Dominio Souls-like: modelado, benchmarks y límites

Emanuel Pacheco, Víctor Ramos

Universidad Carlos III de Madrid

12/01/2026

Motivación

- Modelar un juego tipo *Dark Souls* como problema de planificación automática.
- Dominios complejos: combate, exploración, recursos, upgrades, invocaciones.
- Evaluar planificadores numéricos y proposicionales sobre:
  - Costes acumulados (muertes, descansos, invocaciones).
  - Secuencias largas de acumulación de recursos mediante repetición.
  - Alto branching por presencia de minion.

Objetivos del trabajo

- Diseñar un dominio numérico `darksouls` y una variante con invocaciones `darksouls-invocations`.
- Diseñar una compilación proposicional `darksouls-fd` para planificadores tipo Fast Downward.
- Generar familias de problemas con dificultad creciente.
- Analizar:
  - Calidad de los planes (longitud, coste).
  - Escalabilidad (tiempo, nodos expandidos, estados evaluados).
  - Límites prácticos (timeouts, `OutOfMemoryError`).

Dominio numérico base `dark-souls`

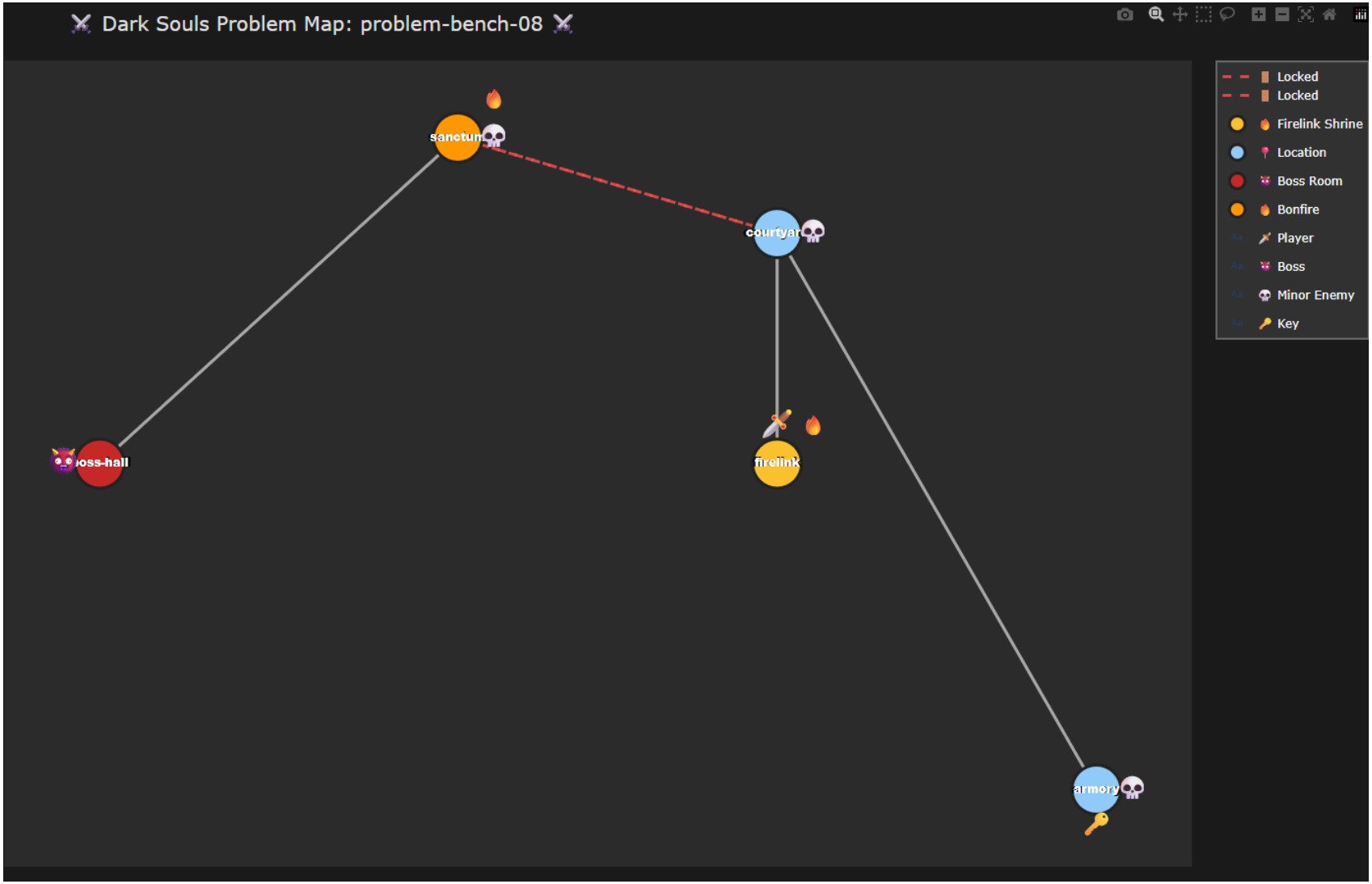
- **Requisitos PDDL**: `:strips`, `:typing`, `:action-costs`, `:adl`, `:fluents`.
- **Modelado del estado**: Ubicaciones, enemigos (menores y jefes), llaves, hogueras y forja. fluents numéricos: salud del jugador, almas, nivel de arma, estus y `total-cost`.
- **Acciones principales**: Movimiento y exploración (`move`, `unlock-door`, `open-shortcut`); Combate (`attack`, `execute-enemy`, `kill-boss`); Progresión y recursos (`pick-up-titanite`, `upgrade-weapon`, `drink-estus`, `rest`); Objetivo (`deposit-soul`).
- **Métrica**: minimizar `(total-cost)`.

Dominio numérico `dark-souls`: Mecánicas Clave

Dominio PDDL numérico de exploración, combate y gestión de recursos:

- **Vida y Muerte**: `attack` intercambia daño. Si `player-health` $\leq 0 \rightarrow$ `player-dead`. El jugador puede curar parte de su vida con un recurso limitado, el estus.
- **Respawn**: Revive en la última hoguera, restaura salud y estus, pierde todas las almas, revive enemigos menores y reinicia la salud de jefes vivos. Es como la acción de descansar solo que altamente penalizada.
- **Combate**: Enemigos debilitados se eliminan con `execute-enemy` (almas). Jefes se derrotan con `kill-boss` (alma + muro de niebla).
- **Progresión**: `level-up-stats` aumenta salud máxima. `upgrade-weapon` incrementa el nivel de arma (requisito de jefes).
- **Exploración y Recursos**: Movimiento, llaves, puertas y atajos permanentes. Hogueras: `rest` restaura recursos y revive enemigos menores. Objetivo: `deposit-soul` en Firelink.

Ejemplo gráfico de problema



Dominio con invocaciones `dark-souls`

Extensión del dominio base con un sistema de invocación persistente:

- **Predicados nuevos**: `at-summon`, `summon-available`, `summon-dead`.
- **fluents del minion**: `summon-health`, `summon-max-health`, `summon-damage`, `summon-cost`.
- **Acciones añadidas**: `summon-minion`, `move-summon`. `summon-attack`, `summon-execute-enemy`.
- **Impacto en la planificación**: Aumenta el espacio de estados (jugador + minion). Permite planes cooperativos y reducción de costo en combate.

Dominio proposicional: Estado del jugador

- **HP discretizado**:
  - `(player-hp ?h)`, `(player-max-hp ?m)`.
  - Daño y curación mediante tablas: `hp-after-attack`, `hp-heal`.
  - `hp-zero` $\Rightarrow$ `player-dead`.
- **Estus (curación limitada)**:
  - Slots desbloqueables al matar bosses y se recuperan al descansar en hogera.
  - Curación: $+5 \Rightarrow$ aumenta hp5 con cap en la vida máxima.
- **Almas discretizadas**:
  - `(player-souls ?s)`, mínimo `soul-min`.
  - Ganancia por enemigos: `soul-after-drop`.
  - Gasto al subir nivel: `soul-spend-for-level`.

Dominio proposicional: Progresión y jefes

- **Combate enemigos normales**:
  Proceso de dos etapas:
  - `attack-minor`: debilita al enemigo pero recibe daño (tablas diferentes para cada enemigo).
  - `execute-enemy`: Con el enemigo debilitado lo "*ejecuta*" para reclamar las almas.
- **Combate contra Jefe (HP del jefe discretizado)**:
  - Fase actual, `boss-current-phase`, representa la cantidad de golpes restantes para debilitar el boss.
  - `attack-boss`: Daña al boss pero recibe daño a cambio.
  - Fase cero: `boss-phase-zero` $\Rightarrow$ jefe vulnerable.
- **Remate del jefe**:
  - Acción `kill-boss`.
  - Otorga alma de jefe y desbloquea estus adicional.

ENHSP para dominios numéricos: Justificación

Seleccionamos **ENHSP** (*Expressive Numeric Heuristic Search Planner*) por su capacidad para manejar la complejidad numérica de *Dark Souls* sin abstracciones artificiales.

- **Gestión Nativa de Variables Numéricas (PDDL 2.1):** ENHSP maneja fluents como `(player-health)` o `(soul-count)` como variables reales, evitando múltiples predicados booleanos.
- **Expresividad en Efectos Complejos:** Soporta efectos condicionales universales (`forall` + `when`) necesarios para mecánicas como el *Respawn*.
- **Búsqueda en Espacio de Estados:** Simula la partida paso a paso, aplicando acciones y verificando precondiciones numéricas (p. ej., vida).

ENHSP: Modos de Operación

ENHSP-SAT (Satisfaciente)

Enfocado en velocidad.

- Motor: Lazy GBFS
- Heurística: h_{mrp} (análoga a h_{ff} para dominios numéricos)
- Helpful Actions

ENHSP-OPT (Óptimo)

Garantiza coste mínimo.

- Motor: A*
- Heurística: h_{rmax}

ENHSP-ANY (Anytime)

Mejora progresiva de soluciones.

- Motor: Lazy GBFS
- Heurística: h_{mrp} (análoga a h_{ff} para dominios numéricos)
- Helpful Actions

FD: Modos de Operación

LAMA-FIRST (Satisfaciente, ligero)

Enfocado en velocidad.

- Motor: Lazy Greedy Best-First Search (GBFS)
- Heurística: h_{ff} (plan relajado, coste estimado por longitud)
- Heurística: h_{lm} (hechos obligatorios, estima según landmarks)

SEQ-OPT-FDSS-1 (Óptimo)

Garantiza coste mínimo.

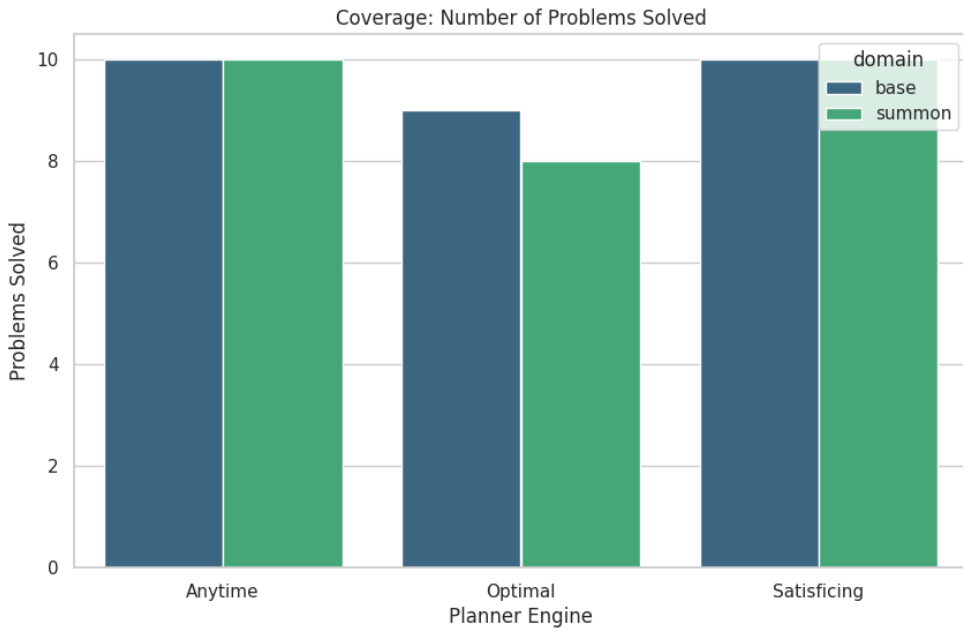
- Motor: A* (portfolio FDSS)
- Heurísticas admisibles: lmcut, hmax, MS

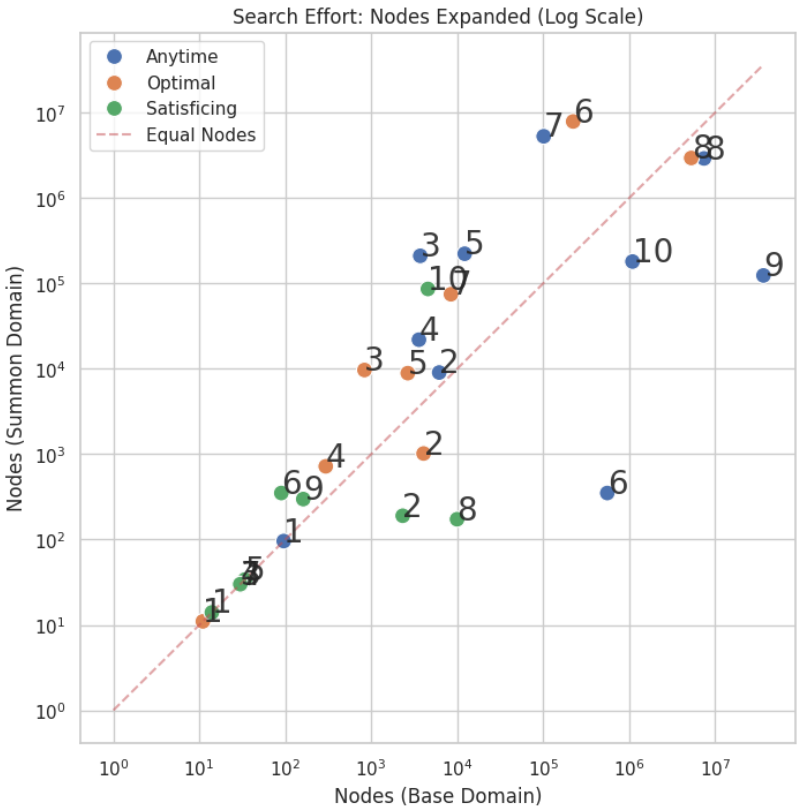
SEQ-SAT-LAMA-2011 (Satisfaciente, anytime)

Mejora progresiva de soluciones.

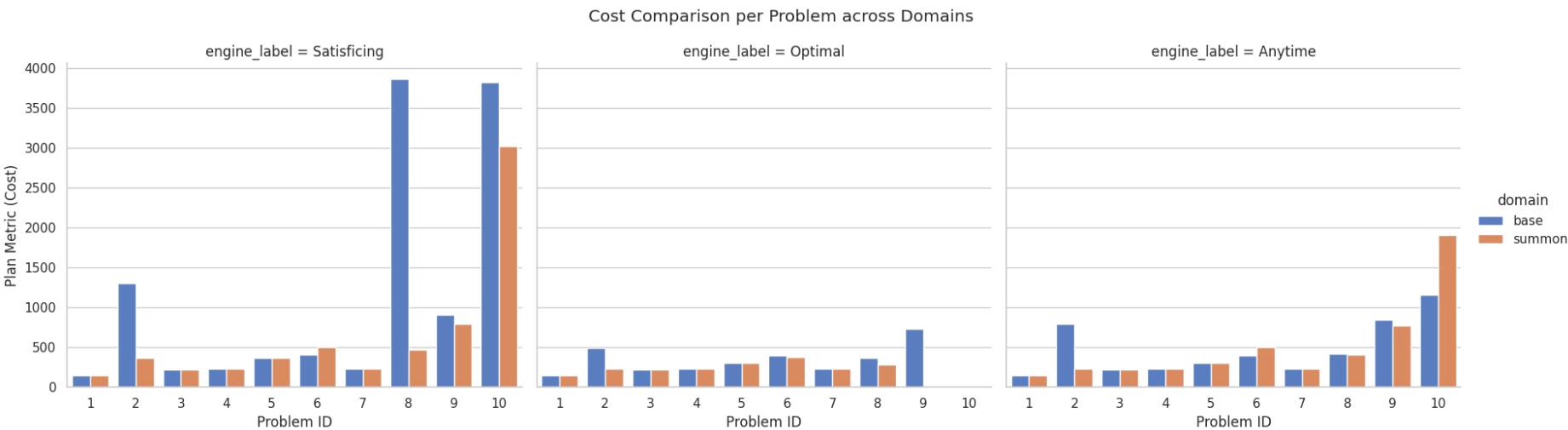
- Lazy Greedy Best-First Search (GBFS)
- Heurísticas: h_{ff} (plan relajado, coste estimado por longitud), h_{lm-sum} (estima según costes para alcanzar lm)

Resultados en dominios numéricos (10 problemas)





Resultados en dominios numéricos (10 problemas)

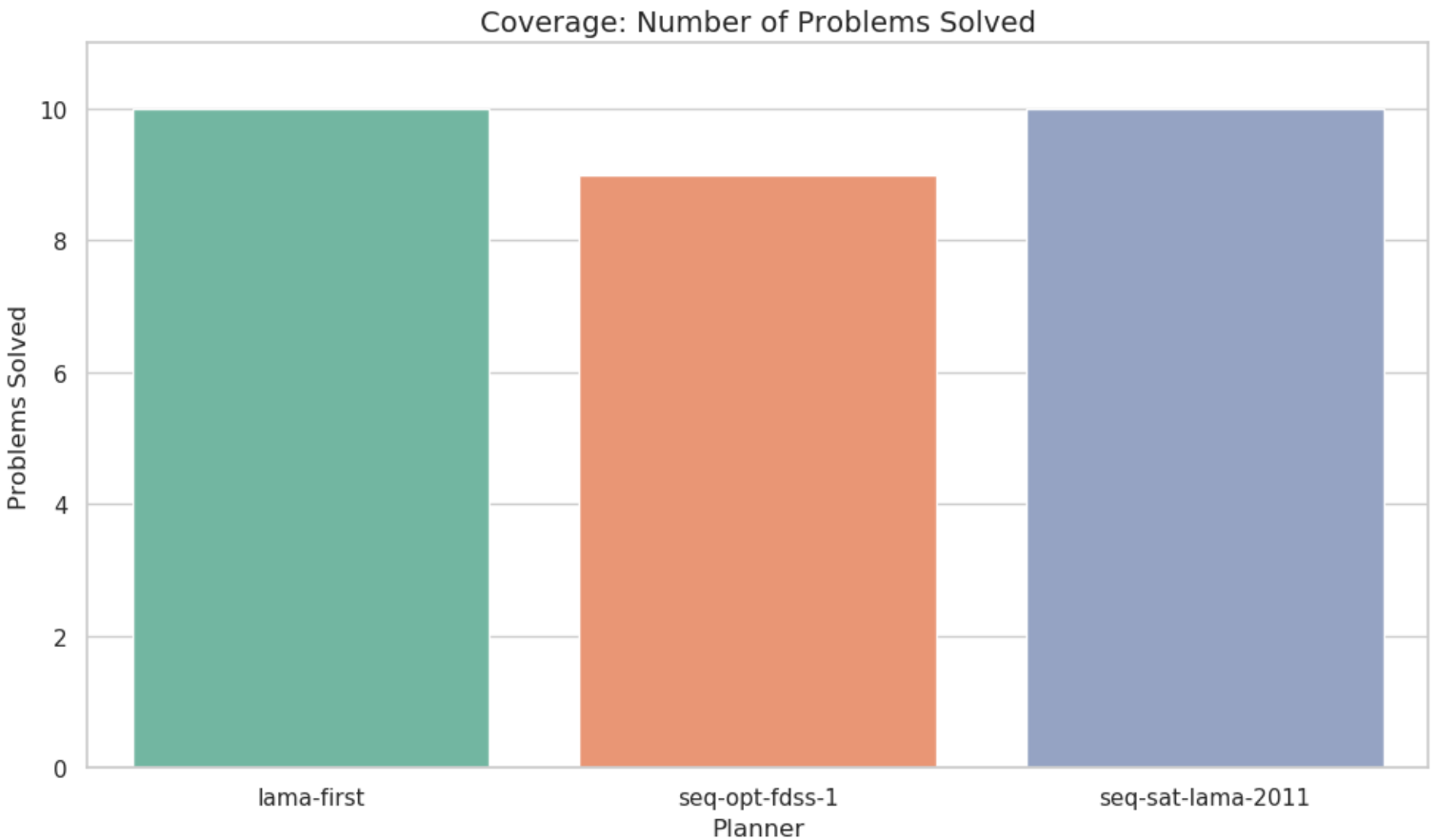


Resultados: Invocaciones (Summon)

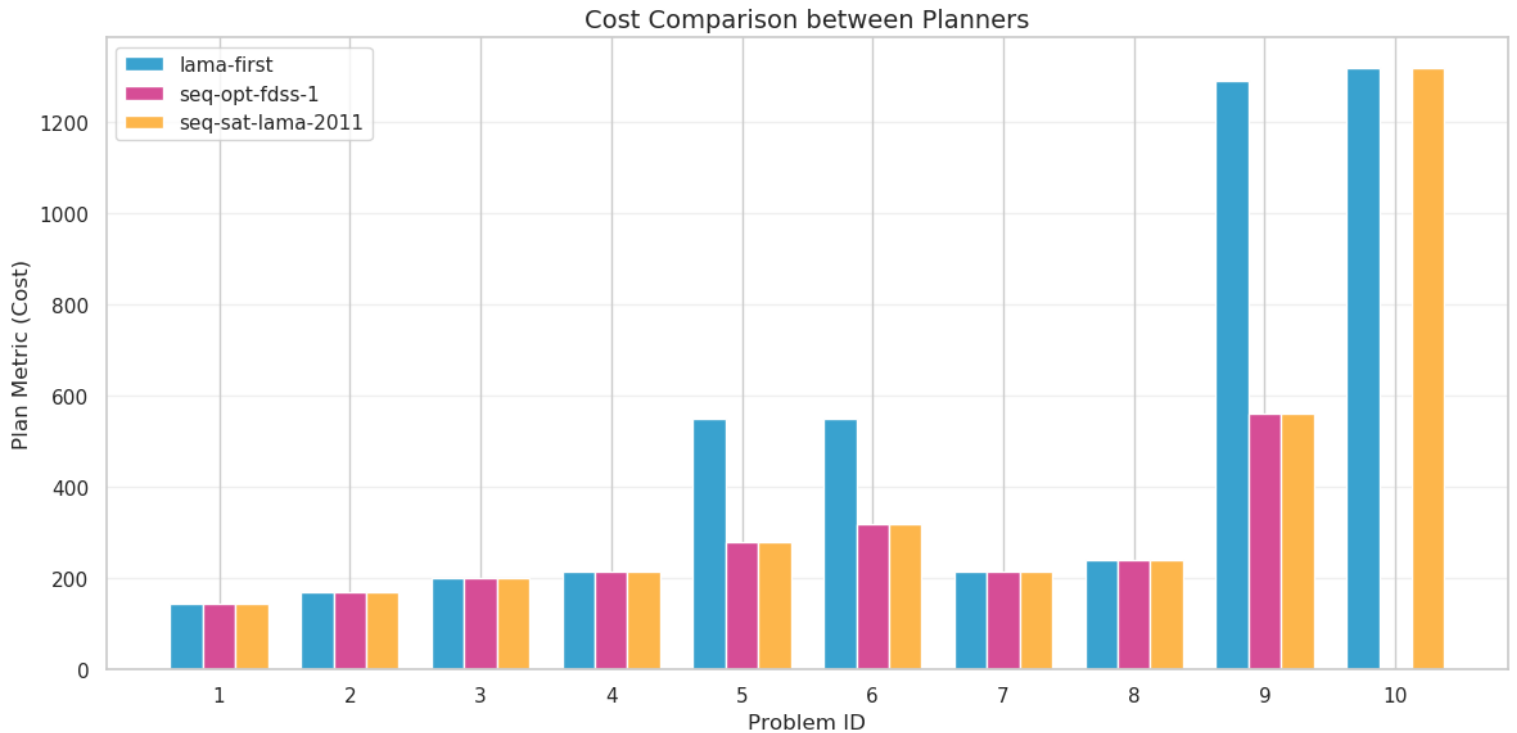
La introducción de un aliado (minion) aumenta la complejidad:

- **Impacto en el Grafo de Búsqueda**:
  - Se duplican las variables fluents (salud, posición).
  - Aumenta el factor de ramificación (acciones del jugador + acciones del minion).
- **Comportamiento observado**:
  - En problemas pequeños, el planificador prefiere **no invocar** para ahorrar el coste de la acción `summon`.
  - En problemas que requieren la subida de nivel, el uso del minion elimina esta necesidad llevando a menores costos.
  - En problemas complejos, el modo Anytime no logra optimizar el coste incluso con la invocación, esto es debido al alto factor de ramificación que no le permite alcanzar el mismo coste que su variante sin invocación, alcanzando el **TIMEOUT** o **OOM** antes.

Resultados en dominio proposicional (10 problemas)



Resultados en dominio proposicional (10 problemas)



Resultados: Dominio proposicional

- **Comportamiento observado**:
  - La estrategia dominante es un *boss rush* con progresión mínima solo cuando es obligatoria.
  - Las subidas de nivel solo aparecen en problemas complejos donde la supervivencia lo requiere ya que el uso de estus es menos costoso.
  - En el caso peculiar del problema 10 al ser más complejo se pueden observar las mecánicas de lvl-up y respawn.

Conclusiones

- Dominio numérico base:
  - Expresivo y razonablemente tratable para ENHSP en instancias pequeñas/medias.
- Dominio con invocaciones:
  - Aumenta la expresividad y las estrategias posibles.
  - Penaliza fuertemente el rendimiento (tiempos largos, OOM).
- Dominio FD:
  - Permite el uso de planificadores proposicionales maduros y heurísticas consolidadas.
  - Interesante para comparar soluciones numéricas vs. discretizada
  - Se aprecian estrategias adaptativas *boss-rush* en problemas simples y progresión explícita solo cuando la supervivencia lo requiere.

Preguntas

¿Preguntas?