Planificació Automàtica de Menús amb PDDL

Visió general

Aquest projecte mostra una cadena completa per generar i resoldre problemes de planificació en **PDDL** que construeixen menús setmanals (dilluns–divendres) sota diferents nivells de restricció. S'utilitza el planificador **Fast-Forward (FF)** i la seva variant **Metric-FF** per treballar amb fluents i optimització de costos.

Nivell	Predicats addicionals	Restriccions	Nota màx.
`basicb`	—	*Compatibilitat plat 1 ↔ plat 2*	6
`ext1b`	`usat`	No repetir cap plat en tota la setmana	7
`ext2b`	`diaPrimerTipus`, `diaSegonTipus`	No repetir tipus en dies consecutius	8
`ext3b`	`platObligatori`	Alguns plats han de sortir un dia determinat	9
`ext4b`	funcions `calories`	1000 ≤ kcal ≤ 1500 per dia	10
`ext5b`	funció `preu`, `metric`	Minimitzar el cost total del menú	11

> **Directori de treball**: totes les rutes de la guia assumeixen que estàs situat/da al directori arrel del repositori.

Estructura de carpetes

```
.
├── binaries/            # executables FF (linux/ i windows/)
├── data/                # catàlegs de plats, calories, preus,
incompatibilitats
├── problems/
│   ├── basicb/
│   ├── ext1b/ … ext5b/  # cada extensió amb el seu domain.pddl
│   │   └── test-cases/  # problemes generats (menu-*.pddl)
├── scripts/
│   ├── generator.py     # genera problemes aleatoris
│   └── run_all.sh       # llança tots els test-cases i dóna estadístiques
└── README.md            # aquesta guia
```

Requisits

- **Python ≥ 3.8** – per executar `scripts/generator.py`.
- **FF** i **Metric-FF**:
  - **Linux**: ja hi ha versions compilades a `binaries/linux/ff` i `binaries/linux/metric-ff`.
  - **Windows**: es proporcionen `binaries/windows/ff.exe` i `binaries/windows/metricff.exe` (compilades amb Cygwin).
  - Si prefereixes compilar-los tu mateix:

```
# Exemple (Linux)
sudo apt install flex bison g++            # dependències
tar -xzf FF-v2.3.tgz
cd FF-v2.3 && make
```

Generar casos de prova

```
# 5 casos per a totes les extensions amb llavor fixa
python3 scripts/generator.py --all --cases 5 --seed 42

# 2 casos només per a ext2 i ext4
python3 scripts/generator.py --ext 2 4 --cases 2
```

Es creen fitxers `menu-<ext>N-tc<M>.pddl` dins `problems/<extNb>/test-cases/`.

Execució manual d'un cas

Linux

```
# Extensió 3, cas 1
binaries/linux/ff \
  -o problems/ext3b/domain.pddl \
  -f problems/ext3b/test-cases/menu-ext3b-tc1.pddl
```

Windows (PowerShell o CMD)

```
REM Extensió 4 (Metric-FF) – recorda afegir -O només a ext5
binaries\windows\metricff.exe ^
  -o problems\ext4b\domain.pddl ^
  -f problems\ext4b\test-cases\menu-ext4b-tc1.pddl
```

Si treballes dins WSL i vols cridar les versions Windows per a proves de rendiment, converteix les rutes amb `wslpath -w`.

`scripts/run_all.sh`

Què fa

1. Recorre `basicb` i totes les extensions (`ext1b` … `ext5b`).
2. Localitza tots els `.pddl` dins de `test-cases/`.
3. Tria automàticament l'executable adequat:
   - `ff` per `basicb`, `ext1b`, `ext2b`, `ext3b`.
   - `metric-ff` per `ext4b`.
   - `metric-ff -O` per `ext5b` (optimització).
4. Executa cada combinació *domini* + *problema* i comprova el `exit code`.
5. Mostra un resum final amb nombre de tests passats (%) i desa tot el log a `scripts/run_all.log`.

Ús

```
chmod +x scripts/run_all.sh   # un sol cop
./scripts/run_all.sh          # llança totes les proves
```

Exemple de sortida abreujada:

```
=== Extension basicb ===
→ menu-basicb-tc1.pddl    ✔ OK
...
──────────────── RESULTATS ────────────────
Total tests    : 30
Passats        : 29
Percentatge    : 97 %
```

Preguntes freqüents

- **P: FF s'atura amb "predicate INCOMPATIBLE is declared to have 2 (not 3) arguments".**
  R: Assegura't que tots els noms de plats són ASCII pur (`[A-Za-z0-9_-]`). Accents o espais trenquen la tokenització.
- **P: Metric-FF no troba el pla òptim?**
  R: La cerca `best-first` pot acabar amb un pla vàlid però no mínim. Torna a executar-lo o ajusta els paràmetres d'heurística si cal.