به نام خدا

پروژه دوم درس آزمایشگاه هوش مصنوعی

استاد راهنما : استاد زاد ضیابری

گردآورنده : مهدی شیرازی

بهار **1404**

چکیده

این پروژه به طراحی و پیاده‌سازی یک سیستم هوشمند زمان‌بندی وظایف برای کاوشگر مریخی "آستروبات" می‌پردازد. با توجه به محدودیت‌های سخت‌افزاری و عملیاتی در محیط مریخ، نیاز به یک سیستم برنامه‌ریزی دقیق برای مدیریت بهینه منابع انرژی و زیرسیستم‌ها احساس می‌شود.

در طراحی و پیاده سازی این پروژه از زبان برنامه نویسی پایتون نسخه 3.12.6 استفاده شده است.

**در ادامه به شرح کامل پروژه می پردازیم ...**

**روش شناسی و تحلیل**

- تخصیص بهینه ۵ وظیفه اصلی به ۵ بازه زمانی

- رعایت دقیق محدودیت‌های انرژی

- جلوگیری از تداخل عملکردی زیرسیستم‌ها

- تضمین انجام تمام وظایف در بازه‌های مشخص

**چالش‌های کلیدی:**

- محدودیت شدید منابع انرژی

- چالش‌های محاسباتی و الگوریتمی

- چالش‌های عملیاتی

- چالش‌های پیاده‌سازی

الف) ساختار داده‌ها و ورودی‌ها:

tasks = ['T1', 'T2', 'T3', 'T4', 'T5']

subsystems } =

'T2': 'Navigation',

'T2': 'Sampling',

'T2': 'Communication',

'T2': 'Navigation',

'T2': 'Sampling'

{

power\_needs } =

'T1': 5, 'T2': 4, 'T3': 6, 'T4': 7, 'T5': 3

}

power\_limit = [10, 6, 12, 8, 10]

ب) هسته الگوریتم (تابع backtrack):

def backtrack(assignment, tasks, subsystems, power\_needs, power\_limits):

if len(assignment) == len(tasks):

return assignment

for task in tasks:

if task not in assignment:

for time\_slot in range(1, 6):

if is\_valid\_assignment(assignment, task, time\_slot, subsystems, power\_needs, power\_limits):

assignment[task] = time\_slot

result = backtrack(assignment.copy(), tasks, subsystems, power\_needs, power\_limits)

if result is not None:

return result

break

return None

ج) بررسی محدودیت‌ها (تابع is\_valid\_assignment):

def is\_valid\_assignment(assignment, task, time\_slot, subsystems, power\_needs, power\_limits):

# بررسی تخصیص تکراری

if time\_slot in assignment. values() :

return False

# بررسی محدودیت انرژی

if power\_needs[task] > power\_limits[time\_slot - 1]:

return False

# بررسی تداخل زیرسیستم

current\_subsystem = subsystems[task]

for adj\_time in [time\_slot - 1, time\_slot + 1]:

if adj\_time in assignment.values() :

for t, slot in assignment.items() :

if slot == adj\_time and subsystems[t] == current\_subsystem:

return False

return True

تحلیل عملکرد الگوریتم

\* مکانیزم عملکرد :

۱. انتخاب وظیفه: سیستم به صورت متوالی هر وظیفه را انتخاب می‌کند

۲. تخصیص زمانی: بررسی تمام بازه‌های ۱ تا ۵ برای هر وظیفه

۳. بررسی محدودیت‌ها: سه فیلتر اصلی اعمال می‌شود

۴. تکرار بازگشتی: فرآیند برای وظایف باقیمانده تکرار می‌شود

۵. عقبگرد: اگر مسیری به جواب نرسد، به مرحله قبل برمی‌گردد

\* مثالی از اجرای الگوریتم :

۱. ابتدا T1 به بازه ۱ تخصیص می‌یابد (ناوبری - انرژی ۵)

۲. سپس T2 به بازه ۲ (ناوبری - انرژی ۴) رد می‌شود. (تداخل زیرسیستم)

۳. T2 به بازه ۳ منتقل می‌شود. بررسی انرژی و ...

۴. نمونه خروجی و تفسیر

}

'T1': 1,

'T2': 3,

'T3': 2,

'T4': 4,

'T5': 5

{

تفسیر برنامه زمانی و چالش ها

\* تفسیر برنامه زمانی:

- بازه ۱: ناوبری (T1) - مصرف ۵ از ۱۰ واحد

- بازه ۲: ارتباطات (T3) - مصرف ۶ از ۶ واحد

- بازه ۳: ناوبری (T2) - مصرف ۴ از ۱۲ واحد

- بازه ۴: نمونه‌برداری (T4) - مصرف ۷ از ۸ واحد

- بازه ۵: نمونه‌برداری (T5) - مصرف ۳ از ۱۰ واحد

۱. چالش‌های محاسباتی و الگوریتمی

\* انفجار ترکیبی (Combinatorial Explosion):

- با افزایش تعداد وظایف (n)، پیچیدگی محاسباتی به صورت فاکتوریل (O(n!)) رشد می‌کند.

- برای ۵ وظیفه: ۱۲۰ حالت ممکن

- برای ۱۰ وظیفه: ۳٫۶ میلیون حالت

\* بهینه‌سازی مصرف حافظه :

- نیاز به ذخیره‌سازی تمام تخصیص‌های موقت در حین بازگشت

۳. چالش‌های عملیاتی

- تداخل زیرسیستم‌ها:

- مثال واقعی: سیستم ناوبری هنگام نمونه ‌برداری دقت کمتری دارد.

- نیاز به افزودن محدودیت‌های غیرخطی (مثلاً: اگر T4 فعال باشد، T1 حداکثر ۳ واحد انرژی می‌تواند مصرف کند)

**\*راهکارهای پیشنهادی برای چالش‌ها\***

۱. استفاده از هیوریستیک‌ها :

- اولویت‌دهی به وظایف با انرژی کمتر در بازه‌های محدود

۲. پیاده‌سازی نسخه موازی:

- تقسیم مسئله بین چند پردازنده

۳. افزودن یادگیری ماشین:

- پیش‌بینی مصرف انرژی واقعی بر اساس داده‌های تاریخی

**محدودیت ها**

الف) مقیاس پذیری :

**\*** برای n>8 غیر عملی می شود.

**\*** پیچیدگی زمانی بسیار بالاx n!) O (n2

**\*** پیچیدگی فضایی نسبتا مناسب O(n) [python Dictionary O(n2)]

ضمیمه ها

کد پروژه :

**def is\_valid\_assignment(assignment, task, time\_slot, subsystems, power\_needs, power\_limits):**

**if time\_slot in assignment.values():**

**return False**

**if power\_needs[task] > power\_limits[time\_slot - 1]:**

**return False**

**current\_subsystem = subsystems[task]**

**for adj\_time in [time\_slot - 1, time\_slot + 1]:**

**if adj\_time in assignment.values():**

**for t, slot in assignment.items():**

**if slot == adj\_time and subsystems[t] == current\_subsystem:**

**return False**

**return True**

**def backtrack(assignment, tasks, subsystems, power\_needs, power\_limits):**

**if len(assignment) == len(tasks):**

**return assignment**

**for task in tasks:**

**if task not in assignment:**

**for time\_slot in range(1, 6):**

**if is\_valid\_assignment(assignment, task, time\_slot, subsystems, power\_needs, power\_limits):**

**assignment[task] = time\_slot**

**result = backtrack(assignment.copy(), tasks, subsystems, power\_needs, power\_limits)**

**if result is not None:**

**return result**

**break**

**return None**

**tasks = ['T1', 'T2', 'T3', 'T4', 'T5']**

**subsystems = {**

**'T1': 'Navigation',**

**'T2': 'Navigation',**

**'T3': 'Communication',**

**'T4': 'Sampling',**

**'T5': 'Sampling'**

**}**

**power\_needs = {**

**'T1': 5,**

**'T2': 4,**

**'T3': 6,**

**'T4': 7,**

**'T5': 3**

**}**

**power\_limits = [10, 6, 12, 8, 10]**

**assignment = backtrack({}, tasks, subsystems, power\_needs, power\_limits)**

**if assignment:**

**print("Task assignment found:")**

**for task, time\_slot in assignment.items():**

**print(f"{task}: Time slot {time\_slot} (Subsystem: {subsystems[task]}, Power: {power\_needs[task]})")**

**else:**

**print("No solution found")**

**منابع و ماخذ**

**\*** یادگیری و مرور الگوریتم Backtracking :

[**https://en.wikipedia.org/wiki/Backtracking**](https://en.wikipedia.org/wiki/Backtracking)

[**https://www.geeksforgeeks.org/backtracking-algorithms/**](https://www.geeksforgeeks.org/backtracking-algorithms/)

[**https://www.youtube.com/watch?v=DKCbsiDBN6c**](https://www.youtube.com/watch?v=DKCbsiDBN6c)

**\*** بررسی پیچیدگی های زمانی و فضایی و چالش های کد :

**Deep seek AI Chatbot**

**Copilot AI Chatbot**