar dichas posiciones, esto con el fin de preservar el fitness que fue calculado para dicho individuo.

# 5. Resultados de las pruebas

# 6. Conclusiones