UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Tetris Solver
Jugando al Tetris con técnicas metaheurísticas

TRABAJO TÉCNICAS METAHEURÍSTICAS

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Miquel Gómez

Resumen

```python
def generate_prompt(df_train: pd.DataFrame, df_batch: pd.DataFrame, genres: str)
    ↪ -> str:
    prompt = f"""
    Classify the following movie in any of these genres. More than one genre can
    ↪ be assigned.
    Genres: {genres}

    ============================================
    """

    for _, row in df_train.iterrows():
        prompt += f"Movie title: {row['movie_name']}\n"
        prompt += f"Plot: {row['description']}\n"
        prompt += f"Actual genres: {row['genre']}\n"
        prompt += "-------------------\n"

    prompt += """
    ============================================
    Now, classify the following movies returning a structured JSON
    response with the movie names and their genres.
    """

    for _, row in df_batch.iterrows():
        prompt += f"Movie title: {row['movie_name']}\n"
        prompt += f"Plot: {row['description']}\n\n"

    return prompt
```

Palabras clave: Clasificación de textos; Películas; Transformers; LLMs; Machine Learning

Índice general

Introducción

0.1 Problema a resolver

CAPÍTULO 1
Codificación

En este apartado se habla de como se han codificado los individuos para abordar el problema. Recordar que el objetivo es encontrar, para un set de piezas concreto, la posición y orientación óptima de cada pieza para minimizar el espacio ocupado. A eso hay que añadirle dos reglas del juego: al completar una línea esta se limpia, y las piezas no pueden estar flotando al colocarse.

Con esto, se parte de la idea de 'jugar' al Tetris. La propuesta tras esta premisa no es jugar al juego de verdad, sino conseguir una codificación que permita cumplir las restricciones impuestas, y conseguir soluciones que minimicen el espacio ocupado sin necesidad de descartar individuos inválidos.

La forma de atacar este problema de cobertura, será definiendo una serie de movimiento posibles. Con ellos, se codifica la posición final de cada pieza como una secuencia de estos. Así pues, cada genotipo será una secuencia de movimientos tras otros, de forma que si hay x moviemientos válidos y se disponone de n piezas, la codificación del genotipo será una secuencia $x \times n$ movimientos.

Al codificar los genotipos de esta forma, los individuos resultantes serán siempre válidos, ya que cada pieza se colocará en el tablero siguiendo las reglas del juego. Esto implica también que deberemos ser capaces de simular el juego para poder evaluar cada individuo, ya que la fenotipo de cada individuo será el estado final del tablero tras colocar todas las piezas siguiendo los movimientos indicados en el genotipo.

También, habrá movimiento que resulten en '*no-op*', como intentar mover una pieza a la izquierda cuando ya está en el borde izquierdo del tablero. Más abajo vemos como se gestionan estos casos.

1.1 Movimientos posibles

Cuando decimos codificar los individuos como una secuencia de movimientos, es necesario definir qué podrán hacer estos. Si nos fijamos en el juego original, los movimientos posibles son:

- Mover la pieza a la izquierda.
- Mover la pieza a la derecha.
- Rotar la pieza en el sentido de las agujas del reloj.
- Dejar caer la pieza.
- Bloquear la pieza.

- Intercambiar la pieza actual por la siguiente (o por una ya cambiada anteriormente)

En el juego original cuando se dejaba caer una pieza y esta toca el suelo, se bloqueaba al instante. Sin embargo, en versiones más modernas, esto no es así y aún habiendo tocado el suelo se permite mover la pieza con unas ciertas reglas. En este casi estaríamos hablamos de un Tetris Tetris 99 [**<empty citation>**], donde se permite que las piezas se muevan con más libertad y por tanto, se da más capacidad de representación al jugador. Es por todo esto que elegimos usar esta versión del juego y no la clásica con el objetivo de dotar a los individuos de más capacidad de representación lo que potencialmente, debería permitirnos llegar a mejores soluciones.

Ahora, si se tiene algo de experiencia en el juego, se puede ver que no en todas las situaciones todos los movimientos son necesarios para llegar a una solución concrata. Sin ir más lejos, en casos donde el tablero está casi vacío y no hay piezas creando agujeros, siempre se podrá poner una pieza en cualquiera de las posiciones validas con una *rotación* y un *movimiento lateral*.

Como tampoco queremos eliminar de la experimentación la posibilidad de ver los efectos que tiene el sí hacer más movimientos una vez las piezas han tocado el suelo, se han configurado cuatro sets posibles de movimientos:

- Simple.
    1. Mover la pieza.
    2. Rotar la pieza.
- Double.
    1. Mover la pieza.
    2. Rotar la pieza.
    3. *Dejar caer la pieza*
    4. Mover la pieza.
    5. Rotar la pieza.
- SwapSimple.
    1. Intercambiar la pieza actual.
    2. Mover la pieza.
    3. Rotar la pieza.
- SwapDouble.
    1. Intercambiar la pieza actual.
    2. Mover la pieza.
    3. Rotar la pieza.
    4. *Dejar caer la pieza*
    5. Mover la pieza.
    6. Rotar la pieza.

Donde el paso de dejar caer la pieza NO es configurable por el individuo (es fijo) y el resto de movimientos serán un entero que indicará cuántas veces se realiza ese movimiento. Damos ejemplos de cada uno de los movimientos:

- Intercambiar la pieza actual: $\{0,1\}$ 0 si no se quiere intercambiar, 1 si se quiere intercambiar.
- Mover la pieza: $\{-5,5\}$ un entero positivo o negativo. Si es positivo, se moverá esa cantidad de veces a la derecha, si es negativo, se moverá esa cantidad de veces a la izquierda (o hasta que no se pueda mover más).
- Rotar la pieza: $\{0,1,2,3\}$ un entero entre 0 y 3, indicando cuántas veces se rotará la pieza en el sentido de las agujas del reloj.
- *Dejar caer la pieza*: no es configurable, la pieza caerá hasta que toque el suelo o otra pieza.
- Mover la pieza: $\{-9,...,9\}$, Igual que el anterior.
- Rotar la pieza: $\{0,1,2,3\}$, Igual que el anterior.

Como optimización en este punto se ha propuesto lo siguiente: dado que las piezas aparecen en cierta posición concreta, el rango de movimientos laterales que se hace al principio se ha limitado a un rango de -5 a 5. Esto es, si una pieza aparece en la columna 5 del tablero, no tendría sentido moverla más de 5 veces a izquierda o derecha (ya que se saldría del tablero).

1.2 Tipo de individuos

1.3 Función objetivo

CAPÍTULO 2

Tecnología de Implementación

CAPÍTULO 3
Implementación

3.1 Algoritmo genético

3.2 Enfriamiento simulado

CAPÍTULO 4
Resultado

4.1 Algoritmo genético

4.2 Enfriamiento simulado

4.3 Evolución

CAPÍTULO 5

Conclusiones