**INSTITUTO TECNOLÓGICO DE COSTA RICA**

****

**ANÁLISIS DE ALGORITMOS**

**Sistema de optimización de calendarios de filmación**

**Realizado por:**

**Jose Daniel Amador Salas**

**José Pablo Brenes Alfaro**

**Profesora:**

**Lorena Valerio Solís**

**Santa Clara, 2018**

Tabla de contenido

[Introducción 3](#_Toc516325138)

[Análisis del problema 4](#_Toc516325139)

[Solución del problema 8](#_Toc516325140)

[Función Objetivo 8](#_Toc516325141)

[Estrategias de diseño 8](#_Toc516325142)

[Ramificación y Poda 8](#_Toc516325143)

[Pseudocódigo: 9](#_Toc516325144)

[Algoritmo Genético 11](#_Toc516325145)

[Población inicial: 11](#_Toc516325146)

[Evaluación o función de ajuste: 11](#_Toc516325147)

[Cruces: 12](#_Toc516325148)

[ Cycle Crossover: 12](#_Toc516325149)

[ OX1 Crossover: 13](#_Toc516325150)

[Mutación: 13](#_Toc516325151)

[Escenarios: 14](#_Toc516325152)

[ Escenario #1 14](#_Toc516325153)

[ Escenario #2 14](#_Toc516325154)

[ Escenario #3 14](#_Toc516325155)

[ Escenario #4 14](#_Toc516325156)

[Salidas 14](#_Toc516325157)

[Análisis de Resultados 15](#_Toc516325158)

[Mediciones 18](#_Toc516325159)

[Factores de crecimiento 19](#_Toc516325160)

[Análisis Analítico 20](#_Toc516325161)

[Método Ramificación y Poda 20](#_Toc516325162)

[Método Algoritmo Genético 22](#_Toc516325163)

# Introducción

El cine es uno de los mercados más interesantes y caros de realizar, más aún, si se le agrega el hecho de que no lo produzca una de las grandes empresas cinematográficas. Esto hace que se comiencen a ver problemas de presupuesto, lo que produce una limitante de recursos para implementar en actores, escenas. Es aquí cuando se recalca la necesidad de un sistema de optimización de los recursos, con el objetivo de reducir el coste de producción de una filmación.

El principal problema al organizar la grabación de una película es poder tener disponibilidad del lugar de grabación y de los actores para poder rodar las escenas. Una mala asignación de recursos se traduce como pérdidas para la empresa, pues con cada día que transcurre hasta el último día del contrato de un actor se le debe pagar sus servicios.

Programar jornadas de grabación eficientes es primordial para cualquier empresa que desee grabar una película. Entonces, para dar solución a este problema de optimización se proponen dos algoritmos con los que se plantea encontrar la solución eficiente.

# Análisis del problema

El problema de programación óptima de los horarios de rodaje que se propone está conformado por un conjunto de escenas que deben ser programadas en diferentes jornadas, un conjunto de localidades de grabación donde se realizará cada una de las escenas y la repartición de los actores que debe de participar en el rodaje dichas escenas. Cada actor debe participar en el rodaje de una o varias escenas, las cuales pueden realizarse en localidades y jornadas diferentes. El objetivo es elegir la mejor jornada y día para la grabación de cada una de las escenas para minimizar el costo total de la producción la cual se basa en la siguiente variable: C = mínimo de la sumatoria de todos los recursos del costo por la diferencia en jornadas (L-F + un día); esto con el fin de ahorrar gastos por insumos pagados a los actores que tiene que esperar hasta su próxima participación en la grabación.

Para plantear el problema adecuándose a las necesidades que se proponen, se puede describir de la siguiente manera: se debe programar un sistema que asigne eficientemente los recursos(actores) de grabación en diferentes escenas de películas usando distintas estrategias de diseño como la ramificación y poda e incluyendo también los algoritmos genéticos, todo esto con el fin de reducir el costo de los recursos humanos contratados, además se debe medir empíricamente y analíticamente los algoritmos implementados para poder medir de manera óptima así como su uso de memoria para las estrategias de diseño de algoritmos.

De acuerdo con la situación planteada se requiere la utilización de cuatro variables, estas determinan el eje central del problema y se utilizarán de la siguiente manera:

* **E:** representa las escenas que serán rodadas durante el periodo de grabación, esta será de tipo objeto y tendrá las siguientes características:
  + **Un atributo <<id>>:** un identificador de tipo numérico (int), que diferencia cada escena de la demás.
  + **Un atributo <<id>>:** un identificador de tipo numérico (int), que diferencia cada escena de la demás.
  + **Una lista de actores:** esta contiene la lista de actores que serán asignados a la grabación de esta escena.
  + **Una localidad:** lugar donde se realizará la grabación de la escena en cuestión.
  + **Un atributo <<duración>>:** determina la duración de de la escena, es decir, cuanto tiempo se dispone para realizar la grabación de dicha escena, es de tipo int.
* **R:** representa los recursos (actores) que se pueden asignar al rodaje de una escena. Es de tipo objeto y contiene las siguientes características:
  + **Un atributo <<id>:** variable de tipo int que se utiliza para diferenciar a los recursos de los demás.
  + **Un atributo <<costo por día>>>:** esta representa el coste que tiene el actor por cada día que transcurre desde su contratación hasta la última escena en la que participa, se almacena en una variable tipo int.
  + **Un atributo <<primera Jornada>>:** primer día en el cual el actor realiza su participación en el rodaje de una escena, es una instancia de tipo **J**.
  + **Un atributo <<última Jornada>>:** día en el cual el actor realiza su última participación en el rodaje de una escena, es una instancia de tipo **J**.

Nota\*: Los atributos primera jornada, última jornada se utilizan para determinar el costo de contratación del recurso para el periodo de grabación de la película el cual está determinado por la fórmula **[Last\_Day - First\_Day + 1] \* cost\_per\_day.**

* **L:** representa las localidades donde se llevará a cabo las grabaciones de las escenas, es de tipo objeto y tiene las siguientes características:
  + **Un atributo <<id>>:**  variable de tipo int que identifica una localidad de las demás.
  + **Una lista de jornadas:** estos serán los días en los que dicha jornada estará disponible para ser utilizada.
* **J:** representa los días de grabación disponibles y se dividen en día o noche, es de tipo objeto y contiene las siguientes características:
* **Un atributo <<id>>:** variable de tipo String que permite identificar una jornada de las demás.
* **Un atributo <<número de día>>:** variable de tipo int que simboliza el contador de días que transcurre durante la grabación.
* **Un atributo <<Máximo tiempo de grabación>>:** indica la cantidad de horas que se pueden grabar en esa jornada, esta variable es de tipo int.
* **Un atributo <<is day>>:** variable de tipo bool que permite diferenciar si el día es una jornada diurna o nocturna.

Las características anteriormente planteadas determinan la relación existente entre las diferentes entidades del problema. La interpretación para la implementación de esta es la siguiente:

Tenemos un conjunto N de escenas, las cuales contienen una localidad lugar donde se grabará la escena y una lista de N actores que participarán en esta grabación. Además, cada actor y localidad contienen una lista de N días de filmación en los cuales estos están disponibles.

De estas variables se establecen algunas relaciones importantes en las cuales se definen en la siguiente lista:

★ Si el recurso(actor) debe participar en la escena e.

★ Si la escena debe ser filmada en la localización l.

★ Si la localización está disponible en la jornada j.

★ Si el recurso(actor) está disponible en la jornada j.

★ Si la escena se filma de día.

Entre otras características que se deben considerar para realizar las relaciones entre las variables se encuentran los siguientes atributos:

❖ **d=** Duración de la escena, medido en horas.

❖ **C=** Costo del recurso, medido en Unidad monetaria/jornada.

❖ **D=** Número máximo de horas que se puede grabar por jornada.

❖ **Z =** Si la escena ese programa en la jornada j.

❖ **W=** Si el recurso r será utilizado en la jornada j.

❖ **F=** Primera jornada del programa en que el recurso r se utiliza.

❖ **L =** Última jornada del programa en que el recurso r se utiliza.

Como anteriormente se mencionó, los atributos F y L se utilizan para calcular el máximo coste de implementación.

Además, estas relaciones con las variables propuestas tienen que tener restricciones en las cuales evitan hacer inconvenientes entre las relaciones de las variables, todo con el fin de mejorar la planificación del rodaje de las películas, por ello se definen las siguientes restricciones:

1. Un recurso será utilizado en una jornada, sólo si existe una escena programada para esa jornada que lo requiera.

2. Un recurso no debe ser asignado a una jornada si no se necesita.

3. La primera jornada en que un recurso es utilizado debe ser menor que todas las jornadas en las cuales el recurso es requerido.

4. La última jornada en que un recurso es utilizado debe ser mayor que todas las jornadas en las cuales el recurso es requerido.

5. La duración del total de las escenas programadas en una jornada en particular no puede superar el límite establecido.

6. Una escena debe ser programa sólo en una jornada.

7. Todas las escenas que se filman en una jornada deben ser del mismo tipo (día o noche).

Sólo se pueden programar jornadas del mismo tipo (día o noche) de forma consecutiva. En caso de que se necesite cambiar de tipo de jornada debemos dejar un período libre.

# Solución del problema

Después de analizar la problemática que se nos presenta decidimos realizar la valoración del proyecto en base a dos algoritmos, ramificación y poda y algoritmos genéticos. A continuación, se presentan las diferentes estrategias de diseño. Además, de la función objetivo y otros elementos importantes.

## Función Objetivo

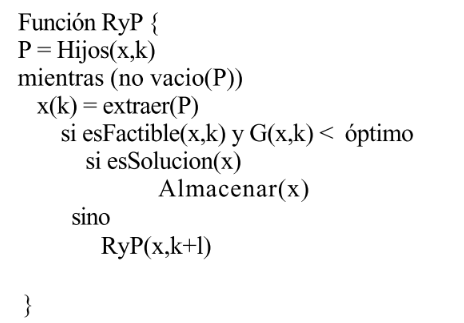
La función objetivo está definida por la Min(Θ) donde theta representa el total de costos que supone realizar una escena con cierto número de actores. Para poder lograr una aproximación a la función objetivo es necesario asignar de manera eficiente los días de rodaje para cada escena. Estas combinaciones que es necesario hacer son las que generaran las estrategias de diseño hasta encontrar la función objetivo.

## Estrategias de diseño

Para resolver el análisis planteado sobre la optimización de horarios de grabación de las películas se desarrollan los siguientes algoritmos, con los cuales se planea poner a prueba la eficiencia en la optimización de los horarios de grabación.

# Ramificación y Poda

Según (Universidad Nacional del nordeste,2015)la técnica de ramificación y poda se suele interpretar como un árbol de soluciones, donde cada rama nos lleva a una posible solución posterior de la actual. En este caso, el algoritmo se encarga de detectar en que ramificación las soluciones dadas ya no están siendo óptimas ósea la minimización efectiva de los costos de rodaje de una película, esto con el fin de <<podar>> esa rama del árbol y no continuar malgastando los recursos en casos de que se alejan de la solución óptima.

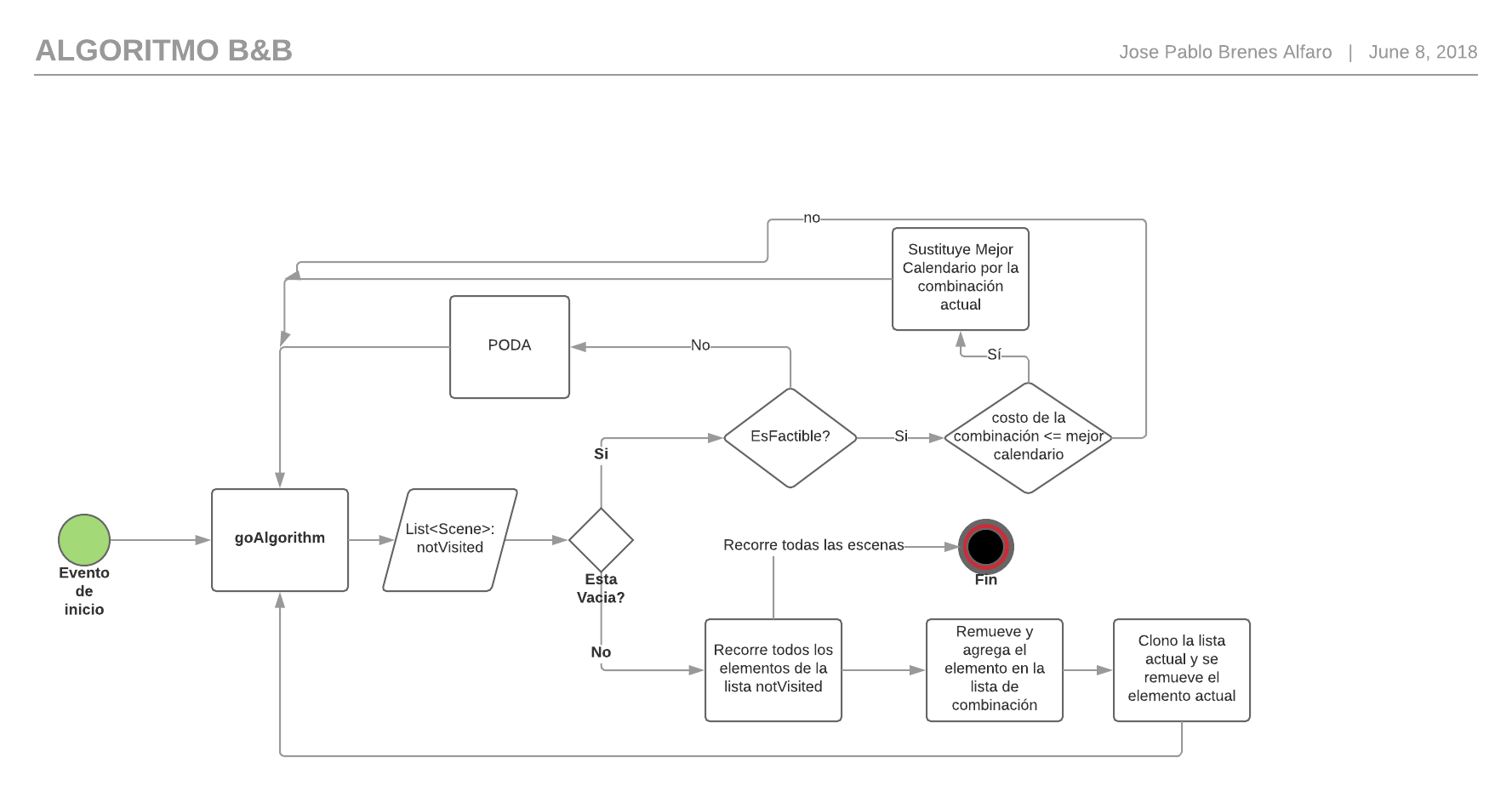
Pseudocódigo:el pseudocódigo del algoritmo de ramificación y poda es el siguiente:

Donde:

* G(X) es la **función de estimación del algoritmo.**
* P es la **pila de posibles soluciones.**
* esFactible es la función que se considera si la propuesta es válida.
* esSolucion es la función que comprueba si se satisface el objetivo.
* optimo es el valor de la función a optimizar evaluado sobre la mejor solución encontrada hasta el momento.
* **Nota:** (<) representa la minimización de costos.

Este algoritmo termina cuando no queden recursos con días ociosos o que se acaben las todas las combinaciones posibles.

**Diagrama de Flujo**

****

**Estrategias de Ramificación:** para la creación de la ramificación del árbol todos los nodos (combinaciones de las soluciones) de un nivel deben ser expandidos antes de alcanzar un nuevo nivel, ya que para poder elegir la rama del árbol que va a ser explorada, se deben de conocer todas las ramas posibles.

Todos los nodos que no han sido explorados se almacenan en lo que se denomina Lista de nodos vivos (lo llamaremos noVisited), nodos pendientes de expandir el algoritmo.

**Estrategia FIFO:** Para esta estrategia la lista de nodos vivos es una cola (en este caso es una lista llamada noVisited que se simula una cola) en la cual su recorrido lo hace por niveles (en anchura), esto con el fin de recorrer y generar todas las combinaciones (estas combinaciones se van generando y modificando en la lista de escenarios llamada visited). Por lo cual este recorrido tiene una medición de tipo factorial (se mejora con la poda).

**Estrategia de Poda:** el objetivo principal de estar podando los nodos es obtener la solución del mínimo costo de la producción que se está dando. Se trabaja con una estrategia que logra un problema de minimización, donde se han recorrido varios nodos, estimando una cuota inferior de restricciones mencionadas anteriormente. Entonces, si se obtiene una posible solución válida para el problema con un beneficio en el cual es la reducción de los recursos y días ociosos, estos siendo menores (los beneficios) que los demás, nos llevará a la conclusión de que se puede podar sin que se pierda ninguna posible solución óptima.

**En resumen:**

El algoritmo busca una solución que tiene asociado un costo en el cual busca un mínimo costo (que es el mínimo de combinaciones que sean permitidas para obtener el mínimo costo de producción) llamado BestCalendar (objeto en el cual guarda la mejor combinación, así como su costo). Además de ramificar una solución padre (branch) en hijos, se trata de eliminar las consideraciones de aquellos hijos cuyos descendientes no son factibles (validación de las restricciones) para así asegurar que dicha combinación si se pueda realizar. La forma de acotar se realiza de la manera mencionada anteriormente, esta reduce el tiempo de búsqueda de la solución óptima al “podar” (pruning) los subárboles de descendientes no factibles.

**1. Selección:** extrae un nodo entre el conjunto de los nodos vivos que están almacenados en una lista.

**2. Ramificación:** se construyen los posibles nodos hijos del nodo seleccionado en la etapa anterior.

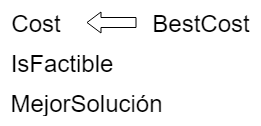
**3. Poda:** Se eliminan algunos de los nodos solución obtenidos de acuerdo con el incumplimiento de las restricciones planteadas. Aquellos nodos no podados pasan a formar parte del conjunto de nodos vivos, y se comienza de nuevo por el proceso de selección.

# Algoritmo Genético

Según (Herrera, F., Lozano, M., & Verdegay, J. L. ,1995) los algoritmos genéticos “...son procedimientos adaptativos para la búsqueda de soluciones en espacios complejos inspirados en la evolución biológica, con patrones de operaciones basados en el principio darwiniano al entorno en el que viven” En síntesis los algoritmos genéticos tratan de generar una población inicial con la que se espera realizar una secuencia de cruces para obtener un ajuste adecuado en la solución. Los nuevos individuos (candidatos a solución) se crean mediante los operadores genéticos y operadores de mutación. Este proceso de alteración de la población inicial se repite hasta que se encuentra la solución adecuada (en nuestro caso la minimización del coste de rodaje de la película).

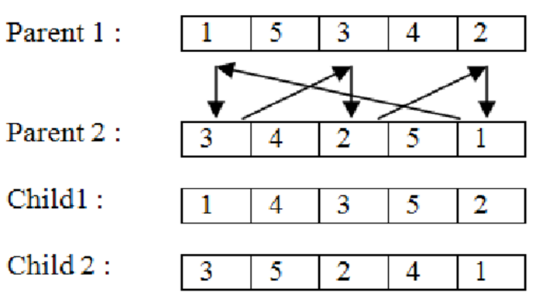
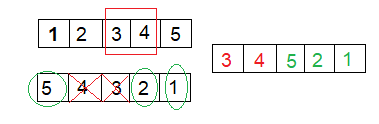
Población inicial:como se mencionó anteriormente los algoritmos genéticos necesita de una población inicial con la cual iniciar el proceso de cruce y procreación de nuevos hijos. Para generar la población inicial tomamos la lista de escenas ordenadas y las asignamos al padre, luego de esto tomamos la lista de escenas inversa y las asignamos a la madre.

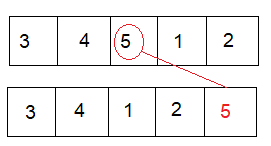
Evaluación o función de ajuste:corresponde a la función evaluadora la cual se encarga de verificar que tan óptimo es un individuo o que tanto se adapta este al medio. Para validar si un individuo es apto para ser solución se tomaron dos factores en cuenta para su calificación: la factibilidad y el coste de implementación. Para la factibilidad se realiza una evaluación para determinar si cada solución generada incumple alguna restricción, y de esta manera se descartan las que no son soluciones. Con respecto al coste, se calcula obteniendo el valor total de cada recurso (actor) y luego se suman obteniendo el total de coste de contratación para todos los actores en calendario de grabación generado y con este se puede verificar cuál de los calendarios generados es el mejor o más óptimo.

Como se muestra en la figura, la validación inicial se realiza sobre el coste del calendario. Si es menor al coste mínimo actual (inicialmente se toma el máximo int) entonces se procede a hacer la validación de su factibilidad donde se toma en cuenta si se incumple alguna de las restricciones. Sí no incumple ninguna de esta, entonces se asigna como el mejor calendario encontrado.

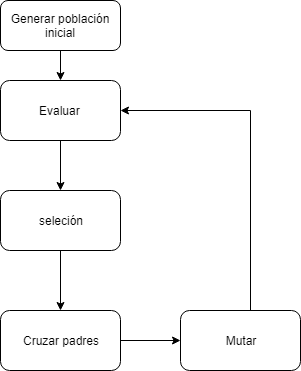
Cruces: el principal objetivo del cruce es mantener una población diversa de soluciones en cada generación. El cruce, también llamado cruce, crea nuevos individuos (hijos) a través de la recombinación de los genes de individuos ya existente (padres).

Se plantea la utilización de dos tipos diferentes de cruce para las poblaciones:

*  Cycle Crossover: el operador Cycle Crossover identifica una serie de ciclos llamados entre dos cromosomas padres. Luego, para formar el Niño 1, el ciclo uno se copia del padre 1, el ciclo 2 del padre 2, el ciclo 3 del padre 1, y así sucesivamente.
*  OX1 Crossover: es un cruce de permutación bastante simple. Básicamente, se toma una franja de alelos del padre uno y se marcan en el padre dos, luego se procede agregar los elementos seleccionados al hijo y se completa con los elementos no marcados del padre dos. Este mismo proceso se hace en sentido contrario para generar el segundo hijo.

Mutación: para el proceso de mutación se eligió la mutación por intercambio, esta se realiza cuando uno de los hijos generados ya se ha generado anteriormente y realiza tomando el valor del centro del arreglo de escenas y agregándolo al final de la lista, esto le brinda variabilidad las poblaciones generadas. Como se ve en la siguiente imagen, la mutación se produce tomando gen de en medio del cromosoma y colocándolo al final de la cadena genética.

En la siguiente imagen se presenta el ciclo evolutivo del algoritmo genético:

Este proceso evolutivo tiene como objetivo encontrar varias generaciones de posibles soluciones. En el proceso de evolución se determina cual solución es la mejor y también termina el algoritmo en caso de haber realizado la cantidad de generaciones solicitada. La selección elige cuáles de los cromosomas son más prometedores y así envía nuevos elementos para realizar los cruces. Estos son dos diferentes como se explica en la parte anterior, su diferencia radica en el algoritmo que utilizan para intercambiar segmentos de alelos del cromosoma de un padre con el otro.

## Escenarios:

Como se ha mencionado anteriormente, los escenarios son el principal factor para optimizar los horarios del rodaje de las películas, por lo que estas adquieren un gran valor y por ello contienen escenas que realizarán los actores, localidades donde se va a grabar las películas, los actores de estas como los roles y las jornadas laborales de los actores.

* Escenario #1: este contiene cuatro escenas, cuatro localidades, cinco actores y cuatro días de grabación. Cada escena tiene asignado dos actores y una localidad. Tanto actores como localidades poseen una lista de días disponibles que será en los días que tienen disponible para grabar.
* Escenario #2**:** este contiene cinco escenas, cinco localidades, seis actores y cinco días de grabación. Cada escena tiene asignado dos actores y una localidad. Tanto actores como localidades poseen una lista de días disponibles que será en los días que tienen disponible para grabar.
* Escenario #3**:** este contiene seis escenas, seis localidades, siete actores y seis días de grabación. Cada escena tiene asignado dos actores y una localidad. Tanto actores como localidades poseen una lista de días disponibles que será en los días que tienen disponible para grabar.
* Escenario #4**:** este contiene siete escenas, siete localidades, ocho actores y siete días de grabación. Cada escena tiene asignado dos actores y una localidad. Tanto actores como localidades poseen una lista de días disponibles que será en los días que tienen disponible para grabar.

## Salidas

La idea es que se desplieguen en pantalla los diferentes escenarios y que por medio de estos se puedan observar las comparaciones de los dos algoritmos en los cuales contienen las variables de medición(memoria) para ver la cantidad de esta consumida por estos; Además, en el caso de los dos algoritmos se debe mostrar la combinación encontrada para los tamaños pequeños. Por parte del algoritmo de ramificación y poda, se muestra la solución parcial de un nivel X en el cual contiene todos los nodos expandidos, también agregar la cantidad de nodos creados en el árbol. Por parte del algoritmo genético, se muestra los cruces realizados y su mutación, así como la puntuación asignada a cada cromosoma dado.

# Análisis de Resultados

En las siguientes tablas se muestran los resultados finales, estos divididos por partes como: descripción, completo, incompleto o defectuoso y el porqué. Esto con el fin de ver más claramente los resultados del sistema con los dos algoritmos: Ramificación y poda al igual que el genético.

Tabla para la **generación de datos** al igual que las **relaciones** entre estos.

|  |  |  |
| --- | --- | --- |
| **Titulo** | **Estado** | **Descripción** |
| Generación de escenas y sus relaciones | Completado |  |
| Generación de recursos y sus relaciones | Completado |  |
| Generación de localidades y sus relaciones | Completado |  |
| Generación de jornadas y sus relaciones | Completado |  |

Tabla para la **Ramificación y Poda**

|  |  |  |
| --- | --- | --- |
| **Descripción** | **Estado** | **Razón** |
| Ramificación tipo FIFO | Completado | Por cuestiones de relaciones entre las entidades, no se puede implementar el LC FIFO. |
| Poda | Completado |  |
| Generación del mejor Calendario | Completado |  |
| Función evaluadora | Completado |  |
| Obtener los costos de las combinaciones | Completado |  |
| Impresión de la medición empírica | Completado |  |
| Obtener la cantidad de nodos creados | Completado |  |
| Obtener la combinación en un nivel X | Completado |  |
| Obtener la cantidad de memoria consumida | Completado |  |

Tabla para la **Algoritmo Genético**

|  |  |  |
| --- | --- | --- |
| **Titulo** | **Estado** | **Descripción** |
| Generación de la población inicial | Completado | Se generan dos padres para comenzar el proceso evolutivo |
| Función de ajuste (Evaluación) | Completado | Se evalúan las soluciones para ver si son factible y es la mejor |
| Selección | Completado | se seleccionan los nuevos operadores de cruce |
| Aplicación del primer Cruce Genético | Completado | Se utilizó el cycle crossover |
| Aplicación del segundo Cruce Genético | Completado | Se utilizó el order one crossover |
| Mutación | Completado | Mutación por intercambio de genes |
| Optimización de la función objetivo | Completado | Se encuentra el calendario con menor coste |
| Generación de descendientes | Completado | Se generan hasta 10 generaciones para las pruebas |
| Mediciones | Completo |  |

# Mediciones

* Medición Empírica del Ramificación y Poda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operaciones** | **Tamaño de la Lista** | | | |
| **4** | **5** | **6** | **7** |
| **Asignaciones** | **2447** | **17075** | **133193** | **1006597** |
| **Comparaciones** | **1181** | **8766** | **70981** | **591868** |
| **Memoria** | **10** | **35** | **158** | **968** |

* Medición Empírica del Algoritmo Genético (Cycle Crossover)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operaciones** | **Tamaño de la Lista** | | | |
| **4** | **5** | **6** | **7** |
| **Asignaciones** | **3448** | **4546** | **5004** | **1018** |
| **Comparaciones** | **1866** | **2521** | **2824** | **647** |
| **Memoria** | **116** | **125** | **167** | **171** |

* Medición Empírica del Algoritmo Genético (Orden One)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operaciones** | **Tamaño de la Lista** | | | |
| **4** | **5** | **6** | **7** |
| **Asignaciones** | **1589** | **301** | **4441** | **3133** |
| **Comparaciones** | **1016** | **183** | **2832** | **2144** |
| **Memoria** | **89** | **97** | **109** | **124** |

# Factores de crecimiento

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Talla** | **Factor talla** | **Factor Asig** | **Factor Comp** | **Factor Memoria** |
| **De 4 a 6** | **1.5** | **54.43** | **60.10** | **15.8** |
| **De 4 a 7** | **1.75** | **411.35** | **501.15** | **96.8** |
| **De 5 a 6** | **1.2** | **7.80** | **8.09** | **4.51** |
| **De 5 a 7** | **1.4** | **58.95** | **67.51** | **27.65** |
| **De 4 a 5** | **1.25** | **6.97** | **7.42** | **3.5** |

Factor Crecimiento en Ramificación y Poda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Talla** | **Factor talla** | **Factor Asig** | **Factor Comp** | **Factor Memoria** |
| **De 4 a 6** | **1.5** | **1.45** | **1.51** | **1.43** |
| **De 4 a 7** | **1.75** | **0.29** | **0.34** | **1.47** |
| **De 5 a 6** | **1.2** | **1.10** | **1.12** | **1.33** |
| **De 5 a 7** | **1.4** | **0.22** | **0.25** | **1.36** |
| **De 4 a 5** | **1.25** | **1.31** | **1.35** | **1.077** |

Factor Crecimiento en el Algoritmo Genético (Cycle Crossover)

|  |  |
| --- | --- |
| **Clasificación del comportamiento de las asignaciones** | En general tienen un comportamiento **O(N)**, pero fluctúa un poco más arriba del factor talla o un poco menos. |
| **Clasificación del comportamiento de las comparaciones** | En general tienen un comportamiento **O(N)**, pero fluctúa un poco más arriba del factor talla o un poco menos. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Talla** | **Factor talla** | **Factor Asig** | **Factor Comp** | **Factor Memoria** |
| **De 4 a 6** | **1.5** | **2.79** | **2.78** | **1.22** |
| **De 4 a 7** | **1.75** | **1.97** | **2.11** | **1.39** |
| **De 5 a 6** | **1.2** | **14.75** | **15.47** | **1.12** |
| **De 5 a 7** | **1.4** | **10.40** | **11.71** | **1.58** |
| **De 4 a 5** | **1.25** | **0.18** | **0.18** | **1.089** |

Factor Crecimiento en el Algoritmo Genético (Orden One)

|  |  |
| --- | --- |
| **Clasificación del comportamiento de las asignaciones** | En general se comporta **O(N)**, este fluctuando un poco más arriba y abajo de la talla, solo en determinados casos si se incrementa bastante. |
| **Clasificación del comportamiento de las comparaciones** | En general se comporta **O(N)**, este fluctuando un poco más arriba y abajo de la talla, solo en determinados casos si se incrementa bastante. |

# Análisis Analítico

## Método Ramificación y Poda

|  |  |
| --- | --- |
| **public void goAlgorithm(List<Scene> notVisited, ,bool isCombination) {** |  |
| **if (notVisited.Count == 0) {** | **1** |
| **if (evaluation.isFactible(visited)) {** | **20N^2-4N** |
| **if (evaluation. getCostCalendar(visited) <= bestCalendar.bestCost ) {** | **6N^2+10N+3** |
| **changeBestCalendar(visited); return;}}}** | **1** |
| **foreach(Scene scene in notVisited) {** | **N+1** |
| **this.visited.Remove(scene); this.visited.Add(scene);** | **2N** |
| **List<Scene> auxScene = evaluation.shallowClone(notVisited);** | **1\*N** |
| **auxScene.Remove(scene);** | **1\*N** |
| **goAlgorithm(auxScene,isCombination);}}** | **26N^2+11N+6** |
| **Total:** | **52N^2+22N+12** |

**Método isFactible**

|  |  |
| --- | --- |
| **public bool isFactible(List<Scene> listScenes) {** |  |
| **for (int i = 0; i < listScenes.Count; i++) {** | **2N** |
| **listScenes[i].dayF = stage.filmingDays[i]; }** | **2N-1** |
| **foreach (Actor a in stage.actors) {** | **N+1** |
| **for (int i = 1; i < listScenes.Count-1; i++){** | **N\*(2N)** |
| **if (listScenes[i].listActors.Contains(a)) {** | **1\*(2N-1) \*N** |
| **if (listScenes[i - 1].listActors.Contains(a)) {** | **1\*(2N-1) \*N** |
| **if (listScenes[i].dayF.isDay) {** | **1\*(2N-1) \*N** |
| **if (!listScenes[i - 1].dayF.isDay){** | **1\*(2N-1) \*N** |
| **return false;}}** |  |
| **else{**  **if (listScenes[i - 1].dayF.id.Equals(listScenes[i].dayF.id))**  **return false;}}** | **1\*(2N-1) \*N** |
| **if (listScenes[i + 1].listActors.Contains(a)){** | **1\*(2N-1) \*N** |
| **if (listScenes[i].dayF.isDay){** | **1\*(2N-1) \*N** |
| **if (listScenes[i + 1].dayF.id.Equals(listScenes[i].dayF.id))**  **return false;}** | **1\*(2N-1) \*N** |
| **else{**  **if (listScenes[i + 1].dayF.isDay)}}}}}** | **1\*(2N-1) \*N** |
| **Return true;** |  |
| **Total:** | **20N^2-4N** |

**Método getCostCalendar**

|  |  |
| --- | --- |
| **public int getCostCalendar(List<Scene> scenes ){** |  |
| **for (int i = 0; i < listScenes.Count; i++) {** | **2N** |
| **listScenes[i].dayF = stage.filmingDays[i];}** | **2N-1** |
| **int totalCost = 0;** | **1** |
| **setCostActores(scenes);** | **6N^2+4N+1** |
| **foreach (Actor a in stage.actors){** | **N+1** |
| **totalCost += a.costTotal; asig++;}** | **N** |
| **return totalCost;** | **1** |
| **Total:** | **6N^2+10N+3** |

**Método setCostActores**

|  |  |
| --- | --- |
| **private void setCostActores(List<Scene> scenes) {** |  |
| **foreach (Actor a in stage.actors) {** | **N+1** |
| **bool asigned=false;** | **N** |
| **foreach (Scene e in scenes) {** | **N\*(N+1)** |
| **if (e.listActors.Contains(a)) {** | **1\*N\*(N)** |
| **if (!asigned) {** | **1\*N\*(N)** |
| **a.firstDay = e.dayF;** | **1\*N\*(N)** |
| **asigned = true;}** | **1\*N\*(N)** |
| **a.lastDay = e.dayF; asig++;}}** | **1\*N\*(N)** |
| **a.costTotal = ((a.lastDay.numDia - a.firstDay.numDia) + 1) \* a.costXDay; asig++;** | **1\*N** |
| **Total:** | **6N^2+4N+1** |

## Método Algoritmo Genético

**Método generar la población inicial**

|  |  |
| --- | --- |
| **private void generateStartPopulation(){** |  |
| **father = stage.scenes;** | **1** |
| **List<Scene> aux = evaluating.shallowClone(stage.scenes);** | **1** |
| **aux.Reverse();** | **1** |
| **mother = aux;** | **1** |
| **Total:** | **4** |

**Método Evaluación**

|  |  |
| --- | --- |
| **public void evaluation(){** |  |
| **int cost = evaluating.getCostCalendar(father);** | **6N^2+10N+3** |
| **if (cost < bestOne.bestCost){** | **1** |
| **if (evaluating.isFactible(father)){** | **20N^2-4N** |
| **bestOne.listScenes = evaluating.shallowClone(father);**  **bestOne.bestCost = cost;}}** | **2** |
| **cost = evaluating.getCostCalendar(mother);** | **6N^2+10N+3** |
| **if (cost < bestOne.bestCost) {** | **1** |
| **if (evaluating.isFactible(mother)){** | **20N^2-4N** |
| **bestOne.listScenes = evaluating.shallowClone(mother);**  **bestOne.bestCost = cost;}}** | **2** |
| **if (sonOne != null && sonTwo != null){** | **2** |
| **cost = evaluating.getCostCalendar(sonOne);** | **6N^2+10N+3** |
| **if (cost < bestOne.bestCost){** | **1** |
| **if (evaluating.isFactible(sonOne)){** | **20N^2-4N** |
| **bestOne.listScenes = evaluating.shallowClone(sonOne);**  **bestOne.bestCost = cost;}}** | **2** |
| **cost = evaluating.getCostCalendar(sonTwo);** | **6N^2+10N+3** |
| **if (cost < bestOne.bestCost){** | **1** |
| **if (evaluating.isFactible(sonTwo)){** | **20N^2-4N** |
| **bestOne.listScenes = evaluating.shallowClone(sonTwo);**  **bestOne.bestCost = cost;}}}** | **2** |
| **if (runner != 3){** | **1** |
| **runner += 1;** | **1** |
| **Selection();** | **8N^2+24N+29** |
| **}** |  |
| **Total:** | **112N^2+48N+57** |

**Método Selección**

|  |  |
| --- | --- |
| **private void selection(){** |  |
| **if (sonTwo != null) {** | **1** |
| **father = evaluating.shallowClone(sonOne);**  **mother = evaluating.shallowClone(sonTwo);** | **2** |
| **if (onCycle)** | **1** |
| **Crossover1();** | **8N^2+20N+3** |
| **else Crossover2();** | **4N+22** |
| **Total:** | **8N^2+24N+29** |

**Método Crossover1**

|  |  |
| --- | --- |
| **private void Crossover1(){** |  |
| **List<List<Scene>> aux = new CycleCrossover().PerformCross(father, mother);** | **8N^2+20N-3** |
| **family.Add(father);** | **1** |
| **family.Add(mother);** | **1** |
| **sonOne = evaluating.shallowClone(mutation(sonOne, "Hijo #1"));** | **1** |
| **sonTwo = evaluating.shallowClone(mutation(sonTwo, "Hijo #2"));** | **1** |
| **family.Add(sonOne);** | **1** |
| **family.Add(sonTwo);** | **1** |
| **evaluation();** |  |
| **Total** | **8N^2+20N+3** |

**Método CycleCrossover**

|  |  |
| --- | --- |
| **public List<List<Scene>> PerformCross(List<Scene> parent1, List<Scene> parent2){** |  |
| **var cycles = new List<List<int>>();** | **1** |
| **List<Scene> offspring1 = createNew(parent1.Count);** | **2N** |
| **List<Scene> offspring2 = createNew(parent1.Count);** | **2N** |
| **for (int i = 0; i < parent1.Count; i++){** | **2N** |
| **if (!cycles.SelectMany(p => p).Contains(i)) {** | **2N-1** |
| **var cycle = new List<int>();** | **2N-1** |
| **CreateCycle(parent1, parent2, i, cycle);** | **14** |
| **cycles.Add(cycle);}}** | **2N-1** |
| **for (int i = 0; i < cycles.Count; i++){** | **2N** |
| **var cycle = cycles[i];** | **2N-1** |
| **if (i % 2 == 0){** | **2N-1** |
| **CopyCycleIndexPair(cycle, parent1, offspring1, parent2, offspring2);}** | **8N-2\*(N-1/2)** |
| **else{**  **CopyCycleIndexPair(cycle, parent1, offspring2, parent2, offspring1);}}** | **8N-2** |
| **return new List<List<Scene>>() { offspring1, offspring2 };** | **2** |
| **Total** | **8N^2+20N-3** |

**Método Crossover2**

|  |  |
| --- | --- |
| **private void Crossover2(){** |  |
| **sonOne = new OrderOne().OrderOneCrossover(father, mother);** | **2N+8** |
| **sonTwo = new OrderOne().OrderOneCrossover(mother, father);** | **2N+8** |
| **family.Add(father);** | **1** |
| **family.Add(mother);** | **1** |
| **sonOne = evaluating.shallowClone(mutation(sonOne, "Hijo #1"));** | **1** |
| **sonTwo = evaluating.shallowClone(mutation(sonTwo, "Hijo #2"));** | **1** |
| **family.Add(sonOne);** | **1** |
| **family.Add(sonTwo);** | **1** |
| **evaluation();** |  |
| **Total** | **4N+22** |

**Análisis empírico del algoritmo de Ramificación y poda:**

**52N^2+22N+12**

**Análisis empírico del algoritmo genético(CycleCrossover):**

**112N^2+44N+35**

**Análisis empírico del algoritmo genético(OrderOne):**

**104N^2+28N+60**

# Conclusiones

Según las mediciones realizadas con los algoritmos de Ramificación y Poda y genético, se determina por medio de las mediciones anteriormente mostradas que el genético es el más eficiente ya que su comportamiento no es factorial sino es lineal O(N) con pocas fluctuaciones. Sin embargo, el algoritmo de ramificación y poda siempre va a dar la solución correcta entre todas ya que este recorre todas las posibles combinaciones por lo cual se ejecuta de manera adecuada también.

Otro punto importante es la siguiente pregunta: ¿Conforme crece la talla cuál algoritmo se va haciendo más eficiente?, para responder a la pregunta dada, hay que observar detalladamente las mediciones anteriores; por lo que el algoritmo genético muestra es, por ejemplo: con un tamaño de lista de siete, se redujo bastante por lo que tiende a ser cada vez mejor si el tamaño de la misma se va incrementando.

# Recomendaciones

* Se recomienda para este tipo de problemas usar la ramificación LC FIFO/LIFO, ya que este tiende a ser mejor que las utilizadas anteriormente, esto por su forma de ramificar con un sentida más óptimo.
* Si hay cruces mejores que los utilizados en el algoritmo genético, no dudar en usarlos, estos mejoran el algoritmo en sí; de esta manera se mejoran las combinaciones para cada generación.
* Se recomienda implementar otro tipo de mutación, ya que esta puede mejorar las generaciones y, a su vez, el costo también.

# Referencias

Herrera, F., Lozano, M., & Verdegay, J. L. (1995). Algoritmos genéticos: Fundamentos, extensiones y aplicaciones. *Arbor*, *152*(597), 9.

Universidad Nacional del nordeste. (2015). Análisis de algoritmos. 223/5/2018, de Universidad Nacional del nordeste Sitio web: <http://exa.unne.edu.ar/informatica/programacion1/public_html/archivos/tema10_algoritmos.pdf>