METTI A CONFRONTO DIVERSI PROBLEMI

**REPORT**

Il dominio è stato testato utilizzando come planner **OPTIC**. Verranno messi a confronto i *plan* di diversi *problem* al fine di evidenziare le performance del planner in scenari differenti.  
Saranno analizzati anche tre valori che permettono di valutare la qualità e l’efficienza di un planner:

* **Makespan**: è il tempo totale che intercorre dall’inizio della prima azione fino alla fine dell’ultima.
* **Metric**: il valore minimizzato o ottimizzato in base alla metrica specificata.
* **States evaluated**: rappresenta quanti stati distinti (configurazioni del mondo) il planner ha valutato prima di trovare un piano valido. Maggiore è il numero di stati valutati, maggiore sarà il lavoro del planner. Questo valore ci indica quindi l’efficienza del planner: un buon planner troverà piani ottimali esplorando pochi stati.
* **Tempo di pianificazione**: è il tempo di calcolo effettivo che il planner ha impiegato per esplorare lo spazio degli stati, applicare euristiche e restituire una (o più) soluzioni. Un piano buono, ma che richieda troppo tempo per essere calcolato, potrebbe risultare inutile in un sistema operativo in tempo reale.

**Problem 0\_5**

Durante l’esecuzione del *problem0\_5*, il planner ha trovato 2 piani temporali:  
• Piano 1: 30.005 s  
• Piano 2: 30.005 s

| **Parametro** | **Valore** |
| --- | --- |
| Metric / Makespan | 30.005 |
| Stati valutati | 35 |
| Tempo di pianificazione | 10.113 s |

* **Metric / Makespan**: il piano ha una durata di 30.005 secondi.
* **States evaluated**: valore basso, indica che il nostro dominio è semplice o che il planner è molto efficiente.
* **Tempo**: 10 secondi è un buon valore per la pianificazione.

Questi risultati suggeriscono che OPTIC ha trovato più volte la stessa soluzione ottimale, evidenziando una certa stabilità rispetto al problema.

[Spiegati i passaggi del problema 0\_5. Non verranno approfonditi in egual modo tutti i problemi: verranno evidenziati solo i passaggi più significativi.]

Analizziamo ora le azioni del piano da un punto di vista strutturale:  
Il piano prevede come prima azione il movimento dei due robot (**move\_two\_robot\_to\_crate robot2 robot1 crate1 base1**) verso la crate1 (pesante 70 kg).  
Una volta effettuato il *drop* della crate1 alla base, inizia il *loading* sul nastro.  
Successivamente, il robot1 si muove e carica la crate2 (leggera, 20 kg), fino al *drop* alla base e al conseguente *loading*.  
Dopo il primo *drop*, il robot1 (e il robot2) ha una batteria residua di 12 unità. È quindi necessario che passi dalla **charging bay** per ricaricarsi e successivamente riprenda la propria attività. Questo passaggio è necessario per accumulare abbastanza energia per affrontare le azioni future: *move\_to\_crate* e *move\_back*.  
Stessa situazione per il robot2: anche se non viene utilizzato per le azioni successive, è necessario che venga ricaricato passando dalla **charging bay**.

Questi grafici forniscono una visione dinamica di alcune variabili chiave durante l’esecuzione del piano. Mostrano come lo stato della batteria dei robot vari in relazione all’utilizzo e alla ricarica, come la priorità di un'entità cambi nel tempo e come la disponibilità della base di ricarica sia influenzata dall’attività dei robot.

È importante definire il significato di **priority** e **free\_base**.  
Nonostante in questo problema la *priority* non abbia avuto un ruolo centrale, essa è utile per caricare crate appartenenti allo stesso gruppo. Nel nostro problema passa in secondo piano poiché, disponendo solo di due crate, la priorità non è necessaria.  
Invece, *free\_base* evidenzia nel tempo gli istanti in cui la base è libera o occupata da una crate, permettendo quindi l’avvio del processo di *loading* sul nastro trasportatore o il *drop* alla base da parte dei robot, qualora essa sia libera.

**Problem 1**

Il planner ha trovato due plan :

* plan1 = 50.674
* plan2 = 48.674

| **Parametro** | **Valore** |
| --- | --- |
| Metric / Makespan | 50.674 |
| Stati valutati | 68 |
| Tempo di pianificazione | 33.758 s |

* **Makespan**: durata maggiore rispetto al problema precedente. Tuttavia, la gestione delle casse è più articolata e una cassa risulta fragile, richiedendo un *loading\_leggero* che impiega più tempo (da 4 a 6 unità di tempo). L’aumento del makespan è quindi giustificato dalla maggiore complessità.
* **States evaluated**: il numero di stati valutati è quasi raddoppiato rispetto al problema precedente, principalmente a causa della maggiore complessità e del numero più elevato di azioni ricorrenti. Il planner, operando su uno spazio di ricerca più ampio, ha valutato più alternative valide. Resta comunque un valore ottimale.
* **Tempo di pianificazione**: aumentato significativamente, ma proporzionale al numero di stati valutati e alla complessità del problema.

[Caricati *plan1* e *plan2*]

Il planner ha trovato due piani: *piano1* con makespan 50.674 e *piano2* con makespan 48.674. Ciò suggerisce un’ottimizzazione nella scelta delle azioni da parte del *piano2*.

La differenza principale è nell’ordine in cui le crate vengono processate:

* *Piano1*: crate1 e crate2 (gruppo A), infine crate3
* *Piano2*: prima crate2, poi crate1, infine crate3

Il *piano2* processa subito la crate fragile, sfruttando il fatto che i robot sono carichi, ottimizzando quindi il trasporto più delicato sin dall’inizio.

Un’altra differenza riguarda i tipi di *loader* utilizzati:

* **Piano1**
  + crate2 usa *loading\_fragile*
  + loader coinvolto: **loader1**
* **Piano2**
  + crate2 usa *loading\_leggero\_fragile*
  + loader coinvolto: **loader\_leggero1**

Anche la ricarica avviene in modo diverso:  
Nel *piano1* è intervallata tra le operazioni, mentre nel *piano2* avviene dopo ogni trasporto, evidenziando un *pattern* regolare (dopo crate1 e crate2, ma non dopo crate3).

Il *piano2* ha quindi ottimizzato la gestione di alcune azioni rispetto al *piano1*, garantendo un tempo di esecuzione inferiore.

**Problem 2**

| **Parametro** | **Valore** |
| --- | --- |
| Metric / Makespan | 82.008 |
| Stati valutati | 98 |
| Tempo di pianificazione | 35.189 s |

* **Makespan**: doppio rispetto al *piano2* e quasi triplo rispetto al *piano1*. Questo è atteso, poiché ora sono presenti 4 casse da trasportare e caricare, con conseguenti operazioni seriali aggiuntive.
* **States evaluated**: il numero cresce, ma non in modo esplosivo. Ciò suggerisce che il dominio sia ben strutturato e riesca a contenere lo spazio di ricerca.
* **Tempo di pianificazione**: nonostante il numero maggiore di stati, il tempo rimane stabile. La complessità non impatta significativamente il tempo di calcolo.
* Il planner ha trovato un solo piano, a differenza dei casi precedenti. Questo può indicare:
  + minore flessibilità nella sequenza delle azioni (poche alternative valide), oppure
  + che un eventuale secondo piano superasse una soglia critica di tempo/makespan e quindi sia stato scartato.

**Problem 3**

Sono stati trovati dal planner tre piani:

* plan1 = 156.012
* plan2 = 154.012
* plan3 = 146.012

| Parametro | Valore |
| --- | --- |
| Metric / Makespan | 156.012 |
| Stati valutati | 1376 |
| Tempo di pianificazione | 32.912 s |

* Makespan: Medesimo numero di casse rispetto al *problem2*, ma makespan raddoppiato. Ciò è dovuto principalmente alla presenza di casse molto pesanti, che comportano tempi di trasporto molto lunghi e movimenti molto lenti.
* States evaluated: Il planner ha esplorato molti più percorsi rispetto ai casi precedenti. Le diverse durate, le ricariche necessarie e le casse pesanti hanno aumentato l'entropia della ricerca, rendendo il compito molto complesso.
* Tempo di pianificazione: Nonostante la complessità, il planner è riuscito a trovare 3 plan che soddisfano i requisiti in un tempo ottimale (32.912 s).

Nonostante i 3 plan trovati, si può notare un errore comune: nel processare la *crate3*, si riscontra un problema nella batteria dei robot. Infatti, con una batteria carica al massimo (20 unità), non sarebbe possibile completare il processo della *crate3* in nessuno dei plan.  
L’azione move\_two\_robot\_to\_crate robot2 robot1 crate3 base1 dura 3 unità, e move\_back\_two\_robot crate3 base1 robot2 robot1 dura 18 unità, per un totale di 21 unità. Questo risulta impossibile da eseguire con una batteria da 20 unità.  
Per risolvere questa situazione, è necessario aumentare la capacità della batteria dei robot da 20 a 25 unità. Questo incremento consente di processare la *crate3* senza riscontrare anomalie nei livelli della batteria.

**Problem 4**

Plan trovati dal planner:

* Plan1 = 101.681
* Plan2 = 89.681

| Parametro | Valore |
| --- | --- |
| Metric / Makespan | 101.681 |
| Stati valutati | 2449 |
| Tempo di pianificazione | 36.748 s |

* Makespan: Nonostante la notevole complessità del problema, otteniamo un valore molto buono, coerente con i problemi precedenti. Rispetto al *problem3*, non si verificano problematiche legate alla ricarica o a processi di movimento lenti, il che comporta una diminuzione del makespan.
* States evaluated: Valore molto elevato rispetto ai problemi precedenti, a causa dell’aumento delle combinazioni di azioni e risorse disponibili. Questo evidenzia una moderata complessità del problema, che comporta un aumento del tempo e della memoria necessari per la pianificazione, ma anche una maggiore efficienza nel piano eseguibile.
* Tempo di pianificazione: Valore accettabile e proporzionale alla complessità del problema, considerando lo stato iniziale, i goal e l’aumento delle risorse rispetto ai casi precedenti.

Questi valori indicano un piano efficiente, frutto di una ricerca del planner eseguita in tempi ragionevoli, con un makespan contenuto e un utilizzo efficace delle risorse**.**

**Nota sul Ground Size**

**Il *Ground size*** indica il numero totale di azioni istanziate, predicati istanziati e oggetti considerati nel problema PDDL dopo l’istanziazione completa da parte del planner.  
Per il suo calcolo, servono:

* il numero e i tipi di oggetti (robot, crate, base, ecc.) nel file problema
* il numero di azioni nel dominio e i loro parametri
* il numero di combinazioni compatibili per ciascuna azione