Stardew Valley HDDL

1st Gabriel Mariano da Silva
Faculdade do Gama
Universidade de Brasília
Brasília, Brasil
200018167

2nd Diógenes Dantas Lélis Júnior
Faculdade do Gama
Universidade de Brasília
Brasilia, Brasil
190105267

3rd Danilo César Tertuliano Melo
Faculdade do Gama
Universidade de Brasília
Brasília, Brasil
221031149

4th Rodrigo Edmar Wright Dos Santos
Faculdade do Gama
Universidade de Brasília
Brasília, Brasil
200027158

I. Introdução

Stardew Valley é um jogo indie de RPG desenvolvido por Eric Barone (mais conhecido pelo seu nickname ConcernedApe). No jogo, há a simulação de um ambiente rural a ser explorado e desenvolvido pelo jogador. O objetivo do jogo é restaurar a fazenda e torná-la um negócio próspero.No jogo é possível cultivar plantações, criar animais, explorar cavernas e fazer amizade com os moradores da cidade.

Para o desenvolvimento do domínio para o presente trabalho, buscaremos simular o processo necessário para a coleta de itens, a consequente construção de estruturas e, enfim, a criação de animais no jogo.

O grupo está interessado em mostrar os passos essenciais para a construção das estruturas necessárias para abrigar animais presentes no jogo, como galinhas, patos, galinhas vazias, dinossauros, galinhas douradas, coelhos, vacas, cabras, porcos, avestruzes e ovelhas.

No domínio definido está incluso tipos, predicados e tarefas relacionadas à criação e obtenção de vários animais no jogo, como galinhas, patos, galinhas nulas, dinossauros, galinhas douradas, coelhos, vacas, cabras, porcos, avestruzes e ovelhas. Ele também inclui tarefas abstratas para obter recursos como madeira, pedra e ouro, bem como construir e atualizar estruturas como galinheiros e celeiros.

O grupo definiu os arquivos de problema como um conjunto de tarefas que representam os animais que a fazenda deve conter. Então, a partir desse problema, será gerado um plano com os passos para a construção das respectivas estruturas que os abrigarão os animais.

II. Design do Cenário

Para o cenário do domínio, o grupo propôs em apresentar um modelo hierárquico focado em mostrar os passos para construir estruturas essenciais para abrigar os animais.

Os passos para construir as estruturas são divididos em subtarefas, criando um contexto hierárquico para o problema, o qual se inicia em estruturas de baixo nível e finaliza com um conjunto de tarefas que geram uma estrutura de alto nível

Em contexto geral, é possível representar o modelo hierárquico a partir de uma estrutura simplista, na qual o planejamento se concentra nas ações de construção das estruturas para abrigar os animais

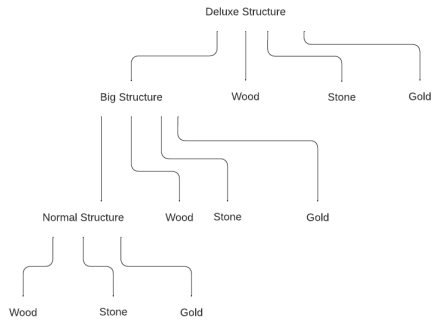

Fig. 1. Modelos com Níveis Hierárquicos

O código abaixo consiste em um trecho do nosso domínio que trata as construções das estruturas e a sua respectiva evolução

```
CONSTRUIR ESTRUTURAS
 ; ========================

 (:method buildcoopabstract-1
  :parameters (?s - structure ?i - inventory)
  :task (build-coop-abstract ?s ?i)
  :precondition (and
    (not (is-barn ?s))
    (is-nothing ?s)
  )
  :subtasks (and
   (build-coop ?s ?i)
  )
 )

 (:action build-coop
  :parameters (?s - structure ?i - inventory)
  :precondition (and
```

```
   (not (is-barn ?s))
  )
  :effect (and
   (not (is-nothing ?s))
   (not (has-wood ?i))
   (not (has-stone ?i))
   (not (has-gold ?i))
   (is-normal ?s)
   (is-coop ?s)
  )
 )

 (:method buildbarnabstract-1
  :parameters (?s - structure ?i - inventory)
  :task (build-barn-abstract ?s ?i)
  :precondition (and
    (not (is-coop ?s))
    (is-nothing ?s)
  )
  :subtasks (and
   (build-barn ?s ?i)
  )
 )

 (:action build-barn
  :parameters (?s - structure ?i - inventory)
  :precondition (and
   (not (is-coop ?s))
  )
  :effect (and
   (not (is-nothing ?s))
   (not (has-wood ?i))
   (not (has-stone ?i))
   (not (has-gold ?i))
   (is-normal ?s)
   (is-barn ?s)
  )
 )

; ========================
; EVOLUIR ESTRUTURAS
; ========================

 (:method upgradeabstract-1
  :parameters (?s - structure ?i - inventory)
  :task (upgrade-abstract ?s ?i)
  :precondition (and
    (is-normal ?s)
  )
  :subtasks (and
   (upgrade-structure-big ?s ?i)
  )
 )

 (:method upgradeabstract-2
  :parameters (?s - structure ?i - inventory)
  :task (upgrade-abstract ?s ?i)
  :precondition (and
    (is-big ?s)
  )
  :subtasks (and
   (upgrade-structure-deluxe ?s ?i)
  )
 )

 (:action upgrade-structure-big
  :parameters (?s - structure ?i - inventory)
  :precondition (and
   (is-normal ?s)
  )
  :effect (and
   (not (has-wood ?i))
   (not (has-stone ?i))
   (not (has-gold ?i))
   (not (is-normal ?s))
   (is-big ?s)
  )
 )

 (:action upgrade-structure-deluxe
  :parameters (?s - structure ?i - inventory)
  :precondition (and
   (is-big ?s)
  )
  :effect (and
   (not (has-wood ?i))
   (not (has-stone ?i))
   (not (has-gold ?i))
   (not (is-big ?s))
   (is-deluxe ?s)
  )
 )
```

Abaixo, um modelo que exemplifica tanto o contexto das estruturas construídas quanto os animais que elas podem abrigar.

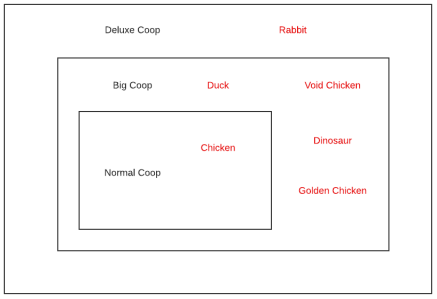

Fig. 2. Modelo estrutura e animais

III. Modelos de Planejamento Hierárquico

Um planejador hierárquico é um tipo de sistema de planejamento em inteligência artificial que organiza as ações em níveis hierárquicos para facilitar o planejamento em problemas complexos. Ele divide o problema em subtarefas mais gerenciáveis, tornando o processo de planejamento mais eficiente e escalável.

A hierarquia geralmente consiste em níveis abstratos (ou superiores) e níveis concretos (ou inferiores). No nível superior, estão as decisões estratégicas de alto nível, enquanto os níveis mais baixos detalham a execução prática e específica. Essa abordagem permite a reutilização de planos para tarefas similares e simplifica a representação e resolução de problemas complexos.

Para gerar os planos, o grupo decidiu utilizar dois planejadores HTN: O **HYPERTENSION** e o **PANDA**. Os dois planejadores foram competidores no **International Planning**

Competition (IPC), o PANDA no IPC 2023 e o Hypertension no IPC 2020.

IV. Experimentos

No código a seguir temos o problema gerado para se obter uma galinha:

```
(define (problem problem-gen)
    (:domain stardewvalley)
    (:objects
        a - animalcatalog
        coop barn - structure
        i - inventory
    )

    (:htn :parameters ()
        :ordered-subtasks (and
   (create-chicken a coop i)

        )
    )
    (:init
   (is-nothing coop)
   (is-nothing barn)
    )
)
```

Nesse código, temos o predicado "a" que representa o catálogo de animais que já foram obtidos pelo planejador, e o predicado "i" que são os itens que estão disponíveis no inventário. Esses itens podem ser madeira (wood), pedra (stone) e ouro (gold), que serão necessários para construir as estruturas. Neste domínio, não foram utilizados valores numéricos pois, após pesquisa, foi encontrado apenas um planejador que lida com numéricos cuja linguagem de trabalho não é HDDL, sendo ele o HyperTensionU. Por fim temos as structure coop (galinheiro) e barn (celeiro), que determinam as estruturas que podem ser criadas na fazenda.

Dito isso a task que deve ser resolvida nesse problema é a "create-chicken a coop i" que diz para o planejador que ele deve obter uma galinha no "a" (animal catalog) utilizando o inventário "i". As variáveis "coop" e "barn" foram inicializadas com "is-nothing" que representam que o viveiro (coop) e celeiro (barn) ainda não foram construídos. Ao executar esse programa com o HyperTension foram obtidos os seguintes resultados:

```
 ----------------------Plan--------------------
0: get_wood(i)
1: get_stone(i)
2: get_gold(i)
3: build_coop(coop i)
4: get_chicken(a)
```

Nele podemos ver a sequência de ações que o planejador encontrou para solucionar o problema. Nesse caso ele decidiu pegar os recursos necessários para construir um galinheiro (coop) normal, e posteriormente pegar a galinha. Considerando que o domínio é hierárquico então essa é a única solução possível, e é a solução ótima.

V. Discussões

A. Resultados

Utilizando o HyperTension executamos os problemas disponíveis obtendo os seguintes resultados:

TABLE I
Tabela benchmark

problem	time	steps
p001	0.0003287	4
p002	0.0004730	24
p003	0.0001213	33
p004	0.0000813	13
p005	0.0005028	25

Podemos reparar que os tempos da tabela mesmo para as soluções mais extensas são pequenos. Isso ocorre porque o domínio é relativamente simples, considerando que a real dificuldade do domínio é construir as estruturas (coop e barn), e depois de construídas o planejador só precisa pegar os animais necessários para concluir a task.

Sendo assim, a complexidade do domínio não está em adquirir os animais, e sim na ordem em que eles são adquiridos. Isso porque se for necessário pegar um animal que necessita de uma construção de luxo (deluxe) antes de pegar um animal que necessita de uma construção normal, o planejador poderá construir a construção de maior nível restando apenas pegar os animais necessários. Dessa forma resultando em um tempo menor mesmo com muitos animais, como no exemplo do problema p003.

B. Dificuldades

As maiores dificuldades enfrentadas pela equipe foi lidar com os numéricos. Isso porque atualmente não há muitos planejadores que lidam com numéricos em HDDL. Dessa forma foi necessário simplificar o domínio para algo que não necessitasse de numéricos para obter o mínimo de 2 (dois) planejadores que pudessem lidar com numéricos.

Além disso, o domínio gerado foi muito extenso, mesmo contendo uma baixa complexidade. Isso ocorreu devido a cada tipo de animal gerado, uma task específica para aquele animal deveria ser gerada também. Consequentemente gerando uma sequência de métodos muito semelhantes entre as tasks. Sendo esse o maior ponto de melhoria que pode ser feito, simplificar todas as tasks e métodos.

References

[1] https://github.com/ipc2023-htn/PandaDealer. Acesso em: dezembro de 2023
[2] https://github.com/Maumagnaguagno/HyperTensioN. Acesso em: dezembro de 2023
[3] https://pt.stardewcommunitywiki.com/Stardew_Valley_Wiki. Acesso em: dezembro de 2023
[4] https://ipc2020.hierarchical-task.net/results/results⟩. Acesso em: novembro de 2023
[5] https://ipc2023-htn.github.io/results. Acesso em: novembro de 2023