Planning Project - A Good Snowman is Hard to Plan

Matteo Paparo - Mat 264387

Introduzione

Il seguente elaborato consiste in uno studio rigaurdante l'uso di *PDDL Numeric*, nel dettaglio è stato affrontato il quesito "A Good Snowman is Hard to Plan". Il tutto è centrato sulla creazione dei pupazzi di neve, dove al dominio originario sono state effettuate diverse conversioni graduali, ovvero si è scelto di proseguire con una conversione delle componenti più importanti del dominio. L'obiettivo di questa analisi è verificare l'impatto dell'uso di pddl numeric nel quesito proposto.

Domain

La rappresentazinoe del dominio avviene su una griglia (non obbligatoriamente regolare) $n \times m$ in cui ogni singola cella può essere innevata, e in base al problema proposto saranno presenti dalle 3 alle 6 palle di neve per la creazione dei pupazzi di neve. In riferimento all'obiettivo del problema sono stati identificati 3 domini differenti, rispettivamente per la creazione di 1, 2 pupazzi di neve.
Le regole da rispettare sono:

- Se una palla di neve passa sopra una cella innevata, se la palla di neve non è delle dimensioni massime, allora essa aumenterà di grandezza;
- É possibile creare una pila di palle di neve, se la palla in cima è di dimensioni più piccole rispetto la palla sottostante;
- Per spostare una palla di neve, il giocatore deve stare in una cella adiacente alla palla di neve. Se la cella di destinazione è libera, o contiene una palla di neve di dimensioni maggiori, allora la palla di neve può essere spostata e il giocatore si sposterà sulla cella precedentemente occupata dalla palla.

Types, objects and fluents
Gli oggetti cosiderati sono i seguenti: *loc, dir* e *ball*. Il tipo *loc* rappresenta la posizione di una cella della griglia. Il tipo *dir* enumera le quattro diezioni in cui il personaggio e le palle possono muoversi (su, giù, sinistra, destra) e il tipo *ball* definisce tutte le palle presenti nello scenario.

Per modellare lo stato corrente, nel dominio di base, sono stati defini i seguenti predicati:

- (snow ?l - loc): location l is covered in snow;
- (next ?from ?to - loc ?d - dir): locations from and to are adjacent; location to is in direction d relative to from;
- (occ ?l - loc): location l contains at least one ball;
- (char at ?l - loc): character is at location l;
- (ball at ?b - ball ?l - loc): ball b is at location l;
- (ball size s ?b - ball): ball b has small size;
- (ball size m ?b - ball): ball b has medium size;
- (ball size l ?b - ball): ball b has large size;
- (goal): the goal is satisfied.

Actions
Le azioni considerate sono le seguenti: *move_ character*, che come suggerisce il nome, permette al personaggio di spostarsi in una cella adiacente, *move_ ball*, che permette di spostare una palla di neve in una cella adiacente, e *goal* che permette di verificare se il goal è stato raggiunto.

In riferimento alla versione originale, si hanno i seguenti risultati[1]:

Domain	Problems	Solved
1	33	25
2	13	1

Conversione a PDDL Numeric

L'adattamento del dominio a PDDL Numeric è stato effettuato in modo graduale, partendo dal dominio base si è proseguito con la conversione più funzionale, ovvero con la conversione dei predicati riguardanti la ball_size in una funzione numerica. Nel dettaglio abbiamo che per identificare le tre dimensioni (small, medium e large) delle palle di neve, sono state assegnati tre valori differenti, ovvero 1 per small, 2 per medium e 3 per large. Successivamente, si è proseguito con un adattamento della metodologia di definizione della griglia, convertendo il predicato relativo in due funzioni numeriche, in modo da identificare la cella con delle coordinate. Per concludere una conversione con una metodologia differente di verifica per la creazione dei pupazzi di neve.

In sintesi, identifichiamo 3 versioni differenti:

1 **Goal** : che comprende la conversione dei predicati riguardanti le dimensioni delle palle di neve e del goal;

2 **Coord** : che in aggiunta alla conversione precedente, identifica le location con delle coordinate numeriche;

3 **Count_ball** : che in aggiunta alla prima versione, tiene traccia del numero di palle di neve in ogni location.

Goal - version

Nel dominio originale, per differenziare la grandezza di una palla di neve venivano utilizzati tre predicati differenti. Con la conversione a PDDL Numeric, essi sono stati sintetizzati in una sola funzione numerica denominata *ball_ size*. Essa può assumere tre valori differenti, ovvero 1 per le palle piccole, 2 per le palle medie e 3 per le palle grandi. In fine, anche il predicato riguardante l'obiettivo è stato convertito in una funzione numerica. Nel caso della creazione di un pupazzo di neve, assume valore 1, se il pupazzo di neve è stato creato. Questa conversione assume un significato differente per i domini predisposti alla creazione di più pupazzi, che tiene conto del numero di pupazzi creati, infatti è stata rinominata in *snowman_ built*.
In merito a questo caso, sono stati aggiunti due predicati significativi :

[1]Per la risoluzione dei problemi è stato utilizzato il solver *enshp-20* e sono stati risolti su un dispositivo avente come processore Apple Silicon M3 16Gb di RAM

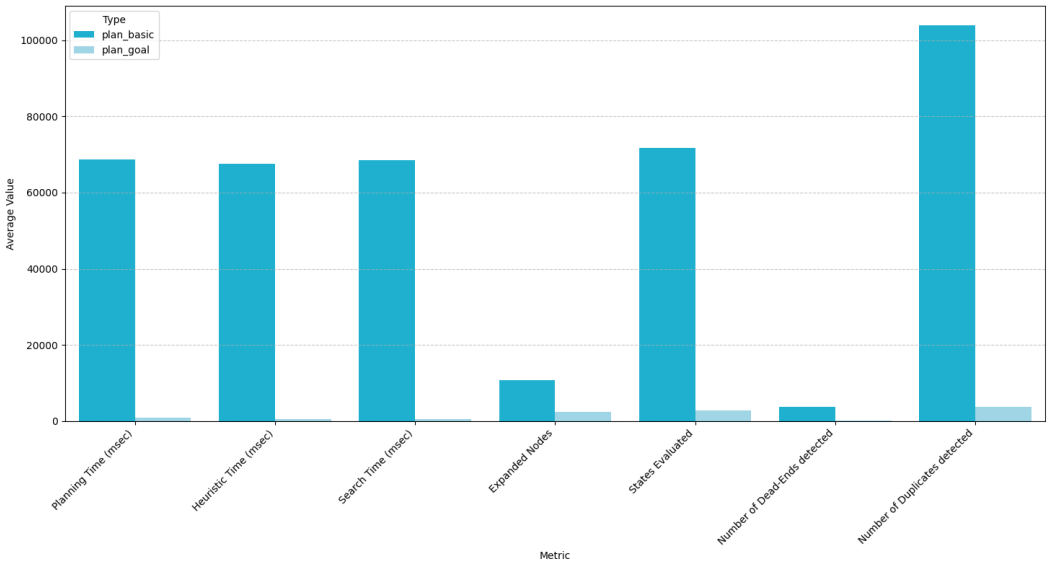
- (snowman_at ?l - loc): location l contains a snowman;
- (ball_used_in_snowman ?b - ball): ball b is used in a snowman.

Questa scelta è stata fatta per rendere l'obiettivo del problema più chiaro, poichè nel dominio originale, per la creazione di più pupazzi di neve, si verificava se tutte le palle di neve erano distinte tra di loro e in gruppi da tre occupavano location differenti. Per come è stato implementato il dominio, esso è estendibile a problemi predisposti per la creazione di più pupazzi di neve.

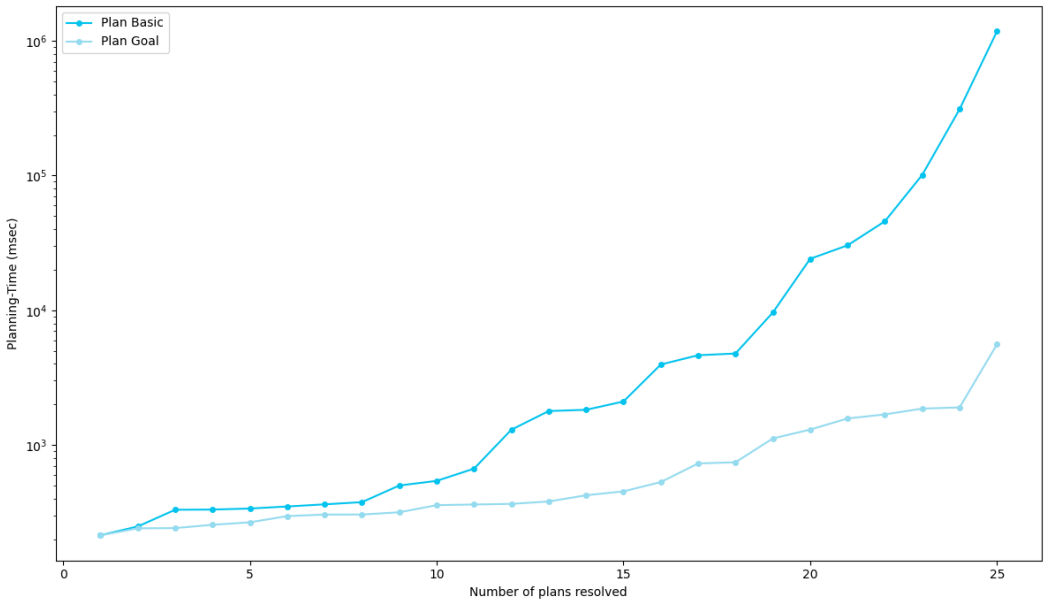
Questa è la conversione più semplice che ha riportato dei risultati molto soddisfacenti. Dei 46 problemi proposti, ben 33 hanno dato una soluzione. Nel dettaglio abbiamo che :

Domain	**Problems**	**Solved**
1	33	31
2	13	2

Rispetto la versione originale si è presentato un leggero incremento del grounding time, ma una riduzione di tutte le altre metriche.
In media è stato riportato un **Grounding Time** per la versione **Basic** di **71,44 msec** e per la versione **Goal** di **104,76 msec** ma una riduzione del **Plan-Length** di **11 steps**.



Dal grafico possiamo evincere una riduzione drastica del numero di **Stati** e **Duplicati** valutati, che comportano una diminuzione dei tempi di risoluzione, quindi un'ottimizzazione del dominio originale.



Dal seguente grafico[2] si può notare come la versione **Goal** abbia il **Planning Time**, all'aumentare dei piani risolti, un andamento molto più contenuto rispetto alla versione **Basic**, che ha un andamento esponenziale. Questo è dovuto al fatto che la versione **Goal** ha un numero di stati e duplicati molto più contenuto.

Coordinate - version

La versione **Coordinate** è stata realizzata con l'obiettivo di ridurre la complessità di inizializzazione della griglia. Nella versione **Basic** il collegamento tra le varie locatione viene effettuata con l'uso del predicato

(next ?from ?to - loc ?d - dir)

comportando quindi, per ogni singola cella della griglia, la definizione di un predicato per ogni direzione. In questa versione, invece, si è scelto di identificare ogni singola cella della griglia con delle coordinate numeriche, in modo da poter definire le location con una funzione numerica.

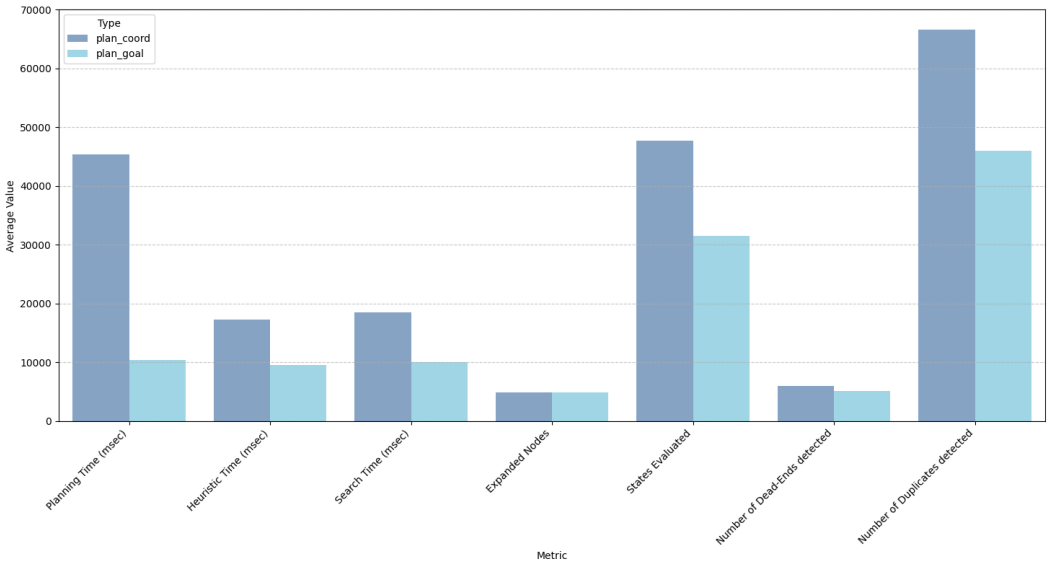
(x-coord ?l - location) and (y-coord ?l - location)

Questa modifica è significativa per la riduzione del numero di predicati, poichè per problemi definiti su griglie di dimensioni elevate, il numero di predicati cresce in modo esponenziale. Questa versione è un'estensione della versione **Goal**, in quanto mantiene le modifiche precedenti. In riferimento a questa versione, dei 46 problemi proposti, ben 30 hanno dato una soluzione. Nel dettaglio abbiamo che :

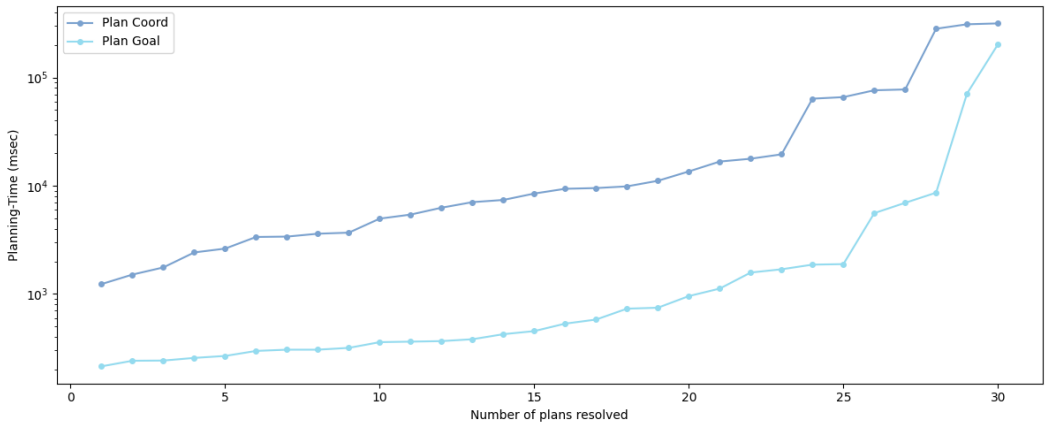
Domain	Problems	Solved
1	33	28
2	13	2

[2]Non è una comparazione tra i piani degli autori

In media è stato riportato un **Grounding Time** per la versione **Goal** di **113,93 msec** e per la versione **Coordinate** di ben **3170,13 msec**, un valore molto importante da valutare. Mentre la lunghezza del piano non ha subito variazioni significative



Dal grafico possiamo notare un incremento,in generale, dei tempi. Il dato più significativo sta nella disparità dei differenti **Planning Time**. Nel dettaglio abbiamo, in media, per la versione **Goal** un tempo meglio di **4474,25 msec**, contro i **46574,40 msec** della versione **Coordinate**, riportando valori **10** volte maggiori.



Il seguente grafico evidenzia la differenza tra il **Planning Time** delle due versioni evidenziando la differenza tra i tempi e il loro andamento.

Count_ball - version

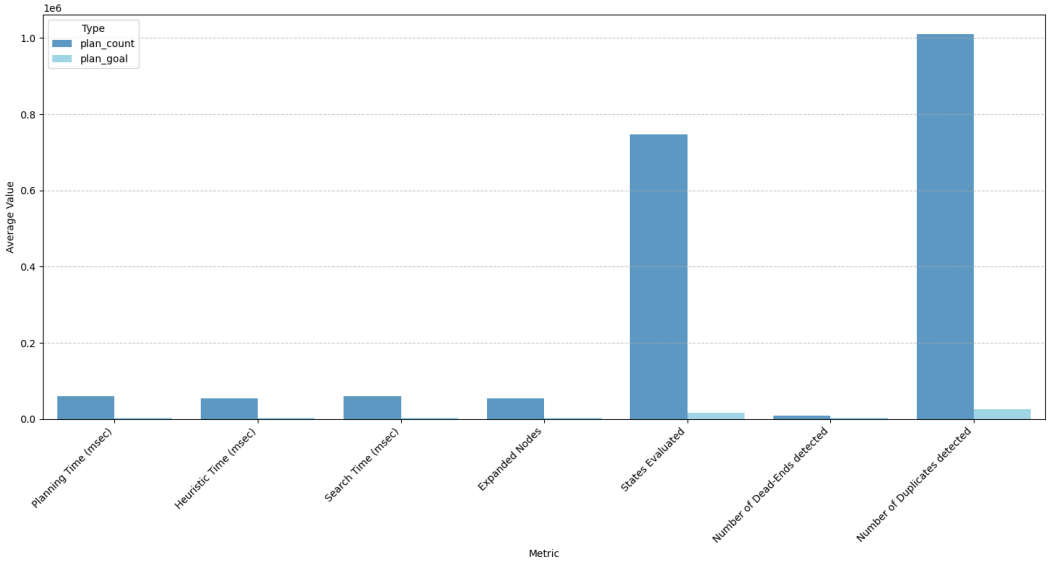
La versione **Count_ball** è stata realizzata con l'obiettivo di tenere traccia del numero di palle di neve in ogni location, con il secondo fine di evitare la verifica di soddisfacibilità dell'obiettivo. Per come è strutturato il dominio, se tre palle di neve si trovano in una sola location, esse saranno ordinate in senso decrescente, dal basso verso l'alto, predisposte per la realizzazione di un pupazzo di neve. Con l'introduzione della funzione

(ball_count ?l - loc)

evitiamo verifiche superflue, riducendo i tempi di risoluzione. Questa versione è un'estensione della versione **Goal**, in quanto mantiene le modifiche precedenti. In riferimento a questa versione, dei 46 problemi proposti in 25 hanno dato una soluzione

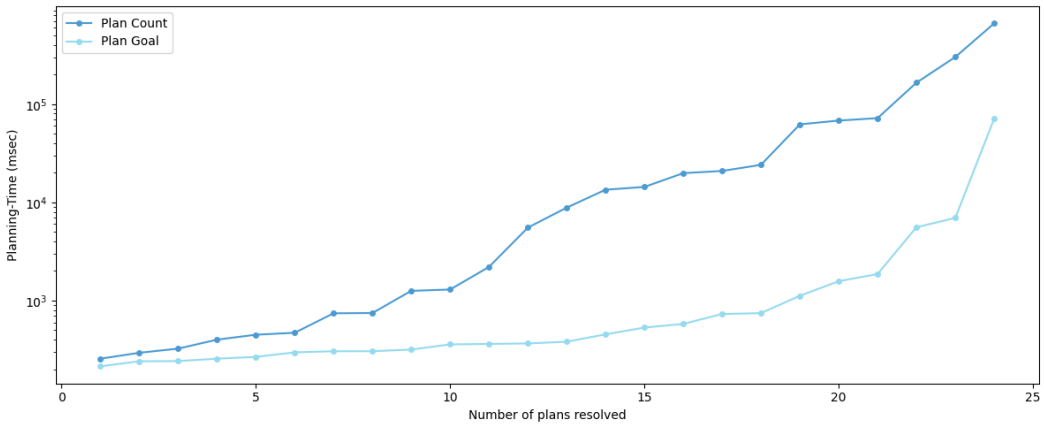
Domain	Problems	Solved
1	33	22
2	13	3

Rispetto la verione **Goal** si è riportato una riduzione del **Grounding Time**, rispettivamente per la versione **Goal** di **105,17 msec** e per la versione **Count_ball** di **79,25 msec**. Inoltre è stato riportato un incremento del **Plan-Length**, in media di abbiamo che per la versione **Goal** un valore di **52 steps** e per la versione **Count_ball** di **66 steps**.



Dal seguente grafico possiamo evidenziare la disparità di **States Evaluated** e **Duplicated States Evaluated** tra le due versioni. Nel dettaglio abbiamo che:

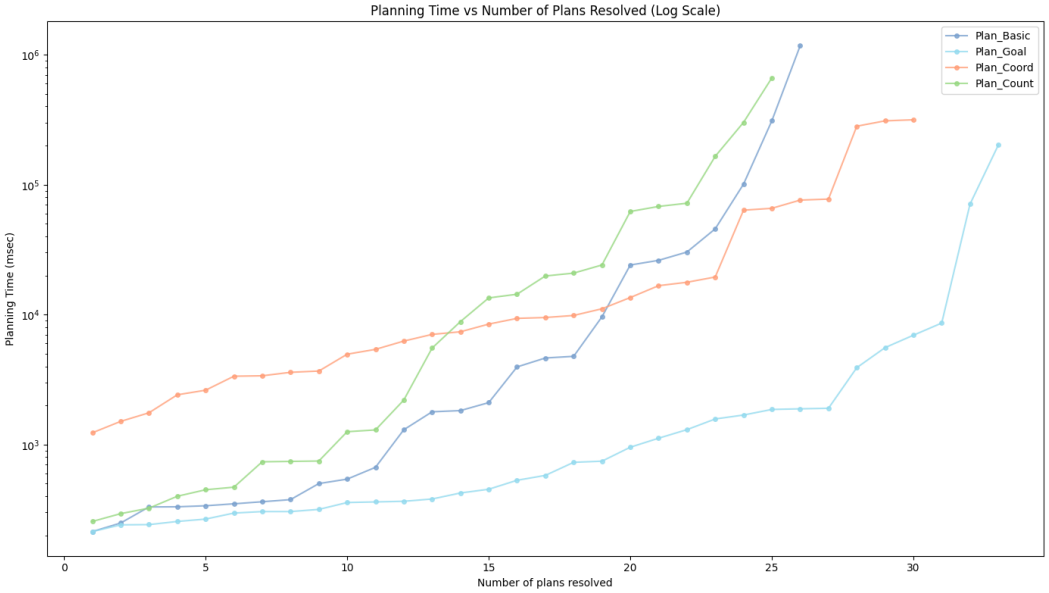
Version	States Evaluated	Duplicated States Evaluated
Goal	16.561	748.227
Count_ball	25.266	1.010.668



Il seguente grafico evidenzia l'andamento del **Planning Time** per le due versioni, evidenziando un andamento più contenuto per la versione **Goal** rispetto alla versione **Count_ball** che al contrario presenta dei tempi più alti già ad un numero di livelli relativamente basso.

Analisi generale

In generale, l'uso di PDDL Numeric ha portato a dei risultati molto soddisfacenti, in quanto si è ottenuto un incremento del numero di problemi risolti e una riduzione dei tempi di risoluzione. In particolare, la versione Goal ha riportato un incremento del numero di problemi risolti rispetto alla versione originale, mentre le sue estensioni, ovvero la Coordinate e la Count_ball hanno risportato dei lati negativi che purtroppo non li rendono preferibili rispetto alla versione Goal.
Dalla versione Basic alla versione Goal sono stati risolti 9 problemi in più e dal grafico successivo, evidenziamo i tempi ridotti rispetto le altre verisoni.



La seguente tabella riassume il numero di problemi risolti per ogni versione:

Version	**Problem 1 and 2**	**Solved**
Basic	33 - 13	25 - 1
Goal	33 - 13	31 - 2
Coordinate	33 - 13	28 - 2
Count_ball	33 - 13	22 - 3

In definitiva, ogni versione evidenzia delle caratteristiche significative :

- **Basic** con piani particolarmente lunghi e un numero di duplicati relativamente alto;
- **Goal** ha una pianificazione molto veloce con pochi nodi espansi e duplicati, ma poco adatto a scenari complessi;
- **Coordinate** è equilibrato, ha un piano medio-lungo riuscendo a mantenere sotto controllo i duplicati, con tempi di pianificazione moderati;
- **Count_ball** è adatto a piani articolati, ma ha un altissimo umero di duplicati e tempi lunghi;