Agent Development (P01)

Artificial Intelligence, 2022-23

Pedro Simões (21140), Gonçalo Cunha (21145), João Apresentação (21152)

1. Introdução

Este trabalho prático, relativo à unidade curricular de Inteligência Artificial, visa impulsionar a melhoria de performance do trabalho em grupo, num desafio que irá explorar as necessidades de conhecimento relativo ao desenvolvimento de agentes inteligentes e algoritmos CSP.

1.1. Contextualização

Como referido anteriormente, este trabalho foi realizado no âmbito da disciplina de inteligência artificial e propende aplicar os conhecimentos relativos a algoritmos de busca CSP implementando-os num agente inteligente capaz de resolver um problema relativo a agendamento de horários de aulas.

1.2. Problema a resolver

Entre uma lista de problemas propostas pelo professor, decidimos escolher o tema que achávamos que nos traria mais criatividade para explorar e pensar em soluções diversas.

No cenário atual de crise energética pretendeu-se então construir um agente capaz de remarcar os horários para os cursos de graduação da escola de tecnologia, a fim de reduzir o deslocamento para o campo e a necessidades de utilização de ar condicionado.

1.3. Objetivos

Desenvolver um agente que implemente o algoritmo de busca CSP, com objetivo de:

- Todas as aulas que tenham 2 horas estejam dispersas durante a semana;
    - (10 aulas por semana sendo que pelo menos 1 aula é online);
- Uma turma não deve ter mais que 3 aulas por dia;
- As aulas online não podem ser colocadas após aulas presenciais;
- Apenas 2 aulas podem ocorrer de manhã e até 2 à tarde;
- Cada turma tem de 2 a 4 aulas em uma sala específica.

1.4. Objetivo da formulação

A formulação do objetivo deste agente, baseia-se em, após receber os dados de um horário (as respetivas disciplinas, salas, dias e horas para cada aula de cada turma), desenvolver um horário que cumpra com as restrições fornecidas no enunciado

1.5. Funções e estruturação do agente

1.5.1. Atributos do agente (PEAS)

- **Medidas de Desempenho -** Horário mais eficiente ao nível de restrições implícitas;
- **Ambiente -** Turmas, UC's, Salas;
- **Atuadores -** Código python desenvolvido;
- **Sensores -** Dados inseridos.

1.5.2. Características de ambiente

- **Totalmente observável** – acesso ao estado completo em cada momento;
- **Determinístico** – o próximo estado será determinado pelo atual;
- **Agente único** – ambiente consiste em apenas um agente;
- **Sequencial** – a decisão que o agente toma irá afetar as futuras tendo em conta as restrições presentes no problema;
- **Estático** – O ambiente não altera enquanto o agente toma decisões;
- **Discreto** – Finito número de ações para obter o resultado;
- **Conhecido** – Todos os resultados são fornecidos pelas ações do agente.

1.5.3. Domínio

- Domínio (Turma[n]) -> { a, b }
- Domínio (UC[n]) -> { matemática, ciências, inglês, história }
- Domínio (Sala[n]) -> {201, 202, 104, 108, online}
- Domínio (Aulas[i]) -> {Aula: Object }
    - Aula é um objeto composto por:
        - Turma[n]
        - Uc[n]
        - Sala[n]
        - Dia_semana[n]
        - Duração[n]
        - inicioAula[n]

i – Corresponde à posição da aula em causa
n – Corresponde à posição das restantes variáveis

1.5.4. Formulação do problema

- **Variáveis:** Aula0.turma, Aula0.duração, Aula0.sala, Aula0.dia_semana, Aula0.inicioAula, Aula0.uc, Aula1.turma, …, Aula[n].uc
- **Domínio:**

  Cada membro de Aula[n].turma: { 1, 2 }

  Cada membro de Aula[n].duração: { 2 }

  Cada membro de Aula[n].sala: { 1, 2, 3, 4, 5 }, sendo sala 5 a aula online

  Cada membro de Aula[n].dia_semana: { 1, 2, 3, 4, 5 },

  Cada membro de Aula[n].inicioAula: { 9, 10, 11, 12, 13, 14, 15, 16, 17}

  Cada membro de Aula[n].uc: { 1, 2, 3, 4, 5 }

Os valores do domínio de cada membro são as keys para os tuplos criados para cada variável

- **Restrições:** [carregar aqui]

1.6. Variáveis

Para a construção deste agente utilizamos as seguintes variáveis:

- **Turma** - armazenar dados relativos às respetivas turmas existentes na escola de tecnologia
- **UC** - armazenar dados das disciplinas frequentadas pelas respetivas turmas
- **Sala** - armazenar dados relativos às salas que as turmas vão ocupar
- **Dia_Semana** - armazenar dados relativos à disposição das UCs ao longo da semana
- **Duracao** - armazenar dados relativos ao tempo que cada UC demora a ser realizada
- **InicioAula** - armazenar dados relativos à hora de início da aula

1.7. Restrições

- **One_uc_per_timeslot** - restringe que, para um determinado slot* de horário, (em diferentes horários) não pode ter a mesma disciplina sobreposta.
  One_classroom_per_timeslot - restringe que, para um determinado slot* de horário, diferentes horários não podem ter a mesma sala.
- **One_class_per_timeslot** - restringe que, para um determinado slot* do horário, (em diferentes horários), não podem ter a mesma aula.
- **online_class_not_after_presencial_class_after** - restringe que, as aulas online não podem ser reservadas imediatamente após uma aula presencial.
- **Atmost_three** - restringe que a turma não dever ter mais que 3 aulas por dia.

*Slot foi o nome de utilizamos para descrever cada espaço (bloco de memória) que uma aula ocupa no horário.

1.8. Estado final

O estado final é o horário preenchido com todas as aulas para as respetivas turmas (cumprindo assim o número de aulas semanais que é 10 x turmas), cumprindo assim também com todas as restrições que foram impostas

https://github.com/L0ud3r/CSP_Artificial_Intelligence

2. Conclusão

Este projeto foi possível recorrendo ao uso do Visual Studio Code, desenvolvido com linguagem python, GitHub para manipulação do projeto ao nível da equipa e Notebook.

Relativamente aos resultados obtidos foram os melhores possíveis dentro de tudo que foi possível ser implementado, com falta de uma restrição (Cada turma tem de 2 a 4 aulas em uma sala de aula específica.).