Any-Angle Pathfinding for Multiple Agents Based on SIPP Algorithm

Konstantin Yakovlev[(1),(2)] and **Anton Andreychuk**[(3),(1)]

yakovlev@isa.ru andreychuk@mail.com

(1) Federal Research Center "Computer Science and Control" of Russian Academy of Sciences
(2) National Research University Higher School of Economics
(3) Peoples' Friendship University of Russia

This work was supported by the Russian Science Foundation (Project No.16-11-00048)

1. Problem Statement

Environment
- Bounded rectangle
  - Tessellated to the square grid
  - Grid cell size is $2r$

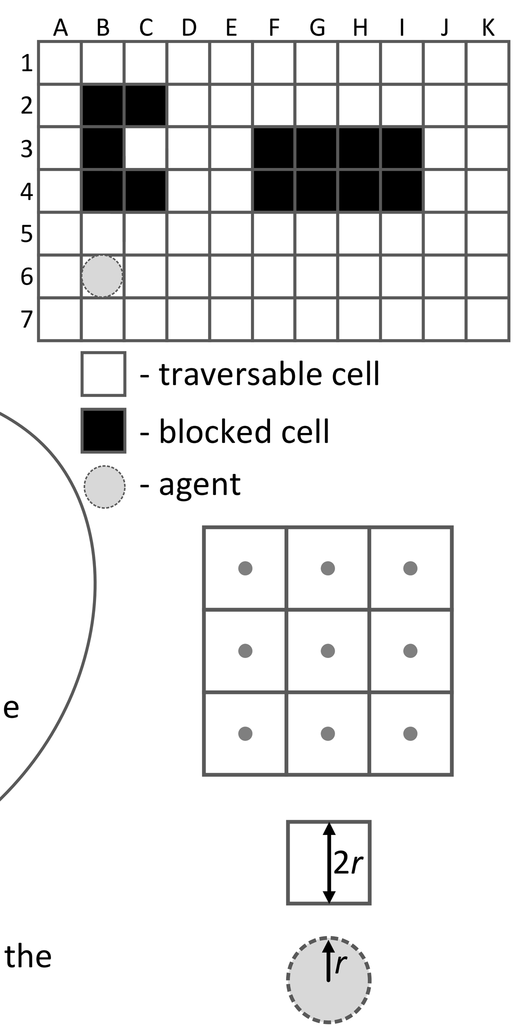
Agents
- Open disks of radii r
- Agents positions = centers of the cells

Actions
- Move
  - Any-angle moves allowed
  - Moving speed is constant and equals $2r$ per time step
    Time is equivalent to distance traveled
  - Inertial effects are neglected
- Wait (before the move)
  - An arbitrary amount of time
  - Occupying the cell

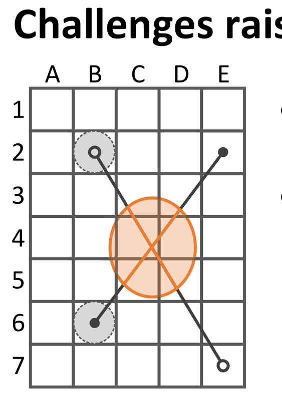
Task

Given start and goal locations find a set of feasible conflict-free trajectories

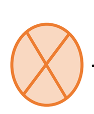
Solution cost = sum of individual costs



Challenges raised by any-angle moves



- Conflict is not associated with the exact grid cell or edge
- Conflict may occur at any moment of contiguous timeline


– continuous conflict zone induced by any-angles moves

2. Suggested Approach

Decoupled, e.g. **prioritized** planning
- Agents are assigned unique priorities
- Trajectories are planned one-by-one

  Previously planned trajectories = moving obstacles

Individual planner (**AA-SIPP**)

Builds on top of SIPP algorithm [1]

Handles any-angle moves

Incorporates "avoid low-priority start locations" strategy [2]

In order to guarantee the completeness of the multi-agent planner in specific cases

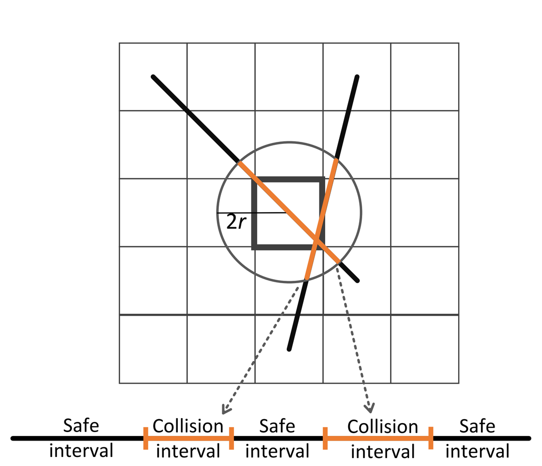
Resultant multi-agent planner **AA-SIPP(m)**
- Is complete under well-defined conditions, which often hold in practice

  Is not compete in general

3. SIPP/AA-SIPP Search Space

Search node identifier ***s*=[*cfg*, *interval*]**
- *cfg* – configuration (grid cell)
- *interval* – the contiguous period of time, during which an agent may safely occupy *cfg*

Search node associated data *g*(*s*), *h*(*s*), *parent*(*s*), *time*(*s*)
- *time* – earliest possible arrival time to *cfg* via *parent* (must be within *interval*)
- *g* – cost of the best path to *s* found so far
- *h* – heuristic estimate of cost-to-go



4. Any-Angle SIPP (AA-SIPP)

Algorithm: AA-SIPP

```
g(s_start) = 0; OPEN = ∅;
insert s_start into OPEN with f(s_start) = h(s_start);
while s_goal is not expanded do
    remove s with the smallest f-value from OPEN;
    for each cfg in NEIGHBORS(s.cfg) do
        successors = getSuccessors(cfg, s);
        if cfg is reachable from parent(s).cfg then
            successors = successors ∪
              getSuccessors(cfg, parent(s));
        for each s' in successors do
            if s' was not visited before then
                f(s') = g(s') = ∞;
            if g(s') > g(s) + c(s, s') then
                g(s') = g(s) + c(s, s');
                updateTime(s');
                insert s' into OPEN with
                  f(s') = g(s') + h(s');

Function getSuccessors(cfg, s)
    successors = ∅;
    m_time = time to reach cfg from s.cfg;
    start_t = time(s) + m_time;
    end_t = endTime(interval(s)) + m_time;
    intervals = get all safe intervals for cfg;
    for each safe interval i in intervals do
        if startTime(i) > end_t or
          endTime(i) < start_t then
            continue;
        t=earliest arrival time from s to cfg during
          interval i with no collisions;
        if t does not exist then
            continue;
        s'=state of configuration cfg with interval i
          and time t;
        insert s' into successors;
    return successors;
```

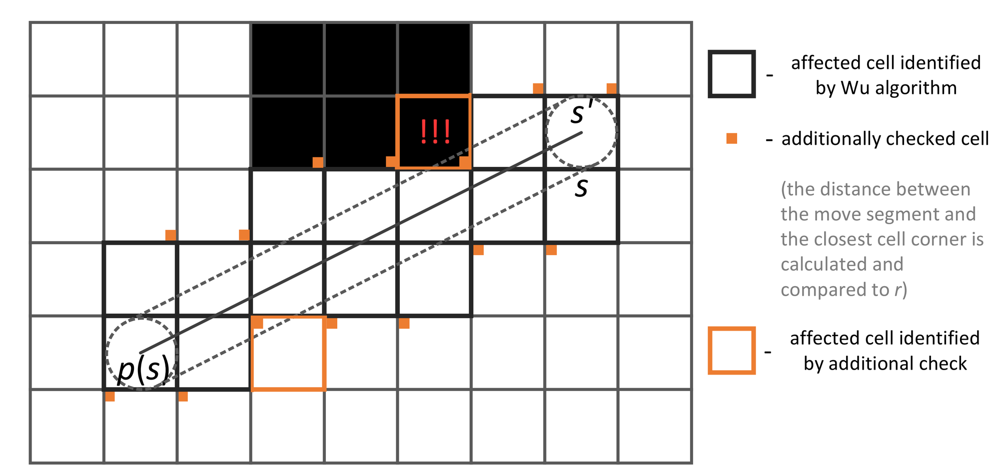
AA-SIPP

- Modified SIPP algorithm
  - Uses shortcuts (like Theta* [3])
- Is complete w.r.t. grid discretization of the workspace
- Finds solutions which cost does not exceed the cost of regular SIPP solutions

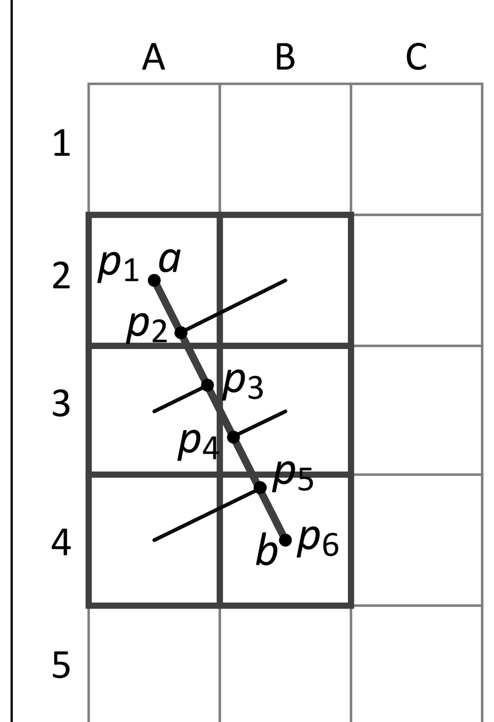
Validating the shortcut

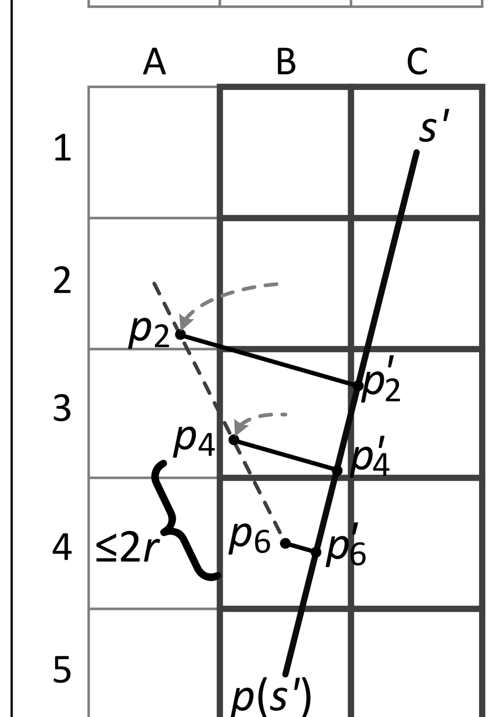
- Identify all cells affected by the moving agent
  - Wu algorithm [4]

    plus
  - Additional checks
- Check whether the identified cells are all unblocked
  - If they are – the shortcut move is valid
  - If they are not – the shortcut move is invalid



5. Estimating Earliest Arrival Times





Representation of moving obstacles

- Moving obstacles' trajectories are split into the sequences of constraints: *constraint* = [*p*, *time*(*p*)],

  p – point on the trajectory

  time(*p*) – moment of time when the obstacle passes *p*
- Constraints are tied to the grid cells affected by moving obstacle
- Several constraints are stored if moving obstacle waits at the cell
- Distance (and time difference) between each two consecutive constraints does not exceed $2r$

Algorithm : Calculation of EAT

```
CELLS = get all cells affected by <p(s'), s'>;
INTERVALS = ∅;
for each cell in CELLS do
    for each constraint ∈ cell do
        if distance(constraint, <p(s'), s'>) < 2r then
            offset = distance(s', the closest to constraint point on <p(s'), s'>);
            collision_interval = [time(constraint) − 4r +
              offset, time(constraint) + 4r + offset];
            INTERVALS = INTERVALS ∪ collision_interval;
        else
            continue;
t = time(p(s')) + distance(p(s'), s');
INTERVALS = combine all intersecting collision intervals in INTERVALS;
for each collision_interval in INTERVALS do
    if t ∈ collision_interval then
        t = endTime(collision_interval);
if t ∉ safe interval of s' then
    s' is not reachable from p(s') during the consistent safe interval;
else
    t is the earliest arrival time;
```

6. AA-SIPP(m)

Algorithm: AA-SIPP(m)

```
PATHS = ∅;
determine agents' priorities;
for each agent w.r.t priorities do
    path = find path for agent with AA-SIPP;
    if path ≠ ∅ then
        PATHS = PATHS ∪ path;
        transform path into the sequence of constraints
          and add them to the search space;
return PATHS;
```

AA-SIPP(m) is complete in well-formed infrastructures (WFIs)

WFI is a planning instance having the following property:

Individual path for each agent exists:
- with at least *r*-clearance with respect to the static obstacles
- with at least 2*r*-clearance to any other start or goal location

7. Experimental Evaluation

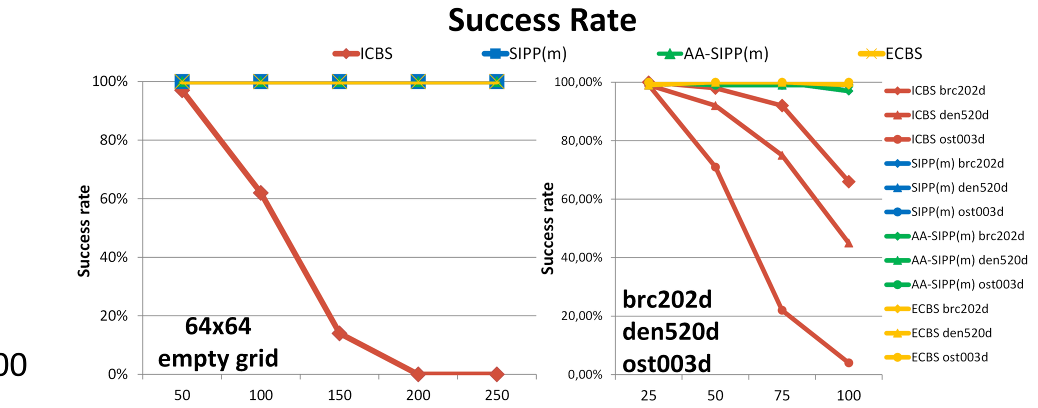
Algorithms
- ICBS [5]
- SIPP(m)
- AA-SIPP(m)
- ECBS [6]

The 1st experiment
- Empty 64x64 grid
- Number of agents: 50, 100, 150, 200 and 250
- 100 instances per each number of agents (500 in total)
- Each instance is a well-formed infrastructure
- Runtime limit – 5 minutes

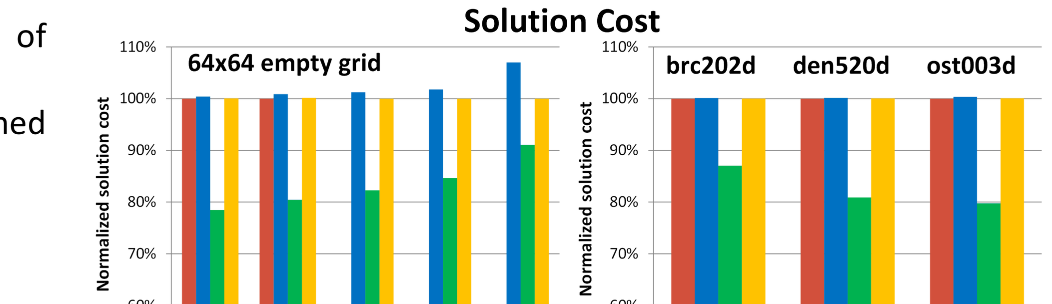
The 2nd experiment
- 3 grid maps from MovingAI benchmark set (Dragon Age: Origins):

  brc202d den520d ost003d
- Number of agents: 25, 50, 75 and 100
- 100 instances per map per number of agents (1200 in total)
- Runtime limit – 5 minutes

Success Rate



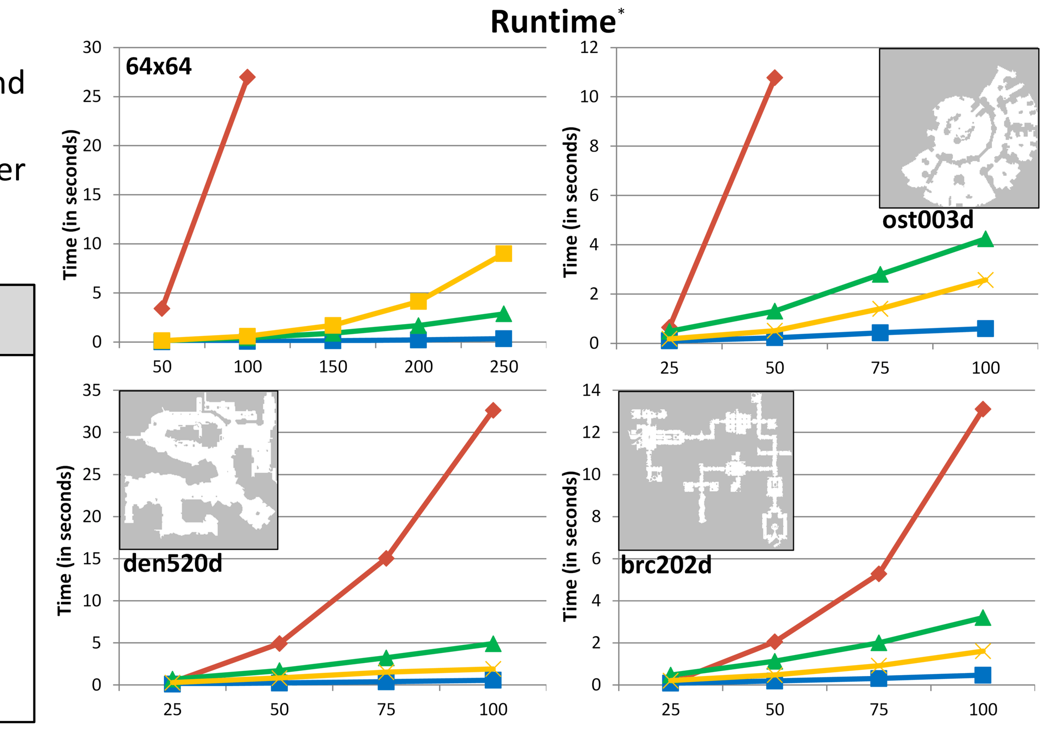
Solution Cost



	50	100	150	200	250
ICBS	100,00%	100,00%	–	–	–
SIPP(m)	100,39%	100,85%	101,23%	101,77%	107,00%
AA-SIPP(m)	78,48%	80,44%	82,27%	84,66%	91,04%
ECBS	100,05%	100,15%	100,00%	100,00%	100,00%

	brc202d	den520d	ost003d
ICBS	100%	100%	100%
SIPP(m)	100,09%	100,14%	100,35%
AA-SIPP(m)	87,03%	80,88%	79,74%
ECBS	100,02%	100,04%	100,07%

Runtime*



* runtimes are not directly comparable due to different implementations of SIPP(m), AA-SIPP(m) and ICBS, ECBS.

Results

- AA-SIPP(m) solutions are up to 22% better (solution costs are lower)
- AA-SIPP(m) scales well to large number of agents
- AA-SIPP(m) success rate exceeds 97%

AA-SIPP(m) sources (C++, GitHub)

Key References

[1] Phillips, M.; and Likhachev, M. 2011. SIPP: Safe interval path planning for dynamic environments. In Proceedings of The 2011 IEEE International Conference on Robotics and Automation (ICRA-2011), 5628–5635.

[2] Čáp, M.; Novák, P.; Kleiner, A.; and Selecký, M. 2015. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. IEEE Transactions on Automation Science and Engineering 12(3):835–849.

[3] Nash, A.; Daniel, K.; Koenig, S.; and Felner, A. 2007. Theta*: Any-angle path planning on grids. In Proceedings of The 22nd AAAI Conference on Artificial Intelligence (AAAI-2007), 1177–1183.

[4] Wu, X. 1991. An efficient antialiasing technique. In Proceedings of The 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH-1991), 143–152.

[5] Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Betzalel, O.; Tolpin, D.; and Shimony, E. 2015. ICBS: Improved conflict-based search algorithm for multiagent pathfinding. In Proceedings of The 24th International Joint Conference on Artificial Intelligence (IJCAI-2015), 740–746.

[6] Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In Proceedings of The 7th Annual Symposium on Combinatorial Search (SoCS-2014), 19–27.