Inteligencia Artificial

Algoritmo de 2 pasos con Tabu Seach para el Problema de Generación de Mazmorras

Javier Hormaechea D'Arcangeli

24 de noviembre de 2024

Resumen

Este informe aborda el problema de la generación de mazmorras, un desafío común en el desarrollo de videojuegos "Roguelike", los que se caracterizan por ofrecer niveles o mazmorras generados de manera aleatoria. El documento presenta un análisis de distintos enfoques utilizados para la generación de mazmorras, destacando los algoritmos evolutivos y los enfoques de dos pasos. Se exploran las fortalezas y limitaciones de estos enfoques, subrayando la necesidad de generar estructuras más complejas y desafiantes que puedan ser aplicadas en casos reales. El documento plantea un algoritmo de dos pasos con Tabu Search para generar y optimizar mazmorras, separando la generacion y optimización de la estructura y la de las llaves y barreras. Resultados experimentales muestran que el algoritmo tiene un menor tiempo de ejecución al compararlo con otros metodos, pero para esto sacrifica calidad de la solución lo que sugiere su uso potencial en generar soluciones iniciales para modelos más refinados.

1. Introducción

La industria del videojuego a demostrado un crecimiento exponencial, con unos ingresos esperados de 187,7 mil millones de dolares en 2024 y proyectando unos ingresos de 213,3 mil millones para 2027 [6]. Dentro de la industria, se han destacado ultimamente los juegos del género "Roguelike" que se caracterizan por permitir al jugador explorar niveles o mazmorras generadas aleatoriamente permitiendo un alto grado de rejugabilidad. Esto a llevado al género y sus variantes a posicionarse en los puestos 6, 10 y 16 de géneros con más juegos vendidos en 2023 [10]. Uno de los desafios que presenta este genero es el de generar mazmorras con estructuras desafiantes para los jugadores, si a esto se le suma que el género es principalmente utilizado por desarrolladores independientes que no siempre tienen los recursos para desarrollar un sistema de generación de mazmorras especifico para su titulo, surge la necesidad de algoritmos que permitan generar mazmorras aleatorias dados ciertos parametros iniciales entregados por el desarrollador.

El presente documento busca sentar las bases necesarias para la implementacion de una solución al problema de generación de mazmorras. Para esto, se definen las variables y restricciones del problema, luego se presentan las soluciones existentes a la fecha para finalmente plantear un modelo matemático cuyas soluciones correspondan a las soluciones del problema.

2. Definición del Problema

El problema de generación de mazmorras consiste en generar la estructura de una mazmorra asi como los obstaculos que se interpondran en el avance del jugador, para simplificar el problema se considerara la estructura de la mazmorra como un arbol donde el nodo raiz corresponde a

la habitación inicial, esto, a su vez, implica que no existen ciclos, se consideran los siguientes parametros:

- Número de habitaciones(rooms): corresponden a los nodos del arbol, puden contener llaves y pueden tener entre 0 y 3 hijos, que son las habitaciones a las que estan conectadas.
- Número de barreras(barriers): se ubican en conecciones entre 2 habitaciones y bloquea dicha conección hasta que se obtiene la llave que la libera
- Número de llaves(keys): se ubican en habitaciones, cada llave tiene una barrera correspondiente que abre.
- Llaves necesarias(needed_keys): es el número mínimo de llaves que se deben recojer para completar una mazmorra en particular.
- Coeficiente lineal(lcoef): es el promedio de los coeficientes lineales de las habitaciones, los que corresponden al número de hijos de la habitación omitiendo a las que no tienen hijos.

El objetivo de este problema entonces es: dados unos valores de los parametros anteriores entregados por un desarrollador o usuario generar, en el menor tiempo posible, una mazmorra donde los valores reales de dichos parametros sean los más cercanos posibles a los entregados. Si bien se trabajara esta simplificación basada en el trabajo de Felipe Dumont [3], existen trabajos más complejos [5] donde se consideran mazmorras continuas con más elementos.

3. Estado del Arte

El problema de generación de mazmorras tiene sus origenes con "Rogue" [8] el videojuego que da nombre al genero "Roguelike", el juego tiene un sistema de generación simple que crea entre 6 y 10 habitaciones al azar con tamaños aleatorios de entre 3x3 y 10x10, luego de esto determina los corredores entre ellas representando la conectividad en un grafo donde, nuevamente, las habitaciones se conectan a sus vecinas de manera aleatoria [9]. Pese a la simpleza de estos origenes los trabajos contemporaneos ulizan multiples tecnicas, destacando el uso de algoritmos evolutivos como el caso de Cerny [2] que lo utiliza en conjunto con un algoritmo greedy o Ashlock [1] que los emplea para generar estructuras tipo cuevas, otra estrategia a mencionar son los algoritmos en 2 pasos que separa la generación en 2 para poder optimizar cada parte por separado, destacando el trabajo de Gellet [4] donde primero genera una descripción del nivel en una gramatica libre de contexto para luego generar el espacio físico y el de Dumont [3] que primero genera la estructura de la mazmorra para luego colocar las llaves y las barreras. Caben destacar los trabajos de Pereira [7] y Dumont [3] debido a que estos trabajan con un modelo similar al utilizado en este documento.

4. Modelo Matemático

El modelo matemático a utilizar busca minimizar la siguiente función [3] [7]:

$$f = 2(\Delta_{rooms} + \Delta_{keys} + \Delta_{barriers} + \frac{rooms}{10} * \Delta_{lcoef}) + \Delta_{needed\_keys}$$

- Δ: diferencia entre el valor entregado por el usuario y valor real de la estructura generada.

Para el planteamiento de las restricciones y el espacio de busqueda se definen las siguientes variables binarias:

* *Como se menciona en la descripción del problema, la estructura de la mazmorra se interpreta como un arbol siendo la habitación inicial la raiz.*

- x_{ij}: es 1 si existe la habitación i y es hija de la habitación j, $i \in \{1, 2, \ldots, N\}$; $j \in \{0, 1, \ldots, N\}$ (j=0 significa que es la raiz, N es el doble de las habitaciones deseadas)
- b_{ijl}: es 1 si existe una barrera i entre las habitaciones j y l, $i \in \{1, 2, B\}$; $j, l \in \{1, 2, \ldots, N\}$ (i se utiliza para emparejar con las llaves, B es el doble de las barreras deseadas)
- k_{ij}: es 1 si existe una llave i en la habitacion j, $i \in \{1, 2, K\}$; $j \in \{1, 2, \ldots, N\}$ (i se utiliza para emparejar con las barreras, K es el doble de las llaves deseadas)

  Este modelo esta sujeto a las siguientes restricciones:

  - $\sum_{i=1}^{N} x_{ij} \leq 3$, $\forall j \in \{0, 1, \ldots, N\}$, una habitación no puede tener más de 3 hijos.
  - $\sum_{j=1}^{N} k_{ij} = \sum_{j=1}^{N} \sum_{l=1}^{N} b_{ijl} \leq 1$, $\forall i \in \{1, 2, \ldots, K\}$ las barreras y llaves solo pueden estar en 1 lugar, cada barrera tiene solo una llave y viceversa (es menor o igual porque la barrera podria no existir).
  - $\sum_{j=0}^{N} x_{ij} \leq 1$, $\forall i \in \{1, 2, \ldots, N\}$ una habitación solo tiene 1 padre (es menor o igual porque la habitación podria no existir).
  - $\sum_{i=1}^{N} x_{i0} = 1$, existe solo una habitación inicial.
  - $b_{ijl} \leq x_{jl} + x_{lj}$, $\forall i \in \{1, 2, \ldots, B\}$, $\forall j, l \in \{1, 2, \ldots, N\}$ una barrera solo puede estar en una conexión que existe.
  - $\sum_{i=1}^{N} k_{ij} \leq 100000 x_{jl}$, $\forall j, k \in \{1, 2, \ldots, N\}$ solo pueden haber llaves en habitaciones que existen.

  Para determinar el espacio de busqueda tienen: $N(N+1)$ variables x, BN^2 variables b y KN variables k. Por lo tanto, se tienen $2^{(B+1)N^2+(K+1)N}$ soluciones posibles.

5. Representación

Para la representación del problema se decidio trabajar la mazmorra como un arbol donde cada habitación corresponde a un nodo que puede tener 1 llave y/o 1 barrera. Dentro de la clase mazmorra se almacena, a su vez, 2 listas que corresponden a los nodos que tienen llaves y la los que contienen barrera, luego, la llave que le corresponde a cada barrera es la que tiene su misma posicion en la otra lista.

```
struct Nodo {
public:
    int id;
    int profundidad;
    vector<Nodo*> hijos;
    int id_barrera;
    int id_llave;
}

class Dungeon {
    public:
    Nodo* raiz;
    Nodo* fin;
    vector<Nodo*> recorrido; // por conveniencia es de atras hacia adelante
    int num_habitaciones;
    int num_llaves;
    int llaves_actual;
    int num_barreras;
```

```
    int barreras_actual;
    int llaves_necesarias;
    int llaves_necesarias_actual;
    int habitaciones_generadas;
    double coef_lineal;
    double coef_lineal_actual;
    int nodos_con_hijos;
    vector<Nodo*> nodos_barreras;
    vector<Nodo*> nodos_llaves;
}
```

6. Descripción del algoritmo

El algoritmo que se trabajo consiste en un *2-Step Tabu Search*, este comienza colocando las habitaciones por medio de un greedy, donde para cada nodo, se colocan la cantidad de hijos que le permiten acercarce más al coef_lineal, luego de esto, las habitaciones se optimizan con tabu search donde el movimiento consiste en agregar una hoja o quitar una hoja al arbol. Despues de esto, se colocan las llaves y barreras con la restriccion de que una llave no puede estar más profunda que su barrera. Finalmente se optimizan las llaves y barreras de la mazmorra por medio de otro tabu search, esta vez siendo el movimiento el mover una barrera a otro nodo permitido.

7. Experimentos

Los experimentos realizados consistieron en ejecutar el algoritmo 100 veces para 8 instancias distintas en una maquina virtual corriendo Ubuntu 22.04.2 LTS con 8 GB de ram DDR4 a 3200 MHZ y un procesador AMD Ryzen 5 5600X, se compararan los resultados con los del algoritmo presentado en [3] debido a que ambos algoritmos generan y optimizan la estructura de la mazmorra para luego colocar y optimizar las llaves.

8. Resultados

Los resultados presentados a continuacion corresponden al promedio de las 100 ejecuciones de cada instancia.

	Habitaciones		Llaves		Barreras	
Instancia	2-Step TS	2-Step EA	2-Step TS	2-Step EA	2-Step TS	2-Step EA
(15, 3, 2, 2.0, 2)	14.65	**15.0**	1.89	**3.0**	**2.0**	**2.0**
(20, 4, 4, 1.0, 4)	19.8	**20.0**	3.95	**4.0**	3.96	**4.0**
(20, 4, 4, 2.0, 4)	20.18	**20.0**	3.64	**4.0**	3.92	**4.0**
(25, 8, 8, 1.0, 8)	27.0	**25.0**	**8.0**	**8.0**	8.3	**8.0**
(30, 4, 4, 2.0, 4)	31.10	**30.0**	3.73	**4.0**	4.23	**4.0**
(30, 6, 6, 1.5, 6)	29.64	**30.0**	5.63	**6.0**	**6.0**	**6.0**
(100, 20, 20, 1.5, 20)	92.96	**99.93**	10.52	**20.0**	18.44	**20.0**
(500 100 100 1.5 100)	495.0	**498.88**	50.84	**100.0**	99.33	**99.99**

	Coeficient lineal		Funcion objetivo		Tiempo(s)	
Instancia	2-Step TS	2-Step EA	2-Step TS	2-Step EA	2-Step TS	2-Step EA
(15, 3, 2, 2.0, 2)	1.85	**1.99**	5	**0.02**	**0.06**	3.37
(20, 4, 4, 1.0, 4)	**0.99**	**1.01**	0.36	**0.02**	**0.09**	4.41
(20, 4, 4, 2.0, 4)	1.59	**1.89**	5.57	**0.21**	**0.07**	3.46
(25, 8, 8, 1.0, 8)	**1.0**	1.04	4.16	**0.31**	**0.18**	4.01
(30, 4, 4, 2.0, 4)	1.63	**1.91**	7.44	**0.18**	**0.08**	3.71
(30, 6, 6, 1.5, 6)	1.34	**1.52**	6.05	**0.05**	**0.1**	3.9
(100, 20, 20, 1.5, 20)	1.2	**1.48**	45.86	**0.34**	**0.24**	5.85
(500 100 100 1.5 100)	1.24	**1.44**	240.47	**4.6**	**4.13**	17.34

Se puede ver que 2-Step TS es bastante peor que [3] en lo que respecta a la calidad de la solución, sin embargo, con lo que respecta al tiempo de ejecución se puede ver que e 2-Step TS es entre 40 y 50 veces más rápido para instancias pequeñas, disminullendo la proporción a medida que aumenta el tamaño de la instancia.

9. Conclusiones

En el presente informe se planteo un algoritmo de 2 pasos con tabu search para generar mazmorras dividiendo el proceso en primero, generar la estructura correspondiente y optimizarla, seguido de la colocación de llaves y barreras para su posterior optimización, permitiendo que se optimizen los distintos parametros por separado. Una novedad dentro de este proceso es la introducción de una funcioń objetivo en la que el peso del coeficiente lineal se ve afectado seguń el número de habitaciones.

Los resultados obtenidos permiten ver que la calidad de la mazmorra generada no es muy buena, sin embargo, su tiempo de ejecución sugiere que podria ser una buena herramienta para generar soluciones iniciales para otros algoritmos como [3].

10. Bibliografía

Referencias

[1] Daniel Ashlock. Evolvable fashion-based cellular automata for generating cavern systems, 2015. IEEE Conference on Computational Intelligence and Games (CIG). doi:10.1109/cig.2015.7317958.

[2] Vojtech Cerny and Filip Dechterenko. Rogue-like games as a playground for artificial intelligence - evolutionary approach, 2015. Lecture Notes in Computer Science, 261-271. doi:10.1007/978-3-319-24589-8_20.

[3] Felipe Dumont and María Cristina Riff. 2-step evolutionary algorithm for the generation of dungeons with lock door missions using horizontal symmetry, 2024. In . ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654153.

[4] Alexander Gellel and Penny Sweetser. A hybrid approach to procedural generation of roguelike video game levels, 2020. In International Conference on the Foundations of Digital Games (FDG '20), September 15-18, 2020, Bugibba, Malta. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3402942.3402945.

[5] Antonios Liapis. Multi-segment evolution of dungeon game levels, 2017. In Proceedings of GECCO '17, Berlin, Germany, July 15-19, 2017, 8 pages. DOI: http://dx.doi.org/10.1145/3071178.3071180.

[6] Newzoo. Newzoo's global games market report 2024, Agosto 2024. https://newzoo.com/resources/trend-reports/newzoos-global-games-market-report-2024-free-version. [Último acceso 12 de Octubre de 2024].

[7] Leonardo Tortoro Pereira, Paulo Victor de Souza Prado, Rafael Miranda Lopes, , and Claudio Fabiano Motta Toledo. Procedural generation of dungeons' maps and locked-door missions through an evolutionary algorithm validated with players, 2021. Expert Systems with Applications 180 (2021), 115009.

[8] Michael Toy, Glenn Wichman, and Ken Arnold. Rogue, 1980. Game [Atari]. Apyx.

[9] Michael Toy, Glenn Wichman, and Ken Arnold. Rogue: Código fuente, 1980. https://github.com/Davidslv/rogue. [Último acceso 13 de Octubre de 2024].

[10] Zukalous. What are the top selling indie games of 2023?, Enero 2024. https://howtomarketagame.com/2024/01/25/what-are-the-top-selling-indie-games-of-2023/. [Último acceso 13 de Octubre de 2024].