

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“GENERACIÓN AUTOMÁTICA DE HISTORIAS SECUNDARIAS DE NPCs UTILIZANDO ALGORITMOS EVOLUTIVOS”

RENATA PAZ MELLA GODOY

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Elizabeth Montero Ureta
Profesor Correferente: María Cristina Riff

Noviembre - 2018

DEDICATORIA

Este trabajo está dedicado a mi familia, quienes me han apoyado durante todo este tiempo. En especial a mi madre y mi padre que han sido quienes me han dado todas las herramientas para estar acá y sin ellos no sería la persona que soy ahora.

AGRADECIMIENTOS

En primer lugar quisiera agradecer a mis padres María Paz y Pedro, quienes me apoyaron durante todo el proceso. A mi madre le agradezco que siempre se preocupara de mi y mi salud, encargándose que nunca dejara de almorzar (aunque eso significara quedarse cocinando hasta la noche). A mi padre, que siempre se preocupó de que llegara bien a mi hogar, yéndome a buscar a la universidad incluso hasta en la madrugada luego de largas jornadas de estudio. También agradezco a mi hermana y a mi abuela por su preocupación.

A mi pololo, Ariel, quien me dio ánimos en los momentos más difíciles y que creyó en mi hasta el final.

A mis amigos de la universidad, los golpistas. Agradezco todas nuestras noches de estudio y el apoyo que nos brindábamos mutuamente cuando se acercaban los certámenes. También agradezco a todos los compañeros que conocí a lo largo de las asignaturas; me alegraban el día cuando compartíamos fuera de clases y de la universidad. Todos ellos hicieron que fuera más fácil llevar la carga.

A mis profesores de la universidad, quienes tuvieron fe en mí. En especial a la profesora Valeria del Campo, quien fue la primera persona que me motivó a ser ayudante y vio en mi cualidades especiales. Agradezco también a la profesora Elizabeth Montero, quien fue mi profesora guía en el trabajo final y de la cual aprendí mucho. Estoy muy agradecida de su preocupación y la disposición que siempre tuvo para ayudarme a terminar mi trabajo.

A mis amigos del colegio, quienes han estado ahí conmigo. A pesar de que no nos vemos muy seguido, los momentos que compartimos siempre son de calidad.

Por último agradecer a todas las personas que conocí en el camino y que ayudaron a hacer más grata mi estadía en la universidad. Todos ellos de una u otra manera marcaron este proceso y dejaron una huella que no olvidaré.

RESUMEN

Resumen— En el mundo actual, el área de la inteligencia artificial aplicada en videojuegos crece a pasos agigantados: desde un programa capaz de jugar ajedrez hasta un personaje de un juego capaz de tomar decisiones, la inteligencia artificial ha estado presente en el desarrollo de los videojuegos casi desde su inicio. En esta memoria se busca aportar al desarrollo de videojuegos optimizando el modelo de tres capas propuesto por [García-Ortega et al., 2016] para la generación de historias de *Non-Player Characters (NPCs)* (a través del surgimiento del *monomito*). Para esto se propone mejorar la capa de computación evolutiva reemplazando el algoritmo genético utilizado actualmente por una estrategia evolutiva, encontrando mejores soluciones en menos evaluaciones.

Palabras Clave— inteligencia artificial; computación evolutiva; videojuegos; monomito.

ABSTRACT

Abstract— In today's world, the area of artificial intelligence applied in video games is growing by leaps and bounds: from a program capable of playing chess to a character in a game capable of making decisions, artificial intelligence has been present in the development of video games since the beginning. This thesis seeks to contribute to the development of video games by optimizing the three-layer model proposed by [García-Ortega et al., 2016] for the generation of *NPCs* stories (through the emergence of monomyth). For this, it is proposed to improve the layer of evolutionary computation replacing the genetic algorithm currently used by an evolutionary strategy, finding better solutions in fewer evaluations.

Keywords— artificial intelligence; evolutionary computing; videogames; monomyth.

ACRÓNIMOS

AE Algoritmos Evolutivos **Glosario:** [algoritmos evolutivos](#) .

CUDA *Compute Unified Device Architecture* **Glosario:** [CUDA](#).

GPU *Graphics Processing Unit*. **Glosario:** [GPU](#).

HEFTI *Hybrid Evolutionary-Fuzzy Time-based Interactive*.

NPC *Non-Player Character*. **Glosario:** [NPC](#) .

PCG *Procedural Content Generation* **Glosario:** [PCG](#).

PRNG *Pseudo-Random Number Generator* **Glosario:** [PRNG](#).

RPG *Role Playing Game*.

GLOSARIO

Algoritmos Evolutivos Son algoritmos que se basan en el concepto de la evolución natural como un proceso de optimización. Se basan en una población de puntos de búsqueda inicializada arbitrariamente la cual evoluciona hacia mejores regiones del espacio de búsqueda por medio de procesos tales como la selección, mutación y recombinación. [[Bäck et al., 1993](#)].

Arquetipo Es un modelo o ejemplo de ideas o conocimiento del cual se derivan otros tantos para modelar los pensamientos y actitudes propias de cada individuo, de cada conjunto, de cada sociedad, incluso de cada sistema. Por ejemplo, dentro de la literatura existen arquetipos marcados y que se repiten con frecuencia como lo son el *héroe* y el *villano* (conocido formalmente como la *sombra*). El primero suele ser el protagonista que emprende una aventura con el objetivo de cumplir un ideal, aceptando cualquier desafío e inclusive sacrificándose a sí mismo, mientras que el segundo representa la fuerza que se opone al *héroe* y que normalmente busca destruirlo o derrotarlo.

Búsqueda Evolutiva Similar a los [algoritmos evolutivos](#), estos se basan en la evolución para moverse en un espacio de búsqueda con el objetivo de encontrar mejores soluciones.

CUDA Es una plataforma de computación en paralelo que incluye un compilador y un conjunto de herramientas de desarrollo creadas por *nVidia*¹ que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en [Graphics Processing Unit \(GPU\)](#) de *nVidia*.

¹Empresa multinacional especializada en el desarrollo de unidades de procesamiento gráfico y tecnologías de circuitos integrados para estaciones de trabajo, ordenadores personales y dispositivos móviles.

GPU Es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.

Monomito También conocido como el viaje del héroe, es un término acuñado por el antropólogo y mitólogo estadounidense Joseph Campbell para definir el modelo básico de muchos relatos épicos de todo el mundo. Este patrón está descrito por Campbell en su conocida obra El héroe de las mil caras. [[Campbell, 2008](#)].

NPC Personaje automatizado dentro de un videojuego cuya principal característica es que el jugador no puede controlar sus acciones directamente.

PCG Método para crear contenido (data) de forma automática a través de algoritmos.

PRNG Es un programa escrito y usado en aplicaciones de probabilidad y estadísticas cuando se necesitan grandes cantidades de dígitos aleatorios. La mayoría de estos programas producen cadenas interminables de números de un solo dígito, por lo general en base 10.

Videojuegos de mundo abierto Son aquellos juegos que ofrecen al jugador la posibilidad de moverse libremente por un mundo virtual y alterar cualquier elemento a su voluntad. Pese a su denominación, muchos videojuegos de mundo abierto mantienen ciertas restricciones en algunos puntos de la historia, ya sea debido a limitaciones absolutas de diseño del juego o limitaciones temporales (tales como zonas bloqueadas) impuestas por la linealidad del videojuego.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
ACRÓNIMOS	V
GLOSARIO	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	X
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Objetivo general	7
1.2 Objetivos específicos	8
CAPÍTULO 2: MARCO CONCEPTUAL	9
2.1 Estado del arte	9
2.1.1 Generación de contenido procedimental	9
2.1.2 Generación de historias a través de técnicas procedimentales	10
2.1.3 Sistemas de Drama Interactivos	11
2.1.4 Generación de historias para NPCs mediante métodos de generación de contenido procedimental y algoritmos de inteligencia artificial	14
2.2 Arquitectura del sistema	16
2.2.1 Capa ABM	16
2.2.2 Capa lógica	18
2.2.3 Capa de computación evolutiva	21
2.3 Marco Teórico	23
2.3.1 Algoritmos Genéticos	23
2.3.2 Estrategias Evolutivas	25
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	27
3.1 Representación de la solución	28
3.2 Función de Evaluación	29
3.3 Estructura General	30
3.3.1 Inicialización	31
3.3.2 Mutación	32
3.3.3 Criterios de Reinicio	34
3.3.4 Criterio de Término	35
3.3.5 Exploración del algoritmo	35

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	37
4.1 Detalles del código	37
4.2 Objetivos	37
4.3 Instancias	37
4.4 Algoritmo Genético	39
4.5 Estrategia Evolutiva	43
4.5.1 Máscara 100 %	44
4.5.2 Máscara 70 %	46
4.5.3 Máscara 50 %	49
4.5.4 Inicialización en Algoritmo Genético	50
CAPÍTULO 5: CONCLUSIONES	52
5.1 Trabajo Futuro	53

ÍNDICE DE FIGURAS

1	Arquitectura del sistema (alto nivel).	17
2	Dependencia de los predicados en la teoría lógica propuesta del monomito: los nodos blancos representan los predicados que pueden ser generados por la capa ABM y los nodos grises son predicados inferidos por el Razonador. Los nodos gris oscuro son llamadas facilitadores y conforman el monomito.	19
3	Estructura del individuo utilizado en la capa evolutiva y el significado de cada uno de sus 52 genes.	22
4	Cruzamiento BLX- α	24
5	Dos ejecuciones del algoritmo genético con función de aptitud A.	27
6	Resumen general del significado de los genes del individuo original. Se destacan los que son parte de la representación a utilizar.	28
7	Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.	41
8	Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud AD para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.	42
9	Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud M para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.	43
10	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	44
11	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud AD para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	45

12	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud M para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	46
13	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	47
14	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud AD para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	48
15	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud M para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	48
16	Gráficos de los resultados de la estrategia evolutiva usando una máscara del 50 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	49
17	Gráficos de los resultados del algoritmo genético usando la mejor solución de todas en la población inicial. Esto se realiza para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	50
18	Gráficos de los resultados del algoritmo usando la mejor solución de todas en la población inicial y aplicando elitismo. Esto se realiza para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.	51

ÍNDICE DE TABLAS

1	Comparación teórica de los arquetipos presentes en el monomito y su interpretación en el modelo.	20
2	Ejemplo de un individuo cualquiera.	29
3	Ejemplo de valores que podría tener el promedio de veces que apareció cada arquetipo durante las simulaciones. Los valores son reales y puede tomar cualquier valor positivo incluyendo el 0.	30
4	Ejemplo de una máscara. Los valores no pintados se establecen de manera aleatoria.	32
5	Resumen del significado de cada parámetro utilizado en la estrategia evolutiva. .	36
6	Características de las instancias que se utilizaron para las pruebas.	37
7	Valores de los parámetros que se utilizaron finalmente.	38
8	Resultados Algoritmo Genético con las tres instancia y semillas.	39
9	Promedio de aparición de cada arquetipo para una ejecución del algoritmo genético con la instancia 1 y semilla 1.	40

INTRODUCCIÓN

En videojuegos, los personajes no jugables (o también conocidos como [NPCs](#)) son un tipo de personaje que vive en el mundo del juego para proveer una experiencia más inmersiva al jugador e incluso, en algunos casos presenta un desafío al jugador. Los juegos modernos de rol conocidos como [Role Playing Games \(RPGs\)](#) incluyen cientos de estos personajes. El esfuerzo necesario para crear un buen guión interactivo va en directa proporción con la cantidad de estos en el juego, por lo que mientras más personajes haya, mayor será el costo de crear los guiones. Usualmente, los NPCs se limitan a conversaciones escritas de antemano (por ejemplo, para comprar y vender objetos al jugador), o incluso tienen comportamientos preestablecidos en los que interactúan con el jugador sólo si este realiza una acción determinada: por ejemplo, si el jugador roba algo de la ciudad, el guardia vendrá a atacarlo. Debido a que estos personajes están frecuentemente programados para interactuar con el jugador siguiendo una secuencia guiada de actividades sólo con este propósito, normalmente no interactúan entre ellos. En un mundo con tal cantidad de personajes, la interacción colectiva de estos podría mejorar significativamente la experiencia de juego, llevando a un entorno más inmersivo. Por ejemplo, los habitantes hambrientos podrían convertirse en ladrones, siendo perseguidos por los guardias.

Es importante destacar además que la industria de los videojuegos ha tenido un crecimiento considerable, así como también la complejidad de estos. Existe una vasta variedad de juegos que hace difícil presentar algo que sea innovador y no le resulte conocido o repetitivo al jugador. A esto hay que agregar que la duración de un juego suele estar limitada al tamaño del mundo o la cantidad de etapas que este posee; es por esto que hoy en día se han hecho popular los juegos de [mundo abierto](#) como *The Legend of Zelda: Breath of the Wild* u *Horizon Zero Dawn*, pues estos tienen una mayor duración y le permiten al jugador explorar y disfrutar el juego por la cantidad de tiempo que guste. Otros juegos como *Minecraft* y *Super Mario Maker* le permiten al jugador crear sus propios niveles y elementos, de manera que el límite de diversión solo recaiga en la imaginación del usuario, supliendo un poco la limitación de autogeneración de contenidos que tienen los videojuegos actualmente.

La excesiva pero necesaria cantidad de [NPCs](#) y la obligación de innovar han motivado a que se busquen métodos para generar estas historias de fondo que mejoren la experiencia de juego y hagan que el mundo virtual sea más interactivo. La generación de contenido procedimental conocida como [Procedural Content Generation \(PCG\)](#) es una línea de investigación que lleva vigente desde principios de 2000. Uno de sus ejemplos es el juego llamado “*kkrieger*”, el cual fue lanzado en 2006 y se destacaba por usar técnicas procedimentales para generar texturas, mallas y sonidos para crear un juego más inmersivo y complejo. Pero esto es solo un ejemplo del uso de algoritmos para crear contenido automáticamente; ya en 2002 hubo un acercamiento a crear historias de manera automática mediante el software llamado *The virtual storyteller* [[Theune et al., 2002](#)]. Otras aproximaciones que utilizan un enfoque de creación de contenido procedimental son los sistemas de dramas interactivos. Entre estos destacan el trabajo de Magerko con su software *Interactive Drama Architecture* (abreviado

IDA) [Magerko, 2010], en el cual el usuario se encuentra su propio cuerpo muerto y como fantasma debe descubrir a su asesino. En 2011, Malik Nairat y compañía [Nairat et al., 2011] crearon un drama interactivo utilizando un modelo basado en agentes y algoritmos genéticos para evolucionar la interacción entre estos. Posteriormente en 2014 se presentó un trabajo que utilizó algoritmos genéticos para encontrar la combinación de parámetros y el número de perfiles necesarios para hacer surgir dos **arquetipos**: el de venganza y el de control de natalidad [García-Ortega et al., 2014]. Luego ampliaron su modelo para que lograra emerger distintos tipos de **arquetipos**, utilizando un sistema multi-agente para demostrar la adecuación de su método basado en algoritmos genéticos. Al año siguiente, mejoraron su modelo ampliándolo a **algoritmos evolutivos** (abreviados como **AE**), definiendo una metodología que les permitiera encontrar los mejores valores para los parámetros tanto del AE como de la configuración del mundo ficticio. El año pasado, los mismos autores presentaron un modelo de tres capas: la capa del modelo basado en agentes, la capa de razonamiento lógico y la capa de la computación evolutiva, donde se incluye una implementación del algoritmo genético. Dicho modelo buscaba encontrar parámetros que resultaran apropiados para hacer surgir el **monomito** entre los personajes de un videojuego, de manera que en un futuro se pudieran utilizar para generar interacción entre los **NPCs** sin la necesidad de tener un guión preestablecido.

En este trabajo se estudió el modelo de tres capas con el objetivo de encontrar deficiencias a ser mejoradas posteriormente. La estructura comprende una definición del problema a tratar que incluye los objetivos generales y específicos a abordar en el futuro, un marco conceptual que incluye un estado del arte y un marco teórico. En este último se describe en detalle el funcionamiento del simulador, junto con las técnicas de computación evolutiva que serán empleadas más adelante. En el capítulo “propuesta de solución” se describe el proceso que se llevó a cabo para encontrar una solución adecuada al problema. Luego se valida dicha solución encontrada y se establecen conclusiones en base al análisis para finalmente cerrar con el trabajo futuro y la bibliografía.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

Como se mencionó anteriormente, este trabajo tratará de mejorar las técnicas actuales de generación de contenido procedimental enfocado en la creación de historias para [NPCs](#), con el fin de que, algún día, éstas puedan ser utilizadas en productos comerciales y de forma masiva.

Para hacer esto, la investigación se centrará en el modelo propuesto por [[García-Ortega et al., 2016](#)] de tres capas: la capa del modelo basado en agentes (llamada ABM, por su sigla en inglés), la capa de razonamiento lógico y la capa de computación evolutiva, que incluye la implementación del algoritmo genético. La teoría detrás de este modelo se centra en el [monomito](#), también conocido como el viaje del héroe. En su libro *The Hero with a Thousand Faces* [[Campbell, 2008](#)], Joseph Campbell describe los elementos principales que tienen normalmente los relatos épicos, es decir, establece un patrón común en diversas historias. Posteriormente, Christopher Vogler en su libro *The Writer's Journey* [[Vogler, 1998](#)] toma la idea de Campbell y sintetiza el monomito en 12 etapas descritas a continuación:

1. **Mundo Ordinario:** Se presenta el entorno regular del héroe para luego establecer un contraste. La idea es mostrar el mundo que abandona para realzar el contraste con el nuevo mundo al que será enviado.
2. **El llamado a la aventura:** Se le presenta al héroe un desafío, una aventura que emprender. Una vez que el llamado se presenta, el protagonista no puede permanecer indefinidamente en la comodidad de su mundo. El llamado a la aventura representa la meta, el objetivo que perseguirá el héroe.
3. **La negativa al llamado:** Es el momento de duda. El héroe se enfrenta a sus mayores temores y piensa en dar la vuelta. Se necesita de un cambio en las circunstancias para que el héroe encuentre su motivación definitiva y emprenda la aventura.
4. **Conociendo al mentor:** La función del mentor es preparar al héroe para enfrentar lo desconocido. Puede dar consejos, guía o equipamiento mágico. El mentor solo avanza un poco con el héroe, ya que finalmente este debe avanzar solo hacia lo desconocido. A veces es el mentor quien le da al héroe el empujón necesario para emprender la aventura.
5. **Cruzando el primer umbral:** Ahora el héroe está 100 % comprometido con la aventura y entra a este mundo desconocido, atravesando el primer umbral. Las películas generalmente se desarrollan en 3 actos: La decisión del héroe de actuar, el acto en sí y las consecuencias de su acción. El primer umbral marca el punto de inflexión entre los actos I y II. El héroe decide enfrentar el problema y tomar acción.

6. **Pruebas, aliados y enemigos:** Una vez cruzado el primer umbral, el héroe encuentra nuevos desafíos y pruebas, hace aliados y enemigos y comienza a aprender las reglas del nuevo mundo.
7. **Aproximación a la cueva más interna:** El héroe se aproxima al lugar peligroso donde se encuentra escondido el objeto de la búsqueda. Generalmente los héroes toman una pausa para planear el ataque. Esta fase se conoce como *Approach*². Esta fase cubre todas las preparaciones para entrar a la cueva más interna y confrontar la muerte o el peligro supremo.
8. **La prueba rigurosa:** El héroe se enfrenta a su mayor temor, la posibilidad de morir. Es el “momento negro” para la audiencia, donde se encuentran en suspenso y tensión, no sabiendo si el héroe saldrá vivo o muerto. Es la parte de la historia en donde el héroe debe morir o parecer que lo hace para así nacer de nuevo.
9. **Recompensa (agarrando la espada):** Habiendo sobrevivido a la muerte y vencido el peligro, el héroe toma posesión del tesoro que ha estado buscando, su recompensa. Puede ser un arma especial o incluso hasta sabiduría. Desde el punto de vista del héroe, los miembros del sexo opuesto pueden aparecer como *shapeshifters*³, un arquetipo que representa cambio. En este punto, el héroe se vuelve más atractivo como resultado de haber sobrevivido la prueba. Se ha ganado el título de “héroe” habiendo tomado el riesgo en favor de su comunidad.
10. **El camino de vuelta:** El héroe debe lidiar con las consecuencias de sus actos. Debe reconciliarse con sus padres, el dios, etc. para que las fuerzas violentas no vengan tras él. Marca la decisión de volver al mundo ordinario.
11. **La resurrección:** El héroe resucita y vuelve al mundo ordinario como un nuevo ser con nuevas perspectivas.
12. **El retorno con el elixir:** El héroe retorna al mundo ordinario pero no lo hace solo, pues el viaje habría sido insignificante. El héroe vuelve a su mundo con un elixir, un tesoro o una lección del mundo especial.

Dentro de las etapas mencionadas anteriormente se mencionan diversos [arquetipos](#). Es importante destacar que estos difieren de la personalidad, pues son patrones de la personalidad y no la personalidad en sí. De esta forma, un personaje puede pasar por diferentes arquetipos en distintas etapas. A continuación se detallan los 8 arquetipos presentes en el monomito:

- **Héroe:** El héroe se caracteriza por ser lo que Freud llamó el ego, es decir, esa parte de la personalidad que se separa de la madre, que se considera distinta del resto de la raza humana. El público se siente identificado con el personaje siempre que este parezca

²Se puede traducir como aproximación

³Traducido como cambiaformas

real. El héroe también debe ser capaz de aprender y crecer como persona. Es aquel que toma riesgos y realiza acciones; es bastante activo durante la historia. Está dispuesto a sacrificar su propia vida por un ideal o un grupo y acepta este sacrificio, enfrentando a la muerte. A veces, un personaje que no necesariamente es el héroe, puede actuar heroicamente (por ejemplo, el mentor puede sacrificarse por su discípulo). Las debilidades del héroe hacen que parezca más humano y más cercano al espectador. Existen distintos tipos de héroes: voluntarios e involuntarios, solitarios, orientados a grupos, anti-héroes, héroes trágicos y catalizadores.

- **Voluntarios e involuntarios:** Los héroes voluntarios están comprometidos con la aventura, son valientes, no tienen dudas y son motivados. Los involuntarios en cambio, están llenos de dudas, son pasivos, necesitan motivación de fuerzas externas. Generalmente se necesita que estos últimos cambien.
 - **Anti-héroes:** Puede ser un individuo que no sigue la ley o que es un villano desde el punto de vista social pero con el cual la audiencia simpatiza. Existen dos tipos de anti-héroes: El primero se caracteriza por ser un individuo marginado por la sociedad. A menudo son hombres honorables que se han retirado de una sociedad corrupta. Los segundos tipos de anti-héroes se caracterizan por seguir la idea clásica del héroe trágico. Estos son héroes defectuosos que nunca superan sus demonios internos y son derrotados y destruidos por ellos.
 - **Héroes orientados a grupos:** Son aquellos que se separan de su sociedad hacia una tierra desconocida. Se enfrentan a la decisión de retornar a su mundo ordinario o permanecer en el segundo acto.
 - **Solitario:** Es un héroe distanciado de la sociedad cuyo estado natural es la soledad.
 - **Héroes Catalizadores:** Más que cambiar ellos mismos, su acción es transformar al resto.
- **Mentor:** Usualmente es una influencia positiva. Actúa como la consciencia que nos guía en el camino de la vida cuando no hay nadie para protegernos. El mentor tiene la función de enseñar o entrenar. También es aquel que otorga dones o regalos, los que deben ser ganados por el héroe. Puede ser el que cree los artefactos que le serán otorgados al héroe, como una especie de científico, inventor o artesano. A veces el héroe puede rebelarse ante los mentores que aparecen como consciencia de este. El mentor es también quien motiva al héroe; a veces, este se ve obligado a empujarlo a la aventura. Existen también los mentores oscuros, que llevan al héroe por el camino de la destrucción. Algunos mentores todavía se encuentran en su propio camino del héroe. También puede aparecer el mentor como shaman, es decir, como sanador. Muchas historias no tienen un personaje específico que se identifique como mentor. Normalmente no tiene un lugar específico en la historia; puede aparecer al comienzo o esperar un momento crítico en el acto II y III.
 - **Guardián:** Tienen una función de neurosis: están para probar si el héroe está realmente determinado a aceptar el desafío del cambio. Cuando se confronta una de estas fi-

guras, el protagonista debe resolver un *puzzle* o pasar una prueba. Los guardianas se convierten temporalmente en el enemigo. Los héroes exitosos aprenden a reconocer estos personajes no como enemigos sino como útiles aliados e indicadores de que un nuevo poder se acerca.

- **Heraldo:** Es la fuerza que desafía al héroe. Anuncian un nuevo cambio. Usualmente es el que llama al héroe a la aventura; desbalancea las cosas y ya nada vuelve a ser igual. Llama al cambio. Proveen motivación al héroe al plantearle un desafío. Al igual que otros arquetipos, el heraldo no necesariamente es alguien físico; puede ser una fuerza. El heraldo puede ser una fuerza positiva, negativa o neutra. Un mentor actúa frecuentemente como heraldo que emite un desafío al héroe.
- **Cambiaformas:** Comúnmente es difícil dilucidar este arquetipo debido a su naturaleza cambiante e inestable. Su lealtad o sinceridad está siempre en duda. Expresan la energía femenina y masculina del personaje. A los hombres se les reprime socialmente su lado femenino, mientras que a las mujeres se les oprime el lado masculino. El arquetipo cambiaformas representa esta represión. Pueden ser figuras positivas o negativas que podrían ser útiles o destructivas al héroe. Este arquetipo representa también un catalizador de cambio. Trae duda al héroe y suspenso a la historia. Al igual que otros arquetipos, cualquier arquetipo puede asumir la máscara del cambiaformas. Los villanos o aliados pueden cambiar de forma para seducir o confundir al héroe. Los mentores también pueden aparecer como cambiaformas. Las comedias donde la historia se centra en dos personajes femeninos o masculinos que comparten el rol de héroe también involucran este arquetipo. Usualmente uno es más convencionalmente heroico y más fácil de identificar para la audiencia. El segundo personaje (del mismo sexo que el otro) actúa como el cambiaformas.
- **Sombra:** Representa la energía del lado oscuro, lo inexpresivo, irrealizado o rechazado de algo. Puede ser todas las cosas que no nos gustan de nosotros mismos, todos los oscuros secretos que no admitimos (incluso a nosotros mismos). Estos personajes pueden ser villanos, antagonistas o enemigos. Los villanos y enemigos usualmente se dedican a la muerte, destrucción o derrota del héroe. Los antagonistas en cambio, pueden no ser tan hostiles; podrían ser aliados que persiguen el mismo objetivo pero no están de acuerdo con la postura del héroe. Puede representar el poder de los sentimientos reprimidos como la culpa o un trauma muy profundo. Varias figuras sombrías son también cambiaformas. La función de una sombra es desafiar al héroe y darle un oponente que haga valer la lucha. Crea conflicto y lleva al héroe a una situación de vida o muerte. La sombra puede estar presente en diversos personajes. El héroe mismo podría manifestar un lado oscuro. El mentor puede llevar la máscara de sombra.
- **Aliado:** Los héroes en la literatura necesitan alguien con quien viajar que les sirva de compañía, de consciencia o simplemente los alegren. Realizan muchas tareas mundanas, pero su función no deja de ser importante. No necesariamente existe un único aliado. Este puede ser aquel que le muestre al héroe el nuevo mundo. Los aliados pueden ser no humanos (como espíritus o dioses) e incluso animales. En ocasiones pueden

ser sirvientes. Representa la parte inexpressada o no usada de la personalidad del héroe que debe poner en acción para realizar su trabajo.

- **Bufón:** Este arquetipo encarna la travesura y el deseo de cambio. Los personajes que son payasos o cómicos encarnan este arquetipo. Incluso existe un tipo especializado de héroe que se hace llamar héroe bufón. Tiene como función aterrizar al héroe y bajarle el ego. Este arquetipo libera la tensión y el suspenso a través de la risa. Los personajes bufones son, a menudo, catárticos que afectan la vida de los otros pero no la de ellos mismos.

El autor del modelo tradujo los arquetipos a su propia interpretación de universo y en base a esto armó un esquema de relaciones entre las acciones e interacciones de los personajes el cual, llevado a predicados lógicos, conforman la capa del razonamiento lógico. En esta se infieren si los arquetipos de manera simplificada se manifiestan o no. La idea general del sistema es la siguiente: a partir de parámetros establecidos por la capa evolutiva (específicamente por el algoritmo genético), se echa a andar la simulación (capa ABM) del mundo simplificado descrito por el autor en [[García-Ortega et al., 2016](#)]. Esta simulación genera una serie de eventos entre los personajes, los cuales son procesados por el razonador para determinar si surgen los arquetipos. Posteriormente, en base a la cantidad de arquetipos surgidos se mide la calidad de la solución proporcionada por la capa evolutiva y el proceso se vuelve a repetir sucesivamente.

En general, los [algoritmos evolutivos](#) [[Bäck, 1996](#)] han mostrado excelentes resultados a la hora de generar interacción entre los NPCs [[García-Ortega et al., 2014](#)]. Sin embargo, lograr un nivel de interacción en los NPCs que se asemeje a la vida real es muy complejo, por lo que seguir mejorando los modelos evolucionados basados en agentes es sólo uno de los pequeños pasos que se deben dar para desarrollar algoritmos testeables en juegos reales. Para esto, es imperativo analizar cada componente de los modelos presentados como también es necesario optimizar los algoritmos de testeo que permiten evaluar la función de aptitud.

En este trabajo se centrarán los esfuerzos en la capa evolutiva del modelo, pues uno de los problemas que se detectaron al estudiar el sistema es que si bien un algoritmo genético es capaz de encontrar buenas soluciones, la evaluación de la función de aptitud puede tomar un tiempo considerable (desde segundos o minutos hasta horas), considerando que cada evaluación implica correr la simulación de la capa ABM una cantidad determinada de veces para luego accionar la capa lógica. Además, se detectó que si bien las funciones utilizadas para medir la calidad de una solución son bastante robustas, podrían existir otros detalles que influyen en la calidad de una solución y que no fueron tomados en cuenta.

Objetivo general

Optimizar el modelo de tres capas basado en agentes que se utiliza actualmente para generar historias de NPCs que se acoplen mejor al juego y aumenten el nivel de interacción entre los

personajes.

Objetivos específicos

1. Analizar la línea de investigación actual de la generación de historias automáticas NPCs.
2. Identificar falencias en el modelo evolucionado basado en agentes.
3. Optimizar la capa evolutiva para encontrar mejores soluciones en base a otros criterios y reducir la cantidad de evaluaciones que se necesitan para esto.

CAPÍTULO 2

MARCO CONCEPTUAL

Estado del arte

Generación de contenido procedimental

Asegurar la calidad del contenido de un videojuego antes era una tarea manual. Esto representa nuevos y crecientes desafíos tanto para la población *gamer* como para los productores de juegos, lo que se traduce en un incremento de los costos. En un intento por ajustarse al presupuesto nace la generación de contenido procedimental (PCG por sus siglas en inglés) con el fin de automatizar en cierta medida, el desarrollo de un videojuego [Hendrikx et al., 2013]. A pesar de que la investigación ha avanzado, PCG sigue siendo una tarea difícil, puesto que el generador debe crear contenido que satisfaga las restricciones impuestas por el artista, además de generar instancias que resulten interesantes para los jugadores.

Una de las primeras y más simples aproximaciones es la generación de contenido basada en la generación de números pseudo-aleatoria (*Pseudo-Random Number Generator (PRNG)*). Otro intento de generación utilizando computadores es el juego de exploración espacial “Elite”, que utiliza contenido procedimental para generar el universo. En 1985, Perlin realizó pruebas para generar texturas para objetos [Perlin, 1985]. Desde la mitad de la década de los 80s, el grupo Demonscene [Dig] ha usado técnicas procedimentales para crear entornos virtuales que requieran una baja capacidad de memoria. El juego “.kkrieger” del 2006 es un *shooter* 3D que utiliza la generación de contenido procedimental para crear las texturas, mallas y sonidos que permiten otorgar una experiencia de juego más inmersivo. Para su creación se utilizaron dos técnicas principales: la primera es que no se almacenan las texturas en sí sino su historial de creación (esto hace que el juego ocupe menos memoria); la segunda es que las mallas son creadas a partir de formas sólidas básicas como cilindros y cajas, las que son deformadas para obtener nuevas formas. Existen otras técnicas que se enfocan en obtener resultados estocásticos, más que resultados determinísticos⁴ presentes en “.kkrieger” y “Elite”.

A pesar de la abundancia de técnicas PCG aplicadas a juegos, su aplicación en productos comerciales no es común. Métodos PCG han sido aplicadas exitosamente para generar diversos tipos de contenidos, pero las soluciones existentes no son de propósito general. Se espera que algún día, las técnicas PCG puedan ser aplicadas de forma masiva en diferentes videojuegos.

⁴Un resultado determinístico siempre produce el mismo resultado bajo los mismos parámetros. No así los resultados estocásticos.

Generación de historias a través de técnicas procedimentales

A través de los años, diversas investigaciones han abordado el tema de la generación de historias por medio de técnicas procedimentales. El primero de estos trabajos fue *Tale-Spin* [Meehan, 1976] en 1976. Este sistema produce cuentos de hadas originales que incluyen una enseñanza moral. Están basadas en texto⁵ y tienen un gran número de inconsistencias. La idea es simular un mundo, dándole metas a los personajes y luego informa al lector lo que sucedió con esas metas. Aún así existe interés dramático en las historias generadas por el sistema, de las cuales todas están basadas en un mundo estándar de cuento de hadas (con bosques, ríos y campos). El sistema contiene una gran cantidad de conocimiento acerca de las posibilidades para el mundo virtual, el cual es creado en base a como la historia lo cuenta.

Posteriormente, la planificación fue utilizada para crear infinitas historias tipo comedias dramáticas en UNIVERSE [Lebowitz, 1987]. En este caso fue necesario para el autor darle metas al sistema. El sistema usa estas metas junto con los fragmentos de tramas existentes para crear un resumen de una trama de comedia dramática. Los personajes estereotípicos creados por el sistema son roles dinámicamente asignados en estos fragmentos, con nuevos personajes que se agregan si ningún personaje existente es capaz de asumir un papel en particular. Las relaciones de los personajes son centrales en el argumento.

En el trabajo de Turner [Stede, 1996] se utiliza razonamiento basado en casos para generar historias acerca de caballeros y damicelas en los días del Rey Arturo. Los casos son historias existentes y estas son enlazadas a las historias deseadas y recombinadas para crear nuevas historias. El sistema utiliza su consciencia de lo que es consistente dentro de su mundo para asegurar que las historias generadas tengan esa característica, intentando presentar un giro al final de cada historia. Tanto los personajes como la historia tienen metas, las que son ingresadas por el usuario antes de que comience la generación del relato.

En la década de los 90 se comenzó a investigar la narración y el surgimiento de patrones narrativos con el fin de automatizar el proceso de creación de narraciones. Uno de los primeros trabajos que aplica estos patrones en la creación de narraciones a partir de un algoritmo fue Mariët Theune y compañía [Theune et al., 2002], quienes desarrollaron un narrador de historias virtual. El framework está basado en multi-agentes y busca crear historias o narraciones con sentido, utilizando lenguaje natural. El modelo tiene 4 agentes inteligentes: un director, personajes, un narrador y un presentador. El narrador controla las acciones del personaje para evitar que la trama se desvíe hacia historias no deseadas. Se trata de formar tramas que tengan un inicio, un desarrollo y un final feliz; por esto es importante la acción del director con el fin de evitar finales infelices. El narrador es quien le da el sentido lingüístico a la historia, mientras que el presentador es un ente visual que cuenta la historia producida por el narrador.

A pesar de que no es estrictamente un sistema generador de historias, *Daydreamer* [Mue-ller and Dyer, 1985] es un sistema que crea sueños. La idea es que estos sean generados

⁵La entrada y salida del programa está basada en texto más que en imágenes y sonido.

cuando el computador no está ejecutando ningún programa. Estos sueños se verán afectados por eventos anteriores y se reflejarán en estos o crearán alternativas idealizadas a ellos. Las experiencias en esta etapa son introducidas por el usuario.

Como se puede apreciar, la generación de historias mediante algoritmos es un campo que se investiga hace tiempo, aunque aún no ha sido desarrollado en su totalidad.

Sistemas de Drama Interactivos

Otra cosa interesante de estudiar son los sistemas de drama interactivos. Un sistema de drama interactivo es un sistema en donde un personaje manipulado por una persona, puede interactuar con personajes virtuales insertos en un mundo determinado [Arinbjarnar et al., 2009]. El entorno virtual del drama interactivo y sus personajes dependerán de la trama que se quiera conseguir. Por ejemplo, para representar *Dungeons and Dragons*⁶ será necesario contar con calabozos y monstruos.

A continuación se realizará un recorrido a través de diferentes sistemas de drama interactivo, comenzando por *The Oz Project* [Bates, 1992]. Los investigadores de este proyecto se concentraron en crear agentes creíbles. El usuario podía darle instrucciones a uno de estos personajes y jugar con ellos. Estos personajes interactuaban entre sí en el mundo virtual. El grupo también incluyó la generación de historias interactivas que están basadas en una estructura de grafo de trama. El camino que el usuario tomaba en esta estructura dependía tanto de las elecciones del usuario como de la función de evaluación predefinida, la cual guiaba la historia parcialmente hacia buenas experiencias.

Otro proyecto que obtuvo buenos resultados creando dramas interactivos fue *The Virtual Theater Project* [Hayes-Roth et al.]. Este trabajo utilizó el concepto de “Improvisación dirigida”, en la cual los actores que improvisan siguen ciertas direcciones y restricciones, además de proveer detalle a la historia. Por ejemplo, un actor puede ser instruido para caminar hacia una mesa, y si está representando un personaje enérgico puede que se precipite allí. Los mundos virtuales están poblados por actores que pueden tomar el papel de un personaje. El grupo trabajó en diferentes proyectos; en uno de ellos, el usuario podía destruir la historia, pero no podía comenzar una nueva; sin embargo, era capaz de observar como sus acciones alejaban la trama de su determinado curso. En otro proyecto con otro escenario había un número de clientes y una mesera en un café. El usuario daba instrucciones a uno de los personajes y estos improvisaban para seguir la historia. Las acciones de los personajes, que podían ser dadas por el usuario o por el sistema, eran incorporadas en una estructura de grafo de trama.

En el sistema *Façade* [Mateas and Stern, 2003] el usuario es invitado a la casa de unos amigos, una pareja. Mientras está ahí, se verá inmerso en los problemas maritales y peleas. El usuario es capaz de hablar con los otros personajes, y lo que dicen (y como lo dicen), afectará la

⁶Significa “Calabozos y Dragones”. Es un juego de rol de fantasía heroica.

historia que experimentan. Las acciones del personaje determinan el resultado de la historia y el estado final del matrimonio. Este sistema se compone de un director de drama, ritmos, personajes, historias de valor, acciones y procesamiento del lenguaje natural.

En *Interactive Drama Architecture* (IDA) [Magerko, 2010], el usuario encuentra su propio cuerpo muerto. Como fantasma, debe encontrar quién lo asesino y subsecuentemente manipular a otro personaje para que encuentre el cuerpo y también darse cuenta de quién cometió el crimen. En este sistema, es necesario predefinir: la historia, cualquier dominio de funciones dependientes del director, el ambiente y el contenido de arte, y el comportamiento del personaje. Los personajes son semi-autónomos, en el sentido de que pueden actuar mientras no tengan instrucciones del director, como por ejemplo, beber.

SASCE [Nelson et al., 2006] es un método adaptado de aprendizaje de diferencias temporales (también conocido como *TD-learning*, por su significado en inglés) para dramas interactivos. Este método determina, basado en una función de evaluación predefinida, la mejor ruta aparente para la historia, dependiendo de la acción que se espera que el usuario tome en cada etapa y, por lo tanto, aquello que conducirá a la evaluación más alta. Las rutas para el drama son seleccionadas de las posibles rutas a través de un grafo de tramas predefinido. Las acciones que se esperan que el usuario haga son determinadas por un usuario simulado por computador. Estas simulaciones proveen la data de entrenamiento.

U-DIRECTOR [Mott and Lester, 2006] usa una planificación jerárquica de la red de tareas y redes de decisión dinámicas para implementar una historia de misterio médico que toma lugar en una isla aislada. La historia está preescrita y sigue una trama bastante estricta. Se utiliza un mecanismo de inferencia Bayesiano para decidir cómo manipular al usuario para seguir la trama. Esto permite lograr el final deseado, que es la solución del misterio. El director trata de enganchar al usuario en la trama proveyéndole pistas que lo guiarán al desenlace. Si las pistas no son suficientes, entonces el director se vuelve menos sutil.

PaSSAGE (*Player-Specific Stories via Automatically Generated Events*) [Thue et al., 2007] se enfoca en la adaptación específica del usuario a la historia. Hay un número de posibles personajes que tienen “encuentros” que involucran la interacción con otros. Estos personajes siguen un orden particular dependiendo de su tipo. El personaje que tiene el encuentro depende del tipo de usuario que el jugador ha modelado en base a sus acciones en la etapa introductoria. La forma de los encuentros está definido por un grafo de trama.

IN-TALE (*Interactive Narrative Tacit Adaptive Leader Experience*) [Riedl and Stern, 2006] es un sistema diseñado para el entrenamiento de soldados. El usuario se encontrará a sí mismo en un escenario que podría ocurrir en el frente de batalla. Esto les permite actuar tan libremente como en la realidad y sus acciones determinarán si son capaces de superar exitosamente la situación. El final se adaptará para asegurarse que los eventos problemáticos siempre ocurran, sin importar cómo el usuario decida actuar.

Mimesis [Young and Riedl, 2005] es un sistema creado como parte de un trabajo del grupo *Liquid Narrative*. Está diseñado con una arquitectura general y funciona con cualquier mo-

tor de juego. En este sistema se hace un intento para dar la ilusión de completa libertad. Después de cada acción del usuario que no es acorde al plan actual, el sistema decide si la acción del usuario puede ser “acomodada” o debe ser “intervenida”. Una acción acomodada debe ser incorporada en un nuevo plan para lograr la meta de la historia. Por ejemplo, en un escenario donde se planea robar un banco, si el usuario abre la bóveda del banco (cuando el plan requería que otro personaje lo hiciera), la replanificación puede acomodar esta inconsistencia creando un plan en el cual el usuario no abra la bóveda sino que la encuentre abierta. Si la acomodación no es posible, el sistema interviene en la acción del usuario. Esto puede llevar a que el usuario se pierda algunos hechos.

Cuando el sistema recibe el requerimiento de un plan crea un grafo acíclico dirigido (DAG por su sigla en inglés) para lograr el final de la historia. Dicho grafo es una estructura de grafo de trama en el cual el grafo puede ser redibujado con la narrativa, pero su variación no es suficiente para generar un drama fundamentalmente diferente.

NOLIST (*Non-linear Interactive Storytelling*) [[Bangsø et al., 2004](#)] es un motor de juego que utiliza una red Bayesiana para crear un misterio de asesinato. La red Bayesiana cambia dinámicamente en respuesta a las acciones y observaciones hechas por el usuario. No está predefinido pero combina las acciones del usuario y la inferencia lógica para determinar los detalles de la historia, incluyendo la identidad del asesino. Por ejemplo, si el usuario encuentra un cuerpo y un arma al lado de él, la probabilidad de que dicha arma fuera lo que le causara la muerte aumenta. Así, NOLIST crea el pasado de la historia en respuesta a las interacciones del usuario.

El sistema GADIN (*Generator of Adaptive Dilemma-based Interactive Narratives*) [[Barber and Kudenko, 2009](#)] genera narrativas basadas en dilemas. Estos representan fundamentalmente decisiones difíciles para los personajes en el mundo de la historia, que pueden incluir al usuario. Este sistema usa planificación para lograr los dilemas en el curso de la historia. El usuario es capaz de seleccionar libremente sus propias acciones que son incorporadas donde sea posible en el plan.

El sistema *Erasmatron* [[Crawford, 2013](#)] presenta al usuario un número de acciones posibles, generalmente relacionadas con actos del habla específicos. Una vez el usuario ha hecho su elección, el sistema o un personaje responde apropiadamente. Esta selección de acción basada en turnos continua hasta que la historia termine. El *Erasmatron* es un sistema basado en texto. El sistema tiene un rango de mundos disponibles. Entre estos el usuario puede verse envuelto en la creación de una historia escogiendo opciones de acciones de bajo nivel. Los personajes tienen emociones y personalidades y existe un controlador de drama que actúa como el destino.

Posteriormente, el mismo autor de *Erasmatron* trató de mejorar la idea original y construyó *Storytron* [[Crawford](#)], un sistema que seguía la misma idea de *Erasmatron* pero mejorada. El proyecto no ha logrado surgir debido a que resultaba demasiado complejo utilizarlo (su manual era muy extenso y enredado).

En el sistema DEFACO [Sgouros, 1997] el usuario se vuelve un personaje de la antigua Grecia. El usuario es capaz de especificar sus acciones en ese mundo y así mostrar gráficamente la historia creada. Hasta que se muestra gráficamente, el usuario no conoce las consecuencias de sus acciones. DEFACO permite al usuario participar en historias incorporando asesinatos, matrimonio, sacrificio y dioses. Una serie de reglas controlan la generación del drama en el mundo. Estas historias son dinámicamente creadas en un sistema basado en texto.

OPIATE (*Open ended Proppian Interactive Adaptative Tale Engine*) [Fairclough, 2005] es un sistema que crea historias basadas en la estructura general de Vladimir Propp para los cuentos de hadas [Propp, 1968]. Otros personajes aparte del usuario tienen roles flexibles en la historia. En cada estado, el sistema elige apropiadamente funciones Proppian utilizando planificación basada en casos. El director de la historia guía a los actores dándoles metas relevantes a la función seleccionada. La historia emerge de la interacción de los personajes y los eventos iniciados por el director de la historia. El motor tiene un sistema de “chismes” que conecta a los personajes y esparce noticias y opiniones acerca del usuario y sus acciones. Los personajes también comunican las noticias de los eventos del mundo virtual entre ellos.

DED (*Directed Emergent Drama*) [Arinbjarnar and Kudenko, 2008] es un sistema o motor que tiene un agente director que utiliza esquemas para estructurar un drama emergente. Existe un conjunto de agentes actores que interpretan personajes en el drama que se despliega usando los esquemas como una guía. Los esquemas son estructuras que contienen: metas, un conocimiento base, y acciones para los actores y el usuario del drama. La comunicación entre el director y los actores es a través de esquemas. El director nunca interactúa directamente con el usuario o los actores. El usuario tendrá las mismas opciones de interacciones que tienen los actores y todas las interacciones disponibles para otros personajes también estarán disponibles para el usuario.

Finalmente, el último sistema de drama interactivo que se revisará será el de Malik Nairat y compañía [Nairat et al., 2011], donde utilizan un modelo basado en agentes para integrar al usuario con la creatividad del computador, y algoritmos genéticos para evolucionar la interacción entre los agentes. El primer paso es que el artista cree un conjunto de diferentes personajes. El genoma de un agente representa su personaje mediante parámetros que determinan la dinámica de sus estados internos o emociones y también contiene las acciones que el agente puede realizar. La principal contribución de este trabajo fue la construcción de una representación genética para la representación de la personalidad que fuera usable y apropiada.

Existen otros sistemas de drama interactivos menos desarrollados, pero no serán revisados en este estado del arte para no perder el foco del tema.

Generación de historias para NPCs mediante métodos de generación de contenido procedimental y algoritmos de inteligencia artificial

Hasta ahora, la mayoría de los sistemas que se han revisado en este trabajo utilizan modelos estadísticos, grafos o planificación para generar las historias e interacción entre sus personajes. Aún no se han considerado aquellos que utilizan técnicas del área de la inteligencia artificial (excepto el Malik Nairat y compañía [Nairat et al., 2011]), por lo que a continuación revisaremos los avances del desarrollo de historias para NPCs aplicando técnicas de inteligencia artificial y modelos basados en agentes. En el año 2004, Ong y Leggett [Ong and Leggett, 2004] diseñaron un motor narrativo llamado *Hybrid Evolutionary-Fuzzy Time-based Interactive (HEFTI)* en el que utilizaron algoritmos genéticos en su núcleo para recombinar y evaluar los componentes de la historia generados a partir de un conjunto de plantillas. El sistema realiza la recombinación, mutación y selección que generan variantes lógicamente congruentes a través de la creación y eliminación de elementos (o enlaces) de la historia. Posteriormente en 2010, Neil McIntyre y Mirella Lapata [McIntyre and Lapata, 2010] desarrollaron un generador de historias que aprovecha su propio potencial sin requerir una participación manual extensa. La característica clave de este enfoque es un planificador de historias que se adquiere automáticamente al registrar eventos, sus participantes y sus relaciones de precedencia. Utilizaron técnicas de *búsqueda evolutiva* para explorar el espacio de las posibles historias. Este enfoque fue probado en la creación de cuentos infantiles simples con buen resultado.

Para esta sección revisaremos el trabajo de Rubén García-Ortega, desde su comienzo explorando esta área [García-Ortega et al., 2014] hasta su avance actual [García-Ortega et al., 2016]. El primer objetivo de su trabajo fue crear un entorno donde poder probar las interacciones entre NPCs y estudiar las tramas y sub-tramas que surgían [García-Ortega et al., 2014]. La idea era usar este entorno virtual creado para hacer surgir dos arquetipos: el de “venganza” y “control de natalidad”. Se utiliza un algoritmo genético para encontrar el mejor número de perfiles (o agentes) y configuración de sus parámetros que permitieran la existencia de los arquetipos deseados. Durante el mismo año y basándose en la misma investigación, desarrollaron una metodología general para generar historias cohesivas y coherentes donde los arquetipos deseados (aquellos que son aceptados universalmente) pudieran emerger en sistemas estocásticos complejos [García-Ortega et al., 2014]. Básicamente la metodología es similar al trabajo anterior, pero extendida a diferentes arquetipos (guerra territorial, drama al estilo *Shakespeare*, etc).

Al año siguiente, siguen expandiendo su investigación a la prueba de los parámetros que modelan la historia: el tiempo, el mundo, los personajes y la fuente de conflictos [García-Ortega et al., 2015]. Estos parámetros pueden tener un gran impacto en la generación de la trama, donde pequeñas variaciones podrían alterar sustancialmente los resultados o, por otra parte, algunos parámetros podrían resultar superfluos. Con esto se llegó a diferentes conclusiones como, por ejemplo, los mapas pequeños implicaban usualmente más roces entre los agentes o que mientras más grande sea periodo de tiempo que corre el mundo (los días que transcurren en el entorno virtual), más interacciones se generan entre los personajes y pueden surgir más arquetipos y obtener mejores funciones de aptitud.

El año pasado decidieron extender su modelo a uno de 3 capas: la capa de razonamiento lógico, la capa de computación evolutiva y la capa del modelo basado en agentes [García-Ortega et al., 2016]. La idea del trabajo fue hacer surgir el monomito, o también conocido como el viaje del héroe, un patrón que aparece constantemente en la literatura, películas, videojuegos, etc. Para la realización del trabajo que se presentará en el futuro se usará este modelo para tratar de optimizarlo y sacarle más provecho aún. Ahondaremos en este modelo en la siguiente sección.

Arquitectura del sistema

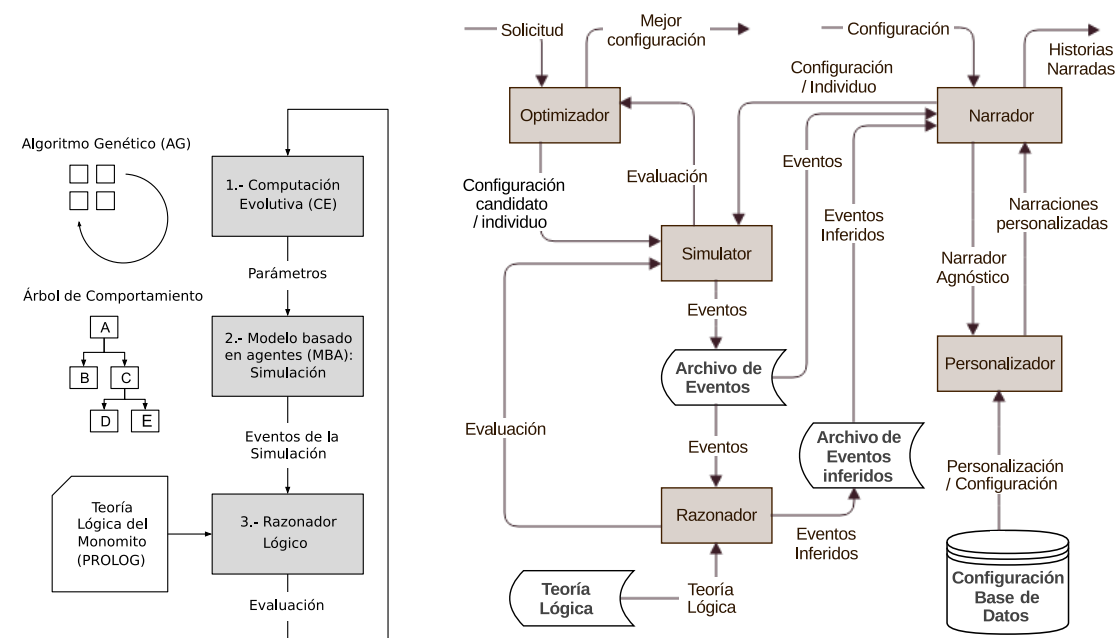
Como lo dice el título de la memoria, el modelo a trabajar consta de tres capas: la capa de computación evolutiva, la capa basada en agentes (ABM) y la capa lógica. La primera está encargada de ejecutar el algoritmo genético para encontrar la mejor configuración del simulador. Dichos parámetros le son entregados a la segunda capa, la cual se encarga de obtener los eventos de la simulación a partir de esa configuración. Tales eventos se le entregan a la capa final, que evalúa si estos siguen o no la línea del monomito. Esta evaluación se le entrega nuevamente a la capa evolutiva y se repite el proceso hasta encontrar la mejor combinación de parámetros que haga surgir el monomito.

Para describir la arquitectura del sistema, se definieron 5 módulos: el optimizador, el simulador, el razonador, el narrador y el personalizador. Cada módulo se describe a continuación:

- **Optimizador:** Formalmente conocida como la capa de computación evolutiva, este módulo utiliza un algoritmo genético cuyos individuos son configuraciones de una simulación y retorna el mejor individuo encontrado, es decir, aquella configuración del simulador que obtiene la mejor función de aptitud.

La función de aptitud de cada individuo consiste en el promedio en varias simulaciones llevadas a cabo por el simulador (15 para efectos de prueba).

- **Simulador:** Este módulo corre el ambiente virtual con cada una de las configuraciones en diferentes pruebas y retorna el promedio (la aptitud) al optimizador.
- **Razonador:** Es el encargado de evaluar los eventos obtenidos por los multi-agentes de la capa ABM, utilizando lógica de predicado para asignar un valor numérico a los eventos siguiendo las pautas extraídas del monomito.
- **Narrador:** Toma una solución dada y la transforma en una narración agnóstica, es decir, una selección de predicados relevantes sin nombres finales para los elementos.
- **Personalizador:** Está encargado de asignarle nombres y estructuras a la narración agnóstica creada por el narrador con el fin de aplicar una configuración literaria específica, por ejemplo, medieval, futurista, etc.



(a) Arquitectura propuesta de tres capas (CE-MBA-RL). (b) Componentes propuestos y diagrama de flujo de datos de toda la arquitectura.

Figura 1: Arquitectura del sistema (alto nivel).
Fuente: [García-Ortega et al., 2016]. Traducción propia.

En la figura 1 se puede ver un diagrama de flujo de la arquitectura del sistema. El lado izquierdo (1a) representa el modelo de tres capas, mientras que el lado derecho (1b) ilustra los distintos módulos y la interacción que tienen estos entre sí.

Capa ABM

El ambiente que se propone para el modelo basado en agentes es **ColorForms** y considera 4 elementos: el mundo, los agentes, los objetos y los eventos.

- **Mundo:** El mundo es el lugar espacio-temporal donde ocurren los eventos. En ColorForms se propone usar un mapa simple: el tablero, que consiste en N cuadrados ($n \times n$). El tiempo se mide en días virtuales. El tamaño n del mundo también es un parámetro que se obtiene del algoritmo genético y su valor fluctúa entre 8 y 16.
- **Agentes:** Los agentes son piezas que pueden ocupar una celda cada uno. Hay tres formas para las piezas: círculos, cuadrados y triángulos. Se establece una suerte de piedra-papel-tijeras para establecer un método para decidir al ganador y perdedor en los conflictos individuales directos con el objetivo de mantener un balance global. Los

agentes pueden cambiar en el tiempo: cada pieza cuenta con un color de fondo y un color de frente que están cambiando constantemente. El color de fondo varía lentamente de manera aleatoria para cada pieza y representa los continuos cambios internos en las necesidades del personaje. El color de frente puede ser cambiado por la pieza o por otras piezas bajo ciertas circunstancias y es usado para mejorar la felicidad del agente: la pieza es más feliz cuando los dos colores son iguales o similares.

- **Objetos:** El objeto por el cual los personajes deben competir o colaborar es el color de la casilla. Los agentes pueden quedarse con el color de la casilla para ser “coloreados”. Los colores de las casillas aparecen y desaparecen aleatoriamente.
- **Eventos:** Los personajes interactúan entre sí y con el ambiente, a pesar de que sus acciones son limitadas: moverse a otra celda libre adyacente, empujar a otro agente dependiendo de la forma, intercambiar el color de fondo y quedarse con una celda de color. Los eventos son generados por dichas interacciones.

El componente social en ColorForms está definido por una matriz de afinidad: cada pieza tiene una afinidad con cada una de las otras piezas en el tablero, lo que va a depender de su forma, del color de fondo y del de frente. Esta afinidad es dinámica e influenciará las decisiones que tome cada pieza, es decir, su felicidad será influenciada por aquellas piezas con mayor afinidad.

Los eventos se presentan en forma de predicado lógico. Un evento existe si un sujeto x tiene la propiedad P al tiempo t . El predicado lógico se denota como $Predicado(t, x_0, \dots, x_n)$ donde t es el tiempo y cada x es un elemento del mundo (un personaje, una celda, etc). Cada predicado tiene un nombre, argumentos y una interpretación que es una descripción del evento en lenguaje natural. Existen dos tipos de predicados: los generados por los agentes en el mundo virtual y aquellos que son inferidos utilizando el módulo Razonador.

Los agentes interactúan en este ambiente para generar eventos que luego se transformarán en historias de fondo. El árbol de comportamiento descrito en la figura (poner cita figura) tiene nodos de condición y de acción: cuando se cumple una condición se genera un evento de la figura (poner cita figura) y cuando la acción es ejecutada correctamente se genera un evento de la figura (poner cita figura).

El simulador toma un arreglo de 52 valores como entrada, los mapea a las variables del ambiente virtual y realiza una ejecución del mundo virtual corriendo secuencialmente el árbol parametrizado de comportamiento de los agentes. Cuando la simulación alcanza una cierta cantidad de días se detiene y los eventos generados son enviados al Razonador para su análisis.

Capa lógica

El Razonador mencionado anteriormente utiliza una teoría lógica en PROLOG, es decir, permite la posibilidad de componer predicados y usar marcos temporales. En la figura 2, se puede ver la dependencia entre los diferentes predicados. A pesar de que el monomito tiene diferentes interpretaciones en la literatura, los autores del simulador decidieron implementar su propia interpretación, la cual fue utilizada para generar el modelo de dependencias expuesto en la figura 2 y es descrita en la tabla 1.

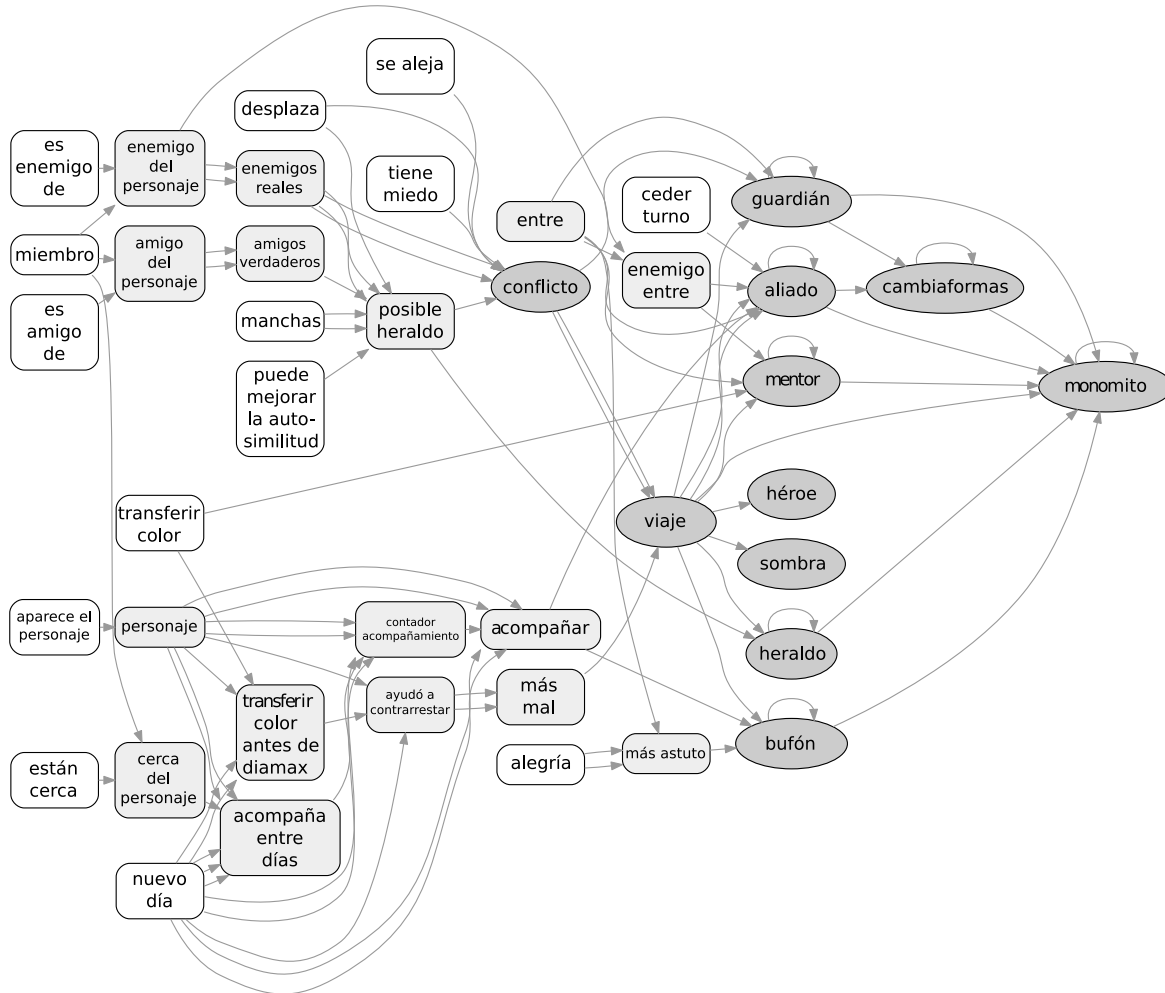


Figura 2: Dependencia de los predicados en la teoría lógica propuesta del monomito: los nodos blancos representan los predicados que pueden ser generados por la capa ABM y los nodos grises son predicados inferidos por el Razonador. Los nodos gris oscuro son llamadas facilitadoras y conforman el monomito.

Fuente: [García-Ortega et al., 2016]. Traducción propia.

Arquetipo	Breve Descripción	Interpretación en el modelo
Conflicto	No está presente en la definición de monomito.	Una secuencia de eventos que tienen éxito en un momento determinado en el que participan dos personajes (el ganador y el perdedor). Un conflicto puede aparecer cuando una pieza asusta o desplaza a otra, o cuando toma la mancha de color que el otro desea. Si dos piezas son amigas y una de ellas es desplazada por una tercera, los dos amigos tienen un <i>Conflicto</i> con esta.
Viaje	No está presente en la definición de monomito.	El intervalo de tiempo entre <i>Conflictos</i> que involucren las mismas piezas. El primer <i>Conflicto</i> debe ser perdido por una de ellas (el héroe) y el segundo debe ser perdido por la otra (la sombra), quien debe ser 'más malo' que el <i>Héroe</i> , es decir, debe haber ayudado a menos amigos.
Héroe	El personaje que está dispuesto a sacrificar sus propias necesidades en beneficio de otros.	La pieza que pierde el primer conflicto y gana el segundo en un <i>Conflicto</i> .
Mentor	El personaje que enseña y protege al <i>Héroe</i> y le da regalos.	Una pieza que tiene gran afinidad con el <i>Héroe</i> durante el <i>Viaje</i> y lo ayuda al menos una vez.
Guardián	La fuerza que el héroe debe superar.	Perdedores de <i>Conflictos</i> contra el <i>Héroe</i> durante el <i>Viaje</i> .
Heraldo	El que llama a la aventura al héroe.	Una pieza con gran afinidad con el <i>Héroe</i> en un <i>Viaje</i> y que pierde el primer <i>Conflicto</i> con la <i>Sombra</i> .
Cambiaformas	El personaje que cambia a través del viaje.	Una pieza que hace de <i>Aliado</i> y de <i>Guardián</i> en el mismo <i>Viaje</i> .
Sombra	El villano o antagonista; el lado oscuro.	El personaje que gana el primer conflicto y pierde el segundo en un <i>Viaje</i> .
Aliado	El personaje que acompaña o ayuda al <i>Héroe</i> a través del <i>Viaje</i> .	Una pieza que le da al <i>Héroe</i> un turno extra para cumplir sus metas o se queda a su lado en el tablero.
Bufón	El que hace travesuras.	Una pieza que se queda al lado del <i>Héroe</i> en el tablero y es más travieso que este (bromeando y haciendo trucos, en otras palabras, teniendo un nivel más alto de alegría).

Tabla 1: Comparación teórica de los arquetipos presentes en el monomito y su interpretación en el modelo.

Fuente: [García-Ortega et al., 2016]. Traducción propia.

Capa de computación evolutiva

Debido al tamaño del individuo, la capa evolutiva debe ser capaz de manejar una gran cantidad de parámetros del simulador. Además, el resultado de las simulaciones es incierto debido a la naturaleza estocástica del mundo virtual y la impredecibilidad de las interrelaciones entre los agentes.

El módulo Optimizador descrito anteriormente utiliza un algoritmo evolutivo para evolucionar las configuraciones del Simulador con el fin de encontrar aquellas con la mejor aptitud, que es calculada en base a las predicciones encontradas por el Razonador. Así, el resultado de la función de aptitud es producto de las interacciones de alto nivel de los personajes que no pueden ser predichas. Esta incertidumbre aumenta por la naturaleza estocástica de la simulación, debido a que diferentes ejecuciones del Simulador proveen distintos resultados que mejoran la re-jugabilidad del videojuego.

Un individuo del algoritmo genético es un cromosoma de 52 genes, donde los primeros 5 son números naturales en diferentes rangos y los siguientes 47 son valores reales en un rango (0,1). En la figura 3 se detalla la estructura de un cromosoma.

La función de aptitud utilizada toma en cuenta el número de arquetipos que han sido deducidos por el Razonador.

Sea A_i un [arquetipo](#) en el conjunto de todos los T predicados deseados A :

$$A = \{\text{Conflicto, Viaje, Sombra, Héroe, Mentor, Herald, Guardián, Aliado, Bufón, Cambiaformas, Monomito}\}$$

y $|A_i|$ el número de ocurrencias de los predicados generados por el Razonador al final de la simulación. Para la función de aptitud se definen tres métricas:

- **Cantidad de Ocurrencias del Monomito encontradas (M):** Cuenta el número de ocurrencias del predicado “monomito”. Es la base de la función aptitud y se define como:

$$F_M = |A_{\text{Monomito}}| \quad (1)$$

- **Cantidad de todas las ocurrencias encontradas (A):** Cuenta el número de ocurrencias de cada predicado que implique llegar al monomito, definido como los “facilitadores”. Se define:

$$F_A = \sum_{i=0}^T |A_i| \quad (2)$$

- **Cantidad de todas las ocurrencias distribuidas por predicados (AD):** Cuenta el número de ocurrencias, pero promueve la aparición de nuevos arquetipos. Esto es, la

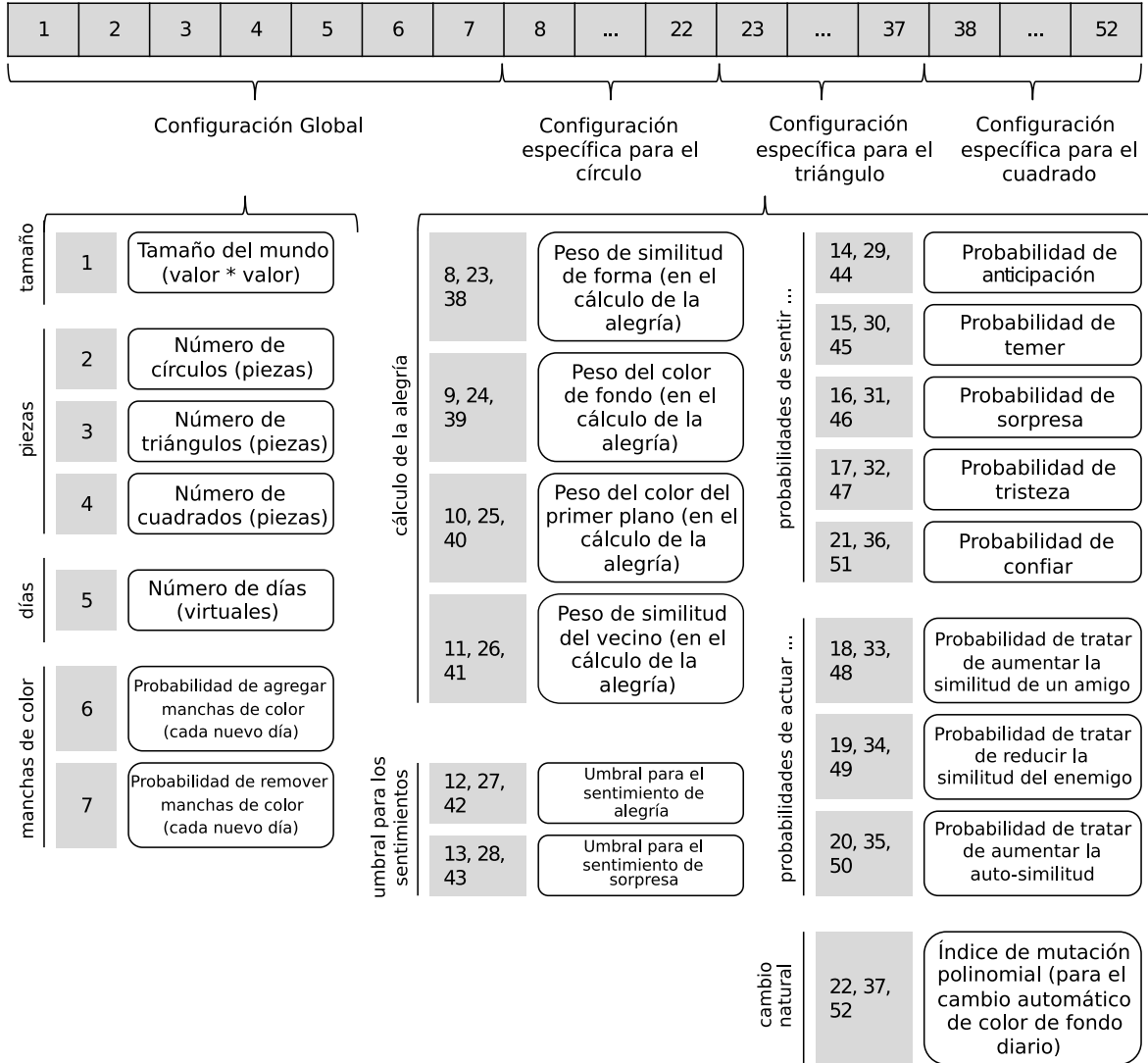


Figura 3: Estructura del individuo utilizado en la capa evolutiva y el significado de cada uno de sus 52 genes.

Fuente: [García-Ortega et al., 2016]. Traducción propia.

ocurrencia de dos predicados diferentes obtiene una mejor aptitud que dos veces la ocurrencia del mismo predicado. Se define:

$$F_{AD} = \sum_{i=0}^T \log_{10}(|A_i| + 1) \quad (3)$$

En palabras simples, si por ejemplo se tiene una ocurrencia de A_{Conflict} y otra de A_{Mentor} , eso resultaría en una evaluación de $F_{AD} = \log_{10}(1 + 1) + \log_{10}(1 + 1) = 2 \log_{10}(2)$. En cambio, si consideramos que sólo ocurre A_{Conflict} dos veces, eso nos da una evaluación

de $F_{AD} = \log_{10}(2 + 1) = \log_{10}(3)$, lo que se traduce en:

$$2 \log_{10} 2 > \log_{10} 3$$

$$\log_{10} 4 > \log_{10} 3$$

A pesar de que podría sólo considerarse F_M , las otras dos métricas se incluyen para medir el efecto de guiar la simulación a través de facilitadores. En los resultados presentados posteriormente queda en evidencia que una función de aptitud que considera solo la aparición de monomitos no es suficiente para hacer evolucionar el sistema. Se necesita de los facilitadores para poder encontrar mejores configuraciones.

Marco Teórico

Para abordar este problema de optimización que se puede considerar como un problema *black box*, se emplearon algoritmos evolutivos. Dichos algoritmos se pueden separar en tres ramas: Programación Evolutiva, Algoritmos Genéticos y Estrategias Evolutivas. Están basados en una población de puntos de búsqueda inicializada arbitrariamente, la cual evoluciona a mejores regiones en el espacio de búsqueda por medio de procesos de selección, mutación y (a veces) recombinación. [Bäck et al., 1993]

Algoritmos Genéticos

Los algoritmos genéticos se basan en la teoría de la evolución de Darwin en la que por medio de selección natural van sobreviviendo los mejores individuos los cuales pasan a formar parte de las futuras generaciones. Durante un período de tiempo y muchas generaciones, la población completa tiende a contener más individuos cuyos cromosomas se traducen en estructuras y comportamientos que les permiten a dichos individuos obtener un mejor desempeño y sobrevivir y reproducirse. [Koza, 1992]

El algoritmo genético generacional que se utiliza en este trabajo consiste en 5 pasos:

1. Se genera la población inicial aleatoriamente.
2. De los individuos pertenecientes a dicha población se seleccionan por medio de torneo binario a aquellos individuos que darán origen a la nueva población.
3. Los individuos seleccionados anteriormente se cruzan entre sí por medio de un cruceamiento BLX- α [Eshelman and Schaffer, 1993].
4. A los mismos individuos seleccionados también se les aplica una mutación utilizando un operador polinomial [Deb and Goyal, 1999].

5. A la nueva población se le aplican los mismos pasos, hasta evolucionar en un número determinado de generaciones.

Cruzamiento BLX- α Este operador de cruzamiento elige uniformemente valores que se encuentran en un rango limitado por dos puntos que contienen los dos padres, pero que puede ser extendido en ambos extremos por igual usando un parámetro α [Eshelman and Schaffer, 1993]. Para explicarlo mejor, considérese p_1 como el valor de un gen perteneciente al padre 1 y p_2 como el valor de un gen perteneciente al padre 2. El intervalo I se define como:

$$I = \max(p_1, p_2) - \min(p_1, p_2) \quad (4)$$

El nuevo valor del gen estará comprendido entre los puntos $\min(p_1, p_2) - \alpha I$ y $\max(p_1, p_2) + \alpha I$. En este caso, el cruzamiento utiliza un $\alpha = 0,5$, lo cual significa que los valores seleccionados están en un intervalo que se extiende entre $\min(p_1, p_2) - 0,5I$ y $\max(p_1, p_2) + 0,5I$

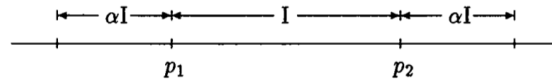


Figura 4: Cruzamiento BLX- α .

Fuente: [Eshelman and Schaffer, 1993].

Mutación Polinomial La principal característica de este operador es que tiene una mayor probabilidad de generar soluciones cercanas al individuo, independientemente de la iteración [Sosa et al., 2014]. Para obtener el nuevo valor del gen, se utiliza la siguiente expresión:

$$x_i^{(t+1)} = x_i^{(t)} + (x_i^{(U)} - x_i^{(L)}) \delta_i \quad (5)$$

donde $x_i^{(t+1)}$ es el valor mutado, $x_i^{(t)}$ es el valor actual, $x_i^{(U)}$ y $x_i^{(L)}$ es el valor máximo y mínimo que puede tener el gen respectivamente y δ_i se calcula en base a una distribución polinomial. Para obtener este último valor se elige un número aleatorio u_i entre 0 y 1 con distribución uniforme de forma tal que δ_i queda definido de la siguiente manera:

$$\delta_i = \begin{cases} (2u_i)^{\frac{1}{n+1}} - 1, & u_i < 0,5 \\ 1 - [2(1 - u_i)]^{\frac{1}{n+1}}, & u_i \geq 0,5 \end{cases} \quad (6)$$

donde n es un valor fijo y representa la perturbación máxima permitida en el valor del gen original. Para este trabajo se usó $n = 20$.

Traducido en pseudocódigo, el algoritmo genético generacional utilizado en el simulador queda de la siguiente forma:

Algoritmo 1 Algoritmo Genético

Entrada: *max_iter*: Cantidad máxima de iteraciones

Entrada: *tamano_poblacion*: Tamaño población

Salida: *mejor_individuo_de_todos*: El mejor individuo de todos

poblacion_inicial $\leftarrow$ Construir poblacion inicial aleatoria

mejor_individuo $\leftarrow$ Encontrar al mejor individuo de la población inicial

mejor_individuo_de_todos $\leftarrow$ *mejor_individuo*

iteracion $\leftarrow$ 0

mientras *iteracion* $\leq$ *max_iter* **hacer**

 Seleccionar individuos para cruzamiento y mutación

 Cruzar dichos individuos

 Mutar dichos individuos

mejor_individuo $\leftarrow$ Encontrar al mejor individuo de la nueva población

si *mejor_individuo* > *mejor_individuo_de_todos* **entonces**

mejor_individuo_de_todos $\leftarrow$ *mejor_individuo*

fin si

iteracion ++

fin mientras

Estrategias Evolutivas

Las estrategias evolutivas parten de la misma idea del algoritmo genético: hacer evolucionar individuos de manera tal que estos sean mejores y mejores a través de las generaciones. Para este trabajo se utilizó específicamente el algoritmo ES-(1+1) con mutación Gaussiana y la regla de 1/5 de éxito de Rechenberg, el cual posee las siguientes características.

- No tiene población.
- Un solo individuo es utilizado para generar una única descendencia.
- Dicha descendencia es generada a través de mutación.

El pseudo-código del algoritmo ES-(1+1) utilizado corresponde a la variante 2 de [Auger, 2009] y se detalla a continuación:

Algoritmo 2 Estrategia Evolutiva

Inicializar $\mathbf{x}_0, \sigma_0$

repetir

$\tilde{\mathbf{x}}_n = \mathbf{x}_n + \sigma_n \mathcal{N}(0, 1)$

si $f(\tilde{\mathbf{x}}_n) > f(\mathbf{x}_n)$ **entonces**

$\mathbf{x}_{n+1} = \tilde{\mathbf{x}}_n$

$\sigma_{n+1} = 1,5 \sigma_n$

sino si $f(\tilde{\mathbf{x}}_n) = f(\mathbf{x}_n)$ **entonces**

$\mathbf{x}_{n+1} = \tilde{\mathbf{x}}_n$

$\sigma_{n+1} = \sigma_n$

sino

$\mathbf{x}_{n+1} = \mathbf{x}_n$

$\sigma_{n+1} = 1,5^{-1/4} \sigma_n$

fin si

hasta Cumplir algún criterio de detención

retornar $\mathbf{x}$

donde $\mathbf{x}$ es el individuo que evoluciona (nuestra solución final) y σ_0 es un valor aleatorio entre 0 y 1. Se considera el caso de maximización para la función de aptitud.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Uno de los principales desafíos a resolver tiene que ver con la estocasticidad⁷ del modelo. Por esta misma razón, se decidió considerar al simulador como una caja negra de la cual solo se sabe que la función de aptitud es estocástica (podría no siempre dar el mismo resultado aún cuando la configuración de los parámetros que usa no cambia).

Por otro lado, el algoritmo requiere altos tiempos de ejecución, lo que empeora a medida que avanzan las iteraciones. Mientras más iteraba el algoritmo, mejores soluciones encontraba pero tomaba mucho más tiempo. En la figura 5 se muestra un gráfico de la función de aptitud para el mejor individuo de todos versus el número de iteraciones; también se puede ver en el mismo gráfico el tiempo de ejecución versus las iteraciones. El eje x representa las iteraciones, mientras que el eje y de la izquierda muestra la aptitud y el de la derecha el tiempo. Se puede apreciar que a medida que aumentan las iteraciones, también aumenta el tiempo de ejecución de cada una. Esto ocurre porque mientras mejor es la solución, más tiempo se tarda el sistema en evaluar la función objetivo pues el resultado tiene una relación directa con la cantidad de eventos que se generan. Además el mismo hecho de tratar el problema como una caja negra hacía imposible reducir el tiempo que tardaban las evaluaciones de la función objetivo, por lo que había que abordar el problema desde otro punto. También es posible notar que a medida que aumenta el número de iteraciones, el valor de la función de aptitud aumenta considerablemente. Esto indica que para obtener mejores resultados con el algoritmo genético, se deben realizar más iteraciones, reforzando todo lo comentado anteriormente sobre el tiempo de ejecución.

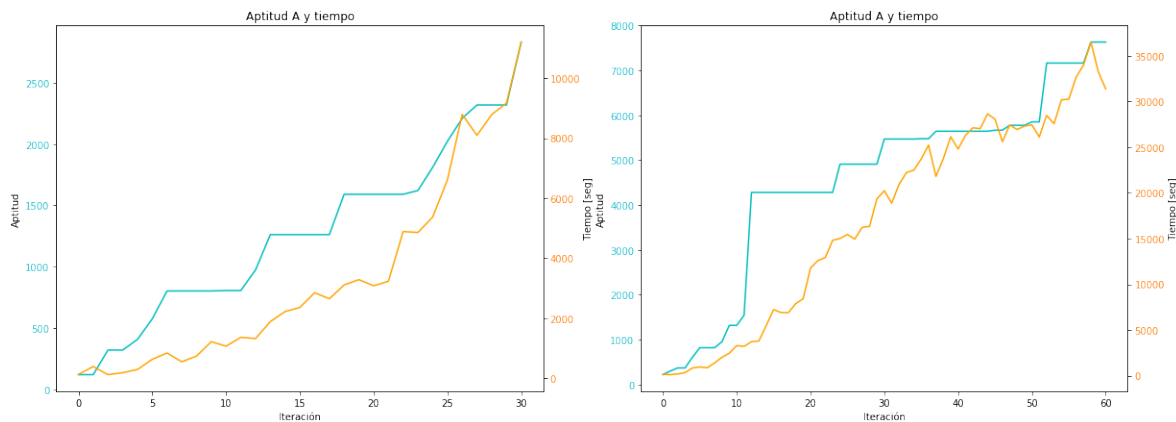


Figura 5: Dos ejecuciones del algoritmo genético con función de aptitud A.

Fuente: Elaboración Propia.

⁷No se tiene certeza del resultado final de la función de evaluación, ya que esta depende de números aleatorios.

El principal foco de investigación se centró en el comportamiento del algoritmo genético. Este algoritmo realiza muchas evaluaciones y no se aprovechan todos los resultados. Además, el operador de cruzamiento es un operador destructivo, lo que significa que al dividir la solución en dos para crear nuevos individuos, podrían perderse las ventajas de los padres y crear individuos que no necesariamente conserven la calidad de sus predecesores. Así, se optó por usar una estrategia evolutiva EE(1+1) que reduce el costo de trabajar con una población de soluciones trabajando con un solo individuo el cual se hace evolucionar mediante mutaciones controladas, eliminando el efecto destructivo del cruzamiento.

Como se describió en el marco conceptual, la estrategia evolutiva utilizada en este trabajo comienza con un solo individuo al cual se le aplican cambios basados en probabilidades (mutación gaussiana), de manera que este vaya evolucionando a un mejor individuo. Al diseño EE(1+1) clásico, se le agregaron elementos extra como la posibilidad de reiniciar el algoritmo o de frenar su ejecución bajo ciertas condiciones; dichos elementos se explicarán más adelante en este capítulo.

Representación de la solución

Para aprovechar el trabajo realizado en [García-Ortega et al., 2016], se utilizó la misma representación pero se consideraron solo los valores reales, pues es lo que se busca optimizar. Recordando el esquema de solución mostrado en la figura 3, el individuo posee 52 genes de los cuales los 5 primeros son enteros, mientras que el resto son números reales cuyo valor está entre 0 y 1. En estos últimos 47 se enfocará la búsqueda.

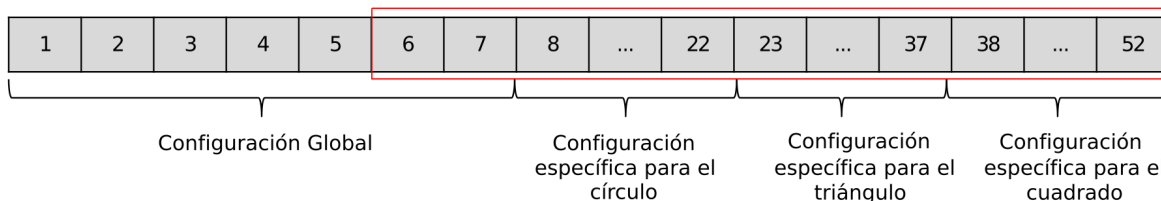


Figura 6: Resumen general del significado de los genes del individuo original. Se destacan los que son parte de la representación a utilizar.

Fuente: [García-Ortega et al., 2016]. Traducción propia.

En la figura 6 se puede apreciar la estructura de la solución y lo que representa cada fragmento del individuo. Los genes reales están encerrados en rojo para destacarlos. Para ejemplificar, suponga que se quiere representar un individuo cuya probabilidad de añadir una mancha de color cada nuevo día es 0.7, la de remover una mancha cada día es 0.3 y el índice de mutación polinomial (para el cambio del color de fondo diario) del cuadrado es 0.5, el individuo tendría los siguientes genes:

Gen	1	2	...	51
Valor	0.7	0.3	...	0.5

Tabla 2: Ejemplo de un individuo cualquiera.
Fuente: Elaboración propia.

Es importante destacar que la búsqueda se realizará solo en estos genes pues los otros representan características generales. Eso si, se utilizarán para crear las instancias de prueba.

Función de Evaluación

Para la función de evaluación se usaron las descritas en el marco conceptual, pues en [García-Ortega et al., 2016] se demostró el propósito de cada una. Además, utilizar las mismas funciones de evaluación permite establecer una comparación igualitaria entre el algoritmo planteado en este trabajo y el utilizado en la literatura. Recordando lo descrito anteriormente, se tenían 3 funciones de evaluación: una mide la cantidad de arquetipos que surgen durante la simulación (función de aptitud A), otra también mide la cantidad de arquetipos surgidos pero aplicando una función logarítmica de tal manera de promover la ocurrencia de distintos arquetipos y no que se repita siempre el mismo (función AD); mientras la tercera función de evaluación mide solo la cantidad de monomitos que aparecen (función M).

Sea A el conjunto de los arquetipos que aparecen en el monomito (*Conflicto*, *Héroe*, *Mentor*, *Sombra*, etc), las tres funciones de aptitud se pueden representar de la siguiente forma:

1. Función A:

$$F_A = \sum_{i=0}^T |A_i| \quad (7)$$

Tiene la ventaja de utilizar los arquetipos del monomito como facilitadores de este, obteniendo buenos resultados.

2. Función AD:

$$F_{AD} = \sum_{i=0}^T \log_{10}(|A_i| + 1) \quad (8)$$

Esta función promueve la aparición de diferentes arquetipos al darle más valor a la aparición consecutiva de dos arquetipos diferentes por sobre dos iguales.

3. Función M:

$$F_M = |A_{\text{Monomito}}| \quad (9)$$

Mide la cantidad de monomitos que aparecen durante la simulación.

Para medir la cantidad de ocurrencias de cada arquetipo, cada simulación se ejecuta 15 veces y se obtiene el promedio de las veces que apareció cada uno. Por ejemplo, si el promedio de aparición de cada arquetipo fueran los de la tabla 3, la función A tendría valor 3570.3, la función AD valdría 21.0 y la función M 66.7.

Arquetipos	Promedio de aparición
Conflicto	865.9
Viaje	97.5
Sombra	97.5
Héroe	97.5
Mentor	102.5
Heraldo	4.5
Guardián	785.3
Aliado	946.7
Bufón	493.1
Cambiaformas	13.1
Monomito	66.7

Tabla 3: Ejemplo de valores que podría tener el promedio de veces que apareció cada arquetipo durante las simulaciones. Los valores son reales y puede tomar cualquier valor positivo incluyendo el 0.

Fuente: Elaboración propia.

Estructura General

El pseudocódigo 3 está basado en la estrategia evolutiva descrita en la sección 2.3.2 y se considera como la solución al problema presentado. A grandes rasgos, comienza con una solución inicial, escoge los genes a modificar en base a una probabilidad y a estos se les aplica una mutación gaussiana que depende de los valores de $\sigma(g_1, \dots, g_n)$. Estos valores se actualizan solo para los genes que mutaron y dicha modificación depende de la calidad del hijo resultante; si el hijo mejora o si un número aleatorio es menor que la probabilidad de aceptar una peor solución, los valores de $\sigma(g_1, \dots, g_n)$ aumentan y el nuevo hijo se convierte en el padre de la siguiente generación; ahora, si el hijo tiene una menor aptitud, los $\sigma(g_1, \dots, g_n)$ disminuyen y el padre se mantiene; finalmente, si padre e hijo tienen la misma aptitud, entonces los $\sigma(g_1, \dots, g_n)$ se mantienen y el hijo pasa a ser el padre. Este proceso se repite hasta que se cumpla alguno de los criterios que se detallan en la sección 3.3.3. Si alguna de estas situaciones ocurre, el algoritmo se reinicia y comienza nuevamente. El proceso completo ocurre hasta completar un límite máximo de evaluaciones definido por el usuario.

Algoritmo 3 Estrategia Evolutiva versión final

Entrada: $probabilidad_mutacion, p, k, probabilidad_aceptar, t, x_0(g_1, \dots, g_n)$
 $evaluaciones \leftarrow 0$
 $mejor_solucion(g_1, \dots, g_n) \leftarrow x_0(g_1, \dots, g_n)$
mientras $evaluaciones \leq p$ **hacer**
 Inicializar $\sigma_0(\sigma_1, \dots, \sigma_n)$
 repetir
 $x_n(g_1, \dots, g_n) \leftarrow$ se mutan los genes
 si $f(\tilde{x}_i(g_1, \dots, g_n)) > f(x_i(g_1, \dots, g_n))$ **o** $random < probabilidad_aceptar$ **en-**
tonces
 $x_{i+1}(g_1, \dots, g_n) = \tilde{x}_i(g_1, \dots, g_n)$
 $\sigma_{i+1} \leftarrow$ se actualizan los valores (aumentan)
 sino si $f(\tilde{x}_i(g_1, \dots, g_n)) = f(x_i(g_1, \dots, g_n))$ **entonces**
 $x_{i+1}(g_1, \dots, g_n) = \tilde{x}_i(g_1, \dots, g_n)$
 $\sigma_{i+1}(\sigma_1, \dots, \sigma_n) = \sigma_i(\sigma_1, \dots, \sigma_n)$
 sino
 $x_{i+1} = x_i(g_1, \dots, g_n)$
 $\sigma_{i+1} \leftarrow$ se actualizan los valores (disminuyen)
 fin si
 si $f(x_i(g_1, \dots, g_n)) \geq f(mejor_solucion(g_1, \dots, g_n))$ **entonces**
 $mejor_solucion(g_1, \dots, g_n) = x_i(g_1, \dots, g_n)$
 fin si
 $evaluaciones ++$
 hasta $f(mejor_solucion(g_1, \dots, g_n))$ no haya cambiado en k iteraciones o haya apa-
 recido un monomito o lleve un tiempo t tratando de resolver un *halting*
fin mientras
retornar $mejor_solucion(g_1, \dots, g_n)$

Inicialización

Para que el algoritmo evolutivo inicie su funcionamiento se requieren los siguientes parámetros: probabilidad de mutación, cantidad de iteraciones que pueden pasar sin que el padre cambie, cantidad máxima de evaluaciones, probabilidad de aceptar una peor solución, tiempo máximo en el que intenta resolver un *halting* y una solución inicial de la cual comenzar. La sintonización de estos parámetros se hizo a través de experimentación y se irá explicando en lo que queda de capítulo.

Debido al tamaño del gen y los posibles valores que pueden tener, el espacio de búsqueda es muy grande por lo que no es bueno comenzar la búsqueda de un punto totalmente aleatorio, pues esto podría significar no encontrar una buena solución, es decir, podría no encontrar un monomito.

Por esta razón se tomaron todas aquellas soluciones que habían encontrado un monomito

para todas las instancias y se trató de buscar un patrón común. La idea de esto era confeccionar una máscara⁸ que determinara qué valores servían para elaborar una solución inicial adecuada. Se decidió de esta manera tomar la mejor solución de todas y aplicarle la máscara, obteniendo el punto de partida buscado.

Para ver cuánto porcentaje de la solución base se necesitaba para lograr buenos resultados, se decidió establecer una máscara que tomara al azar un porcentaje de los genes (como se muestra en la figura 4). Así, la solución inicial fue construida como se muestra en el algoritmo 4. Se comienza recorriendo todos los genes y analizando para cada uno el uso de la máscara. En caso de utilizarlo, el valor de la solución base se hereda en la solución inicial; en otro caso, el gen es inicializado aleatoriamente.

Algoritmo 4 Construcción solución inicial máscara inferior al 100 %

Entrada: $x_0(g_1, \dots, g_n)$, *percentage*
desde $j = 0; j < n; j++$ **hacer**
 si $random < percentage/100$ **entonces**
 $x_0[g_j] = x_0[g_j]$
 sino
 $x_0[g_j] \leftarrow$ número aleatorio entre 0 y 1
 fin si
fin desde
retornar $x_0(g_1, \dots, g_n)$

0.3	0.6	0.1	0.8	0.2	0.7	...	0.35	0.5
-----	-----	-----	-----	-----	-----	-----	------	-----

Tabla 4: Ejemplo de una máscara. Los valores no pintados se establecen de manera aleatoria.
Fuente: Elaboración propia.

El efecto que tiene utilizar una máscara basada en buenas soluciones obtenidas con anterioridad le permite al algoritmo aprender de estos individuos de manera que se obtengan soluciones más rápido y que surjan monomitos en estas.

Mutación

Debido a que el gen es muy grande, los hijos son muy distintos a los padres si todos los genes mutan. Por esta razón se limita la mutación de los genes en base a una probabilidad, es decir, el gen muta solo si el valor de un número al azar es menor a dicha probabilidad. En el algoritmo 5 se selecciona el número aleatorio, se verifica que sea menor a la probabilidad

⁸Arreglo de 0 y 1 que determina qué valores de una solución determinada se van a utilizar para armar la solución inicial y cuáles serán fijados al azar.

de mutación y si resulta ser así se muta. En caso de que el gen cambie, se guarda su índice en un arreglo que posteriormente será utilizado para saber qué valores de σ se van a actualizar.

Algoritmo 5 Mutación para ciertos genes

```

indice_mutacion( $i_1, \dots, i_n$ )  $\leftarrow \vec{0}$ 
/* Se modifican solo los genes flotantes */
desde  $j = 0; j < n; j++$  hacer
    si  $random < probabilidad\_mutacion$  entonces
         $\tilde{x}_i[g_j] = x_i[g_j] + \sigma_i[\sigma_j]\mathcal{N}(0, 1)$ 
         $indice\_mutacion[i_j] = 1$ 
    sino
         $\tilde{x}_i[g_j] = x_i[g_j]$ 
    fin si
fin desde

```

Un detalle es que si el cambio produce un valor que excede el máximo, el gen adquiere el valor máximo posible y lo mismo ocurre con el mínimo. El arreglo $\sigma(\sigma_1, \dots, \sigma_n)$ se inicia con valores aleatorios. Para fijar la probabilidad de mutación se probaron diferentes valores y se decidió establecerla en 0.1, pues fue la que arrojó mejores resultados.

También es importante mencionar que tal como se dijo antes, los valores de $\sigma(\sigma_1, \dots, \sigma_n)$ se actualizan solo para aquellos genes que mutaron. En el algoritmo 6 se muestra cómo cambian estos valores en caso de que la aptitud del hijo sea mejor que la del padre. Primero se verifica si σ_j debe ser actualizado; de ser así, se multiplica su valor por 1,5. De lo contrario, se mantiene intacto su valor.

Algoritmo 6 Aumento de los valores de σ según los genes mutados

```

desde  $j = 0; j < n; j++$  hacer
    si  $indice\_mutacion[i_j] = 1$  entonces
         $\sigma_{i+1}[\sigma_j] = 1,5 \sigma_i[\sigma_j]$ 
    sino
         $\sigma_{i+1}[\sigma_j] = \sigma_i[\sigma_j]$ 
    fin si
fin desde

```

Ahora bien, si la función de aptitud del hijo no superó a la del padre, los valores de $\sigma(\sigma_1, \dots, \sigma_n)$ deben disminuir según se muestra en el algoritmo 7; se realiza un proceso similar al anterior pero ahora el valor de σ_j se multiplica por $1,5^{-1/4}$ (en caso que deba ser actualizado).

Algoritmo 7 Dsiminución de los valores de σ según los genes mutados

```

desde  $j = 0; j < n; j++$  hacer
  si  $indice\_mutacion[i_j] = 1$  entonces
     $\sigma_{i+1}[\sigma_j] = 1,5^{-1/4} \sigma_i[\sigma_j]$ 
  sino
     $\sigma_{i+1}[\sigma_j] = \sigma_i[\sigma_j]$ 
  fin si
fin desde

```

Criterios de Reinicio

Comenzar desde una buena solución o incluir una probabilidad de mutación no es suficiente para asegurar que el algoritmo encontrará buenas soluciones. Es necesario introducir más variabilidad en el algoritmo y para esto se decidió agregar reinicios⁹ de manera que si el algoritmo se empieza a alejar de un óptimo, este tenga la oportunidad de comenzar la búsqueda nuevamente.

Para definir un reinicio se consideran los siguientes criterios:

- **Número máximo de iteraciones sin mejorar el resultado:** De manera de prevenir el estancamiento en el proceso de mejoramiento del individuo, se estableció que cumplida una cierta cantidad de iteraciones sin que se mejorara la solución, se reiniciaría el algoritmo para darle la posibilidad de seguir explorando.
- **Aparición de monomitos:** Como lo que se quiere conseguir es que surjan monomitos, lo lógico sería orientar la búsqueda a ese objetivo. Se estableció como otro criterio de reinicio la aparición de un monomito (esto para acelerar los reinicios y darle variabilidad al algoritmo).
- **Tiempo de resolución de *halting***¹⁰: Las buenas soluciones presentan una alta cantidad de *haltings*, pues cuando la calidad de la solución es buena, se genera una cantidad tal de eventos que hacen que el razonador se estanque intentando determinar los arquetipos que surgen. Por esta razón se determinó que si el algoritmo llevaba mucho tiempo tratando de resolver un *halting*, una vez que lo lograra se reiniciaría el algoritmo. Esto porque si el programa se encuentra en esta situación quiere decir que la calidad de la solución que está evaluando es tan buena que difícilmente será mejorada.

⁹cambiar los puntos iniciales del algoritmo y comenzar de nuevo

¹⁰Se refiere a cuando un programa detiene su ejecución. En este caso, se le llama *halting* a cuando el código trata de encontrar la función de evaluación pero no logra resolver los predicados en un tiempo determinado, por lo que se procede a detener la evaluación y ver si hay otras alternativas.

Criterio de Término

Se estableció como criterio de término un número máximo de iteraciones que le permitieran al algoritmo tener la cantidad suficiente de reinicios y encontrar buenas soluciones. Este criterio sirve también para hacer comparable el número de evaluaciones que realiza esta estrategia respecto al algoritmo genético.

Exploración del algoritmo

Otro método de exploración que tiene la propuesta anterior (aparte de los reinicios) es la capacidad de aceptar peores soluciones en base a una probabilidad. Esto significa que a la hora de elegir el padre, no solo fijamos la atención en el valor de la función de aptitud sino que también consideramos la probabilidad de aceptar una solución peor. Entonces, se acepta el hijo como padre si es que la aptitud de este es igual o mejor, o si un número aleatorio resulta menor que la probabilidad. Esto también sirve para mitigar la variabilidad de la función de evaluación, pues si un individuo se evalúa más de una vez, no necesariamente obtiene los mismos resultados. Al aceptar peores soluciones se acepta también la posibilidad de que aunque la evaluación fue más baja, si se hubiera evaluado otra vez tal vez hubiera sido mejor que la aptitud del padre.

Resumen Se plantea una estrategia evolutiva con una cantidad de reinicios variable y sujeta a un conjunto de criterios que le otorgan variabilidad al algoritmo y hacen que no se estanque. Comenzar de una solución inicial buena (o al menos de un porcentaje de esta) proporciona al algoritmo un punto inicial en una zona del espacio de búsqueda donde existe la posibilidad de encontrar buenas soluciones. El mutar solo una cierta cantidad de los genes permite conservar cierto parecido entre padre e hijo, lo cual ayuda a perpetuar las buenas características del predecesor, evitando la destructividad que podría ocasionar el operador.

Parámetro	Descripción
<i>probabilidad_mutacion</i>	Probabilidad en base a la cual se decide si se muta o no un gen.
<i>p</i>	Cantidad máxima de evaluaciones que realiza el algoritmo antes de detener su ejecución.
<i>k</i>	Cantidad de iteraciones seguidas en las cuales no cambia la aptitud del padre, es decir, no se consigue algún hijo con mayor aptitud.
<i>probabilidad_aceptar</i>	Probabilidad en base a la cual se acepta una peor solución.
<i>t</i>	Tiempo máximo en el cual el programa trata de resolver los <i>haltings</i>

Tabla 5: Resumen del significado de cada parámetro utilizado en la estrategia evolutiva.
Fuente: Elaboración propia.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

Detalles del código

El simulador está programado en java versión 1.8 y utiliza maven¹¹ 3.3 para compilar. El código original sobre el cual se trabajó puede ser encontrado en un repositorio github¹². Para aprovechar al máximo el trabajo ya hecho se optó por utilizar las mismas herramientas y versiones. El código propuesto en esta memoria puede ser encontrado también en un repositorio github¹³.

Objetivos

Para validar la solución propuesta se compararon los resultados obtenidos por la estrategia evolutiva modificada con los obtenidos por el algoritmo genético original de [García-Ortega et al., 2016]. Para esto, primero se presenta una evaluación del algoritmo genético en un conjunto específico de instancias y luego se realiza lo mismo para la estrategia evolutiva utilizando diferentes máscaras. Posteriormente se analizan dichos resultados y se hacen las comparaciones correspondientes para finalizar con un resumen de lo concluido.

Instancias

Para las pruebas se utilizaron 3 instancias; la característica de estas es que sus 5 primeros genes están fijos. Dichas instancias se muestran en la tabla 6.

Genes	Instancia 1	Instancia 2	Instancia 3
Tamaño del mundo	8x8	8x8	16x16
Cantidad de círculos	8	4	16
Cantidad de triángulos	8	4	16
Cantidad de cuadrados	8	4	16
Cantidad de días transcurridos	64	64	64

Tabla 6: Características de las instancias que se utilizaron para las pruebas.

Fuente: Elaboración propia.

¹¹Herramienta de software para la gestión y construcción de proyectos Java

¹²<https://github.com/raiben/made>

¹³https://github.com/rmella/made_evolutionary

La instancia 1 es relativamente pequeña; el mundo es del tamaño mínimo considerado por el autor de [García-Ortega et al., 2016] y la cantidad de días es la mitad del máximo considerado por García. La cantidad de personajes hace que no sea muy grande. La instancia 2 es la más pequeña de todas, ya que al igual que la instancia 1 cuenta con el tablero más pequeño, una acotada cantidad de personajes y la misma cantidad de días. La instancia 3 en cambio es la más grande de todas, ya que se utilizaron los valores máximos posibles utilizados por el autor (excepto para el número de días). Estas instancias se eligieron en base a 4 criterios descubiertos durante los experimentos iniciales:

- Mientras más pequeño es el mundo, mejor es la función de aptitud, pues los personajes están más cerca entre sí y es más probable que se produzca interacción entre ellos.
- A mayor cantidad de días, también mejora la función de aptitud, pues el simulador tiene más tiempo para poder generar interacción entre los personajes.
- La función de aptitud aumenta si hay más personajes, pues también aumenta la probabilidad de que interactúen entre sí.
- Mientras más pequeño el mundo, mayor es la aptitud, pues al estar más juntas las piezas se produce mayor interacción entre ellas.

Debido a las características que poseen los videojuegos era necesario probar más de una instancia y que estas reflejaran los parámetros reales que tiene un juego. Además, para probar que la solución propuesta era lo bastante genérica como para aplicarse a cualquier prueba fue necesario crear más de una. Se limitó eso si la cantidad de instancias debido a los altos tiempos de ejecución del código.

Luego de las pruebas realizadas durante la creación de la estrategia evolutiva, se decidió establecer los siguientes valores para los parámetros del algoritmo:

Parámetro	Valor
<i>probabilidad_mutacion</i>	0.1
Cantidad máxima de evaluaciones (p)	900
Cantidad de iteraciones sin cambiar la aptitud (k)	30
<i>probabilidad_aceptar</i> (una peor solución)	0.01
Tiempo máximo de resolución de <i>haltings</i> (t)	35 min

Tabla 7: Valores de los parámetros que se utilizaron finalmente.

Fuente: Elaboración propia.

Los valores de la tabla 7 fueron los que mejor funcionaron con la estrategia evolutiva planteada en el capítulo anterior.

Algoritmo Genético

El código original del algoritmo genético se ejecutó una vez para cada instancia y con tres semillas aleatorias diferentes. Se utilizaron como parámetros para el algoritmo genético los mismos que en [García-Ortega et al., 2016], es decir, una población de tamaño 30, con 30 iteraciones máximo. En la tabla 8 se pueden apreciar los mejores valores de cada una de las 3 funciones de aptitud (A, AD y M) y la máxima cantidad de monomitos encontrados para las 3 semillas obtenidos por el algoritmo original de García planteado en [García-Ortega et al., 2016]. La idea de estos resultados era conocer cual fue la aptitud máxima encontrada por el algoritmo genético (es decir, la obtenida por el mejor individuo de todos) y cual era la cantidad máxima de monomitos que se podían encontrar con este algoritmo. Cabe recordar que la función de aptitud A (ecuación 7) cuenta la aparición de todos los arquetipos, la AD (ecuación 8) también hace lo mismo pero privilegiando el surgimiento de diferentes arquetipos y la M (ecuación 9) cuenta la cantidad de monomitos que aparecen.

	Semilla 1	Semilla 10	Semilla 100
Instancia	Máxima Función de Aptitud A/Máxima cantidad de monomitos		
1	617.1/0	1107.3/0	1164.7/264
2	989.8/0	530.6/0	601.5/70
3	1166.2/0	1118.5/0	1761.6/0
Instancia	Máxima Función de Aptitud AD/Máxima cantidad de monomitos		
1	15.0/0	14.0/0	15.3/0
2	11.0/0	11.3/0	11.9/0
3	17.1/0	16.9/0	16.6/0
Instancia	Máxima Función de Aptitud M/Máxima cantidad de monomitos		
1	0/0	0/0	0/0
2	0/0	41.1/41.1	0/0
3	3.2/3.2	0/0	0/0

Tabla 8: Resultados Algoritmo Genético con las tres instancia y semillas.

Fuente: Elaboración propia.

Analizando los resultados obtenidos en la tabla 8 se puede apreciar que la mayor cantidad de monomitos fueron encontrados con la semilla 100 y que no necesariamente están asociados a una mayor aptitud; por ejemplo, para la instancia 2 la aptitud más alta se obtuvo con la semilla 1 pero esta no produjo ningún monomito (a diferencia de la semilla 100). La función M también encontró monomitos pero en menor cantidad. Es interesante destacar que la máxima cantidad de monomitos encontrados para la instancia 1 y 2 con la semilla 100 no coincide con la máxima aptitud obtenida. Esto revela que no siempre se cumple que a mayor

función de aptitud, más cantidad de monomitos emergen (esto se puede apreciar mejor en las figuras 7, 8, 9).

Como se puede apreciar en la tabla 8, en la mayoría de las ocasiones no se encontró ningún monomito, por lo que un alto valor en la función de aptitud no asegura que surjan. Por esta razón, para medir la calidad de los algoritmos propuestos se buscaron aquellos que hicieran surgir al menos un monomito, sin centrarse tanto en la función de aptitud (esta se utilizó para hacer evolucionar el algoritmo pero no es un factor determinante en la calidad de la solución). Otro punto importante a destacar en cuanto a la comparación de resultados es que no se consideró el tiempo de ejecución como una medida de optimización, sino más bien la cantidad de evaluaciones que se realizan para llegar a soluciones con monomitos. Esta decisión se tomó en base a la imposibilidad de reducir el tiempo de evaluación de la función de aptitud, pues como se mencionó en el capítulo anterior, este depende de la cantidad de eventos que surjan en la simulación. Además, debido a la extensa duración de la ejecución se utilizaron diferentes equipos con diferentes cargas y características para las pruebas, por lo que la información del tiempo de ejecución no es comparable.

Iteración	Aliado	Conflicto	Guardián	Heraldo	Héroe	Viaje	Mentor	Monomito	Sombra	Cambiaformas	Bufón
0	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
1	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
2	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
3	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
4	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
5	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
6	32,3	258,2	102,4	0,9	14,9	14,9	6,9	0	14,9	0	40,6
7	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
8	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
9	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
10	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
11	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
12	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
13	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
14	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
15	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
16	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
17	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
18	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
19	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
20	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
21	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
22	183,9	159,7	93,1	0,3	13,9	13,9	4,7	0	13,9	0	51,8
23	110,8	211,3	104,3	0,1	19,1	19,1	5,6	0	19,1	1,1	54,6
24	110,8	211,3	104,3	0,1	19,1	19,1	5,6	0	19,1	1,1	54,6
25	110,8	211,3	104,3	0,1	19,1	19,1	5,6	0	19,1	1,1	54,6
26	110,8	211,3	104,3	0,1	19,1	19,1	5,6	0	19,1	1,1	54,6
27	121,3	205,1	107,2	0,2	20,5	20,5	8,7	0	20,5	16,9	60,2
28	96,3	237,4	133,6	0,1	17,9	17,9	5,8	0	17,9	0,1	54,9
29	90,2	253,3	143	0,9	21,4	21,4	9,2	0	21,4	0,1	56,3
30	90,2	253,3	143	0,9	21,4	21,4	9,2	0	21,4	0,1	56,3

Tabla 9: Promedio de aparición de cada arquetipo para una ejecución del algoritmo genético con la instancia 1 y semilla 1.

Fuente: Elaboración propia.

En la tabla 9 se muestra el promedio de aparición de cada arquetipo para la ejecución del algoritmo genético con la instancia 1 y semilla 1. En esta se puede apreciar que los arquetipos que más surgen son el *Conflicto*, el *Aliado* y el *Guardián*. Esto significa que el valor de la aptitud en dichos casos se ve influenciado principalmente por estos arquetipos. Otra cosa importante de destacar es que el promedio de aparición para el *Héroe*, la *Sombra* y el *Viaje* es el mismo; esto tiene sentido, pues hay que recordar que el *Viaje* se definía como el tiempo en que ocurrían *Conflictos* entre los mismo dos personajes. También se observa que no aparecen monomitos en ningún momento.

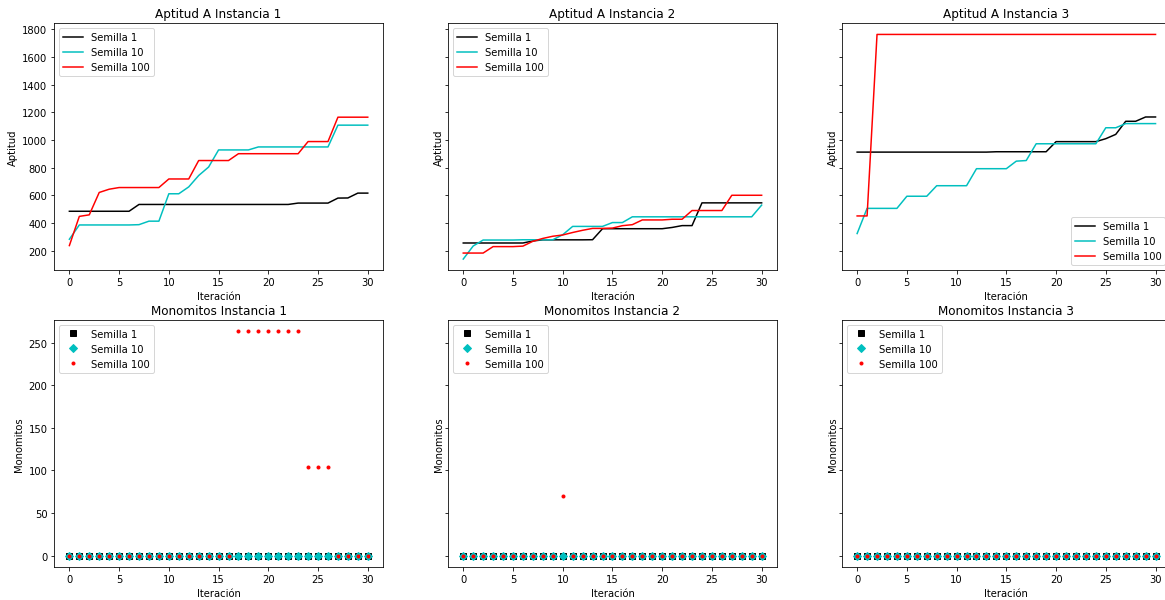


Figura 7: Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.

Fuente: Elaboración propia.

En la primera fila de la figura 7 se muestra para cada instancia estudiada un gráfico de la función de evaluación A de ecuación 7 que cuenta la aparición de todos los arquetipos; dicha función se utilizó para medir la calidad de las soluciones que iba generando el algoritmo genético. El eje x representa la iteración, mientras que el eje y muestra la aptitud del **mejor individuo de todos** hasta ese momento. En estos gráficos se puede apreciar que en el caso de la instancia 2, las 3 semillas tienen resultados similares; esto no ocurre con las otras dos instancias, pues por ejemplo, en el caso de la instancia 3 la semilla 100 tiene un comportamiento muy diferente a las otras. También se puede notar que los gráficos de cada instancia parten desde un punto similar y tienen una subida estable. En la segunda fila se pueden observar los gráficos de la cantidad de monomitos que aparecen en cada instancia para cada semilla. El eje x representa la iteración y el eje y muestra los monomitos generados por el mejor individuo de todos hasta dicha iteración. Se ve que la única semilla que logra hacerlos surgir en baja cantidad es la 100 y esto es solo para las instancias 1 y 2. Es importante

destacar que en la instancia 1, existe una caída en la cantidad de monomitos a medida que la aptitud aumenta; esto refuerza lo dicho anteriormente respecto a que no es posible establecer una relación directa entre la aptitud y la cantidad de monomitos, pues una función con mayor aptitud podría tener menos de estos.

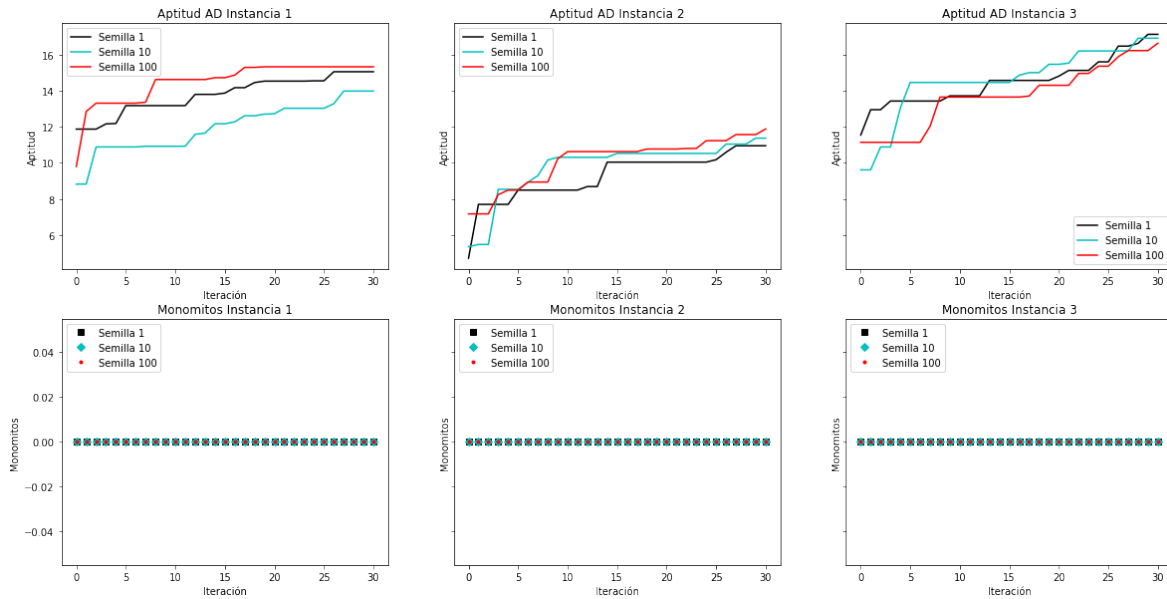


Figura 8: Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **AD** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.

Fuente: Elaboración propia.

En la figura 8 (primera fila) también se muestra para cada instancia un gráfico de la función de evaluación (utilizada por el algoritmo genético) pero en este caso se usa la AD (ecuación 8) que le da énfasis a la aparición de dos arquetipos diferentes al aplicar logaritmo. El valor de la aptitud al igual que antes corresponde al **mejor individuo de todos** hasta ese momento. En ellos se puede notar que las tres semillas tienen comportamientos similares para las tres instancias, aunque en el caso de la segunda los valores obtenidos por la aptitud son más bajos que las demás. En la segunda fila se encuentran los gráficos de la cantidad de monomitos versus la iteración en que se producen. En estos se puede apreciar que con ninguna semilla se encuentra un monomito, lo que podría explicarse con los bajos resultados que genera la función de aptitud; como esta promueve la aparición de nuevos arquetipos y al comienzo suelen aparecer más *Conflictos*, *Aliados* y *Guardianes*, a la población le cuesta más evolucionar.

En la figura 9 (primera fila) se muestran los resultados del algoritmo genético usando la función de evaluación M (ecuación 9) que cuenta la cantidad de veces que aparecen monomitos durante las simulaciones. En la segunda fila se encuentran los gráficos que muestran la aparición de monomitos en cada iteración para el mejor individuo encontrado hasta ese punto.

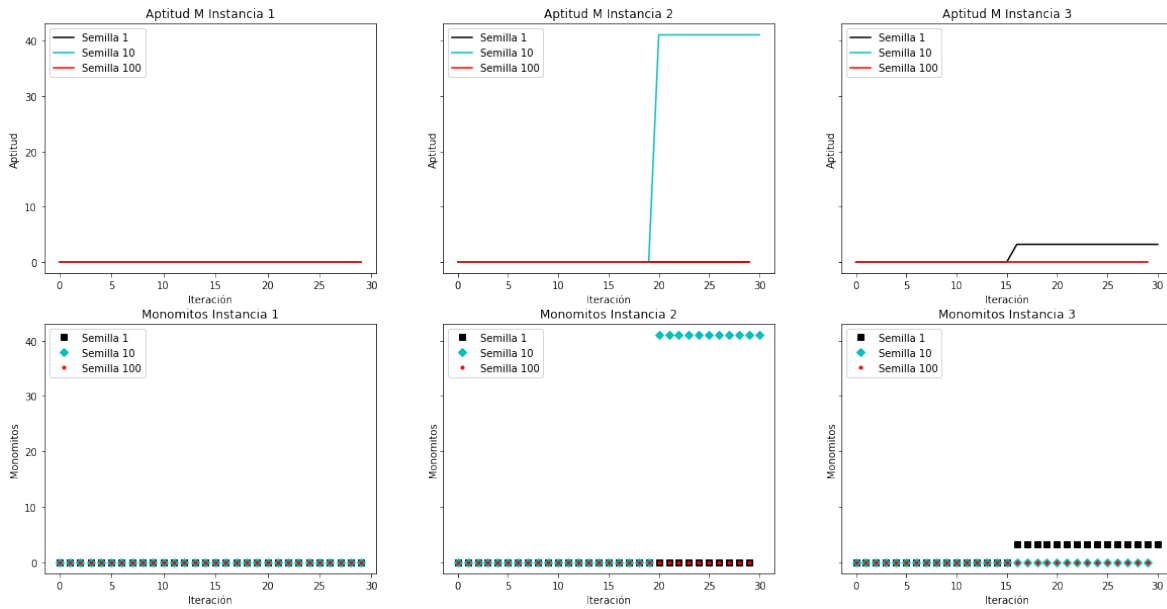


Figura 9: Gráficos de los resultados del algoritmo genético original para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **M** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función.

Fuente: Elaboración propia.

Debido a que la función de aptitud también cuenta la cantidad de veces que surgen los monomitos, los gráficos de la primera fila son iguales a los de la segunda. En estos se aprecia que no se encuentran monomitos salvo en determinados casos. Esto ocurre por una razón similar a lo que sucede con la función AD; como la aptitud M depende de que aparezcan monomitos, en un comienzo es difícil que surjan, por lo que cuesta que el algoritmo evolucione debido a que al comienzo se obtienen muchos resultados de valor 0. Esta búsqueda se podría traducir en una aleatoria más que evolutiva por el tipo de función objetivo, lo que explicaría el comportamiento y los monomitos que encuentra.

Estrategia Evolutiva

Para que los resultados de la estrategia evolutiva propuesta fuesen comparables con los obtenidos por el algoritmo genético, debían realizarse la misma cantidad de evaluaciones. Como en el algoritmo genético se realizaron 30 iteraciones con una población de tamaño 30, la cantidad total de evaluaciones que se realizaron fueron 900. Se consideró el número de evaluaciones como criterio a comparar en vez del tiempo debido a que era un criterio más fiable; la demora excesiva en la ejecución del código, las pruebas en diferentes computadores con diferentes cargas, entre otras cosas, le añadían un exceso de ruido a la medida del tiempo (como se mencionó anteriormente).

Comparar ambos algoritmos requería que se realizaran las mismas pruebas, por lo que se utilizaron las mismas tres semillas (1, 10 y 100) con máximo 900 evaluaciones, para las 3 instancias y las 3 funciones de evaluación. A continuación se evalúa el desempeño de la estrategia evolutiva considerando 3 escenarios de tamaño de máscara de manera de analizar su influencia sobre la solución final y comparar cada una con el resultado obtenido por el algoritmo genético.

Máscara 100 %

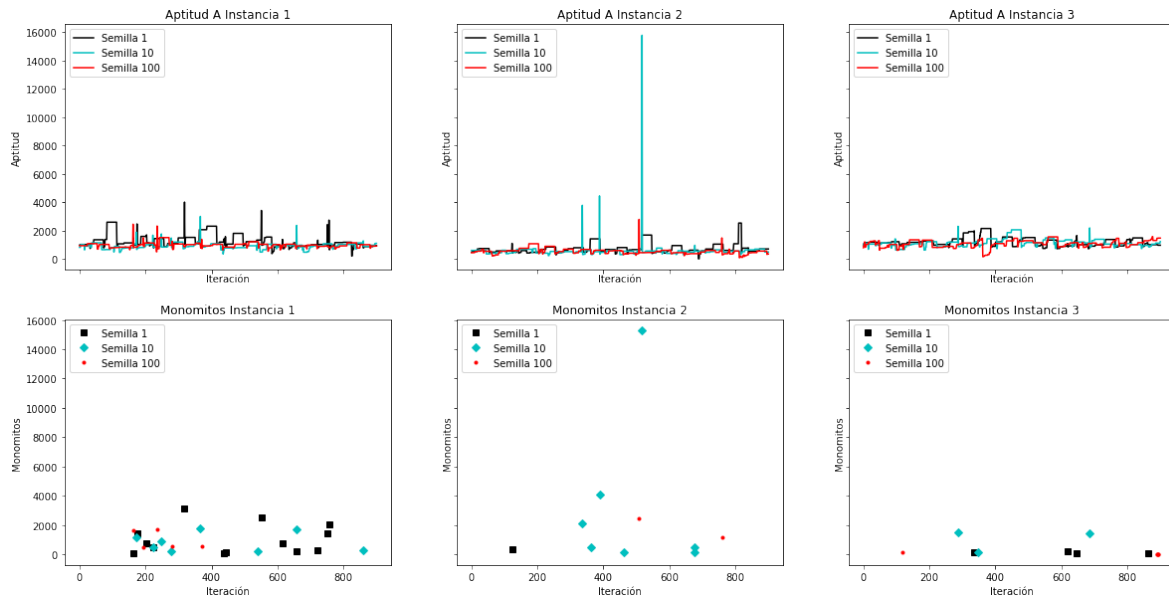


Figura 10: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

La figura 10 al igual que las otras anteriores muestra en la primera fila los gráficos de la función de aptitud A utilizada por la estrategia evolutiva planteada para evolucionar. Se usa una máscara del 100 % para elegir los genes de la solución inicial, es decir, se ocupa el individuo completo. En este caso, el eje x representa la iteración realizada y el eje y muestra la aptitud del padre en dicha iteración. Es importante recordar en este punto que como la estrategia evolutiva solo trabaja con un individuo, la cantidad de iteraciones es igual a la cantidad de evaluaciones realizadas. La segunda fila de la figura muestra la cantidad de monomitos (eje y) surgidos en cada iteración (eje x). En las imágenes se puede apreciar que alrededor de las 200 iteraciones, el algoritmo comienza a encontrar monomitos. Esto significa que si el usuario lo desea, podría terminar la ejecución del programa antes de las 900 iteraciones y aún así obtendría buenas soluciones. Se puede notar que en el caso de la instancia 2, la semilla

10 se destaca por sobre las otras, obteniendo entre 15000 y 16000 monomitos. En cambio, en la instancia 3 el comportamiento de las 3 semillas fue similar (aunque la semilla 10 fue ligeramente mejor).

En la figura 11 se pueden observar los resultados obtenidos por la estrategia evolutiva usando la función de evaluación AD y una máscara del 100 %. En la primera fila se ven los gráficos de la aptitud del padre en cada iteración, mientras que en la segunda aparecen los monomitos. Al igual que para la aptitud A también se aprecia la aparición de monomitos, aunque en el caso de la instancia 2, hay muy pocas ocurrencias. Esto podría explicarse con el tamaño de la instancia y las características de la función objetivo; debido a que existen pocos personajes, es más difícil que interactúen entre sí. Esto, sumado al hecho de que a la función le cuesta empezar a evolucionar debido a la promoción de nuevos arquetipos (como se dijo anteriormente), esta instancia no logra encontrar tantos monomitos con esta función como lo hace con la anterior. De todas maneras es valorable que aún en esas condiciones logre encontrarlos, cosa que no ocurre con el algoritmo genético.

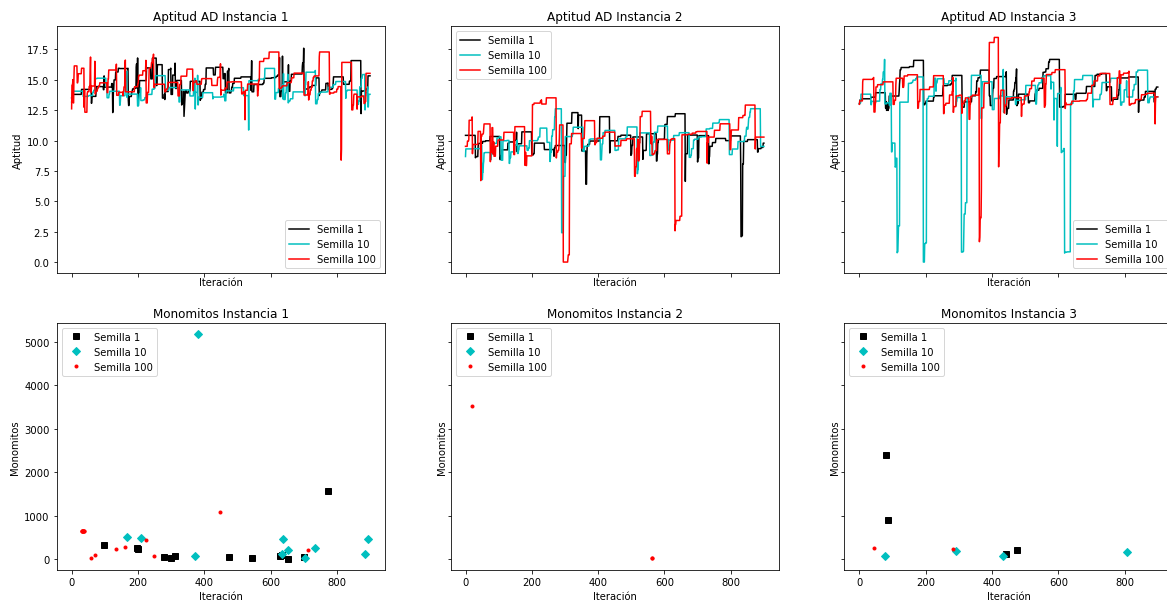


Figura 11: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **AD** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

En la figura 12 se muestran los resultados que se obtuvieron con la estrategia evolutiva usando la máscara del 100 % y la función de evaluación M que como se recordó anteriormente, cuenta los monomitos que aparecen. En la primera fila al igual que en los otros gráficos se muestra la aptitud del padre, mientras que en la segunda se pueden encontrar los monomitos que este genera por cada iteración (omitiendo cuando no encuentra). Ambos gráficos

son iguales debido a lo que representa la función de evaluación utilizada. En estos es posible ver una mejora respecto al algoritmo genético en esta función específica, pero esto se debe a que al partir de una buena solución, se está cerca de un máximo local, es decir, sigue pareciendo una búsqueda aleatoria más que evolutiva pero que tiene la ventaja de estar en una buena zona del espacio de búsqueda. Además, la cantidad de monomitos encontrados supera con creces a los que surgen en el algoritmo genético, lo cual refuerza la teoría del espacio de búsqueda.

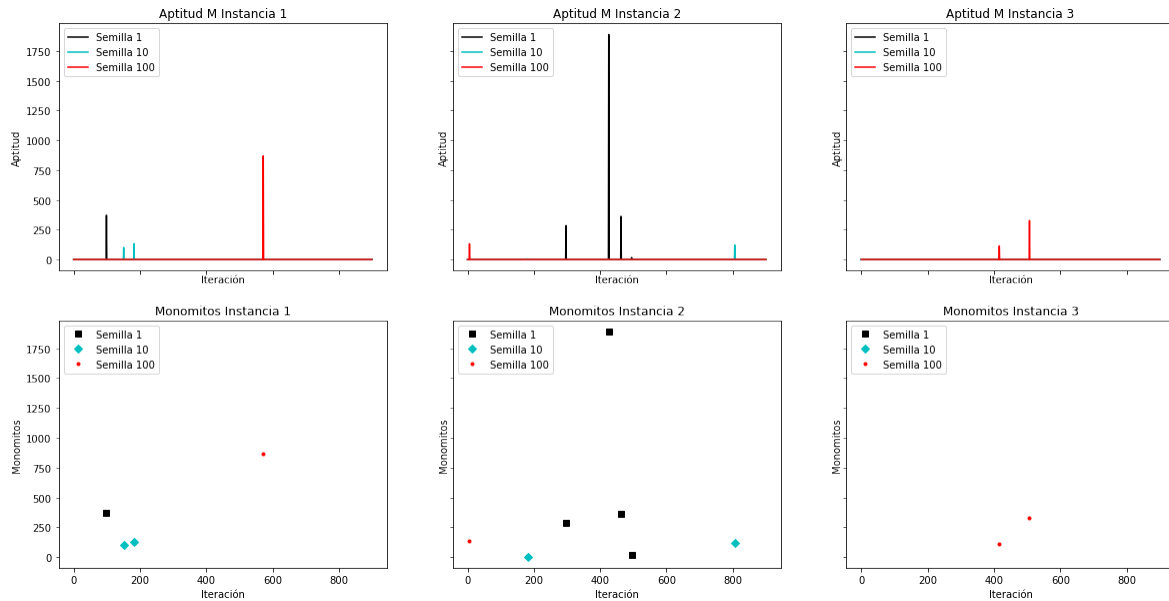


Figura 12: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 100 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **M** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

Máscara 70 %

Para comparar la máscara del 100 % con la del 70 % se realizaron los mismos gráficos también para las 3 instancias, con las 3 semillas y las 3 funciones de evaluación. En el caso de la máscara del 70 %, los resultados de las figuras 13, 14 y 15 son similares a los obtenidos por la máscara del 100 %, pero con valores más bajos. Esto significa que utilizar un menor porcentaje de la solución inicial también entrega buenos resultados, aunque estos son inferiores a los obtenidos con el 100 % de la solución. Usar una máscara más pequeña mejora el tiempo de ejecución del algoritmo que aunque sigue siendo alto, se demora menos tiempo que el algoritmo genético. Un fenómeno interesante que ocurre en la función AD para la máscara 70 % es que la función de aptitud comienza de 0 (cuando se reinicia) en muchas más ocasiones que la máscara del 100 % (ver figura 11 para comparar). Esto aportaría a la disminución en la

calidad de las soluciones, pero no afectaría significativamente a la aparición de monomitos.

Cabe destacar que el tener varias buenas soluciones le entrega diversidad al sistema, es decir, el creador de un videojuego tendrá una gama más amplia de opciones para elegir al momento de utilizar la solución entregada por el modelo.

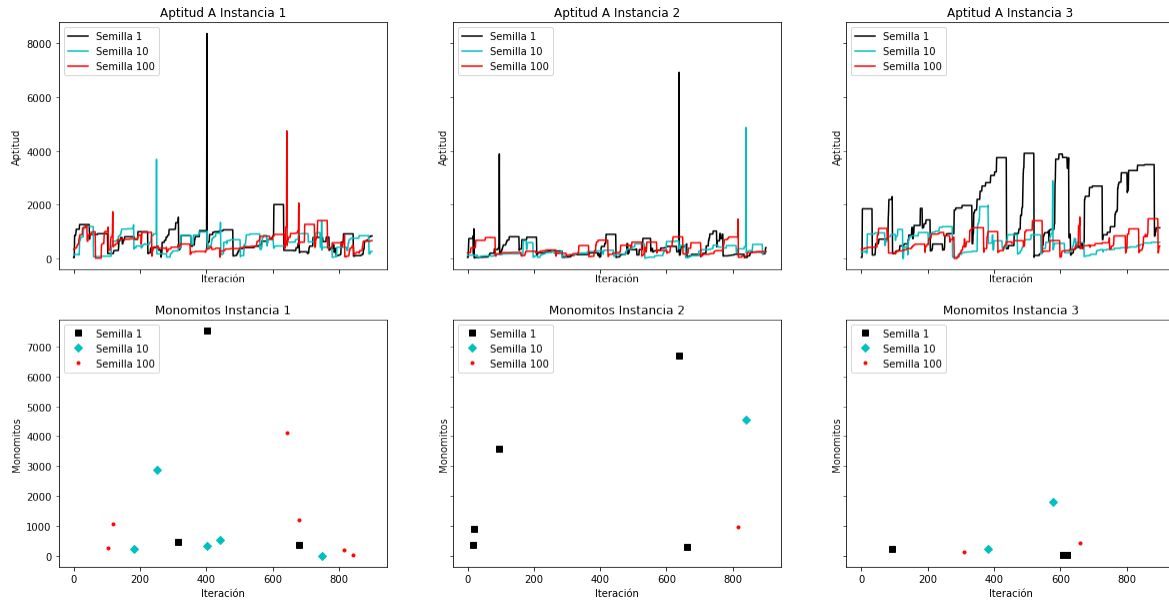


Figura 13: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **A** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

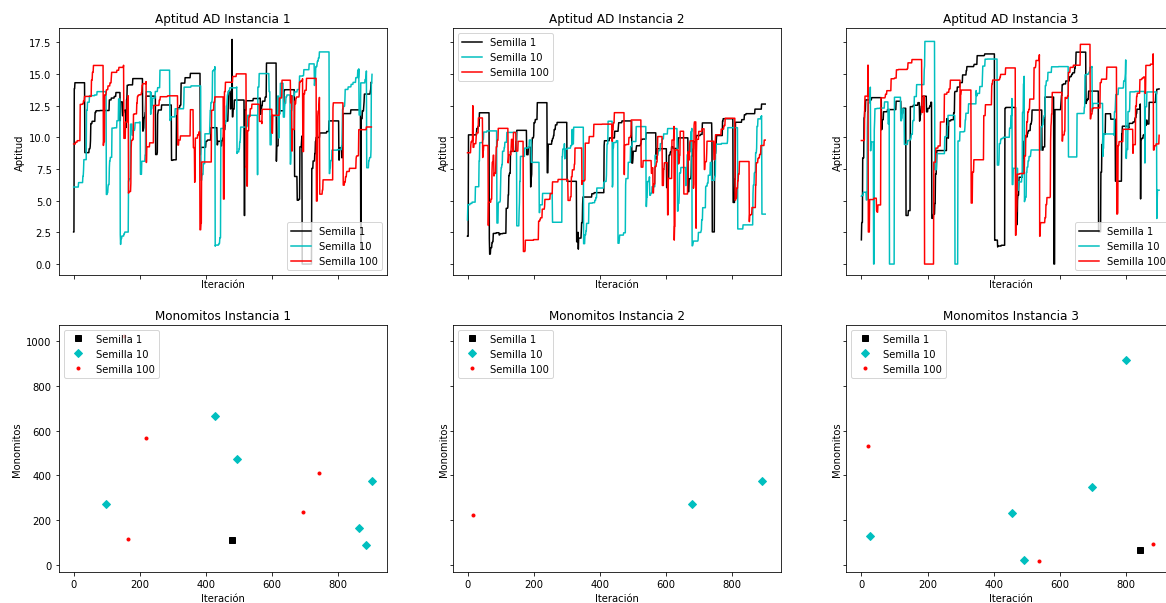


Figura 14: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **AD** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

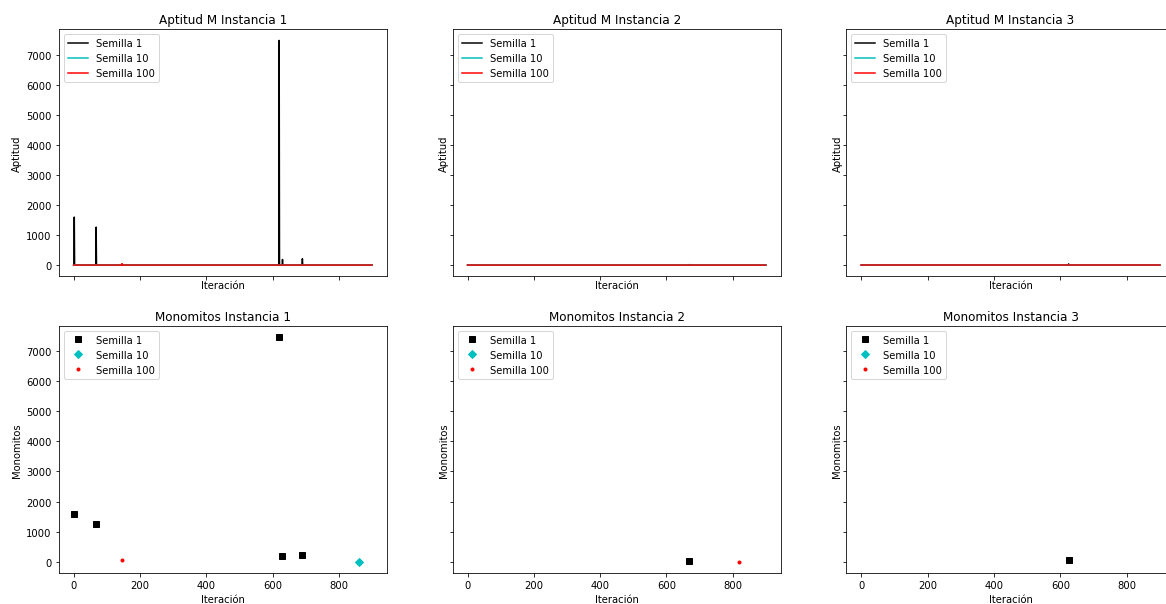


Figura 15: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 70 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **M** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

Máscara 50 %

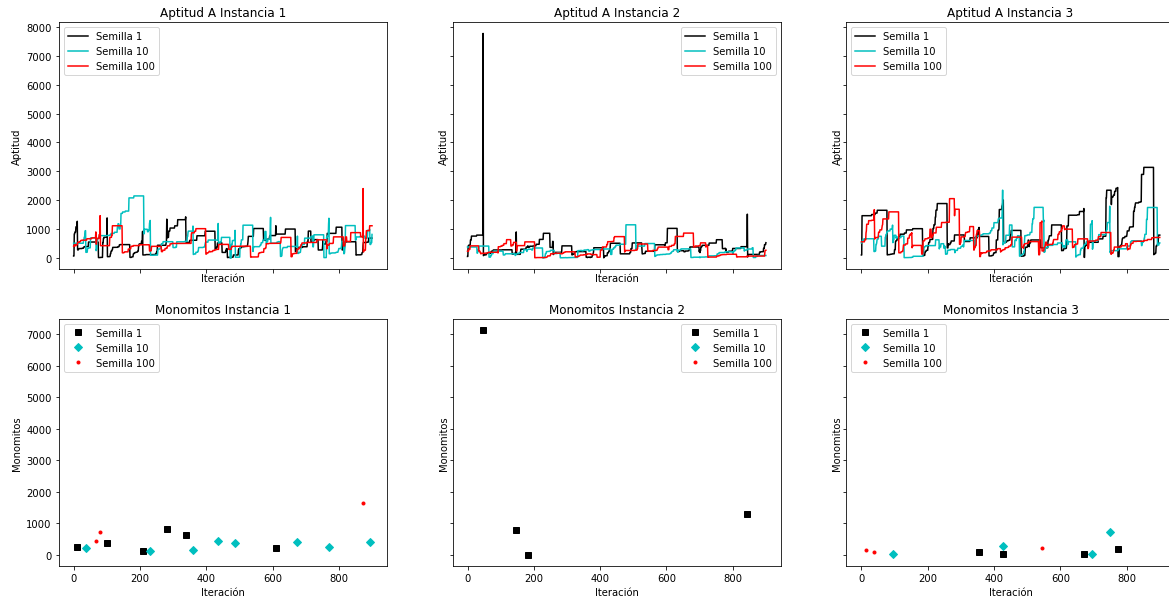


Figura 16: Gráficos de los resultados de la estrategia evolutiva usando una máscara del 50 % para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud A para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

En la figura 16 se muestran los gráficos de la función de aptitud A obtenidos para la estrategia evolutiva utilizando una máscara del 50 % para cada una de las 3 instancias y con las 3 semillas aleatorias. Al igual que se hizo anteriormente, en la primera fila se muestra la aptitud del padre versus las iteraciones, mientras que en la segunda se puede observar el monomito versus las iteraciones. Se puede apreciar que los resultados para la instancia 2 son similares a los obtenidos con la máscara del 70 % para esta función; sin embargo, las aptitudes y monomitos para las otras instancias son más bajas. Esto indica que la cantidad de monomitos encontrados y el valor de la función de aptitud tienen relación con la cantidad de genes que se utilizan de la solución inicial. La ventaja principal de utilizar una máscara más pequeña es la variabilidad de las soluciones, es decir, se obtienen soluciones distintas debido a que no contienen tantos genes pertenecientes a la inicial. También es posible notar que para la instancia 2 al igual que con otros experimentos anteriores se obtuvieron menos veces monomitos; sin embargo, la cantidad encontrada fue bastante buena comparado con otras ejecuciones.

Inicialización en Algoritmo Genético

Las siguientes pruebas tienen por objetivo analizar si la inicialización de la solución inicial en el algoritmo genético original resulta en una mejora de los resultados. Se hicieron dos pruebas: se insertó la solución base en la población inicial y se ejecutó el algoritmo genético original. En la segunda prueba, también se insertó el individuo base en la población inicial pero se ejecutó una versión elitista del algoritmo genético, es decir, el mejor individuo de la población actual pasa a la siguiente.

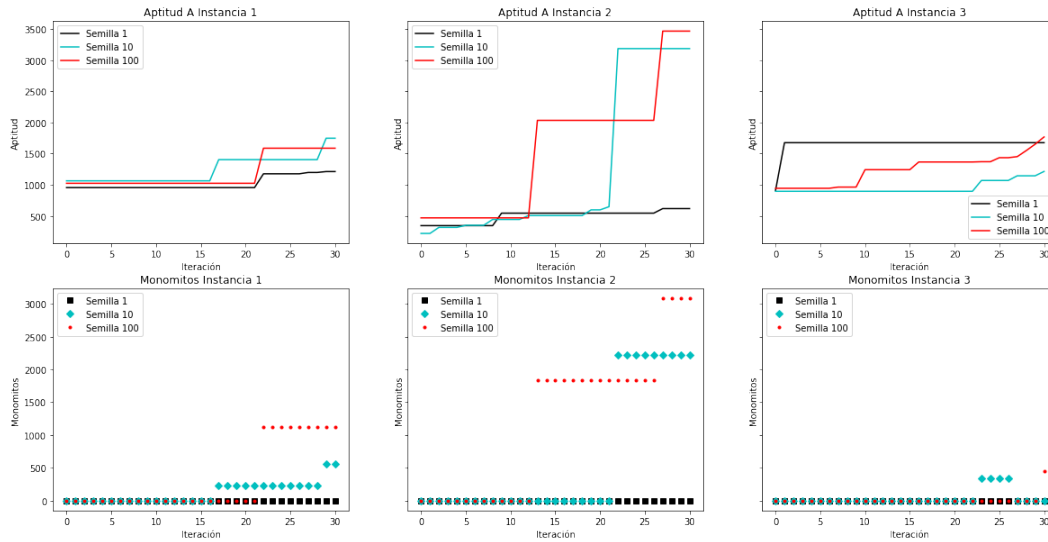


Figura 17: Gráficos de los resultados del algoritmo genético usando la mejor solución de todas en la población inicial. Esto se realiza para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **A** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

En la figura 17 se pueden ver los gráficos de la función de aptitud A obtenidos para el algoritmo genético original (del mejor individuo hasta ese momento), insertando la mejor solución de todas en la población inicial. Si bien esto no debería tener mayor repercusión en los resultados, se observa que para la instancia 2 la aptitud y la cantidad de monomitos mejoran considerablemente respecto del resultado original obtenido mostrado en la figura 7. Esta diferencia se justifica debido a que cuando se inserta el individuo a la población inicial, se está ayudando al algoritmo a acercarse a zonas con mejores soluciones y por esto le es más fácil encontrar monomitos y aptitudes más altas. En el caso de la instancia 1 y 3 también aparecen más monomitos (en comparación de los resultados originales), aunque se reafirma que no consigue mejorar lo obtenido por la estrategia evolutiva.

En la figura 18 se puede visualizar el comportamiento de la función de aptitud A del **mejor individuo de la población** en esa iteración para el algoritmo genético elitista, es decir, aquel

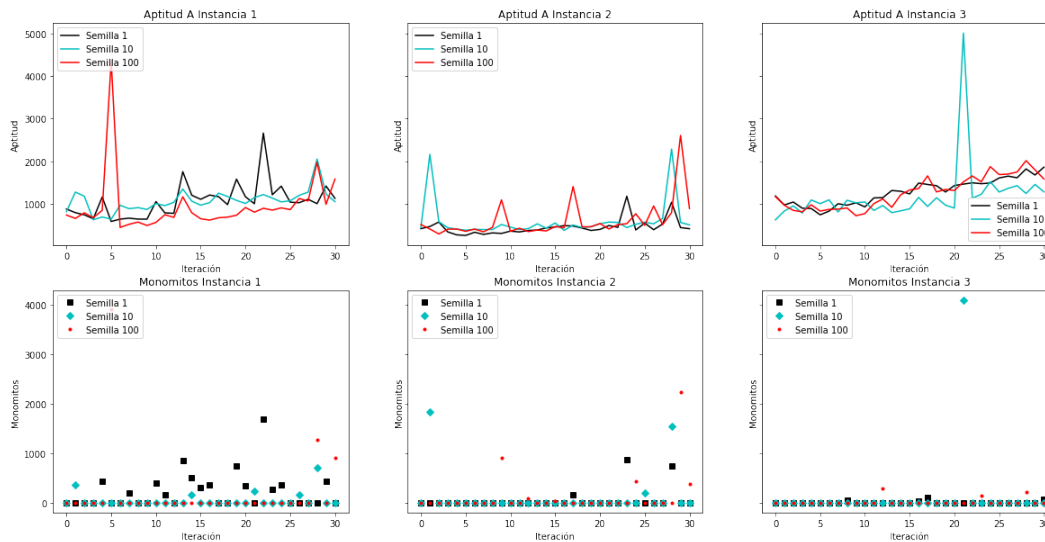


Figura 18: Gráficos de los resultados del algoritmo usando la mejor solución de todas en la población inicial y aplicando elitismo. Esto se realiza para las 3 instancias, con las 3 semillas aleatorias. En la primera fila se muestra el comportamiento de la Función de Aptitud **A** para distintas semillas e instancias. En la segunda se muestran los monomitos encontrados con dicha función, omitiendo los 0.

Fuente: Elaboración propia.

que conserva el mejor individuo de la población actual y lo inserta en la siguiente, reemplazando a un individuo escogido al azar. Es importante destacar que este gráfico es diferente de los demás, pues en la aptitud se considera al mejor individuo de la población y no al mejor de todos, como se hizo en los gráficos anteriores. En esta modificación también se agregó la mejor solución de todas las encontradas previamente en la población inicial. En los gráficos de la segunda fila se puede apreciar que aparecen más monomitos que en la versión original del algoritmo genético, sobre todo para la instancia 1 y 2. Sin embargo, se comprueba nuevamente que este enfoque no es suficiente para superar a la estrategia evolutiva planteada.

Resumen En esta sección se validó la propuesta de la solución mostrando los resultados de los experimentos realizados para ambos algoritmos y comparando la cantidad de monomitos que se encontraban en cada ocasión. Ninguna de las pruebas realizadas con el algoritmo genético (ya sea el original o modificado) lograron encontrar más monomitos que la estrategia evolutiva (aún cuando se utilizaron máscaras inferiores al 100 %).

CAPÍTULO 5

CONCLUSIONES

En este trabajo se presenta una propuesta de solución para la generación automática de historias secundarias para [NPCs](#) basada en estrategias evolutivas. El utilizar enfoque en vez de un algoritmo genético otorga la posibilidad de aprovechar al máximo cada evaluación de la función objetivo. Los algoritmos genéticos son eficaces a la hora de resolver problemas donde la función de aptitud toma un tiempo muy pequeño en evaluarse (del orden de segundos), permitiendo que se puedan realizar muchas iteraciones. A lo largo del escrito se pudo verificar que en casos donde la función de evaluación tarda mucho en ser calculada, estos algoritmos pierden efectividad, al verse forzados a realizar menos iteraciones de las óptimas.

El algoritmo propuesto demostró encontrar resultados de muy buena calidad para el problema estudiado debido a que como la estrategia evolutiva comienza desde un solo individuo para hacerlo evolucionar, se aprovecha mejor cada evaluación. Esto no ocurre en el genético, pues este evalúa por cada población, un conjunto de individuos distintos en cada iteración, de los cuales no todos sirven ni se aprovecha el potencial de las buenas soluciones (el algoritmo original no tenía elitismo).

Utilizar una buena solución inicial demostró ser una buena estrategia para guiar la inicialización del proceso de búsqueda. Partir de una buena solución ayuda a estar más cerca de buenas soluciones. Más aún, el utilizar una máscara de menos del 100 % se le agrega exploración al algoritmo, debido a que lo hace moverse más por el espacio de soluciones.

Algo a destacar sobre la estrategia evolutiva es la capacidad que tuvo para encontrar monomitos. A pesar de que la función de aptitud no tenía una directa proporción con la cantidad de monomitos que se encontraban, el algoritmo evolucionó en la dirección correcta, siendo capaz de sortear el obstáculo de tener un gran espacio de búsqueda. También fue capaz de encontrar soluciones de buena calidad y una de las razones es que estos algoritmos están diseñados para trabajar con valores reales como lo era en nuestro caso. La eficacia del algoritmo genético está probada principalmente para valores binarios y aunque existen adaptaciones para poder trabajar con valores reales (como la que se utilizó en [[García-Ortega et al., 2016](#)]), no se comparan con las estrategias evolutivas.

Es importante recordar que una de las mayores dificultades fue el ruido de la función de evaluación; esto se refiere a que dos evaluaciones de una misma solución pueden dar valores distintos. Esto logró ser mitigado en parte al aceptar peores soluciones, pues se consideraba la posibilidad de que quizás el hijo no era peor que el padre, sino que en esa evaluación no había obtenido una mejor aptitud (pero podría ser que en otra sí). Otra dificultad que se tuvo fue el tamaño del gen. Debido al gran tamaño de este, cualquier cambio en el individuo que se intentara debería ser solo a una porción de este y no al gen completo, pues un cambio en un gen podría afectar fuertemente a otro (epistasia). Por esta razón se utilizaron mutaciones

en base a una probabilidad, es decir, no se muta el gen completo sino solo una parte en cada iteración.

Finalmente pero no menos importante hay que destacar la importancia del trabajo en el contexto del problema. Se debe recordar que el objetivo del modelo de tres capas es generar historias para [NPCs](#) que fuesen congruentes. Para conseguir dicha coherencia, el modelo está basado en la teoría del monomito de Joseph Campbell, quien estableció los patrones comunes que tienen diversas historias en la literatura. El crecimiento de los videojuegos actuales fuerza a los creadores a invertir muchos recursos en la creación y perfección de las historias y el entorno del juego; automatizar estos procesos sería de gran ayuda en un futuro, no solo para ayudar a reducir costos, sino también para crear un mundo virtual más inmersivo, en donde el jugador se sorprenda con las interacciones de los [NPCs](#) y cómo estos crean su propio contexto.

Trabajo Futuro

Se podría intentar reducir el tiempo de ejecución, ya que las evaluaciones siguen tomando mucho tiempo lo que hace que cualquier técnica metaheurística que uno plantee sea difícil de probar. También se podría utilizar [Compute Unified Device Architecture \(CUDA\)](#) de manera de aprovechar la [GPU](#) del computador y hacer que las evaluaciones se ejecuten más rápido.

Algo que también ayudaría a reducir el tiempo de cómputo es mejorar la capa lógica. Debido a que el razonador debe lidiar con muchos eventos, calcular la función de evaluación resulta un proceso que toma un tiempo considerable y que va en aumento conforme crece la cantidad de eventos producidos por la simulación. Sería fundamental mejorar este aspecto ya sea cambiando el modelo o manejando de alguna forma la cantidad de eventos que se producen.

Otra tarea que queda pendiente es extender el modelo. La simplificación de los personajes a tres formas sencillas y el tamaño limitado del mundo hacen que aún no sea aplicable a un juego de gran escala. Además, habría que mejorar el árbol de comportamiento, ya que este es bastante básico también. En cuanto a la parte narrativa, es imperativo mejorar el motor narrativo del sistema, ya que al igual que el árbol de comportamiento, este no soporta historias complejas o su contexto.

Sería interesante construir un videojuego básico que ponga a prueba la solución encontrada en un ambiente real. No tiene porqué ser algo complejo en un principio, pero resulta fundamental encontrar una forma de conectar los parámetros del simulador con motores reales de juegos.

Referencias

- M. Arinbjarnar and D. Kudenko, "Schemas in directed emergent drama," in *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, ser. ICIDS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 180–185. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89454-4_24
- M. Arinbjarnar, H. Barber, and D. Kudenko, "A critical review of interactive drama systems," 2009.
- A. Auger, "Benchmarking the (1+1) evolution strategy with one-fifth success rule on the bbob-2009 function testbed," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 2447–2452. [Online]. Available: <http://doi.acm.org/10.1145/1570256.1570342>
- T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, NY, USA: Oxford University Press, Inc., 1996.
- T. Bäck, G. Rudolph, and H.-P. Schwefel, "Evolutionary programming and evolution strategies: Similarities and differences," in *In Proceedings of the Second Annual Conference on Evolutionary Programming*. Citeseer, 1993.
- O. Bangsø, O. G. Jensen, F. V. Jensen, P. B. Andersen, and T. Kocka, "Non-linear interactive storytelling using object-oriented bayesian networks," in *In Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education*, 2004.
- H. Barber and D. Kudenko, "Generation of adaptive dilemma-based interactive narratives," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 4, pp. 309–326, Dec 2009.
- J. Bates, "Virtual reality, art, and entertainment," *Presence Teleoperators and Virtual Environments*, vol. 1, 01 1992.
- J. Campbell, *The Hero with a Thousand Faces*, ser. Bollingen series. New World Library, 2008, vol. 17.
- C. Crawford, "Storytron: What went wrong," consultado: 2017-06-25. [Online]. Available: <http://www.storytron.com/WhatWentWrong.html>
- C. Crawford, *Chris Crawford on Interactive Storytelling*, 2nd ed. New Riders Games, 2013.
- K. Deb and M. Goyal, "A combined genetic adaptive search (geneas) for engineering design," *Computer Science and Informatics*, vol. 26, 08 1999.

- "The demoscene," Digitale Kultur e.V., consultado: 2017-06-18. [Online]. Available: <http://www.demoscene.info/the-demoscene/>
- L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms*, ser. Foundations of Genetic Algorithms, L. D. WHITLEY, Ed. Elsevier, 1993, vol. 2, pp. 187 – 202. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/B9780080948324500180>
- C. Fairclough, *Story Games and the OPIATE System: Using Case-based Planning for Structuring Plots with an Expert Story Director Agent and Enacting Them in a Socially Simulated Game World*. Trinity College, 2005.
- R. H. García-Ortega, P. García-Sánchez, and A. M. Mora, "My life as a sim: evolving unique and engaging life stories using virtual worlds," in *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, vol. 14, 2014, pp. 3–5.
- R. H. García-Ortega, P. García-Sánchez, J. Merelo, M. I. G. Arenas, P. A. Castillo, and A. M. Mora, "How the world was made: parametrization of evolved agent-based models for backstory generation," in *European Conference on the Applications of Evolutionary Computation*. Springer, Cham, 2015, pp. 443–454.
- R. H. García-Ortega, P. García-Sánchez, J. J. Merelo, A. San-Ginés, and Á. Fernández-Cabezas, *The Story of Their Lives: Massive Procedural Generation of Heroes' Journeys Using Evolved Agent-Based Models and Logical Reasoning*. Cham: Springer International Publishing, 2016, pp. 604–619. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-31204-0_39
- R. H. García-Ortega, P. García-Sánchez, and J. Merelo, "Emerging archetypes in massive artificial societies for literary purposes using genetic algorithms," *arXiv preprint arXiv:1403.3084*, 2014.
- B. Hayes-Roth, P. Doyle, and H. Maldonado, "The virtual theater project," consultado: 2017-06-25. [Online]. Available: <http://www.ksl.stanford.edu/projects/cait/>
- M. Hendriks, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 9, no. 1, pp. 1:1–1:22, feb 2013. [Online]. Available: <http://doi.acm.org/10.1145/2422956.2422957>
- J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.
- M. Lebowitz, *Program of the Ninth Annual Conference of the Cognitive Science Society: 16-18 July 1987, Seattle, Washington*, ser. Program of the annual conference- Cognitive Science Society. Lawrence Erlbaum Associates, 1987, ch. Planning stories.
- B. Magerko, "Story representation and interactive drama," in *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser.

- AIIDE'05. AAAI Press, 2010, pp. 87–92. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3022473.3022489>
- M. Mateas and A. Stern, “Façade: An experiment in building a fully-realized interactive drama,” 2003.
- N. McIntyre and M. Lapata, “Plot induction and evolutionary search for story generation,” in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ser. ACL '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1562–1572. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1858681.1858839>
- J. R. Meehan, “The metanovel: Writing stories by computer.” Ph.D. dissertation, Yale University, New Haven, CT, USA, 1976, aAI7713224.
- B. W. Mott and J. C. Lester, “U-director: A decision-theoretic narrative planning architecture for storytelling environments,” in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: ACM, 2006, pp. 977–984. [Online]. Available: <http://doi.acm.org/10.1145/1160633.1160808>
- E. T. Mueller and M. G. Dyer, “Towards a computational theory of human daydreaming,” in *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*. Cognitive Science Society, 1985.
- M. Nairat, P. Dahlstedt, and M. G. Nordahl, “Character evolution approach to generative storytelling,” in *2011 IEEE Congress of Evolutionary Computation (CEC)*, June 2011, pp. 1258–1263.
- M. J. Nelson, D. L. Roberts, C. L. Isbell, Jr., and M. Mateas, “Reinforcement learning for declarative optimization-based drama management,” in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '06. New York, NY, USA: ACM, 2006, pp. 775–782. [Online]. Available: <http://doi.acm.org/10.1145/1160633.1160769>
- T. Ong and J. J. Leggett, “A genetic algorithm approach to interactive narrative generation,” in *Proceedings of the Fifteenth ACM Conference on Hypertext and Hypermedia*, ser. HYPERTEXT '04. New York, NY, USA: ACM, 2004, pp. 181–182. [Online]. Available: <http://doi.acm.org/10.1145/1012807.1012856>
- K. Perlin, “An image synthesizer,” *SIGGRAPH Comput. Graph.*, vol. 19, no. 3, pp. 287–296, jul 1985. [Online]. Available: <http://doi.acm.org/10.1145/325165.325247>
- V. I. Propp, *Morphology of the Folktale: Second Edition*, ser. American Folklore Society Bibliographical and Special Series. University of Texas Press, 1968.
- M. O. Riedl and A. Stern, “Believable agents and intelligent story adaptation for interactive storytelling,” in *Proceedings of the Third International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, ser. TIDSE'06. Berlin,

- Heidelberg: Springer-Verlag, 2006, pp. 1-12. [Online]. Available: http://dx.doi.org/10.1007/11944577_1
- N. M. Sgouros, "Dynamic, user-centered resolution in interactive stories," in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 990-995. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1622270.1622299>
- H. Sosa, S. Myriam Villagra, and N. Andrea Villagra, "Operadores de mutación en algoritmos genéticos celulares aplicados a problemas continuos," *Informes Científicos - Técnicos UN-PA*, vol. 6, p. 141, 10 2014.
- M. Stede, "Scott r. turner, the creative process. a computer model of storytelling and creativity. hillsdale, nj: Lawrence erlbaum, 1994. isbn 0-8058-1576-7, £49.95. 298 pp." *Nat. Lang. Eng.*, vol. 2, no. 3, pp. 277-285, sep 1996. [Online]. Available: <http://dl.acm.org/citation.cfm?id=974680.974687>
- M. Theune, S. Faas, A. Nijholt, and D. Heylen, "The virtual storyteller," in SIGG, 2002.
- D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach," in *Proceedings of the Third AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'07. AAAI Press, 2007, pp. 43-48. [Online]. Available: <http://dl.acm.org/citation.cfm?id=3022508.3022517>
- C. Vogler, *The Writer's Journey: Mythic Structure for Writers*. M. Wiese Productions, 1998. [Online]. Available: <https://books.google.cl/books?id=Lgzk7EllmugC>
- R. M. Young and M. O. Riedl, "Integrating plan-based behavior generation with game environments," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ser. ACE '05. New York, NY, USA: ACM, 2005, pp. 370-370. [Online]. Available: <http://doi.acm.org/10.1145/1178477.1178553>