

PRIMER PAS: Descripció en PDDL	2
SEGON PAS: IMPLEMENTACIÓ	2
TERCER PAS: HEURÍSTICA	2
QUART PAS: Estudi	4
Factor d'embrancament mig (real) – Map C, BFS	4
Anàlisi comparativa d'algorismes (només amb LNT)	5

PRIMER PAS: Descripció en PDDL

Resultats de domain.pdd i map.pddl :

1. (moure agent2 f1c2 f1c1)
2. (moure-i-recol·lir agent2 f1c1 f2c1 clau-a)
3. (obrir-i-entrar agent1 f3c5 f3c6 porta-a clau-a)
4. (moure agent1 f3c6 f2c6)

SEGON PAS: IMPLEMENTACIÓ

Github:

None

<https://github.com/felixandresn/PROP-ACT1>

TERCER PAS: HEURÍSTICA

Basica:

Aquesta és una heurística básica optimizada que utilitzar el mètode Manhattan per a trobar la clau més propera i o sortida.

Avançada:

Aquesta heurística combina diverses mesures informatives per estimar de manera eficient la distància mínima fins a la meta. L'objectiu és mantenir la funció $h(n)$ admissible (mai sobreestima el cost real) però més informada que la heurística bàsica.

- ◆ Components principals:

Distància mínima agent → clau:

Es calcula la distància Manhattan més petita entre qualsevol agent i qualsevol clau pendent. Serveix per prioritzar els estats on els agents estan més a prop d'obtenir noves claus.

Cost del MST (Minimum Spanning Tree) entre claus:

Es calcula un arbre d'expansió mínima (algorisme de Prim simplificat) entre totes les claus detectades al mapa.

Això representa el cost mínim necessari per connectar totes les claus entre si.

Distància mínima clau → sortida:

Finalment, s'afegeix la distància Manhattan mínima entre qualsevol clau i la sortida (S), que representa el tram final després de recollir totes les claus.

- ◆ Fórmula global:

$$h(n) = \min(\text{agent} \rightarrow \text{clau}) + \text{MST}(\text{claus}) + \min(\text{clau} \rightarrow \text{sortida})$$

- ◆ Avantatges:

Molt ràpida: tot es calcula amb distàncies Manhattan i estructures simples (arrays).

Informada: combina informació sobre agents, claus i sortida, millorant la qualitat de la cerca.

Admissible: mai sobreestima el cost real, ja que totes les distàncies són mínimes possibles.

- ◆ Complexitat:

El càlcul de la heurística és $O(k^2)$ respecte al nombre de claus k , gràcies a la implementació del MST amb arrays.

És molt eficient i escalable per a mapes grans.

QUART PAS: Estudi

Factor d'embrancament mig (real) – Map C, BFS

Càcul del factor d'embrancament mig:

El factor d'embrancament real (b_{real}) es calcula així:

$$b_{real} = \text{Nodes Generats} / \text{Nodes Explorats} = (\text{Nodes Explorats} + \text{Nodes Tallats}) / \text{Nodes Explorats}$$

Amb LNT:

Nodes explorats (BFS) = 32.525

Nodes tallats = 31.675

$$b_{real} = (32.525 + 31.675) / 32.525 \approx 1,974$$

Sense LNT:

Nodes explorats (BFS) = 1.208.951

Nodes tallats = 1.538.532

$$b_{real} = (1.208.951 + 1.538.532) / 1.208.951 \approx 2,273$$

Com a detall principal sense LNT , la memòria emplenar i consulta moltíssims mes nodes per tal , com obviament no està limitat.

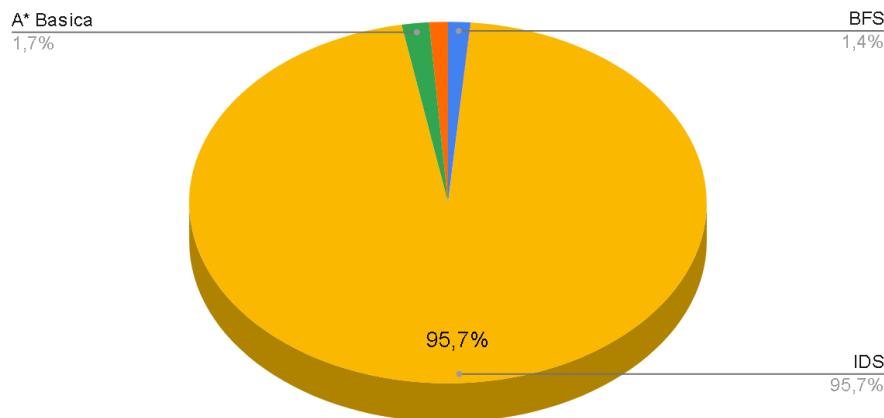
D'això podem treure'n que el LNT sigui molt necessari ja que elimina re-expansions massives i evita duplicats sistemàtics. Per lo tant no omple la memòria amb moviments innecessaris i cercas innecessàries.

Anàlisi comparativa d'algorismes (només amb LNT)

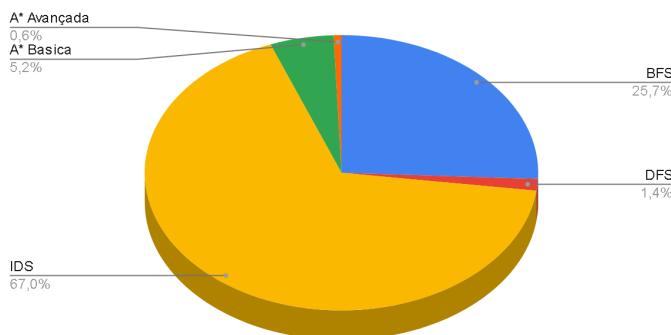
Algorisme	Mapa	Solució?	Optima?	Nodes Expl.	Nodes Tallats	Memoria Pic	Temps (ms)
BFS	A	Si	Si (5)	280	273	474	11
DFS	A	Si	No (14)	15	13	116	1
IDS	A	Si	Si (5)	730	123	55	2
A* Basica	A	Si	Si (5)	57	187	172	3
A* Avançada	A	Si	Si (5)	7	4	58	0
BFS	B	Si	Si (31)	231	227	189	1
DFS	B	Si	No (41)	86	86	130	1
IDS	B	Si	Si (31)	21.121	11.182	179	38
A* Basica	B	Si	Si (31)	151	223	166	1
A* Avançada	B	Si	Si (31)	125	188	150	2
BFS	C	Si	Si (37)	32.525	31.675	15.868	119
DFS	C	No	-	5.332	3.816	1.956	15
IDS	C	Si	Si (37)	3.743.878	2.586.242	12.425	3.224
A* Basica	C	Si	Si (37)	11.164	43.833	12.946	94
A* Avançada	C	Si	Si (37)	5.747	22.248	7.650	53
BFS	D	Si	Si (79)	406.889	406.098	122.398	2.610
DFS	D	No	-	11.228	6.470	2.771	43
IDS	D	No	-	61.810.597	48.169.770	80.449	186.611
A* Basica	D	Si	Si (79)	105.494	651.661	108.095	3.291
A* Avançada	D	Si	Si (79)	75.951	468.051	84.572	2.262

CONCLUSIONS DE RESULTATS:

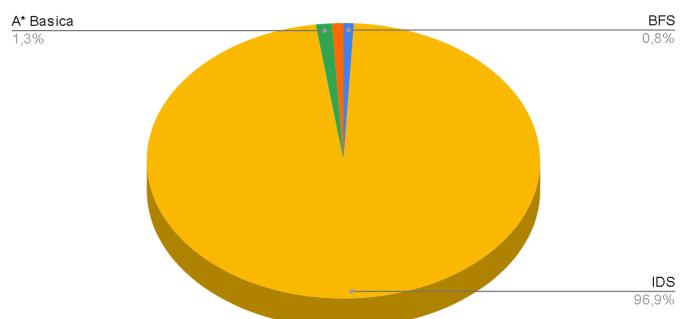
TEMPS:



NODES EXPLORATS:



NODES TALLATS:



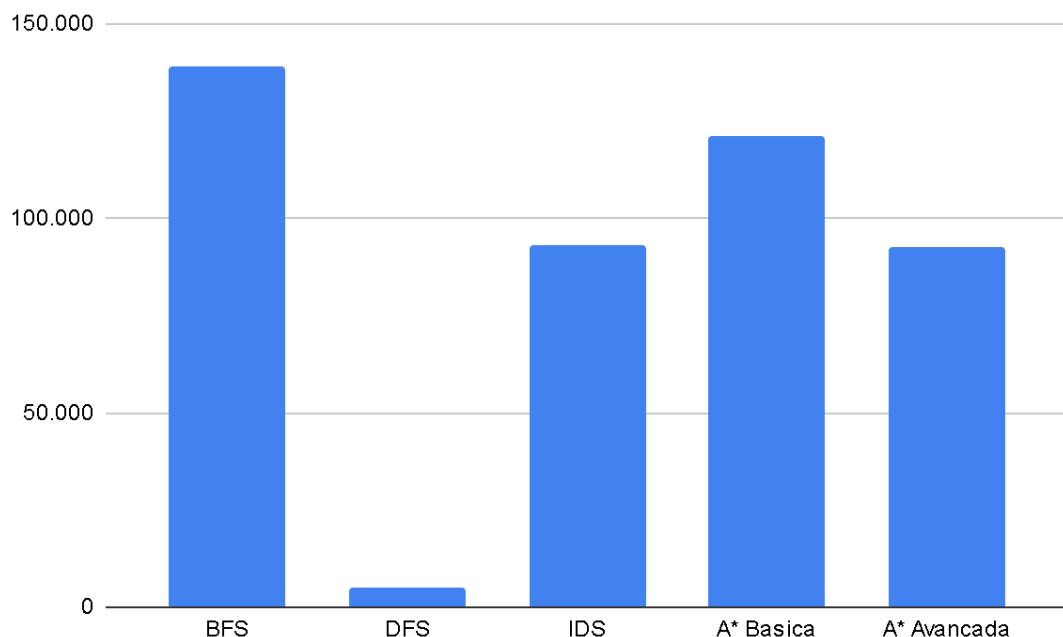
Com podem observar els resultats tant de temps i nodes explorats , la cerca A* Avançada i el DFS son les cerques que menys nodes visita menys nodes tallen, i menys temps triguen en fer-la .

Encara que el més adient es el A* ja que dona una solució en tots els casos .

També perfectament podem veure que el IDS no es el la millor cerca , explora masses nodes y triga molt .

Pero el que no ha donat ninguna solució òptima ha sigut el DFS i el que meny solucions ha troba , només explora molts pocs nodes y triga molt poc temps.

ÚS DE MEMORIA TOTAL:



En quant a utilització de memòria el mes optim segueix sent A* Avançada , encara que el IDS hagi gairebé utilitzat menys memoria .

Ei DFS no utilitza gairebé res de memòria per tant per a tractar mapes molt petits és molt òptim , pero en mapes una mica més grans no es fiable i també comentar que dona resultats no optims.

I per acabar el BFS es veu limitat per Frontera , explora molts més nodes que la A* Basica , pero marca un temps total molt similar .

RESULTAT FINAL :

La cerca més fiable tenint en compte que els mapes poden de ser qualsevol mida i donarà una resposta fiable óptima consumir la memoria justa , recorre els nodes mes necessaris i traigan el mínim de temps es la cerca A* Avançada.

Seguida de el BFS i la cerca A* Basica , el IDS i finalment el DFS el pitjor en casos que no siguin molt a petita escala i resultats òptims.