Who said we need to relax » all variables? »

Written by: Michael Katz, Jörg Hoffmann, and Carmel Domshlak
Presented by: Wyatt Scott

Table of contents

01 »

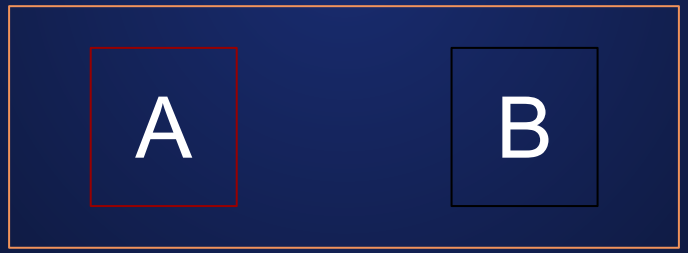
The Problem and Solution

The problem

- Delete relaxation has significant pitfalls in many important classes of planning domains
- It has been challenging to only take some deletes into account for a heuristic

The solution

- We will define red and black variables
  - Red variables take the relaxed semantics in which they accumulate their values rather than switching between them
  - Black variables take the regular semantics
- This is an interpolation between relaxed planning and regular planning



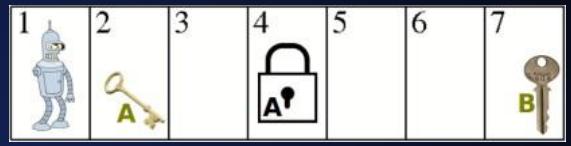
02 »

Context

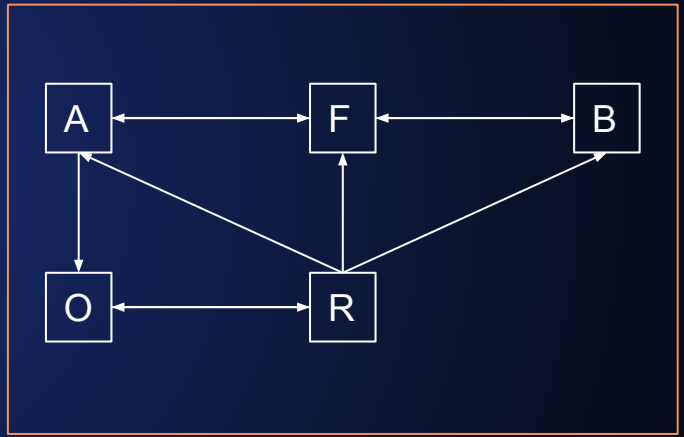
Finite Domain Representation

- A FDR planning task is defined by the following
  - A set of state variables with a finite domain
  - An initial state
  - A goal state
  - A finite set of actions
    - Each action is a pair of preconditions and effects
- PSPACE-Complete
- Generates an accurate plan

Finite Domain Example



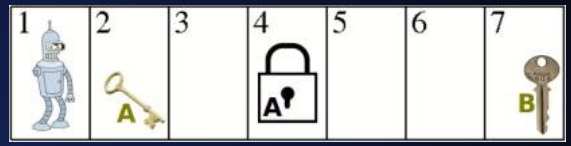
- Goal: Get key B to position 1
- Steps
  - Move to position 2
  - Take key A
  - Move to position 3
  - Open the lock
  - Move to position 7
  - Drop key A
  - Take key B
  - Move to position 1
  - Drop key B



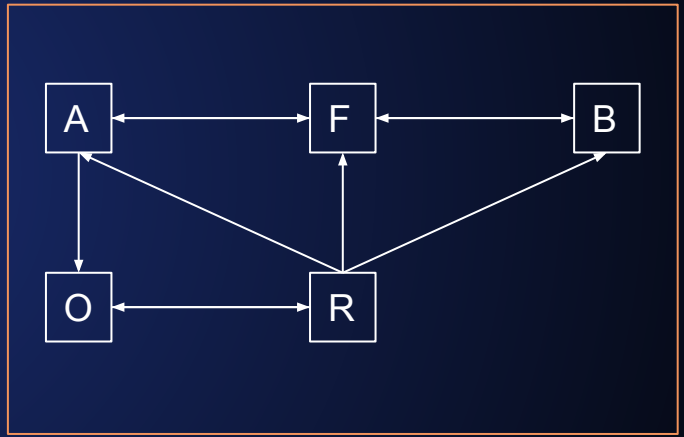
Monotonic Finite Domain Representation

- Polynomial-Time
- Generates a pseudo-plan that ignores variable dependencies
  - A good approximation, but probably incorrect
- Represented by the same items as for FDR
  - Variables, Actions, Initial State, and Goal State
- Instead of each variable only having one value at a time, it accumulates all possible values
- If R is set to 1 and then set to 2, it is now equal to both 1 and 2
  - Similar to how variables persist in Graph Plan
- HSP and FF take advantage of monotonic relaxation
  - This allows them to get a very good heuristic

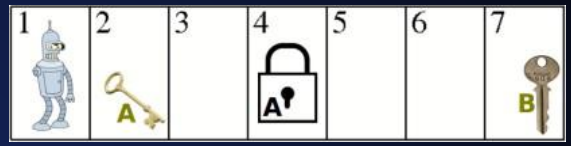
Monotonic Finite Domain Example



- Goal: Get key B to position 1
- Steps
  - Move to position 2
  - Take key A
  - Move to position 3
  - Open the lock
  - Move to position 7
  - Take key B
    - (without dropping key A)
  - Drop key B at position 1
    - (without moving to position 1)

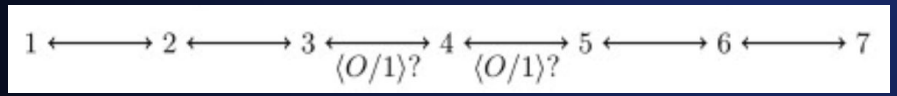


Monotonic Finite Domain Example

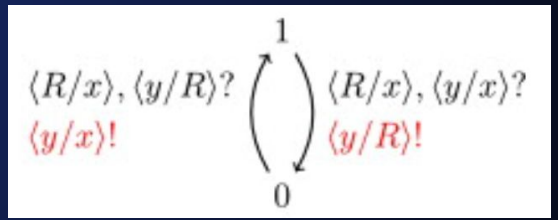


- Internal conditions still need to be met in the domain transition graphs

Robot State



Robot Hand State



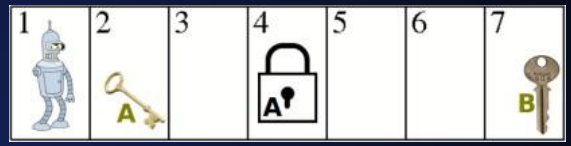
03 »

Red-Black planning

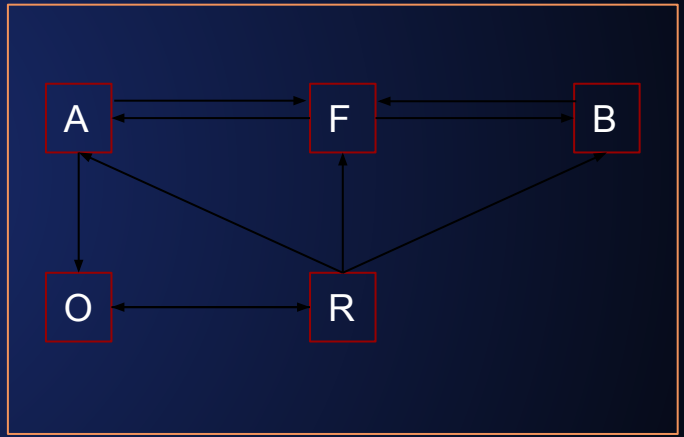
Red-Black Relaxation

- A mix of FDR and MFDR
- Only some state variables are relaxed instead of all or none
- Plan generation for red-black tasks using a fixed number of black variables with a fixed-size domain is polynomial time
  - This becomes exponential when any number of black variables are considered

Red-Black Plan Algorithm

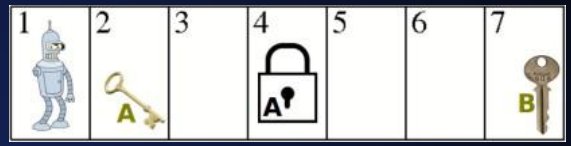


- The red-black plan generation starts with all variables set to red
- If it can't find a plan now, then no plan exists
- If a plan is found, determine if it is valid
- If not, determine which variables need to be black
- Repeat until a valid plan is found

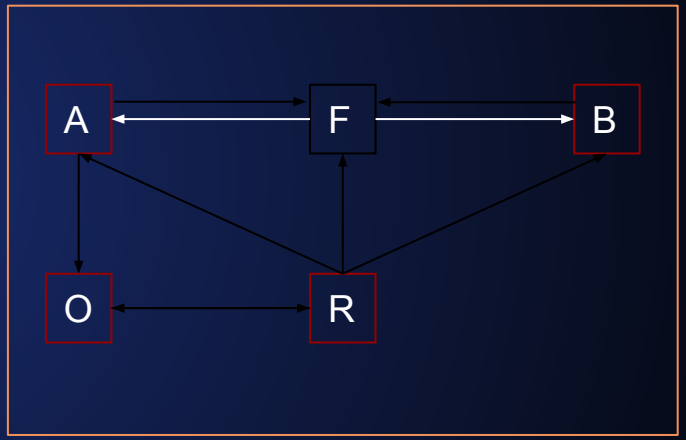


*Black arrows indicate inactive connections
*White arrows indicate active connections

Red-Black Example

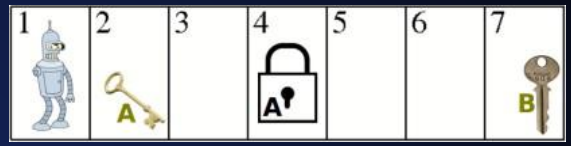


- In our example, F is first set to black
- With F set to black, the robot will need to drop key A before picking up key B

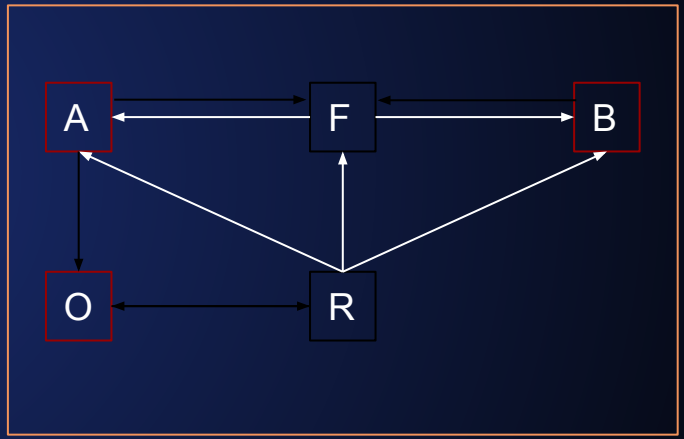


*Black arrows indicate inactive connections
*White arrows indicate active connections

Red-Black Example

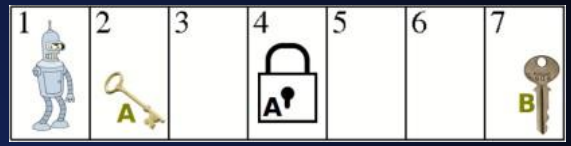


- Since only setting F didn't work, set R to black as well
- With R set to black, the robot will need to move back to position 1 before dropping off key B at position 1

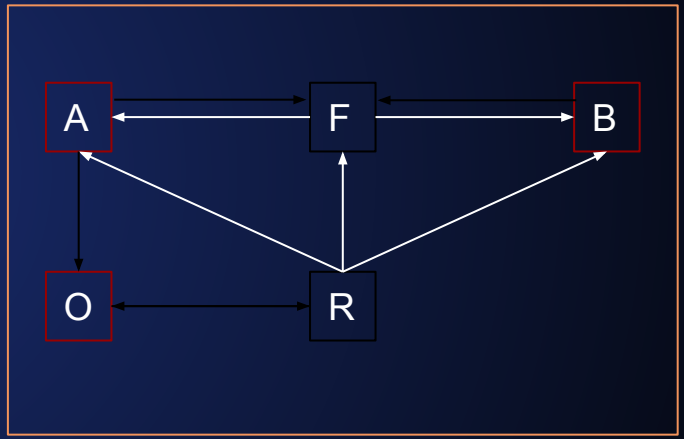


*Black arrows indicate inactive connections
*White arrows indicate active connections

Red-Black Example



- Goal: Get key B to position 1
- Steps
  - Move to position 2
  - Take key A
  - Move to position 3
  - Open the lock
  - Move to position 7
  - **Drop key A**
  - Take key B
  - **Move to position 1**
  - Drop key B



*Black arrows indicate inactive connections
*White arrows indicate active connections

Thanks!

Any Questions?