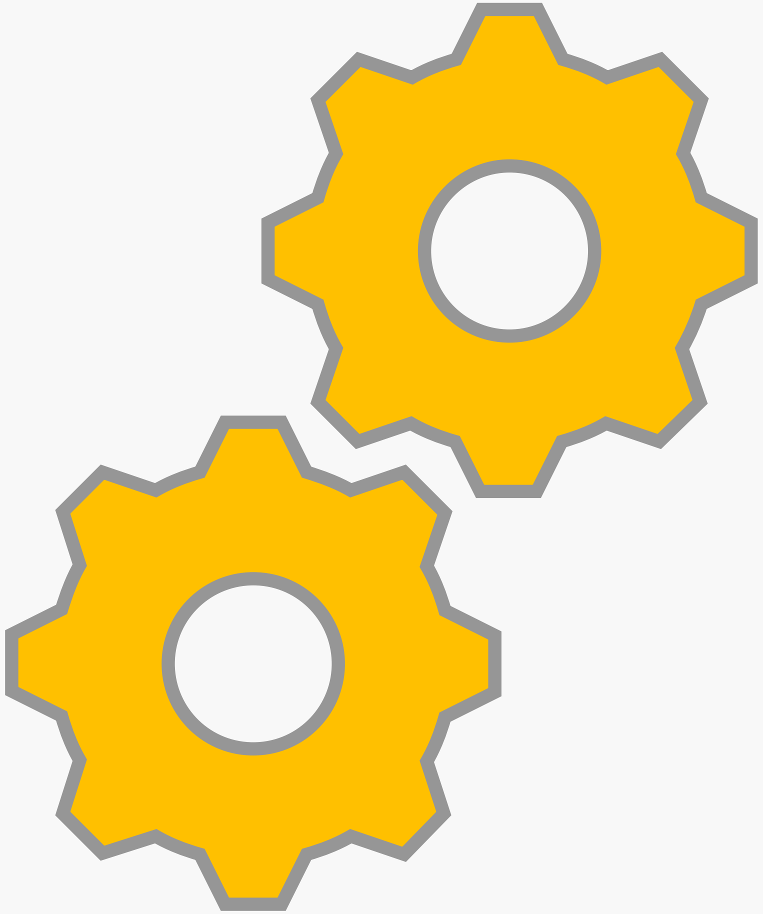
MULTI-AGENT PATHFINDING WITH REAL-TIME HEURISTIC SEARCH

Martin Bakoš

20.4.2022

Presentation structure

- Introduction
- Problem formulation
- Related work
  - *A**
  - *WHCA**
  - *FAR*
  - *RTAA**
- BMAA*
- Experiments
- Conclusion



INTRODUCTION

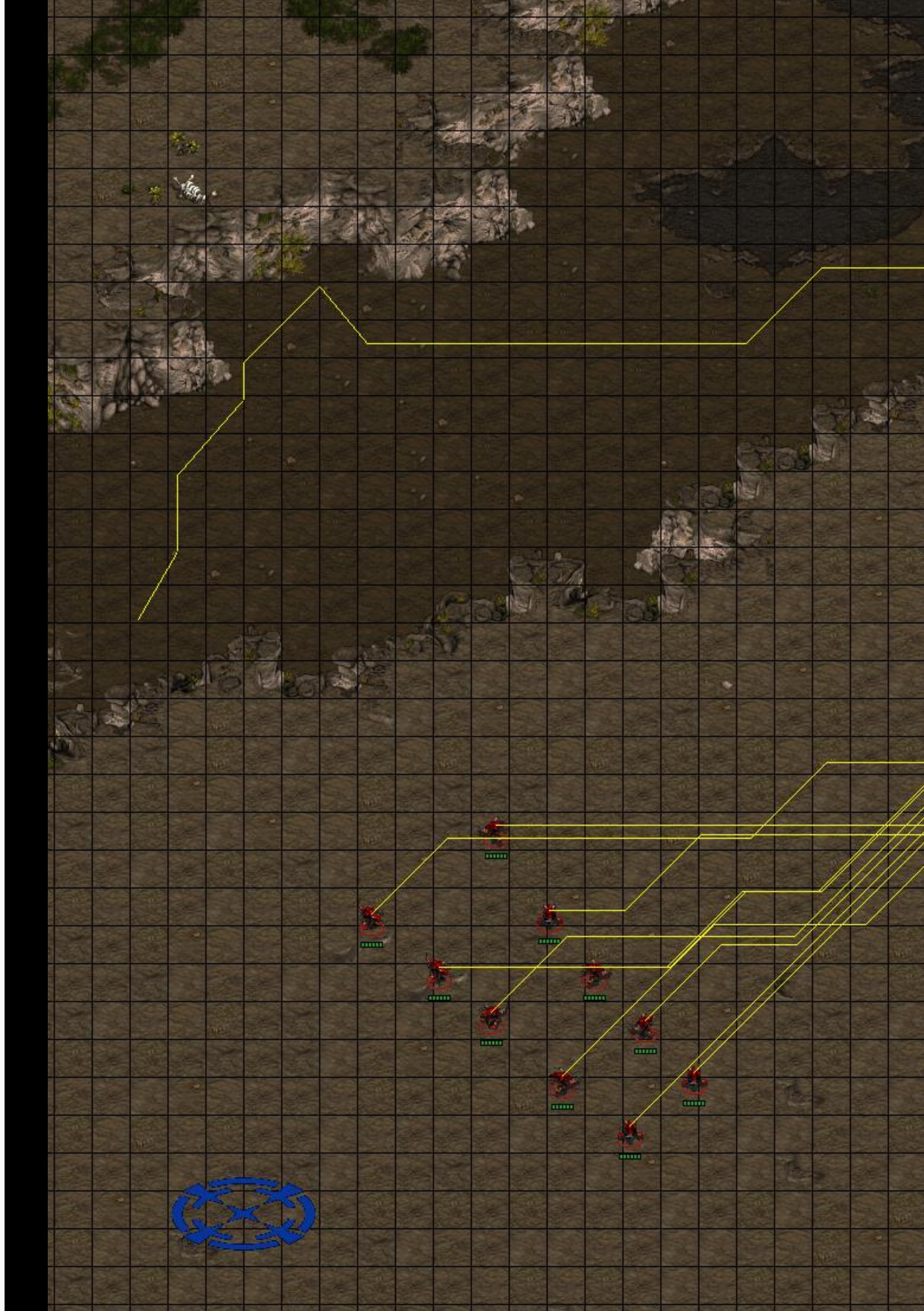
Motivation

Goal:

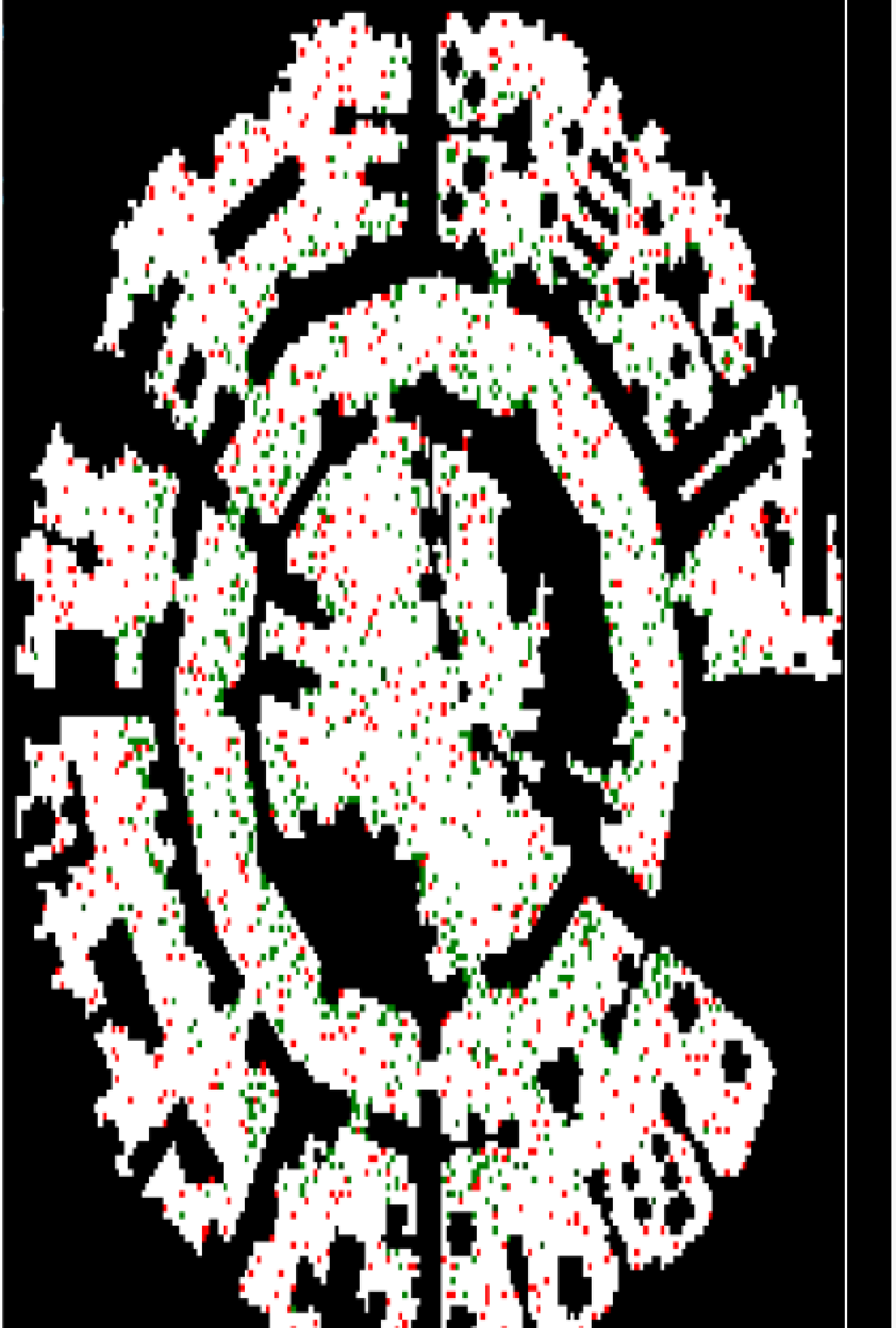
- Suitable MAPF algorithm for NPCs in video games

Requirements:

- Limited amount of time
- Re-tasking
- Unknown map
- Dynamically changing map
- Restricted agent communication
- Non-complete control



PROBLEM FORMULATION



Problem definition

We will define MAPF as pair ***(G, A)*.**

Where:

- ***G*** = *(N, E, c)* – undirected weighted graph
  - ***N*** *– graph nodes*
  - ***E*** $\subseteq$ N × N – *graph edges*
  - ***c***: *E* ⟶ *[0,inf) – cost function*
- ***A*** = *{*a^1*, . . . ,* a^n*}* – agents
  - $\boldsymbol{a}^{i}$ = *(*n^i_{start}*,* n^i_{goal}*) – pair of start and goal node*

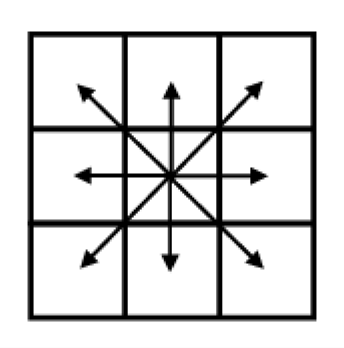
Green – startin location

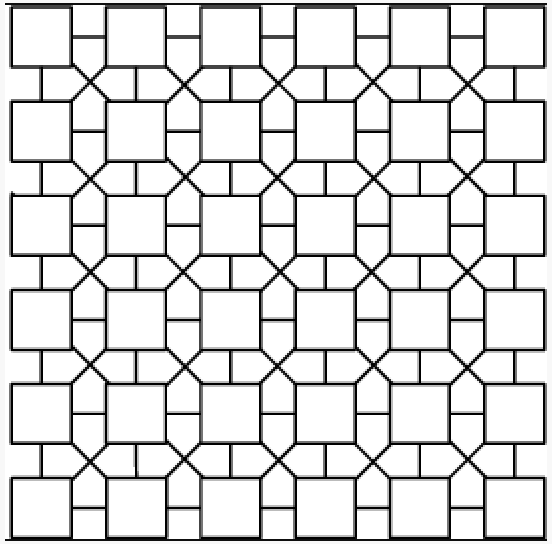
Red – goal location

Graph assumptions

We are assuming graph corresponding to rectangular 8 neighbor grid.

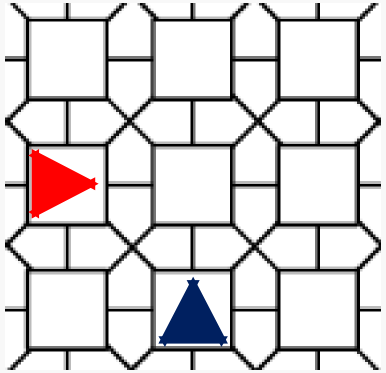
- Node corresponds to cell
  - *Cell can't be blocked by stationary obstacle*
- Neighbors are connected via edge
- Every node has a loop
- Cost of edge is:
  - 1 *between cardinal neighbors*
  - $\sqrt{2}$ *between diagonal neighbors*
  - *0 if it is loop*

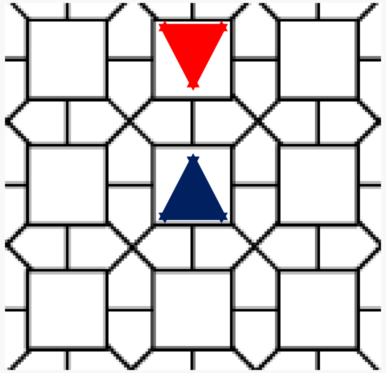




Agent assumptions and colisions

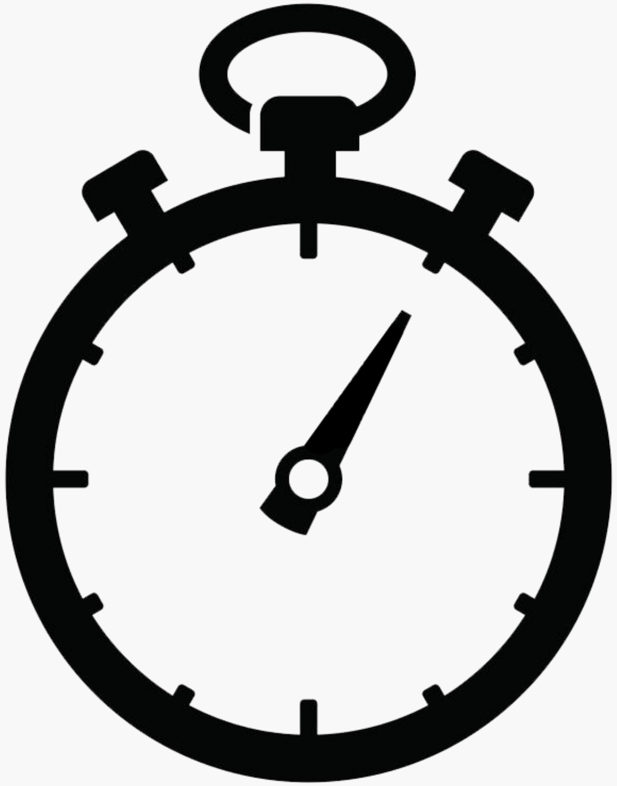
- The time will advance in discrete steps.
- One agent occupies exactly one node
- For every agent a^i we define:
  - $n^i_{curr} \in N$ – current position
  - P – prefix of path to goal
  - $P(n)$ – successor to node n on path
- **Central NPC controller** executes agents movement
  - *Agent a^i is moved from n^i_{curr} to $P(n^i_{curr})$* **or**
  - *Stays in place if*
    - $P(n^i_{curr})$ is not defined
    - Two agents would swap
    - Two agents would move to same node





Performance measures

- **Completion rate:**
  - percentage of agents in their goal locations
- **Completion time** of an agent**:**
  - undefined if agent is not in goal location
  - time step when goal location was last reached
- **Travel distance** of an agent**:**
  - sum of the costs of the edges traversed
- **Completion time** and **Travel distance** for MAPF:
  - Mean of all agent's completion time /travel distance

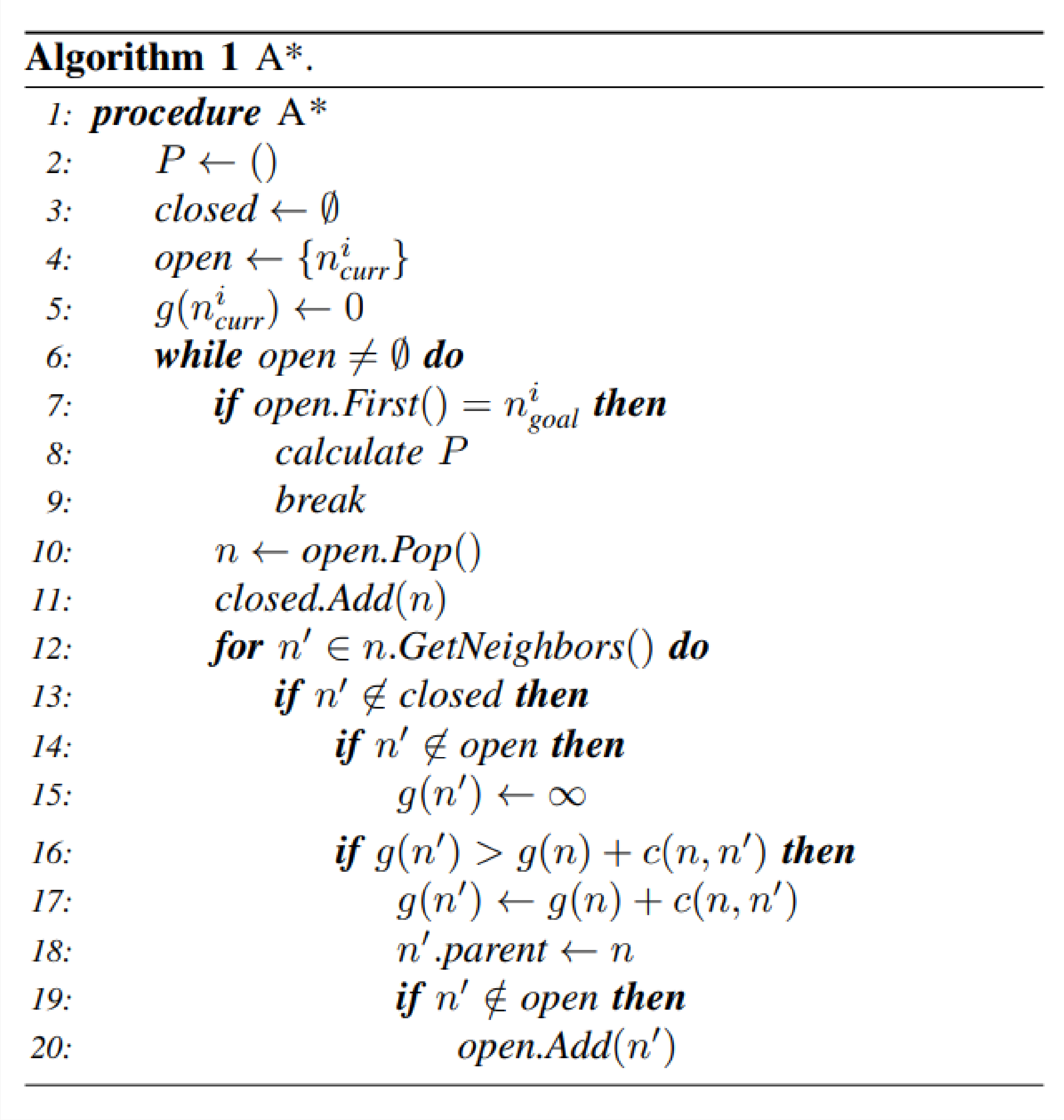


These measures cannot be optimized **simultaneously**!
Completion rate will be our main metric.

RELATED WORK

A*

- **Single-agent** pathfinding
- **Graph** search:
  - *Monotonous (consistent) heuristic is required*
- ***f-value*** for *n* is *f(n) = g(n) + h(n)* where:
  - *g(n)* – minimum path cost from *current* to *n*
  - *h(n)* – heuristic estimate path cost from *n* to *goal*
- **Complete** and **optimal**
- **Foundation** for our algorithm



Algorithm 1 A*.

```
procedure A*
    $P \leftarrow ()$
    $closed \leftarrow \emptyset$
    $open \leftarrow \{n^i_{curr}\}$
    $g(n^i_{curr}) \leftarrow 0$
    while $open \neq \emptyset$ do
        if $open.First() = n^i_{goal}$ then
            calculate $P$
            break
        $n \leftarrow open.Pop()$
        $closed.Add(n)$
        for $n' \in n.GetNeighbors()$ do
            if $n' \notin closed$ then
                if $n' \notin open$ then
                    $g(n') \leftarrow \infty$
                if $g(n') > g(n) + c(n, n')$ then
                    $g(n') \leftarrow g(n) + c(n, n')$
                    $n'.parent \leftarrow n$
                    if $n' \notin open$ then
                        $open.Add(n')$
```

Online MAPF

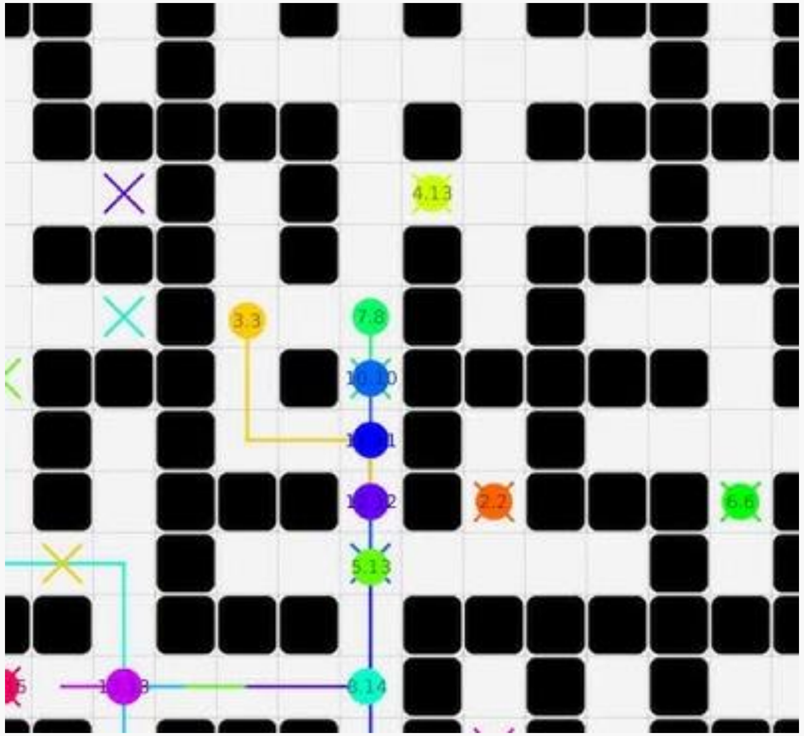
- **Windowed Hierarchical Cooperative A*** (WHCA*)
- **Flow Anotated Replanning** (FAR)

WHCA*

- Plans **collision free** path for **limited** amount of moves

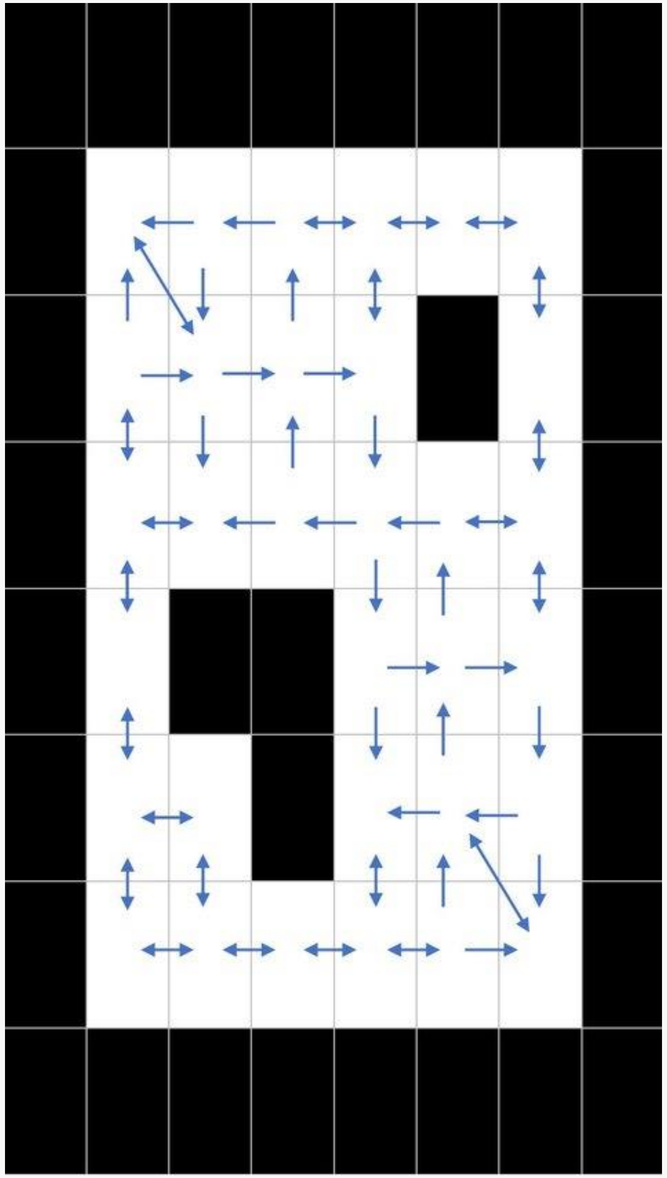
  => window

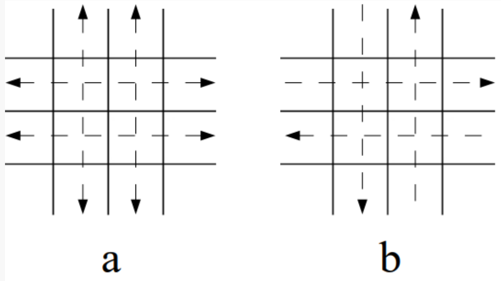
- Uses **reservation table**
  - *This adds time dimension*
- **Limit** must be chosen carefully to
  - *avoid conflicts*
  - *not exceed available time*
- WHCA* requires all agents under **complete** control
- Animation

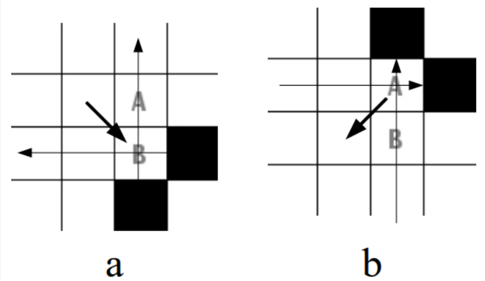


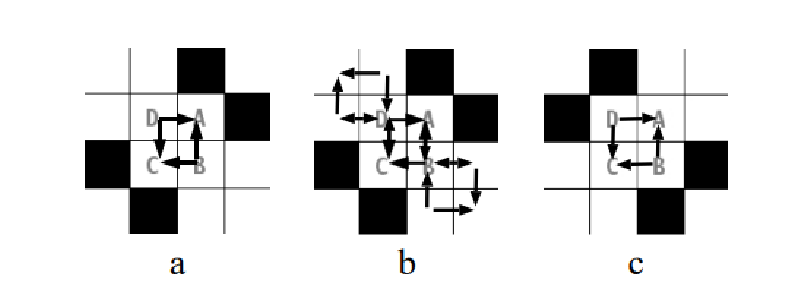
FAR

- Combines WHCA* with **flow annotations**
- Agents must **reserve** their next moves
- Reservations are not incorporated in to planning,
  - *Agent will wait for their turn*
  - *Reservations can cause deadlock*
    - Temporarily move agent from goal location
    - Only a partial solution
- Original graph is transformed to **flow annotated** graph
- A* is then used to find paths









Flow annotated graph

- Lowers the number of collisions
  - Especially **head to head** collisions
- **Undirected** graph → **Directed** graph
  - *move directions*
- Preserves **reachability**

Transformation for grid:

- Rows are alternately connected via westbound and eastbound edges
- Columns are alternately connected via northbound and southbound edges
- *Add diagonal edge to sources and sinks.*
- *Edges in one-cell-wide corridors remain undirected*

Real-Time Heuristic Search (RTHS)

Idea:

- Repeat:
  - *Compute prefix*
  - *Execute first move*
  - *Update heuristic*

Advantages:

- Constant amount of search
- Short computation time
- Small amount of lost search

Real-Time Adaptive A* - (RTAA*)

- RTHS algorithm
- **Implementation:**
  - **A*** with limited number of expansions
  - Move along **path** to node s to be expanded by A*
  - Update heuristic according to *f(n)*

BMAA* - BOUNDED MULTI-AGENT A*

BMAA* - Overview

Idea:

- Every agent **runs RTAA***
- **Central NPC controller** executes moves

Properties:

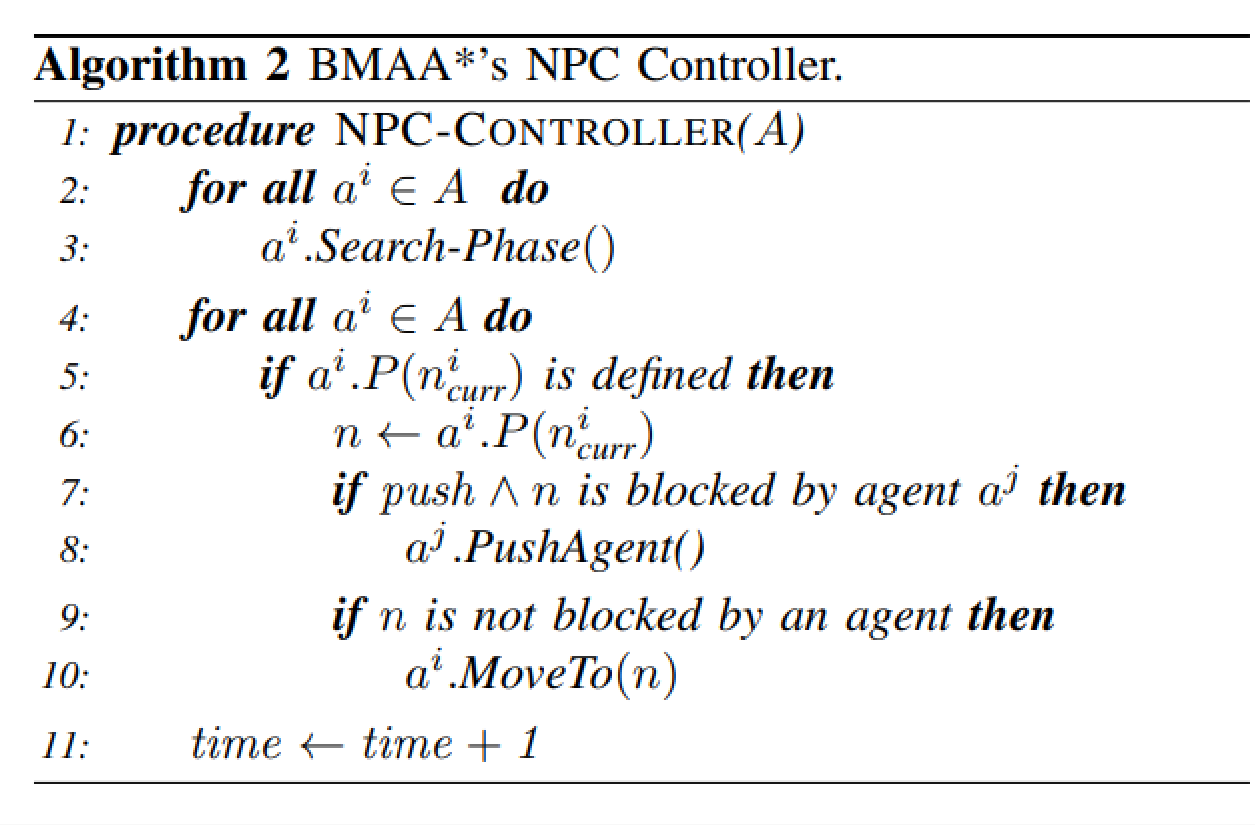
- Modular design
- Works in real-time
- Losses only small amount of search
- No coordination needed
- Complete control not required

Algorithm parameters:

- Expansions
  - *Limit for A* expansions*
- Vision
  - *Agent vision distance*
- Moves
  - *Number of moves before RTAA* re run*
- Push
  - *Whether agent can push other agent*
- Flow
  - *Whether to use flow annotated graph*

BMAA* - NPC-Controller

- *Time* is initialized with *0*
- Invokes in every time stamp
- ***A*** := agents currently under control of system
- Pushed agents will return to their goal positions

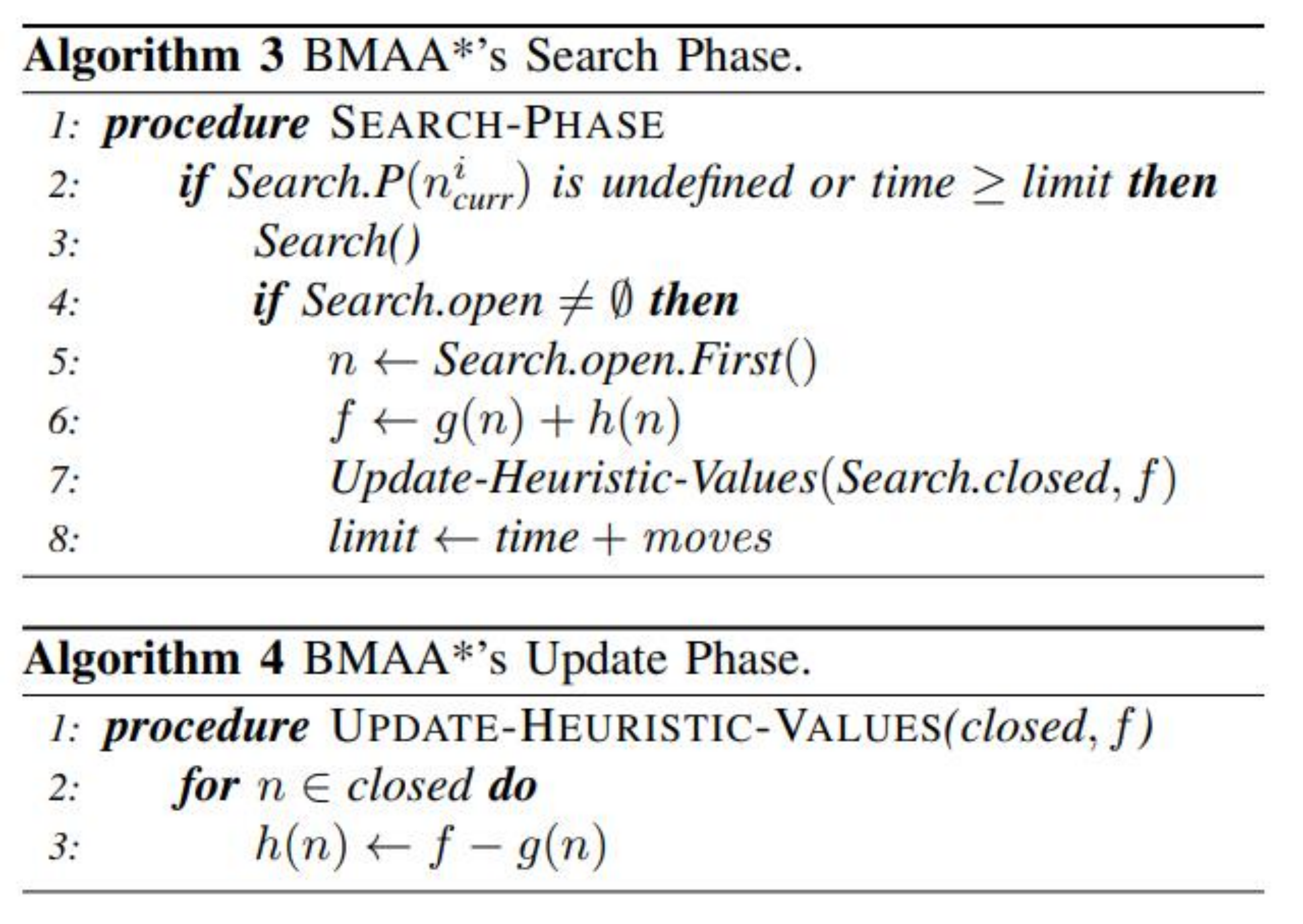


Algorithm 2 BMAA*'s NPC Controller.

```
procedure NPC-CONTROLLER($A$)
    for all $a^i \in A$ do
        $a^i$.Search-Phase()
    for all $a^i \in A$ do
        if $a^i.P(n^i_{curr})$ is defined then
            $n \leftarrow a^i.P(n^i_{curr})$
            if $push \wedge n$ is blocked by agent $a^j$ then
                $a^j$.PushAgent()
            if $n$ is not blocked by an agent then
                $a^i$.MoveTo($n$)
    $time \leftarrow time + 1$
```

BMAA* - Search & RTAA* update

- Find path if:
  - *Path is undefined*
    - Agent was pushed away from path
  - *Executed limited amount of moves*
- Update heuristic by *f-value* of to be expanded node
  - *Admissibility is preserved*
  - *Consistency is preserved*
  - *Goal will be reached*



Algorithm 3 BMAA*'s Search Phase.

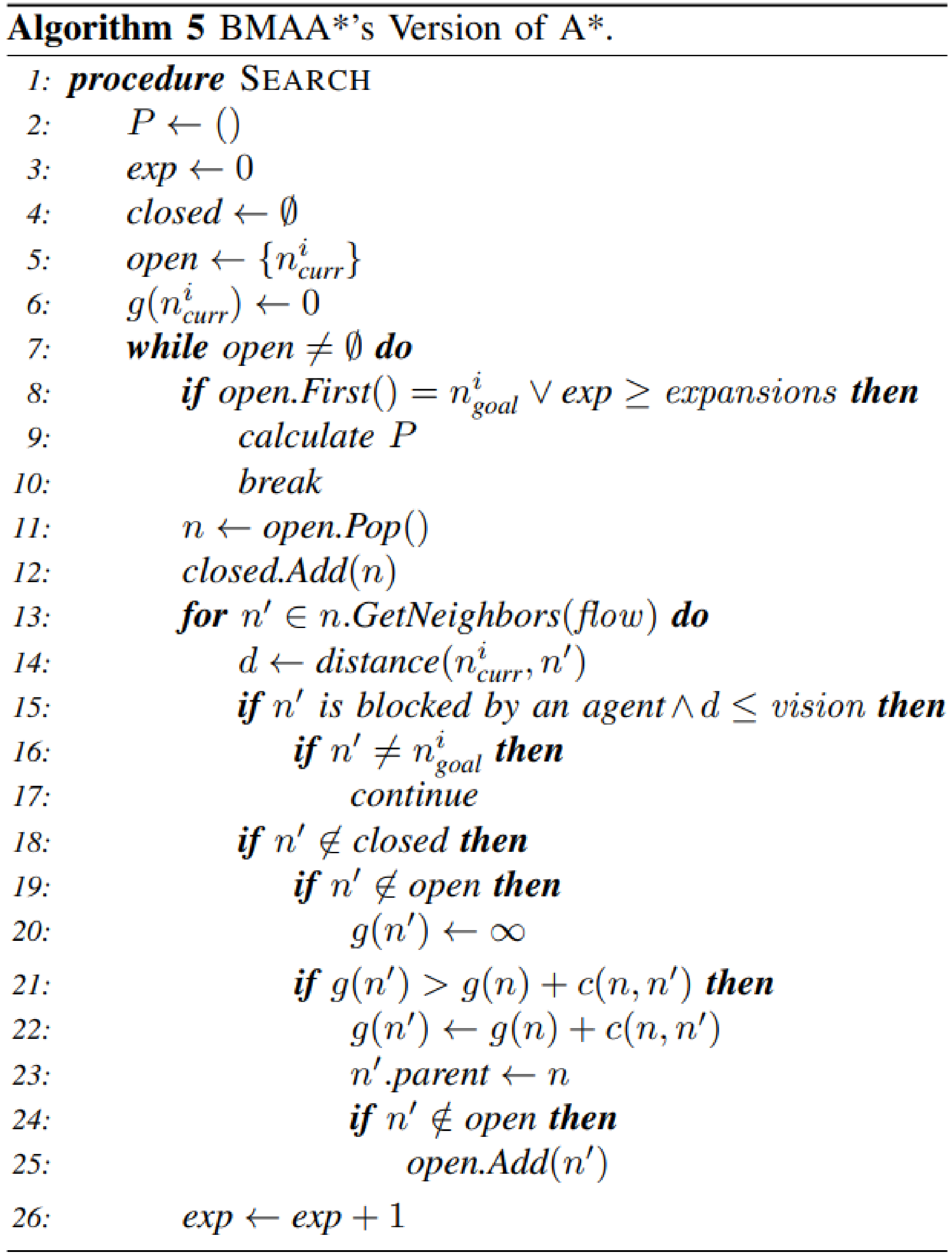
```
procedure SEARCH-PHASE
    if $Search.P(n^i_{curr})$ is undefined or $time \geq limit$ then
        Search()
        if $Search.open \neq \emptyset$ then
            $n \leftarrow Search.open.First()$
            $f \leftarrow g(n) + h(n)$
            Update-Heuristic-Values$(Search.closed, f)$
            $limit \leftarrow time + moves$
```

Algorithm 4 BMAA*'s Update Phase.

```
procedure UPDATE-HEURISTIC-VALUES(closed, f)
    for $n \in closed$ do
        $h(n) \leftarrow f - g(n)$
```

BMAA* - RTAA*

- Each agent has his **own** heuristic values
- Obtained path is only **approximation**
- **Get Neighbors**
  - *Nodes not blocked by stationary obstacle*
  - *If **flow** is **True***
    - Only neighbors from flow annotated graph
    - Generated lazily
    - Cached for later use



Algorithm 5 BMAA*'s Version of A*.

```
procedure SEARCH
    $P \leftarrow ()$
    $exp \leftarrow 0$
    $closed \leftarrow \emptyset$
    $open \leftarrow \{n^i_{curr}\}$
    $g(n^i_{curr}) \leftarrow 0$
    while $open \neq \emptyset$ do
        if $open.First() = n^i_{goal} \lor exp \geq expansions$ then
            calculate $P$
            break
        $n \leftarrow open.Pop()$
        $closed.Add(n)$
        for $n' \in n.GetNeighbors(flow)$ do
            $d \leftarrow distance(n^i_{curr}, n')$
            if $n'$ is blocked by an agent $\land d \leq vision$ then
                if $n' \neq n^i_{goal}$ then
                    continue
            if $n' \notin closed$ then
                if $n' \notin open$ then
                    $g(n') \leftarrow \infty$
                if $g(n') > g(n) + c(n, n')$ then
                    $g(n') \leftarrow g(n) + c(n, n')$
                    $n'.parent \leftarrow n$
                    if $n' \notin open$ then
                        $open.Add(n')$
        $exp \leftarrow exp + 1$
```

EXPERIMENTS

Evaluated Algorithms

Algorithms

- FAR
- A*- Replan
- BMAA*
- BMAA*-c
- BMAA*-f
- BMAA*-f-c

-f => Push = True

-c => Flow = True

Parameters

- Octile heuristic
- Time limit of *30* seconds
- FAR & A* Replan
  - *Reservation size = 3*
- BMAA*
  - *Expansions = 32*
  - *Moves = 32*
  - *Vision = sqrt 2*
  - *Push = False*
  - *Flow = False*

Completion rates

- 3 maps from
  - *Dragon Age: Origins*
  - *WarCraft III*
  - *Baldur's Gate II*
- Number of agents
  - *from 25 to 400 in increments of 25*
  - *from 400 to 2000 in increments of 200*

Observation:

- Noticible change around 200 agents

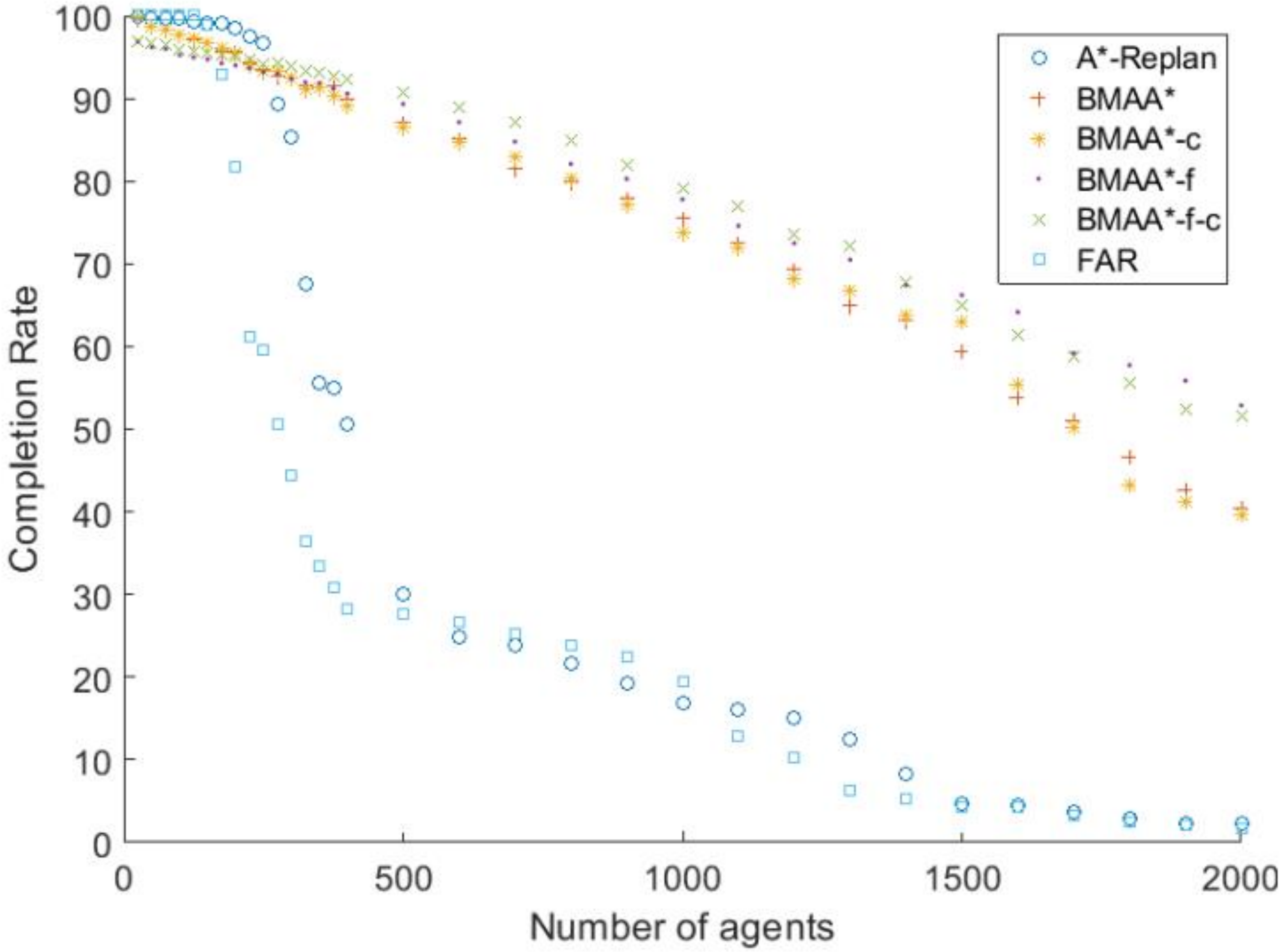


Fig. 2: Completion rates averaged over all MAPF instances.

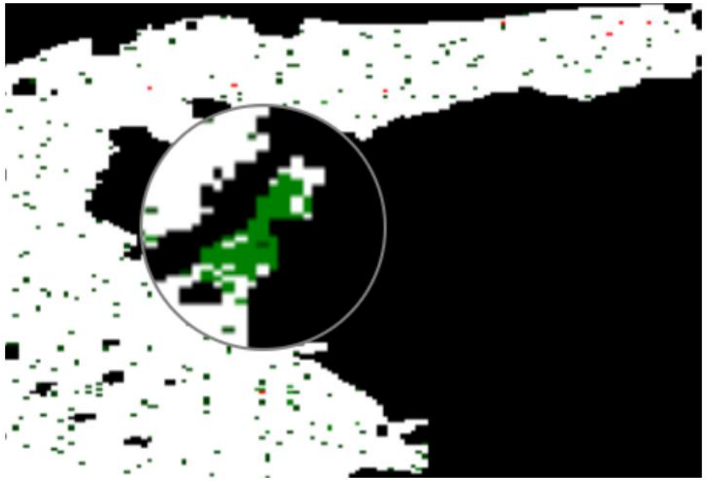


Fig. 4: Issue for BMAA*: Dead ends.

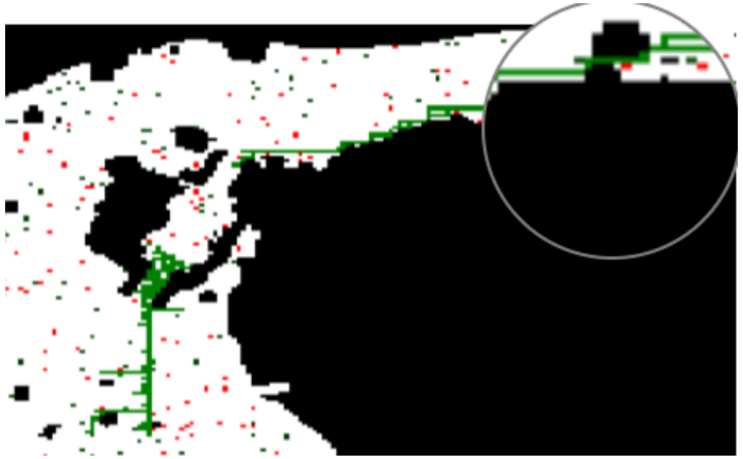


Fig. 3: Issue for FAR: One-cell-wide corridors.

FAR vs BMAA*

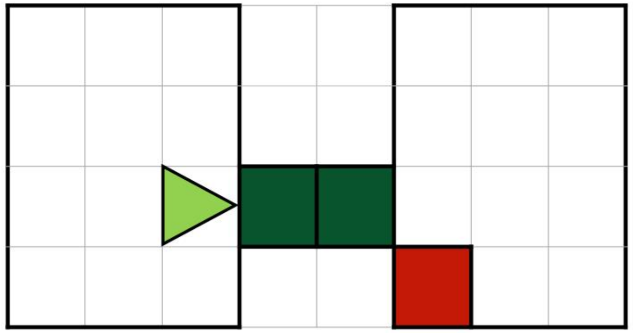


Fig. 5: Unsolvable MAPF instance for the BMAA* versions, where the triangular agent has to move to its red goal location while the green agents are already at their own goal locations in a one-cell-wide corridor.

FAR

- Sharing paths
  - *Congestion in choke points*

BMAA*

- Longer paths
  - *Agents avoids each other*
- Dead ends
- Incompletes

Results

- Best results in each row are in **bolt**
- Results in TABLE II and TABLE III are from runs with **at most *200*** agents**.**
- **Undefined** completion time was set to *30* seconds

Observation:

- BMAA* performs really good on DAO-lak307d map

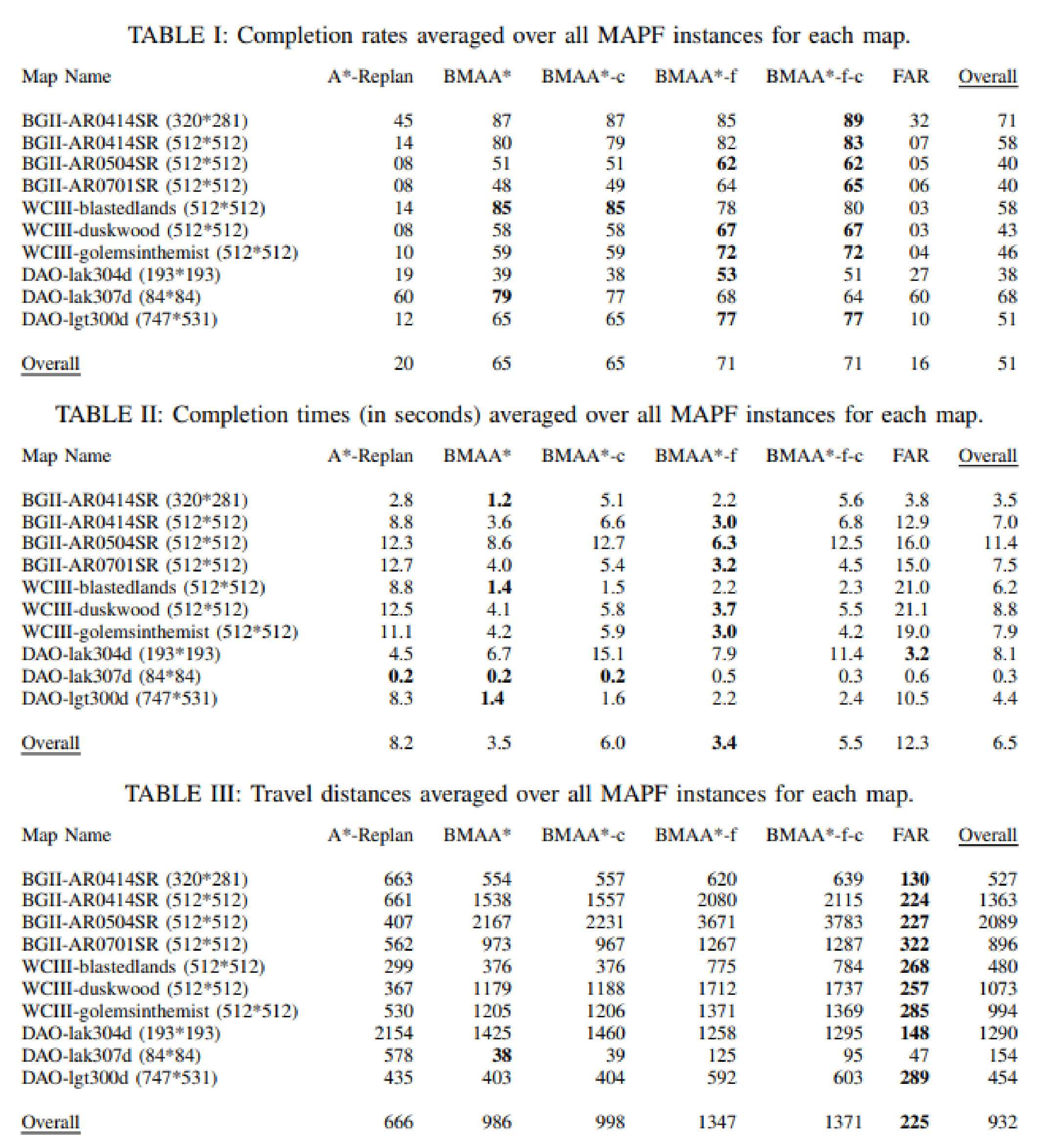


TABLE I: Completion rates averaged over all MAPF instances for each map.

Map Name	A*-Replan	BMAA*	BMAA*-c	BMAA*-f	BMAA*-f-c	FAR	Overall
BGII-AR0414SR (320*281)	45	87	87	85	**89**	32	71
BGII-AR0414SR (512*512)	14	80	79	82	**83**	07	58
BGII-AR0504SR (512*512)	08	51	51	**62**	**62**	05	40
BGII-AR0701SR (512*512)	08	48	49	64	**65**	06	40
WCIII-blastedlands (512*512)	14	**85**	**85**	78	80	03	58
WCIII-duskwood (512*512)	08	58	58	**67**	**67**	03	43
WCIII-golemsinthemist (512*512)	10	59	59	**72**	**72**	04	46
DAO-lak304d (193*193)	19	39	38	**53**	51	27	38
DAO-lak307d (84*84)	60	**79**	77	68	64	60	68
DAO-lgt300d (747*531)	12	65	65	**77**	**77**	10	51
Overall	20	65	65	71	71	16	51

TABLE II: Completion times (in seconds) averaged over all MAPF instances for each map.

Map Name	A*-Replan	BMAA*	BMAA*-c	BMAA*-f	BMAA*-f-c	FAR	Overall
BGII-AR0414SR (320*281)	2.8	**1.2**	5.1	2.2	5.6	3.8	3.5
BGII-AR0414SR (512*512)	8.8	3.6	6.6	**3.0**	6.8	12.9	7.0
BGII-AR0504SR (512*512)	12.3	8.6	12.7	**6.3**	12.5	16.0	11.4
BGII-AR0701SR (512*512)	12.7	4.0	5.4	**3.2**	4.5	15.0	7.5
WCIII-blastedlands (512*512)	8.8	**1.4**	1.5	2.2	2.3	21.0	6.2
WCIII-duskwood (512*512)	12.5	4.1	5.8	**3.7**	5.5	21.1	8.8
WCIII-golemsinthemist (512*512)	11.1	4.2	5.9	**3.0**	4.2	19.0	7.9
DAO-lak304d (193*193)	4.5	6.7	15.1	7.9	11.4	**3.2**	8.1
DAO-lak307d (84*84)	**0.2**	**0.2**	**0.2**	0.5	0.3	0.6	0.3
DAO-lgt300d (747*531)	8.3	**1.4**	1.6	2.2	2.4	10.5	4.4
Overall	8.2	3.5	6.0	**3.4**	5.5	12.3	6.5

TABLE III: Travel distances averaged over all MAPF instances for each map.

Map Name	A*-Replan	BMAA*	BMAA*-c	BMAA*-f	BMAA*-f-c	FAR	Overall
BGII-AR0414SR (320*281)	663	554	557	620	639	**130**	527
BGII-AR0414SR (512*512)	661	1538	1557	2080	2115	**224**	1363
BGII-AR0504SR (512*512)	407	2167	2231	3671	3783	**227**	2089
BGII-AR0701SR (512*512)	562	973	967	1267	1287	**322**	896
WCIII-blastedlands (512*512)	299	376	376	775	784	**268**	480
WCIII-duskwood (512*512)	367	1179	1188	1712	1737	**257**	1073
WCIII-golemsinthemist (512*512)	530	1205	1206	1371	1369	**285**	994
DAO-lak304d (193*193)	2154	1425	1460	1258	1295	**148**	1290
DAO-lak307d (84*84)	578	**38**	39	125	95	47	154
DAO-lgt300d (747*531)	435	403	404	592	603	**289**	454
Overall	666	986	998	1347	1371	**225**	932

SUMMARY

Summary for BMAA*

- Suitable MAPF algorithm for NPCs in video games
- Uses
  - *RTAA**
  - *Central NPC controller*
- Suffers form
  - *Dead ends*
  - *Longer paths*

- Can deal with
  - *Limited amount of time*
  - *Re-tasking*
  - *Unknown map*
  - *Dynamically changing map*
  - *Restricted agent communication*
  - *Non-complete control*

Sources

- https://cpb-us-w2.wpmucdn.com/sites.wustl.edu/dist/b/810/files/2018/08/cig18-bmaa-25hIn9r.pdf
- https://www.aaai.org/Papers/ICAPS/2008/ICAPS08-047.pdf
- https://www.aaai.org/Papers/AIIDE/2005/AIIDE05-020.pdf
- https://www.aaai.org/Papers/Workshops/2006/WS-06-11/WS06-11-010.pdf
- https://github.com/igrek51/coop-pathfinder
- https://starcraft2.com/en-us/
- Images from: https://www.researchgate.net/

THANK YOU FOR YOUR ATTENTION