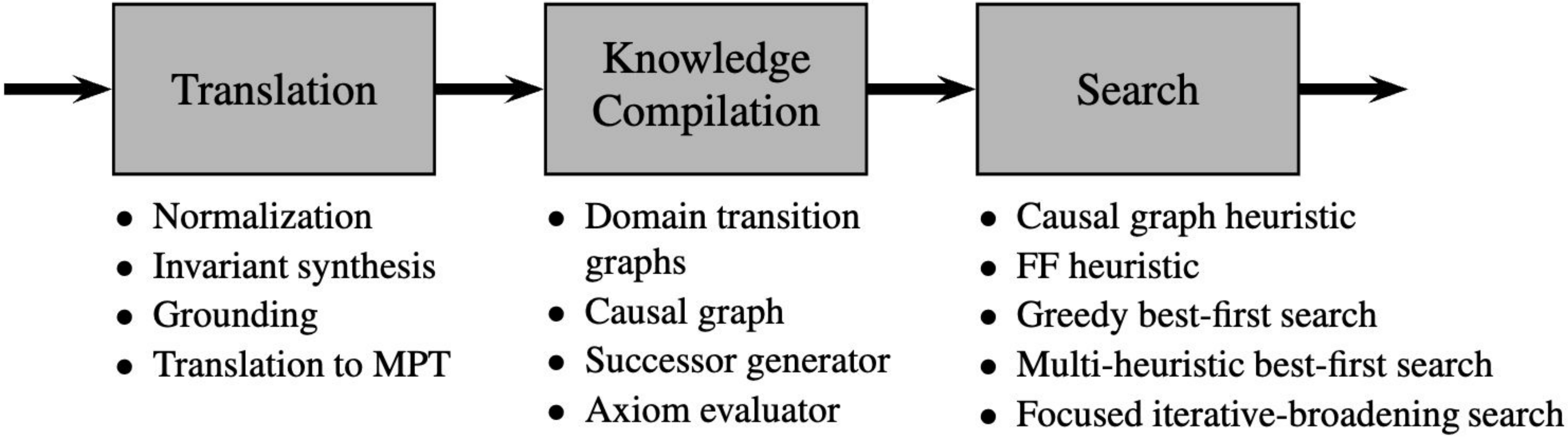
# The Fast Downward Planning System

Malte Helmert, Journal of Artificial Intelligence Research (2006)

Presenter: Cameron Egbert

# The Fast Downward Planning System: Overview



# Knowledge Compilation: Domain Transition Graphs

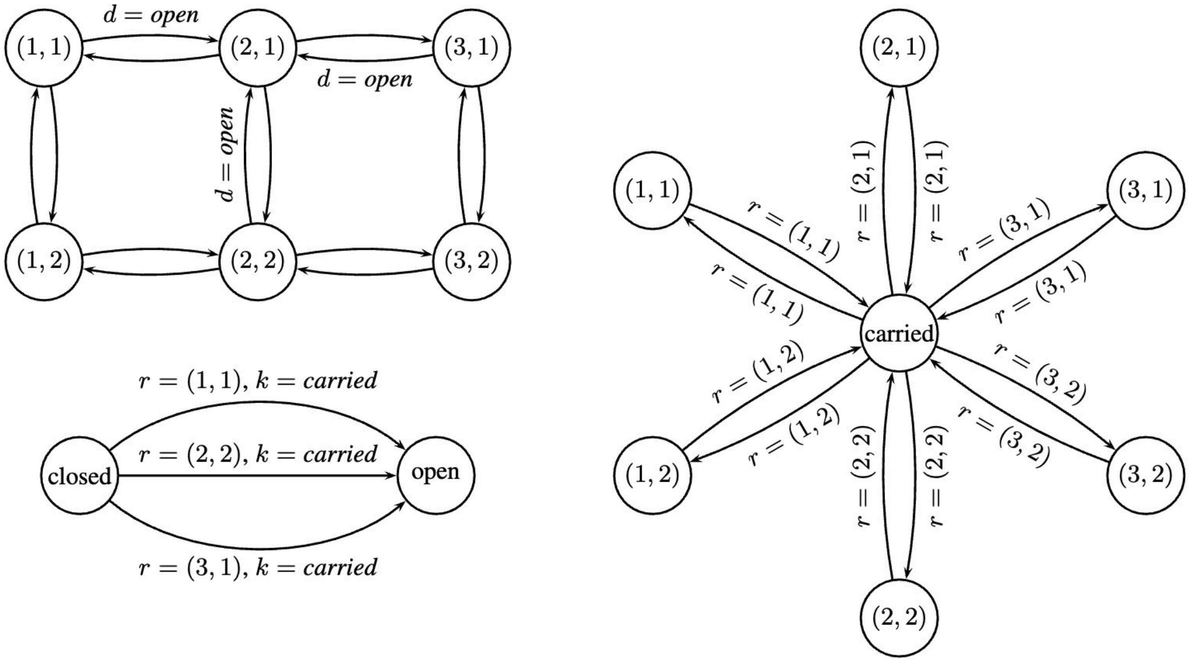



Figure 10: Domain transition graphs of a GRID task. Top left: $DTG(r)$ (robot); right: $DTG(k)$ (key); bottom left: $DTG(d)$ (door).

# Knowledge Compilation: Domain Transition Graphs

**Definition 5** ***Domain transition graphs***
*Let $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$ be a multi-valued planning task, and let $v \in \mathcal{V}$ be a state variable of $\Pi$.*

*The **domain transition graph** of $v$, in symbols DTG$(v)$, is a labelled directed graph with vertex set $\mathcal{D}_v$. If $v$ is a fluent, DTG$(v)$ contains the following arcs:*

- *For each effect cond $\rightarrow v := d'$ of an operator o with precondition pre such that pre $\cup$ cond contains some condition $v = d$, an arc from d to $d'$ labelled with pre $\cup$ cond $\setminus \{v = d\}$.*
- *For each effect cond $\rightarrow v := d'$ of an operator o with precondition pre such that pre $\cup$ cond does not contain the condition $v = d$ for any $d \in \mathcal{D}_v$, an arc from each $d \in \mathcal{D}_v \setminus \{d'\}$ to $d'$ labelled with pre $\cup$ cond.*

*Arcs of domain transition graphs are called **transitions**. Their labels are referred to as the **conditions** of the transition.*

*Domain transition graphs can be weighted, in which case each transition has an associated non-negative integer weight. Unless stated otherwise, we assume that all transitions derived from operators have weight 1 and all transitions derived from axioms have weight 0.*

# Knowledge Compilation: Causal Graphs

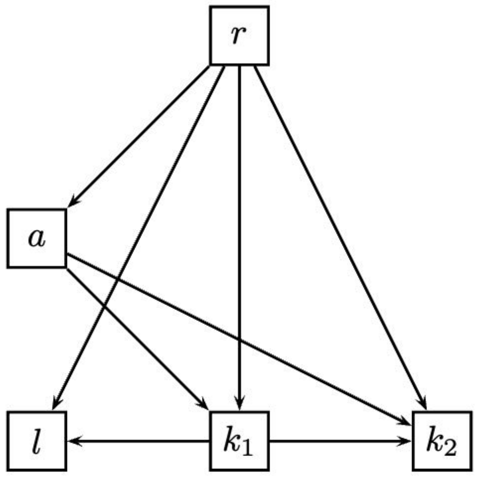
Ex: GRID Task Causal Graph:

r  - robot location

a - arm (holding or free)

l  - locked (or not)

k - key location



# Knowledge Compilation: Causal Graphs

**Definition 6** ***Causal graphs***
*Let $\Pi$ be a multi-valued planning task with variable set $\mathcal{V}$. The **causal graph** of $\Pi$, in symbols $CG(\Pi)$, is the directed graph with vertex set $\mathcal{V}$ containing an arc $(v, v')$ iff $v \neq v'$ and one of the following conditions is true:*

- *The domain transition graph of $v'$ has a transition with some condition on $v$.*
- *The set of affected variables in the effect list of some operator includes both $v$ and $v'$.*

*In the first case, we say that an arc is induced by a **transition condition**. In the second case we say that it is induced by **co-occurring effects**.*

# Knowledge Compilation: Pruning Causal Graphs

- All state variables which are not ancestors of variables mentioned in the goal are completely irrelevant.
    - So eliminate them
- To produce an acyclic graph:
    - Compute strongly connected components
    - The weight of an edge is n if it is induced by n axioms or operators
    - Pick a vertex v with minimal cumulated weight of incoming arcs
    - Set $v \prec v'$
    - Consider $V \setminus \{v\}$ and repeat.
- Only keep edges (v, v') if $v \prec v'$
- Remove dominated conditions

# Example:

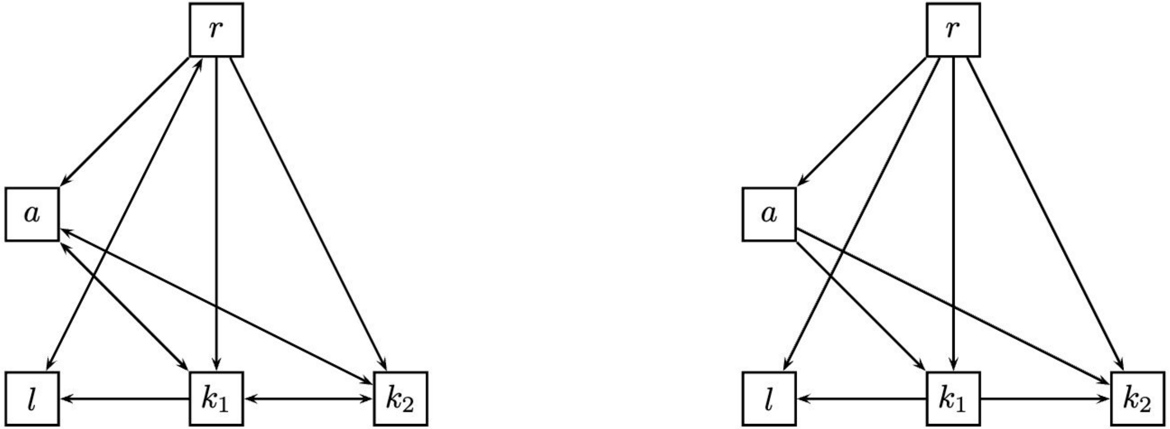



Figure 15: Causal graph of a GRID task (left) and of a relaxed version of the task (right). State variable $r$ encodes the location of the robot, $a$ encodes the status of the robot arm (empty or carrying a key), $l$ encodes the status of the locked location (locked or open), and $k_1$ and $k_2$ encode the locations of the two keys.

# Successor Generation

**Definition 8** ***Successor generators***
*A* ***successor generator*** *for an MPT* $\Pi = \langle \mathcal{V}, s_0, s_\star, \mathcal{A}, \mathcal{O} \rangle$ *is a tree consisting of* ***selector nodes*** *and* ***generator nodes***.

*A selector node is an internal node of the tree. It has an associated variable* $v \in \mathcal{V}$ *called the* ***selection variable***. *Moreover, it has* $|\mathcal{D}_v| + 1$ *children accessed via labelled edges, one edge labelled* $v = d$ *for each value* $d \in \mathcal{D}_v$, *and one edge labelled* $\top$. *The latter edge is called the* ***don't care edge*** *of the selector.*

*A generator node is a leaf node of the tree. It has an associated set of operators from* $\mathcal{O}$ *called the set of* ***generated operators***.

*Each operator* $o \in \mathcal{O}$ *must occur in exactly one generator node, and the set of edge labels leading from the root to this node (excluding don't care edges) must equal the precondition of* $o$.

Avoid checking all operators!

# Search: The Causal Graph Heuristic

Estimates the cost of changing the value of v from d to d′

Use the sum over the costs cost_v( (s(v), s*(v)) ) for all variables v for which a goal condition s*(v) is defined

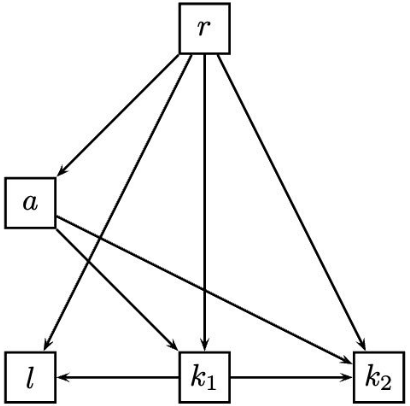
They compute the cost for all d’ using a method that like dijkstra's.

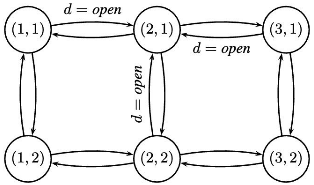
**Helpful Transitions:**
generates a set of applicable operators considered useful for steering search towards the goal.
Helpful operators are used in a relaxed plan and applicable in the current state

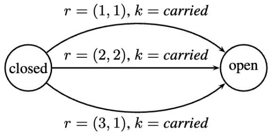
# Search: The Causal Graph Heuristic

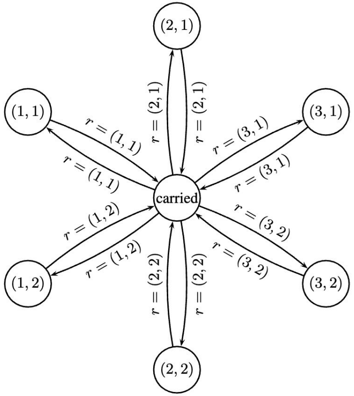
**algorithm** compute-costs($\Pi$, $s$, $v$, $d$):

- Let $\mathcal{V}'$ be the set of immediate predecessors of $v$ in the pruned causal graph of $\Pi$.
- Let $DTG$ be the pruned domain transition graph of $v$.
- $cost_v(d, d) := 0$
- $cost_v(d, d') := \infty$ for all $d' \in \mathcal{D}_v \setminus \{d\}$
- $local\text{-}state_d := s$ restricted to $\mathcal{V}'$
- $unreached := \mathcal{D}_v$
- **while** *unreached* contains a value $d' \in \mathcal{D}_v$ with $cost_v(d, d') < \infty$:
  - Choose such a value $d' \in unreached$ minimizing $cost_v(d, d')$.
  - $unreached := unreached \setminus \{d'\}$
  - **for each** transition $t$ in *DTG* leading from $d'$ to some $d'' \in unreached$:
    - $transition\text{-}cost := 0$ if $v$ is a derived variable; $1$ if $v$ is a fluent
    - **for each** pair $v' = e'$ in the condition of $t$:
      - $e := local\text{-}state_{d'}(v')$
      - **call** compute-costs($\Pi, s, v', e$).
      - $transition\text{-}cost := transition\text{-}cost + cost_{v'}(e, e')$
    - **if** $cost_v(d, d') + transition\text{-}cost < cost_v(d, d'')$:
      - $cost_v(d, d'') := cost_v(d, d') + transition\text{-}cost$
      - $local\text{-}state_{d''} := local\text{-}state_{d'}$
      - **for each** pair $v' = e'$ in the condition of $t$:
        - $local\text{-}state_{d''}(v') := e'$



# Search: The FF Heuristic

**Helpful Actions:**
generates a set of applicable operators considered useful for steering search towards the goal.
Actions applicable the current state of a relaxed plan are helpful.

Compatibility with MVTs looks like variables in superposition

Heuristic: # of operators used in the relaxed plan

# Search: Greedy BFS

**Preferred operators:** Helpful transitions and helpful actions are treated as

Alternate between two successor lists: all and preferred

Deferred Heuristic Evaluation:

- Expand before computing the heuristic
- Keeps from evaluating heuristic for every successor
- Useful for high branching factors
- Less time for more space

# Search: Multi-Heuristic BFS

Different heuristics have different strengths

Keep two open lists for each heuristic.

Alternate between all lists.

# Search: Focused Iterative-Broadening Search

Focuses on one goal at a time.

**Definition 9** ***Modification distances***
*Let $\Pi$ be an MPT, let $o$ be an operator of $\Pi$, and let $v$ be a variable of $\Pi$.*
*The **modification distance** of $o$ with respect to $v$ is defined as the minimum, over all variables $v'$ that occur as affected variables in the effect list of $o$, of the distance from $v'$ to $v$ in $CG(\Pi)$.*

Iteratively broaden the considered operators when modification thresholds prevent solutions.

- Concurrently attempt to satisfy each goal.
- Halt when one is met.
- Concurrently attempt to satisfy remaining goals in addition to previously satisfied goals. (in fact they forbid operators that undo goals)

# Search: Focused Iterative-Broadening Search

**algorithm** reach-one-goal($\Pi$, $v$, $d$, *cond*):

  **for each** $\vartheta \in \{0, 1, \ldots, \mathit{max\text{-}threshold}\}$:

    Let $\mathcal{O}_\vartheta$ be the set of operators of $\Pi$ whose modification distance with respect to $v$ is at most $\vartheta$ and which do not affect any state variable occurring in *cond*.

    Assign the cost $c$ to each operator $o \in \mathcal{O}_\vartheta$ with modification distance $c$ with respect to $v$.

    Call the *uniform-cost-search* algorithm with a closed list, using the operator set $O_\vartheta$, to find a state satisfying $\{v = d\} \cup \mathit{cond}$.

    **return** the plan if uniform-cost-search succeeded.

  **for each** $\vartheta \in \{0, 1, \ldots, \mathit{max\text{-}threshold}\}$:

    Let $\mathcal{O}_\vartheta$ be the set of operators of $\Pi$ whose modification distance with respect to $v$ is at most $\vartheta$.

    Assign the cost $c$ to each operator $o \in \mathcal{O}_\vartheta$ with modification distance $c$ with respect to $v$.

    Call the *uniform-cost-search* algorithm with a closed list, using the operator set $O_\vartheta$, to find a state satisfying $\{v = d\} \cup \mathit{cond}$.

    **return** the plan if uniform-cost-search succeeded.

# Experimental Setups

1. **G**: Use greedy best-first search without preferred operators.
2. **G + P**: Use greedy best-first search with helpful transitions as preferred operators.
3. **G + P$^+$**: Use greedy best-first search with helpful transitions as preferred operators. Use helpful actions as preferred operators in states with no helpful transitions.
4. **M**: Use multi-heuristic best-first search without preferred operators.
5. **M + P**: Use multi-heuristic best-first search with helpful transitions and helpful actions as preferred operators.
6. **F**: Use focused iterative-broadening search.

# Results

| Domain | #Tasks | G | G+P | G+P$^+$ | M | M+P | F | Any | CG | FF | LPG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Blocksworld | 35 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 4 | 0 |
| Depot | 22 | 12 | 13 | 13 | 12 | 8 | 11 | 7 | 14 | 3 | 0 |
| Driverlog | 20 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 3 | 5 | 0 |
| Freecell (IPC2) | 60 | 4 | 4 | 12 | 11 | 12 | 40 | 3 | 2 | 3 | 55 |
| Freecell (IPC3) | 20 | 0 | 0 | 5 | 1 | 2 | 14 | 0 | 0 | 2 | 19 |
| Grid | 5 | 1 | 2 | 1 | 1 | 0 | 4 | 0 | 1 | 0 | 1 |
| Gripper | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Logistics (IPC1) | 35 | 1 | 0 | 0 | 4 | 0 | 26 | 0 | 0 | 0 | 4 |
| Logistics (IPC2) | 28 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Miconic-STRIPS | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Movie | 30 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Mystery | 30 | 1 | 2 | 1 | 0 | 0 | 13 | 0 | 1 | 12 | 15 |
| MPrime | 35 | 0 | 0 | 0 | 2 | 0 | 14 | 0 | 1 | 3 | 7 |
| Rovers | 20 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 3 | 0 | 0 |
| Satellite (IPC3) | 20 | 1 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 |
| Zenotravel | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Total** | 550 | 24 | 21 | 32 | 32 | 22 | 148 | 10 | 25 | 32 | 101 |

Figure 23: Number of unsolved tasks for the STRIPS domains from IPC1, IPC2, and IPC3.

# Results

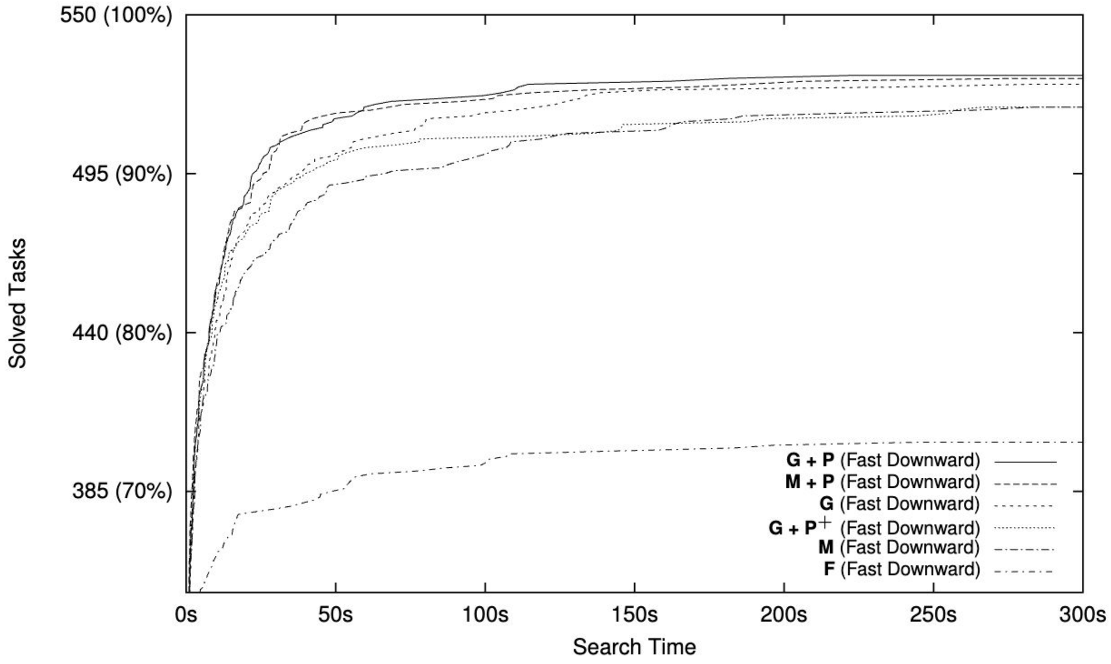



Figure 24: Number of tasks solved vs. runtime for the STRIPS domains from IPC1, IPC2 and IPC3. This graph shows the results for the various configurations of Fast Downward.

# Results

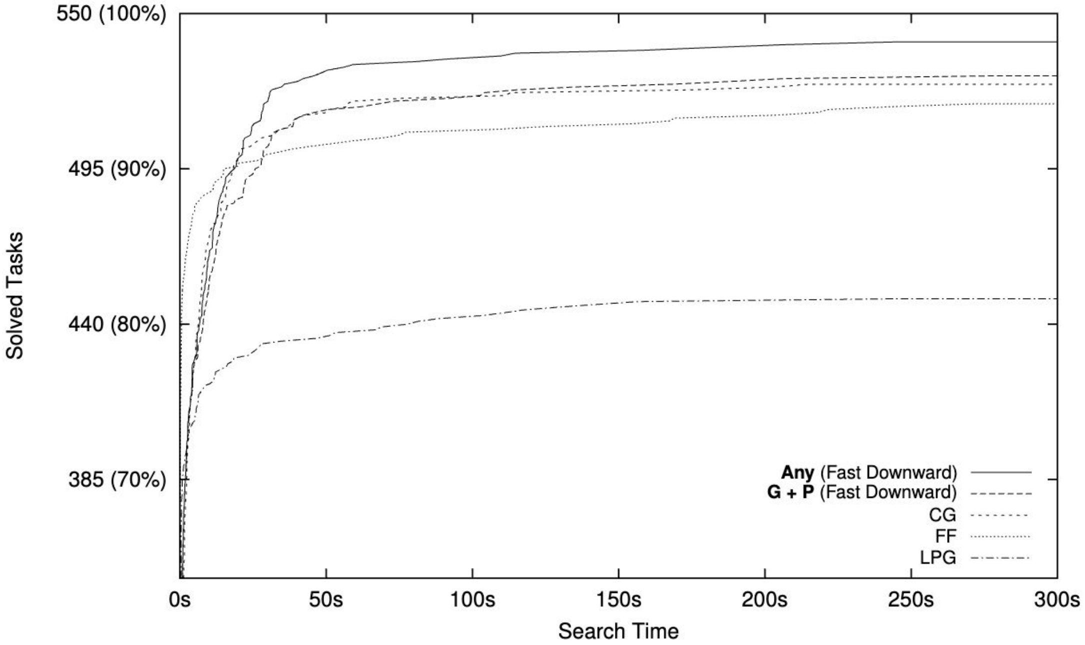



Figure 25: Number of tasks solved vs. runtime for the STRIPS domains from IPC1, IPC2 and IPC3. This graph shows the results for CG, FF and LPG and the hypothetical "Any" planner which always chooses the best configuration of Fast Downward. The result for greedy best-first search with helpful transitions is repeated for ease of comparison with Fig. 24.

# M+P configuration emerges as a clear-cut winner

# The end