Universidade de São Paulo
Mestrado em Ciência da Computação

Instituto de Matemática e Estatística
MAC5788 – Planejamento em IA

Modelamento do ROADEF em PDDL

Christian Danniel Paz Trillo
cpaz@ime.usp.br

13 de Outubro de 2003

1 Restrições

Para a solução planteada se consideram três restrições sobre a descrição do ROADEF:

- as restrições de rádio N/P para os requerimentos de linha de montagem são do tipo $1/P$ para todos os requerimentos,
- as penalidades por cada restrição são associadas ao mesmo valor, no exemplo a apresentar será 1, não se considerará que as restrições podem ter alta ou baixa prioridade, e,
- para o cálculo das penalidades ocasionadas pelos carros do dia anterior, só se levará em consideração o último carro com o requerimento, e não toda a última janela de tempo.

2 Descrição

A solução escolhida consta de ter um pré-processador, que gera, baseado nos arquivos do problema do ROADEF, um arquivo de domínio em pddl, com ações específicas do problema, que serão especificadas daqui a pouco, e um arquivo de problema, definindo as cores usadas no problema, e o estado inicial e meta.
Será usado um contador para o número de carros que pode-se pintar antes de que seja necessária uma limpeza, e uma referência da última cor usada na pintura; do mesmo jeito serão mantidos contadores para cada requerimento indicando o número de carros que faltam para poder montar um carro com esse requerimento sem ter penalidade. Será mantido um contador de limpezas efetuadas e um contador de penalidade, que é incrementada em 1 por cada penalidade cometida.
O domínio consta das seguintes ações por cada carro na lista:

- `paint_carX ?c - color`: processa as possibilidades de montar o carro, acerca da cor dele, é dizer se a cor é a mesma do último carro e ainda não atingiu o limite superior em que precisa-se fazer uma limpeza, ou se a cor não é a mesma, e precisa contar como uma penalidade. Atualiza o contador de cores, assim como a referência à última cor usada.

- `place_carX_featureI`: para cada requerimento do carro, existe uma ação deste tipo que processa as possibilidades acerca do requerimento I, se sua colocação nessa posição faz cair em alguma penalidade ou não. Atualiza o contador do requerimento a $P-1$, porque até não ter $P-1$ carros montados a mais, todos os carros com esse requerimento que sejam montados terão penalidade.

- `mount_carX`: representa o fato de montar o carro, ele decrementa os contadores dos requerimentos não especificados para o carro `X`, além disso habilita as precondições para que outros carros sejam montados.

De fato, estas ações têm que ser efetuadas sequencialmente, é dizer para um carro vermelho `X` com requerimentos `f1` e `f2`, as ações: `paint_carX red`, `place_carX_f1`, `place_carX_f2` e `mount_carX` devem aparecer juntas e nessa ordem no plano gerado pelo planejador. Para isso, são introduzidos predicados que garantem que uma vez iniciada a montagem de um carro, nenhum outra ação seja aplicável além da correspondente ao próximo requerimento, e assim até o carro ser montado.
O estado inicial deve conter as informações de que nenhum carro foi montado até esse momento, inicializar o contador de cor, ao limite superior, a última cor usada depende dos carros montados no dia anterior, os contadores de requerimento dependem do último carro que foi gerado no dia anterior, por exemplo, se o carro foi o terceiro começando pelo último do dia anterior, o contador será inicializado em: $P-3$, onde $1/P$ é o rádio para o requerimento. O contador de penalidade é inicializado em 0, assim como o contador de limpezas.
O estado meta é atingido quando todos os carros tenham sido montados, e poderia ser especificado na meta a penalidade máxima permissível também.
As caraterísticas do PDDL que vão ser usadas são:

- precondições negadas,

- efeitos condicionais, mediante `when`,

- variáveis numéricas, para os contadores descritos e,

- métrica de minimização, para especificar que a penalidade deve ser minimizada.

3 Exemplo

O exemplo do domínio será planteado para as seguintes condições, com três carros a serem montados e dois tipos de requerimentos permitidos (`F1` e `F2`):

- Límite superior para limpeza de sprays de pintura: 3,

- carro1: vermelho, requerimento: `F1`,

- carro2: verde, requerimentos: `F1` e `F2`,

- carro3: vermelho, sem requerimentos,

- rádio para `F1` = 1/2,

- rádio para `F2` = 1/2.

Dos carros do día anterior, só o último teve o requerimento `F1`, o requerimento `F2` não apareceu nos últimos carros.

Assim, o arquivo de domínio deve ser:

```
(define (domain Roadef)
(:requirements :strips :typing :fluents
:negative-preconditions :conditional-effects :equality)
(:types color - object)

(:predicates
    (already_mounted_car1)
    (already_mounted_car2)
    (already_mounted_car3)
    (not_mounted_car1)
    (not_mounted_car2)
    (not_mounted_car3)
    (mounting_car1)
    (mounting_car2)
    (mounting_car3)
    (color_car1 ?c - color)
    (color_car2 ?c - color)
    (color_car3 ?c - color)
    (painted_car1)
    (painted_car2)
    (ready_car1)
    (ready_car2)
    (ready_car3)
    (f1applied_car2)
    (last_color ?c - color)
)

(:functions
    (penalty)
    (cleanings)
    (color_counter)
    (f1_counter)
    (f2_counter)
)

(:action paint_car1
:parameters (?c - color)
:precondition (and (not_mounted_car1)
    (not (mounting_car2)) (not (mounting_car3))
    (color_car1 ?c))
:effect
    (and (last_color ?c)
        (mounting_car1) (not (not_mounted_car1))
        (painted_car1)
        (when
            (and (last_color ?c) (> (color_counter) 0 ) )
            (and (decrease (color_counter) 1))
        )
        (when
            (and (last_color ?c) (= (color_counter) 0 ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1) ))
        )
        (when
```

```
            (and (last_color ?x) (not(= (?x) (?c)) ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1)
                (not (last_color ?x)) (last_color ?c)
                (when
                    (> (color_counter) 0)
                    (and ( (increase (penalty) 1)))
                 )
            ))
         )
    )
)

(:action place_car1_f1
:parameters ()
:precondition (and (painted_car1))
:effect
    (and (not (painted_car1)) (ready_car1)
        (when
            (and (> (f1_counter) 0) )
            (and (increase (penalty) (- (2) (f1_counter))))
        )
        (assign (f1_counter) 1)
    )
)

(:action mount_car1
:parameters ()
:precondition (and (ready_car1))
:effect
    (and (decrease (f2_counter) 1)
        (already_mounted_car1)
        (not (ready_car1))
        (not (mounting_car1))
    )
)

(:action paint_car2
:parameters (?c - color)
:precondition (and (not_mounted_car2)
    (not (mounting_car1)) (not (mounting_car3))
    (color_car2 ?c))
:effect
    (and (last_color ?c)
        (mounting_car2) (not (not_mounted_car2))
        (painted_car2)
        (when
            (and (last_color ?c) (> (color_counter) 0 ) )
            (and (decrease (color_counter) 1))
        )
        (when
            (and (last_color ?c) (= (color_counter) 0 ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1) ))
        )
        (when
```

```
            (and (last_color ?x) (not(= (?x) (?c)) ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1)
                (not (last_color ?x)) (last_color ?c)
                (when
                    (> (color_counter) 0)
                    (and ( (increase (penalty) 1)))
                )
             ))
        )
    )
)

(:action place_car2_f1
:parameters ()
:precondition (and (painted_car2))
:effect
    (and (not (painted_car2)) (f1applied_car2)
        (when
            (and (> (f1_counter) 0) )
            (and (increase (penalty) (- (2) (f1_counter))))
        )
        (assign (f1_counter) 1)
    )
)

(:action place_car2_f2
:parameters ()
:precondition (and (f1applied_car2))
:effect
    (and (not (f1applied_car2)) (ready_car2)
       (when
            (and (> (f2_counter) 0) )
            (and (increase (penalty) (- (2) (f2_counter))))
        )
        (assign (f2_counter) 1)
    )
)

(:action mount_car2
:parameters ()
:precondition (and (ready_car2))
:effect
    (and (already_mounted_car2)
        (not (ready_car2))
        (not (mounting_car2))
    )
)

(:action paint_car3
:parameters (?c - color)
:precondition (and (not_mounted_car3)
    (not (mounting_car1)) (not (mounting_car2))
    (color_car3 ?c))
:effect
```

```
    (and (last_color ?c)
        (mounting_car3) (not (not_mounted_car3))
        (ready_car3)
        (when
            (and (last_color ?c) (> (color_counter) 0 ) )
            (and (decrease (color_counter) 1))
         )
         (when
            (and (last_color ?c) (= (color_counter) 0 ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1) ))
         )
         (when
            (and (last_color ?x) (not(= (?x) (?c)) ) )
            (and ( (assign (color_counter) 2) (increase (cleanings) 1)
               (not (last_color ?x)) (last_color ?c)
               (when
                  (> (color_counter) 0)
                  (and ( (increase (penalty) 1)))
               )
            ))
        )
    )
)

(:action mount_car3
:parameters ()
:precondition (and (ready_car3))
:effect
    (and (decrease (f1_counter) 1) (decrease (f2_counter) 1)
        (already_mounted_car3)
        (not (ready_car3))
        (not (mounting_car3))
    )
)

)
```

E o arquivo de problema deve conter:

```
(define (problem exemplo)
(:domain Roadef)
(:objects red - color green - color)
(:init
    (= f1_counter 1)
    (= f2_counter 0)
    (= penalty 0)
    (= color_counter 2)
    (= cleanings 0)
    (not_mounted_car1)
    (not_mounted_car2)
    (not_mounted_car3)
    (color_car1 red)
    (color_car2 green)
    (color_car3 red)
```

```
)
(:goal
    (already_mounted_car1)
    (already_mounted_car2)
    (already_mounted_car3)
)
(:metric minimize penalty)
```