Final Report

Intelligent Systems Project

Ivan Pavlyk, Abay Kulamkadyr

August 20th, 2022

Improved Conflict-Based Search With Heuristics

Introduction

For our project, we expanded on our previous individual project on Multi-Agent Path Finding algorithms. We decided to improve our high-level conflict-based search algorithm by applying heuristics. The theoretical basis for our project is *Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search* [Li et al.,2019]. We first implemented heuristics based on cardinal conflicts to guide the high-level algorithm, applied the technique for CBS with standard and disjoint splitting, and measured their performances in terms of the number of expanded and generated nodes and the runtime of the algorithms. Then we implemented improved heuristics as proposed in the paper and measured the performance in a similar matter for standard and disjoint splitting.

Implementation

We closely followed the paper [Li et al.,2019] so our implementation of the algorithms closely follows the descriptions of the algorithms in the paper.

ICBS

```
for each agent among agents involved in the problem:
      path, mdd = find initial paths and mdds using a_star
h = compute heuristic
while open list is not empty:
      get the next node from the open list
      if node has no collisions
            return solution
      else
            choose the first collision and convert to a list of
            constraints
            add a new child node to the open list for each constraint
end while
```

CG heuristic

```
The high level of CBS always chooses to expand the CT node
N with the smallest N.cost.
if N.solution contains 1 cardinal conflict then
      h_value = 1
if N.solution contains multiple cardinal conflicts then
      CG = build_conflict_graph
h = minimum_vertex_cover(CG)
```

DG heuristic

```
if N.solution contains 1 cardinal conflict then
      h_value = 1
if N.solution contains multiple cardinal conflicts then
      DG = build_dependency_graph
h = minimum_vertex_cover(DG)
```

WDG heuristic

```
if N.solution contains 1 cardinal conflict then
     h_value = 1
if N.solution contains multiple cardinal conflicts then
     WDG = build_edge_weighted_dependency_graph
h = minimum_weighted_vertex_cover(WDG)
```

build_conflict_graph

```
for each collision among collisions:
      if a cardinal conflict exists between 2 agents for collision:
           conflict_graph(agent_1, agent_2) = ‘conflict’
return conflict_graph
```

build_dependency_graph

```
for each collision among collisions:
      if 2 agents for collision are dependent:
           dependency_graph(agent_1, agent_2) = ‘dependent’
return dependency_graph
```

build_edge_weighted_dependency_graph

```
for each collision among collisions:
     if 2 agents for collision are dependent:
          find sum of minimal conflict-free paths for both agents
          find sum of individual optimal path costs for both agents
          cost = sum_optimal - sum_conflict_free
          weighted_dependency_graph(agent_1, agent_2) = cost
return weighted_dependency_graph
```

minimum_vertex_cover

```
result = empty set
while set E has edges do
      Pick an arbitrary edge (u, v) from set E and add 'u' and 'v'
      to result
      Remove all edges from E which are either incident on u or v
end while
return result
```

minimum_edge_weighted_vertex_cover

```
while set E has edges do
      for i = 1 to n
            for j = 1 to n
                  d(v_i) = Σ(a_ij)
      for i = 1 to n
            for j = 1 to n
                  s(v_i) = d_G(v_j)
      for i = 1 to n
            r(v_i) = s(v_i)d(v_i)/w(v_i)
      add the vertex v_i, having the maximum value of r(v_i),
      into the vertex cover V_c
      V_c =  V_c union v_i
      delete N[v] from G

      for i = 1 to n
            if v_i in V_c
                  v_i = 1
            else
                  v_i = 0
end while
```

Methodology

First experiment: we are going to run ICBS with standard splitting with the following heuristics:

- Prioritizing Cardinal Conflicts
- CG heuristic
- DG heuristic
- WDG heuristic

We are going to compare their running times, the number of nodes expanded and the number of nodes generated.

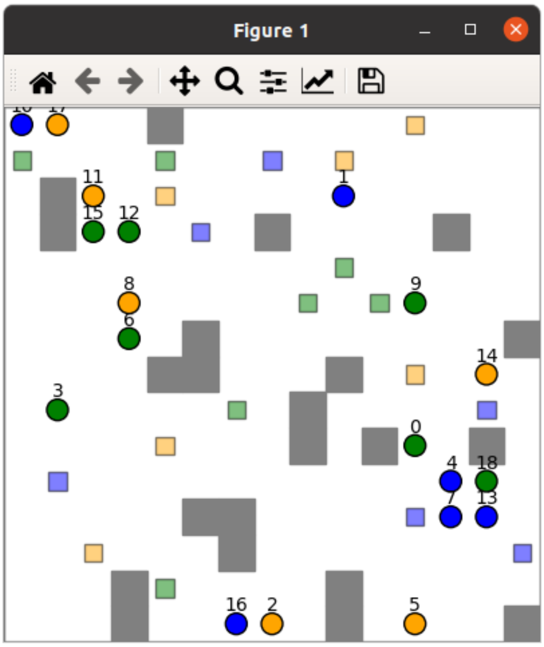
Second experiment: similar to the first but here we are going to use ICBS with disjoint splitting.

Instances that we are going to use are stored in `custominstances` folder.

Experimental Setup

- Programming language: Python 3
- Operating system: Windows 10 Home
- Processor: Intel(R) Core(TM) i7-4790 3.60 GHz
- RAM: 16 GB

Input Instance



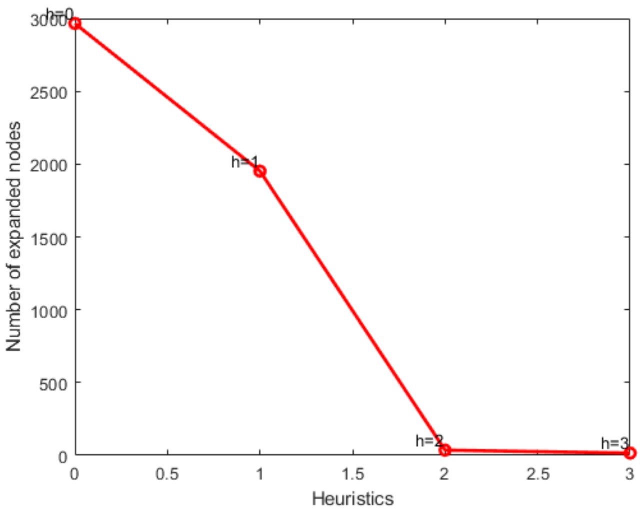
Experimental Results

Experiment 1:

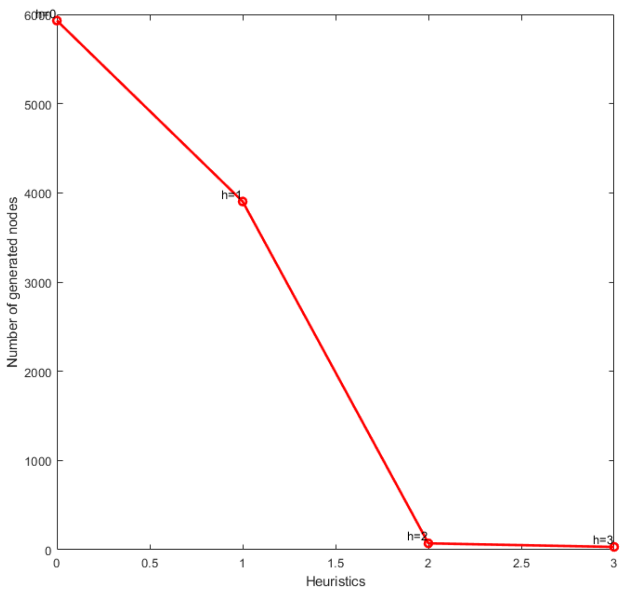
We ran the instance for each heuristic once because for standard splitting the values for running time, expanded nodes and generated nodes would not change because there is no splitting based on randomness.

We were using instance: `custominstances/experiment1.txt`

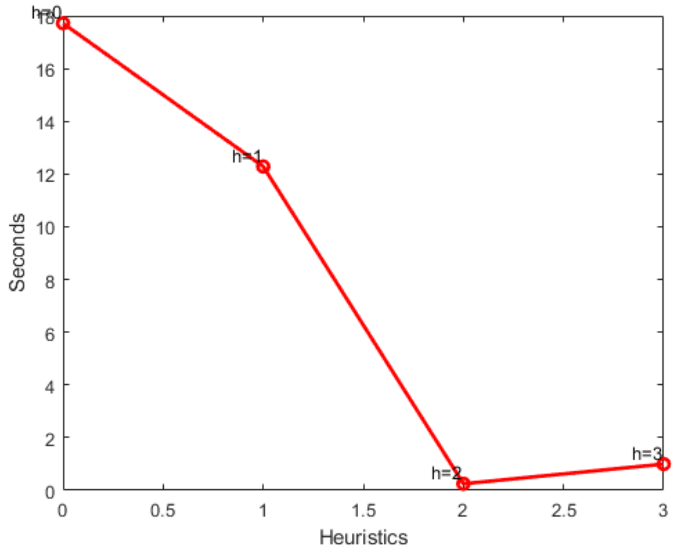
Number of expanded nodes for standard splitting using different heuristics:



Number of generated nodes for standard splitting using different heuristics:



Running time of standard splitting using different heuristics:

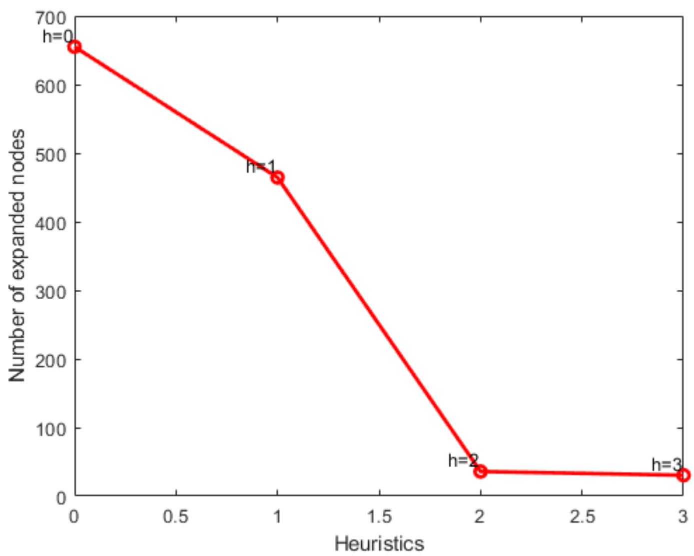


Experiment 2:

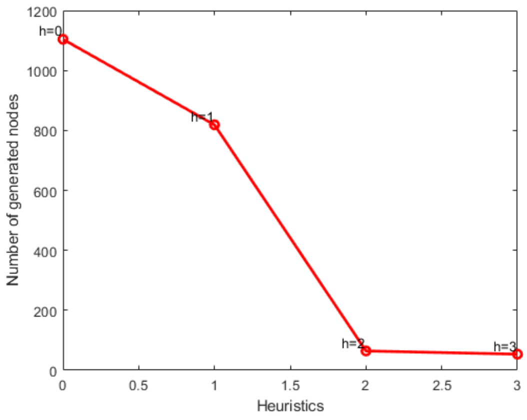
We ran the instance for each heuristic 20 times because there is splitting based on randomness so we need to take average values.

We were using instance: `custominstances/experiment2.txt`

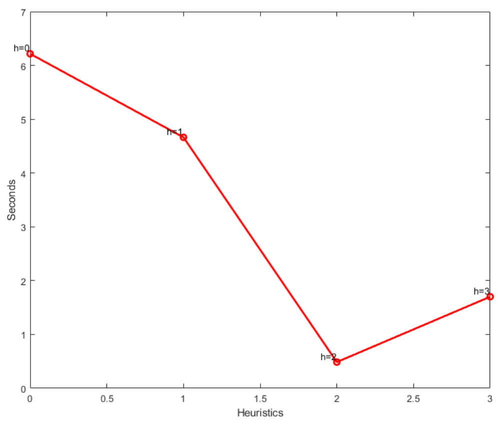
Number of expanded nodes for standard splitting using different heuristics:



Number of generated nodes for standard splitting using different heuristics:



Running time for standard splitting using different heuristics:



Output of the program:

/*Standard Splitting */

standard h = 0:

Run ICBS

Global average cost of paths: 198.0

Time: v

Expanded nodes: 2966

Generated nodes: 5931

Average time: 17.71988296508789

Average expanded nodes: 2966.0

Average generated nodes: 5931.0

Average h-values: 0.0

standard h =1:

Run ICBS

Global average cost of paths: 198.0

Time: 12.284494876861572

Expanded nodes: 1952

Generated nodes: 3903

Average time: 12.284494876861572

Average expanded nodes: 1952.0

Average generated nodes: 3903.0

Average h-values: 0.09684857801691008

standard h =2:

Run ICBS

Global average cost of paths: 198.0

Time: 0.2497403621673584

Expanded nodes: 36

Generated nodes: 71

Average time:

0.2497403621673584

Average expanded nodes: 36.0

Average generated nodes: 71.0

Average h-values: 3.1971830985915495

standard h =3:

Run ICBS

Global average cost of paths: 198.0

Time: 0.9964516162872314

Expanded nodes: 16

Generated nodes: 31

Average time: 0.9964516162872314

Average expanded nodes: 16.0

Average generated nodes: 31.0

Average h-values: 0.6129032258064516

/*Disjoint Splitting */

disjoint h = 0:

Global average cost of paths: 198.0

Time: 124.33529853820801

Expanded nodes: 13103

Generated nodes: 22081

Average time: 6.2167649269104

Average expanded nodes: 655.15

Average generated nodes: 1104.05

Average h-values: 0.0

**Test paths on a simulation

disjoint h = 1:

Global average cost of paths: 198.0

Time: 93.28594708442688

Expanded nodes: 9289

Generated nodes: 16373

Average time: 4.664297354221344

Average expanded nodes: 464.45

Average generated nodes: 818.65

Average h-values: 0.16982363180294804

Test paths on a simulation

disjoint h =2:

Global average cost of paths: 198.0

Time: 9.739197254180908

Expanded nodes: 718

Generated nodes: 1287

Average time: 0.4869598627090454

Average expanded nodes: 35.9

Average generated nodes: 64.35

Average h-values: 3.190864028584857

Test paths on a simulation

disjoint h = 3:

Global average cost of paths: 198.0

Time: 34.03034520149231

Expanded nodes: 609

Generated nodes: 1073

Average time: 1.7015172600746156

Average expanded nodes: 30.45

Average generated nodes: 53.65

Average h-values: 0.3453120667266049

Test paths on a simulation

Conclusion

In this project we improved our high-level conflict-based search algorithm by applying heuristics. We incorporated MDD into our CBS to get ICBS algorithm based on cardinal conflicts to guide the high-level algorithm. We applied the technique for CBS with standard and disjoint splitting. We measured performances of the algorithms in terms of the number of expanded, generated nodes and the runtime. We implemented improved heuristics: CG, DG and WDG. We compared their performances in terms of the number of expanded, generated nodes and the runtime. As we can see from the results h(0) <= h(CG) <= h(DG) <= h(WDG), i.e. the better the heuristic is, the less nodes it expands and generates, while for the WDG it is more expensive to compute the heuristic per node and it performs better on larger input maps.

References

[Li et al., 2019]: J. Li, A. Felner, E. Boyarski, H. Ma, S. Koenig. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *IJCAI-19*, 2019.

[Li et al., 2019]: J. Li. An optimal solver for Multi-Agent Path Finding. https://github.com/Jiaoyang-Li/CBSH2-RTC