PROBLEMA DE PLANEJAMENTO

TIDYBOT

Integrantes do Grupo

- Arthur Mendonça Arruda - 231033737
- Felipe Hermes Fernandes - 180031767
- Gabriel Lopes de Amorim - 231012129
- Henrique Souza Lavarini - 211061888
- João Gabriel David dos Anjos - 232021688
- Lucas Mendonça Arruda - 231035464
- Samuel Nogueira Caetano - 231027186
- Mariana Ribeiro de Santana Gonzaga - 231026993

Tópicos

Do que se trata o domínio?

O problema de planejamento Tidybot foi um desafio do International Planning Competition (IPC) de 2011. Esse domínio simula uma tarefa de organização doméstica em um ambiente 2D, onde robôs precisam mover objetos para locais-alvo. Equipados com um "gripper" para carregar um item de cada vez e um carrinho para transportar vários itens, os robôs organizam objetos em superfícies (como "mesas") e estruturas em U ("armários"). O domínio testa habilidades de planejamento em espaços amplos, exigindo o uso da estrutura geométrica para otimizar a organização.

Existe artigo científico publicado a respeito do domínio ?

01

"TidyBot: Personalized Robot Assistance with Large Language Models"

Publicado na revista Autonomous Robots em 2023. Esse artigo explora como o Tidybot é utilizado em tarefas de organização doméstica, demonstrando como robôs, equipados com modelos de linguagem de grande porte (LLMs), conseguem interpretar e adaptar-se às preferências de organização dos usuários.
O estudo descreve a arquitetura do Tidybot, seu sistema de controle e o uso de LLMs para entender comandos e realizar tarefas em ambientes complexos. A pesquisa também analisa a capacidade dos robôs de lidar com situações desafiadoras, como restrições geométricas e de acessibilidade, visando otimizar a eficiência na organização de objetos.

https://arxiv.org/abs/2305.05658
https://github.com/jimmyyhwu/tidybot

Types

```
(:types robot cart object xc yc xrel yrel)
```

- robot: representa o robô.
- cart: representa o carrinho que o robô pode mover.
- object: representa os objetos que o robô organiza.
- xc, yc: coordenadas x e y no espaço absoluto.
- xrel, yrel: coordenadas relativas ao robô, para movimentação precisa.

PREDICATES

Constant

```
;; Constant preds
(leftof                  ?x1 - xc ?x2 - xc)
(above                   ?y1 - yc ?y2 - yc)
(leftof-rel              ?x1 - xrel ?x2 - xrel)
(above-rel               ?y1 - yrel ?y2 - yrel)
(sum-x        ?x - xc ?xr - xrel ?xsum - xc)
(sum-y        ?y - yc ?yr - yrel ?ysum - yc)
(zerox-rel    ?x - xrel)
(zeroy-rel    ?y - yrel)
(object-goal  ?o - object ?x - xc  ?y - yc)
```

- **(leftof ?x1 - xc ?x2 - xc)**
  - Define que a posição ?x1 está à esquerda da posição ?x2 no eixo X em coordenadas absolutas.
- **(above ?y1 - yc ?y2 - yc)**
  - Define que a posição ?y1 está acima da posição ?y2 no eixo Y em coordenadas absolutas.
- **(leftof-rel ?x1 - xrel ?x2 - xrel)**
  - Define que a posição relativa ?x1 está à esquerda de ?x2 em relação ao robô.
- **(above-rel ?y1 - yrel ?y2 - yrel)**
  - Define que a posição relativa ?y1 está acima de ?y2 em relação ao robô.
- **(sum-x ?x - xc ?xr - xrel ?xsum - xc)**
  - Calcula uma posição absoluta ?xsum somando uma coordenada absoluta ?x e uma coordenada relativa ?xr.
- **(sum-y ?y - yc ?yr - yrel ?ysum - yc)**
  - Similar ao sum-x, mas no eixo Y. Calcula a posição absoluta ?ysum somando uma coordenada absoluta ?y com uma coordenada relativa ?yr.
- **(zerox-rel ?x - xrel)**
  - Indica que ?x é zero em relação ao xrel, ou seja, não há deslocamento horizontal do gripper em relação ao robô.
- **(zeroy-rel ?y - yrel)**
  - Indica que ?y é zero em relação ao xrel, ou seja, não há deslocamento vertical do gripper em relação ao robô.
- **(object-goal ?o - object ?x - xc ?y - yc)**
  - Define a posição de destino de um objeto ?o em coordenadas absolutas (?x, ?y), representando a posição alvo onde o objeto deve ser colocado.

PREDICATES

Base

```
;; Robot base
(parked       ?r - robot)
(base-pos     ?r - robot  ?x - xc  ?y - yc)
(base-obstacle            ?x - xc  ?y - yc)
```

- **(parked ?r - robot)**
  - Indica que o robô ?r está estacionado, ou seja, parado e não em movimento.
- **(base-pos ?r - robot ?x - xc ?y - yc)**
  - Define a posição atual do robô ?r em coordenadas absolutas (?x, ?y).
- **(base-obstacle ?x - xc ?y - yc)**
  - Indica a presença de um obstáculo em uma posição absoluta (?x, ?y), que o robô deve evitar.

Objetos

```
;; Objects
(object-pos   ?o - object ?x - xc  ?y - yc)
(object-done  ?o - object)
(surface ?x - xc ?y - yc)
```

- **(object-pos ?o - object ?x - xc ?y - yc)**
  - Define a posição atual de um objeto ?o em coordenadas absolutas (?x, ?y).
- **(object-done ?o - object)**
  - Indica que o objeto ?o foi corretamente posicionado em sua posição de destino.
- **(surface ?x - xc ?y - yc)**
  - Define que há uma superfície na posição (x, y) onde objetos podem ser colocados.

PREDICATES

Gripper

```
;; Gripper
(holding       ?r - robot ?o - object)
(gripper-empty ?r - robot)
(gripper-rel   ?r - robot ?x - xrel ?y - yrel)
(gripper-obstacle         ?x - xc   ?y - yc)
```

- **(holding ?r - robot ?o - object)**
  - Indica que o robô ?r está segurando o objeto ?o com sua garra.
- **(gripper-empty ?r - robot)**
  - Indica que o gripper do robô está vazio.
- **(gripper-rel ?r - robot ?x - xrel ?y - yrel)**
  - Define a posição do gripper em coordenadas relativas xrel e yrel em relação ao robô.
- **(gripper-obstacle ?x - xc ?y - yc)**
  - Indica um obstáculo na posição onde o gripper pretende atuar, bloqueando sua movimentação.

Carrinho

```
;; Cart
(pushing       ?r - robot ?c - cart)
(not-pushing   ?r - robot)
(not-pushed    ?c - cart)
(cart-pos      ?c - cart ?x - xc ?y - yc)
(on-cart       ?o - object ?c - cart)
)
```

- **(pushing ?r - robot ?c - cart)**
  - Indica que o robô está empurrando o carrinho ?c.
- **(not-pushing ?r - robot)**
  - Indica que o robô não está empurrando o carrinho.
- **(not-pushed ?c - cart)**
  - Define que o carrinho ?c não está sendo empurrado.
- **(cart-pos ?c - cart ?x - xc ?y - yc)**
  - Define a posição atual do carrinho ?c em coordenadas absolutas.
- **(on-cart ?o - object ?c - cart)**
  - Indica que o objeto ?o está sobre o carrinho ?c.

Ações

O domínio Tidybot possui 30 ações, listadas abaixo com seus nomes traduzidos para o português:

1. desestacionar **(unpark)**
2. estacionar **(park)**
3. mover-base-esquerda **(base-left)**
4. mover-base-direita **(base-right)**
5. mover-base-cima **(base-up)**
6. mover-base-baixo **(base-down)**
7. mover-base-com-carrinho-esquerda **(base-cart-left)**
8. mover-base-com-carrinho-direita **(base-cart-right)**
9. mover-base-com-carrinho-cima **(base-cart-up)**
10. mover-base-com-carrinho-baixo **(base-cart-down)**
11. mover-garra-esquerda **(gripper-left)**
12. mover-garra-direita **(gripper-right)**
13. mover-garra-cima **(gripper-up)**
14. mover-garra-baixo **(gripper-down)**
15. agarrar-carrinho-esquerda **(grasp-cart-left)**

1. agarrar-carrinho-direita **(grasp-cart-right)**
2. agarrar-carrinho-cima **(grasp-cart-above)**
3. agarrar-carrinho-baixo **(grasp-cart-below)**
4. desagarrar-carrinho **(ungrasp-cart)**
5. pegar-esquerda **(get-left)**
6. pegar-direita **(get-right)**
7. pegar-cima **(get-up)**
8. pegar-baixo **(get-down)**
9. pegar-do-carrinho **(get-from-cart)**
10. colocar-esquerda **(put-left)**
11. colocar-direita **(put-right)**
12. colocar-cima **(put-up)**
13. colocar-baixo **(put-down)**
14. colocar-no-carrinho **(put-on-cart)**
15. finalizar-objeto **(finish-object)**

Ações - Base Robô

desestacionar (unpark)

```
;; Base movement actions
(:action unpark
 :parameters (?r - robot ?x - xrel ?y - yrel)
 :precondition (and (parked ?r) (gripper-rel ?r ?x ?y) (zerox-rel ?x) (zeroy-rel ?y))
 :effect       (not (parked ?r))
 )
```

estacionar (park)

```
(:action park
 :parameters (?r - robot)
 :precondition (and (not (parked ?r)) (not-pushing ?r))
 :effect       (parked ?r)
 )
```

Ações - Base Robô

mover-base-esquerda (base-left)

```
(:action base-left
 :parameters (?r - robot ?cx - xc ?dx - xc ?y - yc)
 :precondition (and (not (parked ?r))
                    (not-pushing ?r)
                    (leftof ?dx ?cx)
                    (base-pos ?r ?cx ?y)
                    (not (base-obstacle ?dx ?y)))
 :effect       (and (not (base-pos ?r ?cx ?y)) (base-pos ?r ?dx ?y)
                    (not (base-obstacle ?cx ?y)) (base-obstacle ?dx ?y))
 )
```

mover-base-direita (base-right)

```
(:action base-right
 :parameters (?r - robot ?cx - xc ?dx - xc ?y - yc)
 :precondition (and (not (parked ?r))
                    (not-pushing ?r)
                    (leftof ?cx ?dx)
                    (base-pos ?r ?cx ?y)
                    (not (base-obstacle ?dx ?y)))
 :effect       (and (not (base-pos ?r ?cx ?y)) (base-pos ?r ?dx ?y)
                    (not (base-obstacle ?cx ?y)) (base-obstacle ?dx ?y))
 )
```

Ações - Base Robô

mover-base-cima (base-up)

```
(:action base-up
 :parameters (?r - robot ?x - xc ?cy - yc ?dy - yc)
 :precondition (and (not (parked ?r))
                    (not-pushing ?r)
                    (above ?dy ?cy)
                    (base-pos ?r ?x ?cy)
                    (not (base-obstacle ?x ?dy)))
 :effect       (and (not (base-pos ?r ?x ?cy)) (base-pos ?r ?x ?dy)
                    (not (base-obstacle ?x ?cy)) (base-obstacle ?x ?dy))
 )
```

mover-base-baixo (base-down)

```
(:action base-down
 :parameters (?r - robot ?x - xc ?cy - yc ?dy - yc)
 :precondition (and (not (parked ?r))
                    (not-pushing ?r)
                    (above ?cy ?dy)
                    (base-pos ?r ?x ?cy)
                    (not (base-obstacle ?x ?dy)))
 :effect       (and (not (base-pos ?r ?x ?cy)) (base-pos ?r ?x ?dy)
                    (not (base-obstacle ?x ?cy)) (base-obstacle ?x ?dy))
 )
```

Ações - Base Robô

mover-base-com-carrinho-esquerda (base-cart-left)

```
;; Base movement with cart

(:action base-cart-left
 :parameters (?r - robot ?c - cart ?x1 - xc ?x2 - xc ?y - yc ?cx1 - xc ?cx2 - xc ?cy - yc)
 :precondition (and (pushing ?r ?c) (leftof ?x2 ?x1) (leftof ?cx2 ?cx1)
                    (base-pos ?r ?x1 ?y) (cart-pos ?c ?cx1 ?cy)
                    (not (base-obstacle ?x2 ?y)) (not (base-obstacle ?cx2 ?cy)))
 :effect       (and (not (base-pos ?r ?x1 ?y)) (base-pos ?r ?x2 ?y)
                    (not (cart-pos ?c ?cx1 ?cy)) (cart-pos ?c ?cx2 ?cy)
                    (not (base-obstacle ?x1 ?y)) (base-obstacle ?x2 ?y)
                    (not (base-obstacle ?cx1 ?cy)) (base-obstacle ?cx2 ?cy)))
```

mover-base-com-carrinho-direita (base-cart-right)

```
(:action base-cart-right
 :parameters (?r - robot ?c - cart ?x1 - xc ?x2 - xc ?y - yc ?cx1 - xc ?cx2 - xc ?cy - yc)
 :precondition (and (pushing ?r ?c) (leftof ?x1 ?x2) (leftof ?cx1 ?cx2)
                    (base-pos ?r ?x1 ?y) (cart-pos ?c ?cx1 ?cy)
                    (not (base-obstacle ?x2 ?y)) (not (base-obstacle ?cx2 ?cy)))
 :effect       (and (not (base-pos ?r ?x1 ?y)) (base-pos ?r ?x2 ?y)
                    (not (cart-pos ?c ?cx1 ?cy)) (cart-pos ?c ?cx2 ?cy)
                    (not (base-obstacle ?x1 ?y)) (base-obstacle ?x2 ?y)
                    (not (base-obstacle ?cx1 ?cy)) (base-obstacle ?cx2 ?cy)))
```

Ações - Base Robô

mover-base-com-carrinho-cima (base-cart-up)

```
(:action base-cart-up
 :parameters (?r - robot ?c - cart ?x - xc ?y1 - yc ?y2 - yc ?cx - xc ?cy1 - yc ?cy2 - yc)
 :precondition (and (pushing ?r ?c) (above ?y2 ?y1) (above ?cy2 ?cy1)
                    (base-pos ?r ?x ?y1) (cart-pos ?c ?cx ?cy1)
                    (not (base-obstacle ?x ?y2)) (not (base-obstacle ?cx ?cy2)))
 :effect       (and (not (base-pos ?r ?x ?y1)) (base-pos ?r ?x ?y2)
                    (not (cart-pos ?c ?cx ?cy1)) (cart-pos ?c ?cx ?cy2)
                    (not (base-obstacle ?x ?y1)) (base-obstacle ?x ?y2)
                    (not (base-obstacle ?cx ?cy2)) (base-obstacle ?cx ?cy2)))
```

mover-base-com-carrinho-baixo (base-cart-down)

```
(:action base-cart-down
 :parameters (?r - robot ?c - cart ?x - xc ?y1 - yc ?y2 - yc ?cx - xc ?cy1 - yc ?cy2 - yc)
 :precondition (and (pushing ?r ?c) (above ?y1 ?y2) (above ?cy1 ?cy2)
                    (base-pos ?r ?x ?y1) (cart-pos ?c ?cx ?cy1)
                    (not (base-obstacle ?x ?y2)) (not (base-obstacle ?cx ?cy2)))
 :effect       (and (not (base-pos ?r ?x ?y1)) (base-pos ?r ?x ?y2)
                    (not (cart-pos ?c ?cx ?cy1)) (cart-pos ?c ?cx ?cy2)
                    (not (base-obstacle ?x ?y1)) (base-obstacle ?x ?y2)
                    (not (base-obstacle ?cx ?cy2)) (base-obstacle ?cx ?cy2)))
```

Ações - Gripper

mover-garra-esquerda (gripper-left)

```
;; Gripper movement actions

(:action gripper-left
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?cgxrel - xrel ?dgxrel - xrel ?cgxabs - xc ?dgxabs - xc
              ?gyrel - yrel ?gyabs - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?cgxrel ?gyrel)
                    (leftof-rel ?dgxrel ?cgxrel)
                    (sum-x ?basex ?cgxrel ?cgxabs)
                    (sum-x ?basex ?dgxrel ?dgxabs)
                    (sum-y ?basey ?gyrel  ?gyabs)
                    (not (gripper-obstacle ?dgxabs ?gyabs)))
 :effect       (and (not (gripper-rel ?r ?cgxrel ?gyrel)) (gripper-rel ?r ?dgxrel ?gyrel)
                    (not (gripper-obstacle ?cgxabs ?gyabs)) (gripper-obstacle ?dgxabs ?gyabs))
 )
```

mover-garra-direita (gripper-right)

```
(:action gripper-right
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?cgxrel - xrel ?dgxrel - xrel ?cgxabs - xc ?dgxabs - xc
              ?gyrel - yrel ?gyabs - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?cgxrel ?gyrel)
                    (leftof-rel ?cgxrel ?dgxrel)
                    (sum-x ?basex ?cgxrel ?cgxabs)
                    (sum-x ?basex ?dgxrel ?dgxabs)
                    (sum-y ?basey ?gyrel  ?gyabs)
                    (not (gripper-obstacle ?dgxabs ?gyabs)))
 :effect       (and (not (gripper-rel ?r ?cgxrel ?gyrel)) (gripper-rel ?r ?dgxrel ?gyrel)
                    (not (gripper-obstacle ?cgxabs ?gyabs)) (gripper-obstacle ?dgxabs ?gyabs))
 )
```

Ações - Gripper

mover-garra-cima (gripper-up)

```
(:action gripper-up
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc
              ?cgyrel - yrel ?dgyrel - yrel ?cgyabs - yc ?dgyabs - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?cgyrel)
                    (above-rel ?dgyrel ?cgyrel)
                    (sum-x ?basex ?gxrel  ?gxabs)
                    (sum-y ?basey ?cgyrel ?cgyabs)
                    (sum-y ?basey ?dgyrel ?dgyabs)
                    (not (gripper-obstacle ?gxabs ?dgyabs)))
 :effect       (and (not (gripper-rel ?r ?gxrel ?cgyrel)) (gripper-rel ?r ?gxrel ?dgyrel)
                    (not (gripper-obstacle ?gxabs ?cgyabs)) (gripper-obstacle ?gxabs ?dgyabs))
 )
```

mover-garra-baixo (gripper-down)

```
(:action gripper-down
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc
              ?cgyrel - yrel ?dgyrel - yrel ?cgyabs - yc ?dgyabs - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?cgyrel)
                    (above-rel ?cgyrel ?dgyrel)
                    (sum-x ?basex ?gxrel  ?gxabs)
                    (sum-y ?basey ?cgyrel ?cgyabs)
                    (sum-y ?basey ?dgyrel ?dgyabs)
                    (not (gripper-obstacle ?gxabs ?dgyabs)))
 :effect       (and (not (gripper-rel ?r ?gxrel ?cgyrel)) (gripper-rel ?r ?gxrel ?dgyrel)
                    (not (gripper-obstacle ?gxabs ?cgyabs)) (gripper-obstacle ?gxabs ?dgyabs))
 )
```

Ações - Carrinho

agarrar-carrinho-esquerda (grasp-cart-left)

```
;; Cart grasping/ungrasping
(:action grasp-cart-left
 :parameters (?r - robot ?c - cart ?x - xc ?y - yc ?cx - xc)
 :precondition (and (not (parked ?r)) (not-pushed ?c)
                    (base-pos ?r ?x ?y) (cart-pos ?c ?cx ?y)
                    (leftof ?cx ?x) (not-pushing ?r))
 :effect       (and (pushing ?r ?c) (not (not-pushing ?r)) (not (not-pushed ?c))))
```

agarrar-carrinho-direita (grasp-cart-right)

```
(:action grasp-cart-right
 :parameters (?r - robot ?c - cart ?x - xc ?y - yc ?cx - xc)
 :precondition (and (not (parked ?r)) (not-pushed ?c)
                    (base-pos ?r ?x ?y) (cart-pos ?c ?cx ?y)
                    (leftof ?x ?cx) (not-pushing ?r))
 :effect       (and (pushing ?r ?c) (not (not-pushing ?r)) (not (not-pushed ?c))))
```

agarrar-carrinho-cima (grasp-cart-above)

```
(:action grasp-cart-above
 :parameters (?r - robot ?c - cart ?x - xc ?y - yc ?cy - yc)
 :precondition (and (not (parked ?r)) (not-pushed ?c)
                    (base-pos ?r ?x ?y) (cart-pos ?c ?x ?cy)
                    (above ?cy ?y) (not-pushing ?r))
 :effect       (and (pushing ?r ?c) (not (not-pushing ?r)) (not (not-pushed ?c))))
```

agarrar-carrinho-baixo (grasp-cart-below)

```
(:action grasp-cart-below
 :parameters (?r - robot ?c - cart ?x - xc ?y - yc ?cy - yc)
 :precondition (and (not (parked ?r)) (not-pushed ?c)
                    (base-pos ?r ?x ?y) (cart-pos ?c ?x ?cy)
                    (above ?y ?cy) (not-pushing ?r))
 :effect       (and (pushing ?r ?c) (not (not-pushing ?r)) (not (not-pushed ?c))))
```

Ações - Gripper

desagarrar-carrinho (ungrasp-cart)

```
(:action ungrasp-cart
 :parameters (?r - robot ?c - cart )
 :precondition (and (pushing ?r ?c))
 :effect (and (not (pushing ?r ?c)) (not-pushing ?r) (not-pushed ?c)))
```

Ações - Objetos

pegar-esquerda (get-left)

```
;; Object manipulation actions


(:action get-left
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?ox - xc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (gripper-empty ?r)
                    (leftof ?ox ?gxabs)
                    (not (object-done ?o)) (object-pos ?o ?ox ?gyabs))
 :effect       (and (not (object-pos ?o ?ox ?gyabs))
                    (not (gripper-obstacle ?ox ?gyabs))
                    (not (gripper-empty ?r))
                    (holding ?r ?o))
 )
```

pegar-direita (get-right)

```
(:action get-right
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?ox - xc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (gripper-empty ?r)
                    (leftof ?gxabs ?ox)
                    (not (object-done ?o)) (object-pos ?o ?ox ?gyabs))
 :effect       (and (not (object-pos ?o ?ox ?gyabs))
                    (not (gripper-obstacle ?ox ?gyabs))
                    (not (gripper-empty ?r))
                    (holding ?r ?o))
 )
```

Ações - Objetos

pegar-cima (get-up)

```
(:action get-up
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?oy - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (gripper-empty ?r)
                    (above ?oy ?gyabs)
                    (not (object-done ?o)) (object-pos ?o ?gxabs ?oy))
 :effect       (and (not (object-pos ?o ?gxabs ?oy))
                    (not (gripper-obstacle ?gxabs ?oy))
                    (not (gripper-empty ?r))
                    (holding ?r ?o))
 )
```

pegar-baixo (get-down)

```
(:action get-down
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?oy - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (gripper-empty ?r)
                    (above ?gyabs ?oy)
                    (not (object-done ?o)) (object-pos ?o ?gxabs ?oy))
 :effect       (and (not (object-pos ?o ?gxabs ?oy))
                    (not (gripper-obstacle ?gxabs ?oy))
                    (not (gripper-empty ?r))
                    (holding ?r ?o))
 )
```

Ações - Objetos

pegar-do-carrinho (get-from-cart)

```
(:action get-from-cart
 :parameters (?r - robot ?x - xc ?y - yc ?gxrel - xrel
              ?gyrel - yrel ?o - object ?c - cart
              ?cx - xc ?cy - yc)
 :precondition (and (parked ?r) (base-pos ?r ?x ?y)
                    (gripper-rel ?r ?gxrel ?gyrel) (sum-x ?x ?gxrel ?cx)
                    (sum-y ?y ?gyrel ?cy) (gripper-empty ?r) (cart-pos ?c ?cx ?cy)
                    (on-cart ?o ?c))
 :effect       (and (holding ?r ?o) (not (gripper-empty ?r)) (not (on-cart ?o ?c))))
```

colocar-esquerda (put-left)

```
(:action put-left
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?ox - xc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (holding ?r ?o)
                    (leftof ?ox ?gxabs)
                    (not (gripper-obstacle ?ox ?gyabs))
                    (surface ?ox ?gyabs)
                    )
 :effect       (and (not (holding ?r ?o))
                    (object-pos ?o ?ox ?gyabs)
                    (gripper-obstacle ?ox ?gyabs)
                    (gripper-empty ?r)
                    )
 )
```

Ações - Objetos

colocar-direita (put-right)

```
(:action put-right
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?ox - xc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (holding ?r ?o)
                    (leftof ?gxabs ?ox)
                    (not (gripper-obstacle ?ox ?gyabs))
                    (surface ?ox ?gyabs)
                    )
 :effect       (and (not (holding ?r ?o))
                    (object-pos ?o ?ox ?gyabs)
                    (gripper-obstacle ?ox ?gyabs)
                    (gripper-empty ?r)
                    )
 )
```

colocar-cima (put-up)

```
(:action put-up
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?oy - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (holding ?r ?o)
                    (above ?oy ?gyabs)
                    (not (gripper-obstacle ?gxabs ?oy))
                    (surface ?gxabs ?oy)
                    )
 :effect       (and (not (holding ?r ?o))
                    (object-pos ?o ?gxabs ?oy)
                    (gripper-obstacle ?gxabs ?oy)
                    (gripper-empty ?r)
                    )
 )
```

Ações - Objetos

colocar-baixo (put-down)

```
(:action put-down
 :parameters (?r - robot ?basex - xc ?basey - yc
              ?gxrel - xrel ?gxabs - xc ?gyrel - yrel ?gyabs - yc
              ?o - object ?oy - yc)
 :precondition (and (parked ?r)
                    (base-pos ?r ?basex ?basey)
                    (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?basex ?gxrel ?gxabs)
                    (sum-y ?basey ?gyrel ?gyabs)
                    (holding ?r ?o)
                    (above ?gyabs ?oy)
                    (not (gripper-obstacle ?gxabs ?oy))
                    (surface ?gxabs ?oy)
                    )
 :effect       (and (not (holding ?r ?o))
                    (object-pos ?o ?gxabs ?oy)
                    (gripper-obstacle ?gxabs ?oy)
                    (gripper-empty ?r)
                    )
)
```

colocar-no-carrinho (put-on-cart)

```
(:action put-on-cart
 :parameters (?r - robot ?x - xc ?y - yc ?gxrel - xrel
              ?gyrel - yrel ?o - object ?c - cart ?cx - xc ?cy - yc)

 :precondition (and (parked ?r) (base-pos ?r ?x ?y) (gripper-rel ?r ?gxrel ?gyrel)
                    (sum-x ?x ?gxrel ?cx) (sum-y ?y ?gyrel ?cy) (cart-pos ?c ?cx ?cy)
                    (holding ?r ?o))

 :effect (and (not (holding ?r ?o)) (on-cart ?o ?c) (gripper-empty ?r)))
```

Ações - Objetos

finalizar-objeto (finish-object)

```
  (:action finish-object
   :parameters (?o - object ?x - xc ?y - yc)
   :precondition (and (not (object-done ?o)) (object-pos ?o ?x ?y) (object-goal ?o ?x ?y))
   :effect (and (object-done ?o)))
)
```

Problema Resolvido

- Intance1.pddl

Instance1.pddl

Objects

```
(define
  (problem test)
  (:domain TIDYBOT)

  (:objects
   pr2 - robot
   cart - cart
   object0 - object
   object1 - object
   object2 - object
   object3 - object
   x0 - xc
   x1 - xc
   x2 - xc
   x3 - xc
   x4 - xc
   x5 - xc
   x6 - xc
   x7 - xc
   x8 - xc
   y0 - yc
   y1 - yc
   y2 - yc
   y3 - yc
   y4 - yc
   y5 - yc
   y6 - yc
   y7 - yc
   y8 - yc
   xrel-1 - xrel
   xrel0 - xrel
   xrel1 - xrel
   yrel-1 - yrel
   yrel0 - yrel
   yrel1 - yrel
  )
```

Instance1.pddl

Init

```
(:init
 (leftof x0 x1)
 (leftof x1 x2)
 (leftof x2 x3)
 (leftof x3 x4)
 (leftof x4 x5)
 (leftof x5 x6)
 (leftof x6 x7)
 (leftof x7 x8)
 (above y0 y1)
 (above y1 y2)
 (above y2 y3)
 (above y3 y4)
 (above y4 y5)
 (above y5 y6)
 (above y6 y7)
 (above y7 y8)
 (leftof-rel xrel-1 xrel0)
 (leftof-rel xrel0 xrel1)
 (above-rel yrel-1 yrel0)
 (above-rel yrel0 yrel1)
 (sum-x x0 xrel0 x0)
 (sum-x x0 xrel1 x1)
 (sum-x x1 xrel-1 x0)
 (sum-x x1 xrel0 x1)
 (sum-x x1 xrel1 x2)
 (sum-x x2 xrel-1 x1)
 (sum-x x2 xrel0 x2)
 (sum-x x2 xrel1 x3)
 (sum-x x3 xrel-1 x2)
 (sum-x x3 xrel0 x3)
 (sum-x x3 xrel1 x4)
```

```
(sum-x x4 xrel1 x5)
(sum-x x5 xrel-1 x4)
(sum-x x5 xrel0 x5)
(sum-x x5 xrel1 x6)
(sum-x x6 xrel-1 x5)
(sum-x x6 xrel0 x6)
(sum-x x6 xrel1 x7)
(sum-x x7 xrel-1 x6)
(sum-x x7 xrel0 x7)
(sum-x x7 xrel1 x8)
(sum-x x8 xrel-1 x7)
(sum-x x8 xrel0 x8)
(sum-y y0 yrel0 y0)
(sum-y y0 yrel1 y1)
(sum-y y1 yrel-1 y0)
(sum-y y1 yrel0 y1)
(sum-y y1 yrel1 y2)
(sum-y y2 yrel-1 y1)
(sum-y y2 yrel0 y2)
(sum-y y2 yrel1 y3)
(sum-y y3 yrel-1 y2)
(sum-y y3 yrel0 y3)
(sum-y y3 yrel1 y4)
(sum-y y4 yrel-1 y3)
(sum-y y4 yrel0 y4)
(sum-y y4 yrel1 y5)
(sum-y y5 yrel-1 y4)
(sum-y y5 yrel0 y5)
(sum-y y5 yrel1 y6)
(sum-y y6 yrel-1 y5)
```

```
(sum-y y4 yrel1 y5)
(sum-y y5 yrel-1 y4)
(sum-y y5 yrel0 y5)
(sum-y y5 yrel1 y6)
(sum-y y6 yrel-1 y5)
(sum-y y6 yrel0 y6)
(sum-y y6 yrel1 y7)
(sum-y y7 yrel-1 y6)
(sum-y y7 yrel0 y7)
(sum-y y7 yrel1 y8)
(sum-y y8 yrel-1 y7)
(sum-y y8 yrel0 y8)
(zerox-rel xrel0)
(zeroy-rel yrel0)
(object-goal object0 x4 y4)
(object-goal object1 x4 y5)
(object-goal object1 x1 y3)
(object-goal object2 x5 y4)
(object-goal object2 x1 y5)
(object-goal object3 x5 y5)
```

```
 (parked pr2)
 (not-pushing pr2)
 (base-pos pr2 x0 y0)
 (base-obstacle x0 y0)
 (base-obstacle x1 y3)(surface x1 y3)
 (base-obstacle x3 y1)(surface x3 y1)
 (base-obstacle x5 y1)(surface x5 y1)
 (base-obstacle x6 y1)(surface x6 y1)
 (base-obstacle x1 y5)(surface x1 y5)
 (base-obstacle x3 y3)(gripper-obstacle x3 y3)
 (base-obstacle x3 y4)(gripper-obstacle x3 y4)
 (base-obstacle x3 y5)(gripper-obstacle x3 y5)
 (base-obstacle x3 y6)(gripper-obstacle x3 y6)
 (base-obstacle x4 y3)(gripper-obstacle x4 y3)
 (base-obstacle x5 y3)(gripper-obstacle x5 y3)
 (base-obstacle x6 y3)(gripper-obstacle x6 y3)
 (base-obstacle x6 y4)(gripper-obstacle x6 y4)
 (base-obstacle x6 y5)(gripper-obstacle x6 y5)
 (base-obstacle x6 y6)(gripper-obstacle x6 y6)
 (surface x4 y4)
 (surface x4 y5)
 (surface x5 y4)
 (surface x5 y5)

 (cart-pos cart x0 y1)
 (not-pushed cart)
 (base-obstacle x0 y1)

 (object-pos object0 x5 y5)
 (object-pos object1 x5 y1)
 (object-pos object2 x3 y1)
 (object-pos object3 x1 y3)

 (gripper-empty pr2)
 (gripper-rel pr2 xrel0 yrel0)
 (gripper-obstacle x5 y5)
 (gripper-obstacle x5 y1)
 (gripper-obstacle x3 y1)
 (gripper-obstacle x1 y3)
)
```

Instance1.pddl

Goal

```
(:goal
 (and
 (object-done object0)
 (object-done object1)
 (object-done object2)
 (object-done object3)
))
```

Resultados

seq-agl-greedy-2023

(unpark pr2 xrel0 yrel0)
O robô PR2 sai da posição de estacionamento, ficando pronto para se mover.
(base-right pr2 x0 x1 y0)
PR2 se move uma célula para a direita de (x0, y0) para (x1, y0).
(base-right pr2 x1 x2 y0)
PR2 se move mais uma célula para a direita, agora de (x1, y0) para (x2, y0).
(park pr2)
PR2 estaciona temporariamente na posição atual (x2, y0).
(gripper-right pr2 x2 y0 xrel0 xrel1 x2 x3 yrel0 y0)
PR2 move o gripper (pinça) para a direita, estendendo-o de (x2, y0) para (x3, y0) para se aproximar do objeto.
(get-down pr2 x2 y0 xrel1 x3 yrel0 y0 object2 y1)
PR2 usa o gripper para pegar object2, que está localizado na posição (x3, y1).
(gripper-left pr2 x2 y0 xrel1 xrel0 x3 x2 yrel0 y0)
PR2 retrai o gripper para a posição original à esquerda, voltando para (x2, y0).
(unpark pr2 xrel0 yrel0)
PR2 sai novamente do modo de estacionamento e fica pronto para se mover.
(base-down pr2 x2 y0 y1)
PR2 move-se para baixo de (x2, y0) para (x2, y1).
(base-down pr2 x2 y1 y2)
PR2 continua o movimento para baixo, agora de (x2, y1) para (x2, y2).
(base-down pr2 x2 y2 y3)
PR2 continua descendo, indo de (x2, y2) para (x2, y3).
(base-down pr2 x2 y3 y4)
PR2 move-se mais uma vez para baixo, alcançando (x2, y4).

Resultados

seq-agl-greedy-2023

(park pr2)
PR2 estaciona temporariamente na posição atual (x2, y4).
(gripper-left pr2 x2 y4 xrel0 xrel-1 x2 x1 yrel0 y4)
PR2 move o gripper para a esquerda, estendendo-o de (x2, y4) para (x1, y4).
(put-down pr2 x2 y4 xrel-1 x1 yrel0 y4 object2 y5)
PR2 coloca object2 na posição (x1, y5), deixando o objeto no chão.
(finish-object object2 x1 y5)
A tarefa com object2 é concluída na posição (x1, y5).
(get-up pr2 x2 y4 xrel-1 x1 yrel0 y4 object3 y3)
PR2 estende o gripper para cima e agarra object3, que está em (x1, y3).
(gripper-right pr2 x2 y4 xrel-1 xrel0 x1 x2 yrel0 y4)
PR2 move o gripper de volta para a direita, retornando a (x2, y4).
(unpark pr2 xrel0 yrel0)
PR2 sai do modo de estacionamento e fica pronto para novos movimentos.
(base-up pr2 x2 y4 y3)
PR2 sobe uma posição de (x2, y4) para (x2, y3).
(base-up pr2 x2 y3 y2)
PR2 continua subindo de (x2, y3) para (x2, y2).
(base-right pr2 x2 x3 y2)
PR2 se move para a direita, de (x2, y2) para (x3, y2).

(park pr2)
PR2 estaciona temporariamente em (x3, y2).
(gripper-right pr2 x3 y2 xrel0 xrel1 x3 x4 yrel0 y2)
PR2 estende o gripper para a direita, alcançando (x4, y2).
(gripper-up pr2 x3 y2 xrel1 x4 yrel0 yrel-1 y2 y1)
PR2 levanta o gripper na direção de (x4, y1).
(put-left pr2 x3 y2 xrel1 x4 yrel-1 y1 object3 x3)
PR2 coloca object3 na posição (x3, y1).
(get-right pr2 x3 y2 xrel1 x4 yrel-1 y1 object1 x5)
PR2 move o gripper para a direita e agarra object1, que está localizado em (x4, y1).
(gripper-down pr2 x3 y2 xrel1 x4 yrel-1 yrel0 y1 y2)
PR2 move o gripper para baixo, retornando para (x4, y2).
(gripper-left pr2 x3 y2 xrel1 xrel0 x4 x3 yrel0 y2)
PR2 retrai o gripper para a esquerda, voltando para (x3, y2).
(gripper-left pr2 x3 y2 xrel0 xrel-1 x3 x2 yrel0 y2)
PR2 move o gripper mais uma posição à esquerda, para (x2, y2).

Resultados

seq-agl-greedy-2023

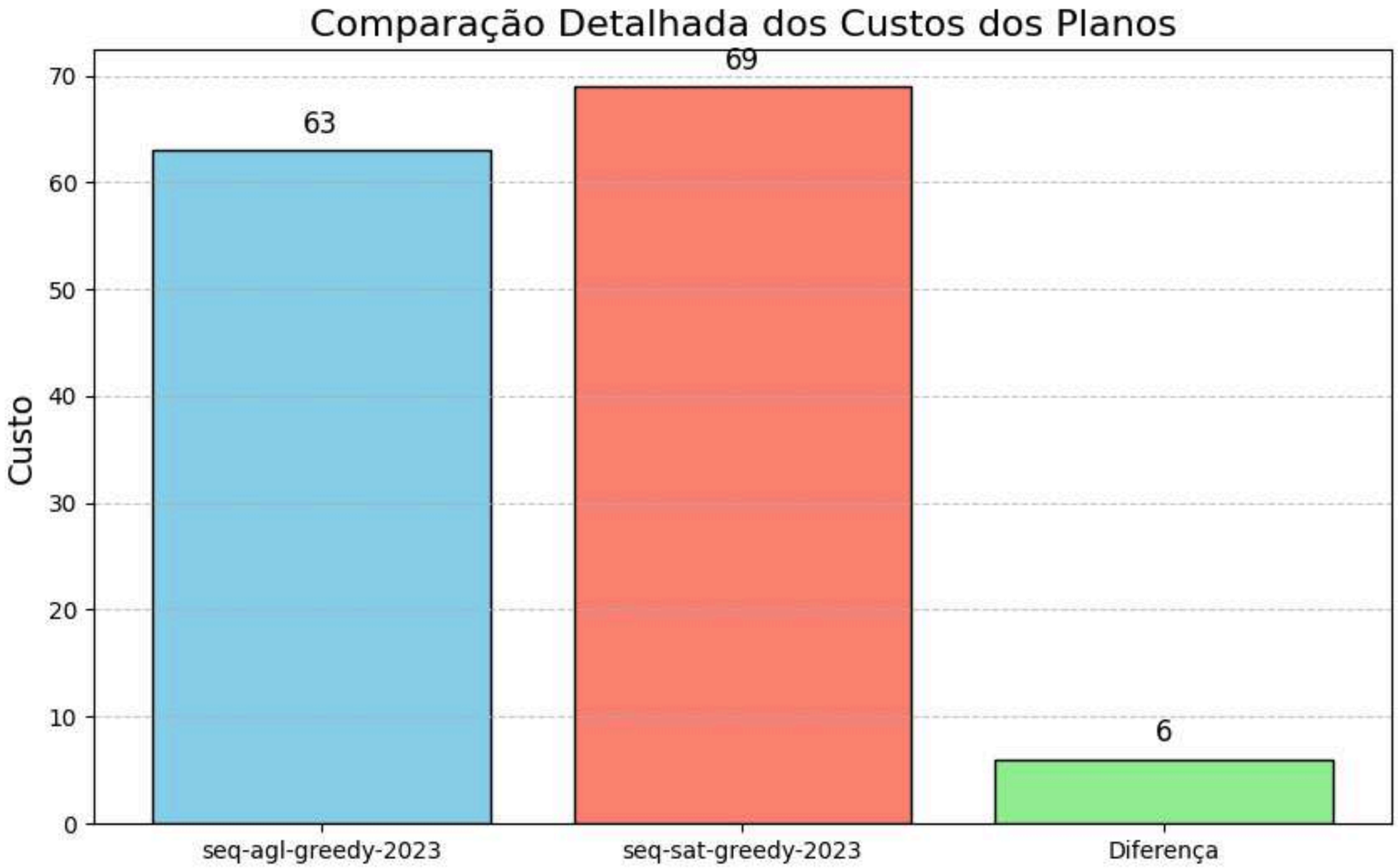
(gripper-down pr2 x3 y2 xrel-1 x2 yrel0 yrel1 y2 y3)
PR2 move o gripper para baixo, de (x2, y2) para (x2, y3).
(put-left pr2 x3 y2 xrel-1 x2 yrel1 y3 object1 x1)
PR2 coloca object1 na posição (x1, y3).
(finish-object object1 x1 y3)
A tarefa com object1 é concluída na posição (x1, y3).
(gripper-up pr2 x3 y2 xrel-1 x2 yrel1 yrel0 y3 y2)
PR2 levanta o gripper de (x2, y3) para (x2, y2).
(gripper-right pr2 x3 y2 xrel-1 xrel0 x2 x3 yrel0 y2)
PR2 move o gripper para a direita, voltando para (x3, y2).
(get-up pr2 x3 y2 xrel0 x3 yrel0 y2 object3 y1)
PR2 estende o gripper para cima e pega object3 em (x3, y1).
(unpark pr2 xrel0 yrel0)
PR2 sai do modo de estacionamento para novos movimentos.
(base-left pr2 x3 x2 y2)
PR2 se move para a esquerda, de (x3, y2) para (x2, y2).
(base-down pr2 x2 y2 y3)
PR2 se move para baixo, de (x2, y2) para (x2, y3).
(base-down pr2 x2 y3 y4)
PR2 continua descendo de (x2, y3) para (x2, y4).
(base-down pr2 x2 y4 y5)
PR2 continua descendo de (x2, y4) para (x2, y5).
(base-down pr2 x2 y5 y6)
PR2 continua descendo de (x2, y5) para (x2, y6).
(base-down pr2 x2 y6 y7)
PR2 continua descendo de (x2, y6) para (x2, y7).
(base-right pr2 x2 x3 y7)
PR2 se move para a direita, de (x2, y7) para (x3, y7).
(base-right pr2 x3 x4 y7)
PR2 se move mais uma vez para a direita, para (x4, y7).
(park pr2)
PR2 estaciona na posição (x4, y7).
(gripper-up pr2 x4 y7 xrel0 x4 yrel0 yrel-1 y7 y6)
PR2 levanta o gripper de (x4, y7) para (x4, y6).
(put-up pr2 x4 y7 xrel0 x4 yrel-1 y6 object3 y5)
PR2 coloca object3 na posição (x4, y5).
(gripper-right pr2 x4 y7 xrel0 xrel1 x4 x5 yrel-1 y6)
PR2 move o gripper para a direita de (x4, y6) para (x5, y6).
(get-up pr2 x4 y7 xrel1 x5 yrel-1 y6 object0 y5)
PR2 levanta o gripper para pegar object0 na posição (x5, y5).
(gripper-down pr2 x4 y7 xrel1 x5 yrel-1 yrel0 y6 y7)
PR2 abaixa o gripper de (x5, y6) para (x5, y7).
(gripper-left pr2 x4 y7 xrel1 xrel0 x5 x4 yrel0 y7)
PR2 move o gripper para a esquerda, retornando para (x4, y7).
(unpark pr2 xrel0 yrel0)
PR2 sai do modo de estacionamento.
(base-up pr2 x4 y7 y6)
PR2 sobe uma posição de (x4, y7) para (x4, y6).
(base-up pr2 x4 y6 y5)
PR2 sobe novamente, indo para (x4, y5).
(park pr2)
PR2 estaciona em (x4, y5).
(put-up pr2 x4 y5 xrel0 x4 yrel0 y5 object0 y4)
PR2 coloca object0 na posição (x4, y4).
(finish-object object0 x4 y4)
A tarefa com object0 é concluída na posição (x4, y4).
(gripper-down pr2 x4 y5 xrel0 x4 yrel0 yrel1 y5 y6)
PR2 abaixa o gripper de (x4, y5) para (x4, y6).
(get-up pr2 x4 y5 xrel0 x4 yrel1 y6 object3 y5)
PR2 levanta o gripper para pegar object3 em (x4, y5).
(gripper-right pr2 x4 y5 xrel0 xrel1 x4 x5 yrel1 y6)
PR2 move o gripper para a direita de (x4, y6) para (x5, y6).
(put-up pr2 x4 y5 xrel1 x5 yrel1 y6 object3 y5)
PR2 coloca object3 na posição (x5, y5).
(finish-object object3 x5 y5)
A tarefa com object3 é concluída na posição (x5, y5).

Resultados

seq-sat-greedy-2023

(unpark pr2 xrel0 yrel0)
(base-right pr2 x0 x1 y0)
(base-right pr2 x1 x2 y0)
(base-down pr2 x2 y0 y1)
(base-down pr2 x2 y1 y2)
(base-down pr2 x2 y2 y3)
(park pr2)
(get-left pr2 x2 y3 xrel0 x2 yrel0 y3 object3 x1)
(unpark pr2 xrel0 yrel0)
(base-down pr2 x2 y3 y4)
(base-down pr2 x2 y4 y5)
(base-down pr2 x2 y5 y6)
(base-down pr2 x2 y6 y7)
(base-right pr2 x2 x3 y7)
(base-right pr2 x3 x4 y7)
(base-up pr2 x4 y7 y6)
(base-up pr2 x4 y6 y5)
(base-right pr2 x4 x5 y5)
(park pr2)
(put-left pr2 x5 y5 xrel0 x5 yrel0 y5 object3 x4)
(unpark pr2 xrel0 yrel0)
(base-left pr2 x5 x4 y5)
(park pr2)
(get-right pr2 x4 y5 xrel0 x4 yrel0 y5 object0 x5)
(put-up pr2 x4 y5 xrel0 x4 yrel0 y5 object0 y4)
(finish-object object0 x4 y4)
(gripper-right pr2 x4 y5 xrel0 xrel1 x4 x5 yrel0 y5)
(get-left pr2 x4 y5 xrel1 x5 yrel0 y5 object3 x4)
(gripper-down pr2 x4 y5 xrel1 x5 yrel0 yrel1 y5 y6)
(put-up pr2 x4 y5 xrel1 x5 yrel1 y6 object3 y5)
(finish-object object3 x5 y5)
(gripper-left pr2 x4 y5 xrel1 xrel0 x5 x4 yrel1 y6)
(gripper-up pr2 x4 y5 xrel0 x4 yrel1 yrel0 y6 y5)
(unpark pr2 xrel0 yrel0)
(base-down pr2 x4 y5 y6)
(base-down pr2 x4 y6 y7)
(base-left pr2 x4 x3 y7)
(base-left pr2 x3 x2 y7)
(base-up pr2 x2 y7 y6)
(base-up pr2 x2 y6 y5)
(base-up pr2 x2 y5 y4)
(base-up pr2 x2 y4 y3)
(base-up pr2 x2 y3 y2)
(base-right pr2 x2 x3 y2)
(park pr2)
(get-up pr2 x3 y2 xrel0 x3 yrel0 y2 object2 y1)
(unpark pr2 xrel0 yrel0)
(base-left pr2 x3 x2 y2)
(base-down pr2 x2 y2 y3)
(base-down pr2 x2 y3 y4)
(base-left pr2 x2 x1 y4)
(park pr2)
(put-down pr2 x1 y4 xrel0 x1 yrel0 y4 object2 y5)
(unpark pr2 xrel0 yrel0)
(finish-object object2 x1 y5)
(base-right pr2 x1 x2 y4)
(base-up pr2 x2 y4 y3)
(base-up pr2 x2 y3 y2)
(base-right pr2 x2 x3 y2)
(park pr2)
(gripper-right pr2 x3 y2 xrel0 xrel1 x3 x4 yrel0 y2)
(gripper-up pr2 x3 y2 xrel1 x4 yrel0 yrel-1 y2 y1)
(get-right pr2 x3 y2 xrel1 x4 yrel-1 y1 object1 x5)
(gripper-left pr2 x3 y2 xrel1 xrel0 x4 x3 yrel-1 y1)
(gripper-down pr2 x3 y2 xrel0 x3 yrel-1 yrel0 y1 y2)
(gripper-left pr2 x3 y2 xrel0 xrel-1 x3 x2 yrel0 y2)
(gripper-down pr2 x3 y2 xrel-1 x2 yrel0 yrel1 y2 y3)
(put-left pr2 x3 y2 xrel-1 x2 yrel1 y3 object1 x1)
(finish-object object1 x1 y3)
; cost = 69 (unit cost)

Gráfico


Comparação Detalhada dos Custos dos Planos
Custo
0
10
20
30
40
50
60
70
63
69
6
seq-agl-greedy-2023
seq-sat-greedy-2023
Diferença