MRPPSim: A Multi-Robot Path Planning Simulation

Ebtehal Turki Saho Alotaibi, Hisham Al-Rawi
Computer Science Department, Al Imam Muhammad Ibn Saud Islamic University
Riyadh, SA

Abstract—**Multi-robot path planning problem is an interesting problem of research having great potential for several optimization problems in the world. In multi-robot path planning problem domain (MRPP), robots must move from their start locations to their goal locations avoiding collisions with each other. MRPP is a relevant problem in several domains, including; automatic packages inside a warehouse, automated guided vehicles, planetary exploration, robotics mining, and video games. This work introduces MRPPSim; a new modeling, evaluation and simulation tool for multi-robot path planning algorithms and its applications. In doing so, it handles all the aspects related to the multi-robot path planning algorithms. Through its working, MRPPSim unifies the representation for the input. This algorithm provides researchers with a set of evaluation models with each of them serving a set of objectives. It provides a comprehensive method to evaluate and compare the algorithm's performance to the ones that solve public benchmark problems inas shown in literature. The work presented in this paper also provides a complete tool to reformat and control user input, critical small benchmark, biconnected, random and grid problems. Once all of this is performed, it calculates the common performance measurements of multi-robot path planning algorithms in a unified way. The work presented in this paper animates the results so the researchers can follow their algorithms' executions. In addition, MRPPSim is designed as set of models, each is dedicated to a specific function, this allows new algorithm, evaluation model, or performance measurements to be easily plugged into the simulator.**

Keywords—component; simulation; modeling; evaluation; multi-robot path planning problem; performance measurements

I. Introduction

Formally, Multi-robot path planning problem (MRPP) consists of a graph and a set of robots. In such problems, each robot has to reach its destination in the minimum time with minimum number of moves. MPP is a relevant problem in a wide range of domains, including; automatic packages inside a warehouse [2], automated guided vehicles [3], planetary exploration [4], robotics mining [5], and video games [6].

There are several variants of MRPP algorithms in the literature with their specialized strengths. However, to have clear vision about these algorithms' performance, these need to be evaluated on a unified robust tool. The multi-robot path planning simulation (MRPPSim) is a modeling, evaluation and simulation tool for multi-robot path planning algorithms and its applications. It handles all the aspects related to the multi-robot path planning algorithms. Hence, the researchers need only to worry about their algorithms. MRPPSim aims to provide the researcher set of evaluation models, each of them serves one of the following objectives;

Objective.1: Test and track the algorithm's behavior for specific cases in fully controlled problems.

Objective.2: Track the algorithm's behavior and compare its performance with the algorithms that are already evaluated on predefined small critical problems in [7].

Objective.3: Test the algorithm's behavior on very large graphs when the occupying ratio (robots number/vertices count) increases.

Objective.4: Compare the algorithm's performance to the performance of the algorithms already tested in publically available large scale problems [1] such as [7-9].

Objective.5: Compare the algorithm's performance to the performance of the algorithms solves biconnected graphs [10], [7, 8]

Objective.6: Test the algorithm's behavior on fixed biconnected graph's size when the occupying ratio increases.

Objective.7: Evaluate the algorithm's performance against the algorithms that solve random graphs.

Objective.8: Track the algorithm's behavior on fixed random graph's size when the occupying ratio increases.

Objective.9: Compare the algorithm's performance to the algorithms that solve grid graphs [7, 8, 10].

Objective.10: Test the algorithm behavior on fixed grid graph's size when the occupying ratio increases.

Objective.11: Evaluate the algorithm's performance on biconnected graphs of fixed occupying ratio when the graph's size increases.

Objective.12: Evaluate the algorithm's performance on random graphs of fixed occupying ratio when the graph's size increases.

Objective.13: Evaluate the algorithm's performance on grid graphs of fixed occupying ratio when the graph's size increases.

This paper is organized as follows; after this introduction, the problem is defined in Section II. In Section III, the multi-robot path planning simulation MRPPSim is introduced. In Section IV, MRPPSim objectives are discussed in details. Finally, conclusions and future works are presented in Section V.

II. Problem Statement

Going through the literature related to MRPP algorithm, we found an urgent need of having a graph-based multi-robot path planning simulator that can cover different types of scenarios and problems. The study of all available algorithms implementations and its performance measurements calculations as well was essential. The simulation is to provide a ready implementation of state-of-art algorithms in the field. During this study, the work started by looking into iRRT simulator [11]. iRRT is a simple Java program for simulating that is widely used for robotics path planning algorithm known as Rapidly-exploring Random Tree or RRT. This algorithm was developed by Sertac Karaman et al. [12] and it works for a single-robot configuration. During the process, several extensions were implemented to fit multi-robot RRT requirements. However, the representation of the robots environment was limited to being continuous environment, while we were also interested in graph-based algorithms. This would enable working with problem scenarios and variations that have not been catered for previously.

III. METHODOLOGY

The proposed work, i.e. MRPPSim is written as an open source C++ code[1] designed to be fully modularized so that researchers can have the ability to plug any modifications.

A. Input

The input parameter for MRPPSim can be classified based on the simulation model which will be described in the next section. In our implementation, the user is initially required to select the simulation model and the algorithm to be used. Based on that, the user can enter or select the main two input for any algorithm. These inputs include; the representation of the environment in the form of undirected graph; $G(V,E)$, whatever the simulation model or the algorithm, the graph that would be sent to the algorithm in XML format representation describing numbered vertices V, and the linking edges E (Fig.1). The second input is the instance; $I(R,$ *Locations*), which is the representation of the number of robots; R, and robots' configuration within the environment; *Locations*, the robots configuration is the definition of the robots start and goal locations in graph's vertices term. (Fig.2).

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
<key id="key0" for="node" attr.name="Coordinate"
attr.type="string" />
<key id="key1" for="edge" attr.name="Weight"
attr.type="double" />
<graph id="G" edgedefault="undirected"
parse.nodeids="canonical" parse.edgeids="canonical"
parse.order="nodesfirst">
<node id="n0">
<data key="key0">0.0 0.0 0.0</data>
</node>
<node id="n1">
<data key="key0">1.0 0.0 0.0</data>
</node>
<node id="n2">
<data key="key0">2.0 0.0 0.0</data>
</node>
<edge id="e0" source="n0" target="n1">
<data key="key1">1</data>
</edge>
<edge id="e2" source="n1" target="n2">
<data key="key1">1</data>
</edge>
<edge id="e3 source="n2" target="n0">
<data key="key1">1</data>
</edge>
  </graph>
 </graphml>
```

Fig. 1. The environment representation is a graph in xml format

```
graph =//Graph.xml
agent: start = 0, goal = 2
agent: start = 2, goal = 3
```

Fig. 2. The robots (agents) start and goal locations (instance) representation is a text file in specific format

B. Simulation models

As already mentioned, MRPPSim is written as a C++ code that offers a set of simulation models, Each of these can serve as set of objectives in our problem definition. This section will present an overview of each simulation model in detail starting with the user input until the algorithm inputs which is the graph and the instance, described earlier, in the following form;

AlgorithmX (G, I)

1) Fully controlled model

In this selection, the user will be able to set the attributes of both graph and the instance to be tested. Once the user enters the graph G(user defined, user defined) attributes and the instance I(user defined) descriptions, MRPPSim then converts the input to the form defined in the subsection A, and send these inputs to the selected algorithm as;

AlgorithmX (G(user defined), I (user defined)).

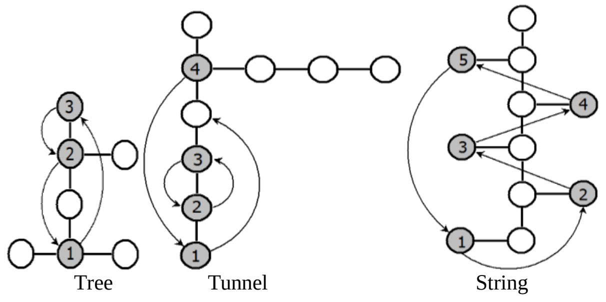
The aim of this model is to provide the user with the ability to track specific cases in fully controlled problem (Objective.1).

2) Small benchmark model

In order to cater for a wider class of instances, MRPPSim has defined six benchmark problems evaluated in [7], a tree with cycle leaves problem evaluated in [11] and a biconnected graph in [10](Fig.3). It is noteworthy that besides the graph G(ready V, ready E), the instance is also defined in these problems as I(ready R, ready *locations*). Therefore, the algorithm input would be;

AlgorithmX(G(ready V, ready E), I(ready R, ready *locations*)).

The aim of this model is to get the user with the ability to compare the algorithm with already evaluated algorithm on these problems (Objective.2).



[1] https://pushandspin.wordpress.com/source-code/

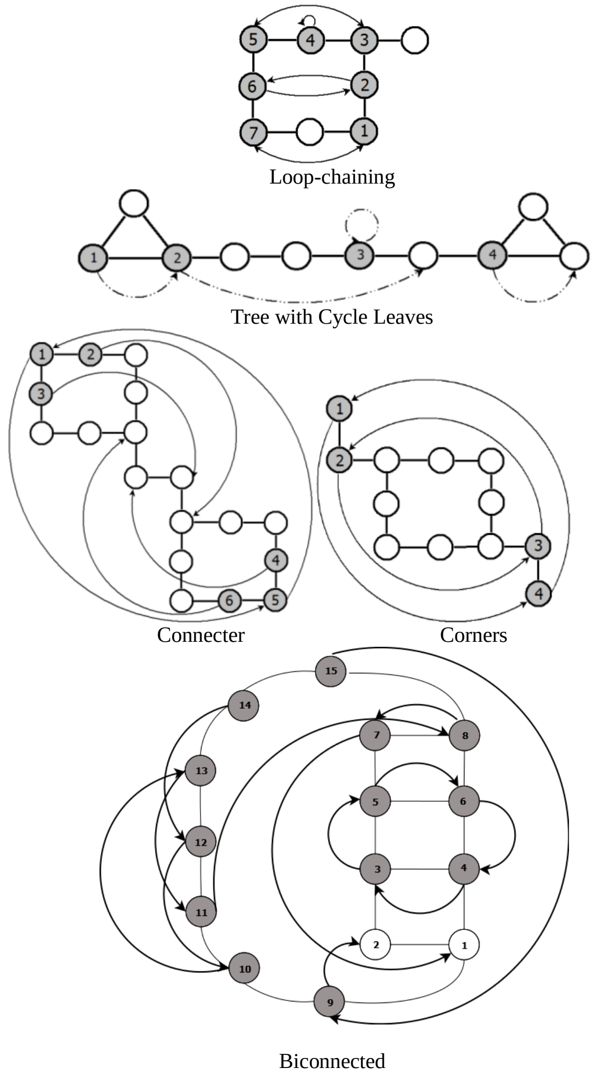


Fig. 3. Small benchmark problems

3) Public benchmark model

Sturtevant in his paper [1] has presented a new repository that has been placed online[2] to improve the evaluation of grid-based problems. The repository allows researchers to use the same problems and test sets, thereby increasing the reproducibility of published results. In the repository, two main sets are described; Commercial Game Benchmarks and Artificial Benchmarks. Each map has been coded in a way to be parsed. The maps are attached with a set of scenarios describing the instances. However, this repository requires a little modification in order to be applicable to the chosen problem domain. This is due to the fact that in some scenarios, there would be multiple robots on the same start and/or goal position. Consequently, instead of using attached scenarios, we have designed random generator. It is used to generate any number of robots for the given graph randomly. The procedures for generating random instances is;

1. Parameter: n: the number of vertices, r: the number of robots.
2. For each r;
    1. Select random node s such that $s \notin$ *start set*
    2. Set s as starting vertex of r.
    3. *start set = start set* ∪ *s*
    4. Select random vertex g such that $s \notin$ *goal set*
    5. Set s as goal vertex of r.
    6. *goal set = goal set* ∪ *g*.

In the Commercial Game Benchmarks, the maps contain areas of land or water and the robots can only move on water. (Fig.4,5). Moreover, in the Artificial Benchmarks, the maps contain areas of passageways and walls. We have designed a parser to read the coded maps and transform them into graphs with each element in the map converted to vertex if it not obstacles (lands/walls). This results in graphs that are characterized by a large set of connected vertices. The parser generates a fixed graph representation; G(coded V, coded E), for every coded map. In addition set of instances describing the start and goal locations would also be generated for each graph randomly by the generator given the number of robots R, minimum number of robots, maximum number of robots and the step size; I (R, $\sum_{i=0}^{max} min + (i * StepSize)$, random locations). Hence, The algorithms input would be;

AlgorithmX (G(coded V,coded E),

$I(R, \sum_{i=0}^{max} min + (i * StepSize)$,random *locations*)).

The aim of this model is to test the algorithm behavior on very large graphs when the occupying ratio (robots number/vertices count) increases (Objective.3) and to compare the algorithm's performance to the performance of the algorithms already tested in these public problems (Objective.4).

4) One-Factor controlled model

While studying Multi-Robot Path Planning problem, we came across two critical factors affecting the algorithms' performance. These are the size of the graph and the occupying ratio (number of robots/ number of vertices). In this section we will overview two type of models; each type controlling one of these critical factor and randomize the other in order to generate graphs and instance files.

a) Fixed-Graph with Variable-Robot model

In this model, number of instances are to be generated for the same graph size. After setting the graph size, the user can set the minimum number of robots, maximum number of robots and the step size to generate required number of instances varying in the occupying ratio.

BICONNECTED GRAPH

The biconnected graph is a connected graph on two or more vertices having no articulation vertices. MRPPSim provides Biconnected graph generator representing $G(V,E_{bi})$ with the number of vertices; V. The generator will generate the biconnected topology by inserting edges E_{bi} between these vertices. In this version of MRPPSim we have used the same biconnected graph generator in [10]. It generates random bi-connected graphs conforming to three parameters; h the number of handles, C_0 the size of the initial cycle, and l the maximum handle length (Fig.6).

[2] http://movingai.com/benchmarks/

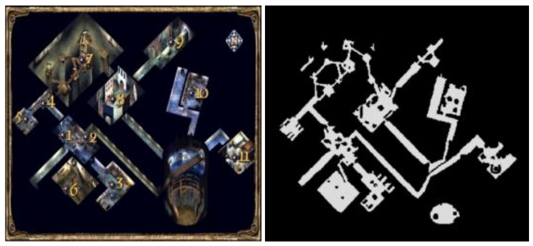


Fig. 4. AR0306SR map of Baldurs Gate II (left), its coded map (right)

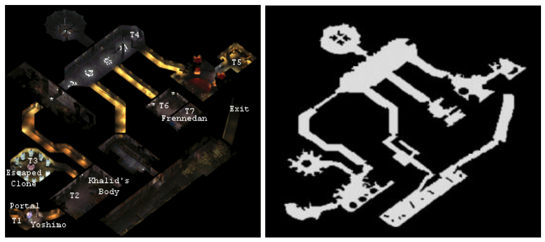


Fig. 5. AR0603SR map of Baldurs Gate II (left), its coded map (right)

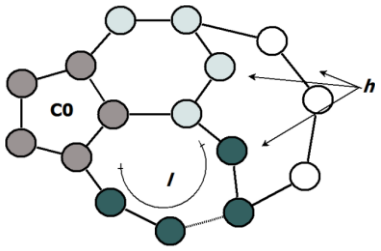


Fig. 6. initial cycle (C0), maximum handle length (l) and number off handles (h) in Surynek grap

Since this model is FG-VR model, the instances *I* need to be generated randomly. The user can set the minimum number of robots, maximum number of robots and the step size to generate a number of instances varying in the occupying ratio; $\sum_{i=0}^{max} min + (i * StepSize)$. However, the robots location; *locations* need to be randomized totally. Furthermore, The algorithm input would be;

AlgorithmX ($G(h, C_0, l)$,

I ($\sum_{i=0}^{max} min + (i * StepSize)$, random *locations*)).

The aim of this model is to provide the researcher with the ability to compare the algorithm's performance to the performance of the algorithms thereby meeting objective 5 and 6 mentioned in earlier sections.

Random Graph

MRPPSim provides random graph generator; $G(V,E)$ given the number of vertices; V, the generator would generate the random topology by inserting edges E_{rand} between these vertices. Since this model is FG-VR model, the instances I will be generated randomly. As has been described earlier, the user can set the minimum number of robots, maximum number of robots and the step size to generate number instances varying in the occupying ratio; $\sum_{i=0}^{max} min + (i * StepSize)$, the robots location; *locations* will be randomized totally. Therefore, the algorithm input would be;

AlgorithmX ($G(V,E_{rand})$,

I ($\sum_{i=0}^{max} min + (i * StepSize)$, random *locations*))

The aim of this model is to provide the researcher with the ability to evaluate the algorithm performance against the performance of algorithms already exusting that solve random graphs (Objective.7) and to track algorithm behavior on fixed random graph's size when the occupying ratio increases. (Objective.8).

Grid Graph

MRPPSim provides random graph generator; $G(V,E)$ given the number of vertices; V, the generator would generate the random topology by inserting edges E_{grid} between these vertices. In this version of MRPPSim, We have used the same generator in [10] to generate grid instances of $n \times n$ size with a constant value of the parameter $(h, C_0, l) = G((n-1)^2-1, 4, 4)$. Since this model is FG-VR model, the instances I would be generated randomly. On the other hand, the user can set the minimum number of robots, maximum number of robots and the step size to generate a number of instances varying in the occupying ratio; $\sum_{i=0}^{max} min + (i * StepSize)$, the robots location; *locations* will be randomized totally. The algorithm input thus would be;

AlgorithmX ($G((n-1)^2-1, 4, 4)$,

I($\sum_{i=0}^{max} min + (i * StepSize)$, random *locations*)).

The aim of this model is to get the researcher with the ability to compare the algorithm's performance to the performance of the algorithms that solve grid graphs, [8], [7]. (Objective.9) and to evaluate the grid graphs when the occupying ratio increase (Objective.10).

b) Fixed-Robot with Variable-Graph model

In this model, a number of graphs will be generated for the same occupying ratio (number of robots/number of vertices). After setting the robot number, the user can set the minimum number of vertices, maximum number of vertices and the step size to generate a number of graphs varying in their size.

Biconnected Graph

Since we have used the same biconnected graph generator as presented in [10], the graph would be generated according to a single variable x that can range from the given minimum number of vertices to the given maximum number of vertices by the given step size. The variable x would be used for all three parameters (number of handles; h, initial cycle size; C_0, and maximum handle length; l). The graph size V hence would be C_0+h*l. The number of empty vertices would be kept fixed to the given occupying ratio. Hence, the instance would be I (*occupying ratio* * V, random *location*). The algorithm input will be:

AlgorithmX ($G(x, x, x)$, I(occupying ratio $*(x+x*x)$, random *location*)),

such that, $x = \sum_{i=0}^{max} min + (i * StepSize)$.

The aim of this model is to get the researcher with the ability to evaluate algorithm's performance on biconnected graphs of fixed occupying ratio when the graph's size increases. (Objective.11).

Random Graph

After the researcher has set the occupying ratio, minimum number of vertices, maximum number of vertices and the step size, the random generator would automatically generate graph *G* ($V_i = \sum_{i=0}^{max} min + (i * StepSize)$, E_{rand}). The instance would be *I* (V_i * *occupying ratio*, random *locations*). Hence, the algorithm input would be:

AlgorithmX (G($V_i = \sum_{i=0}^{max} min + (i * StepSize)$, E_{rand}),

I (V_i* occupying ratio, random locations)).

The aim of this model is to get the researcher with the ability to evaluate the algorithm performance on random graphs of fixed occupying ratio when the graph's size increases. (Objective.12).

GRID GRAPH

After the researcher has set the occupying ratio, the minimum number of vertices, maximum number of vertices and the step size. MRPPSim would automatically generate different grids *G* ($V_i = \sum_{i=0}^{max} min + (\mathrm{i}-1)^2 - 1, 4, 4$), E_{grid}) with the same occupying ratio. The instances would be *I* (V_i* *occupying ratio,* random *locations*). Therefore, the algorithm input would be;

AlgorithmX (G($\sum_{i=0}^{max} min + (\mathrm{i}-1)^2 - 1, 4, 4$), E_{grid}), I(V_i* occupying ratio, random locations)).

The aim of this model is to evaluate the algorithm performance on grid graphs of fixed occupying ratio when the graph's size increases (Objective.13).

C. Output

The most common performance measurements of multi-robot path planning algorithms in [7, 8, 10] are;

1) The path length

Even though some algorithms in literature have implemented parallel implementation, our contribution in this work is to unify the performance parameters allowing the researchers to calculate the path length as the total number of sequential moves.

2) The CPU time

The execution time has been calculated as the real time between the algorithm start time to the end time. In addition, we have ignored the execution time of the preprocess of the algorithms.

3) The makespan

Makespan is the number of time steps required to get all robots to their destination.

4) The optimized path length

These measurements carry any improvement of path length on the original algorithm.

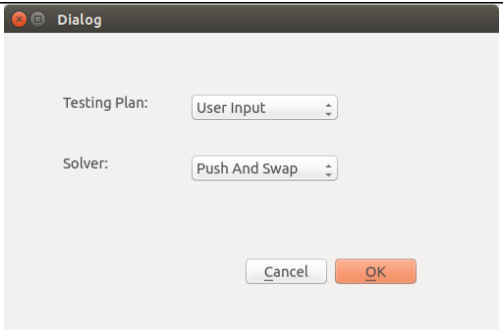
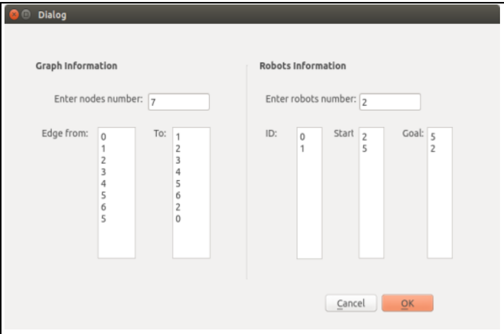
5) The optimized makespan

These measurements carry any improvement of makespan on the original algorithm.

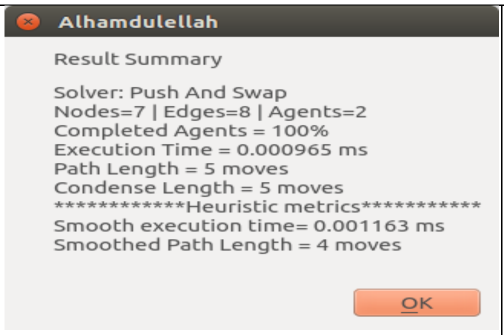
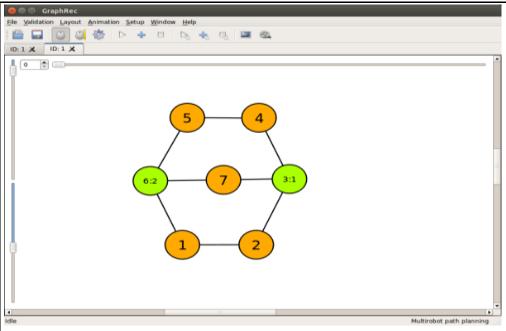
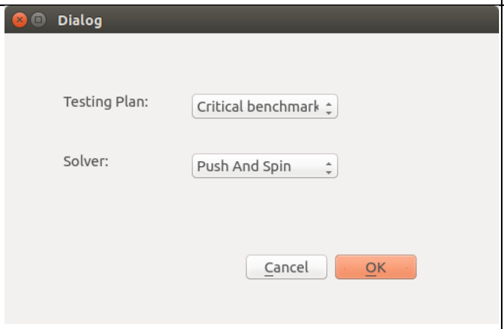
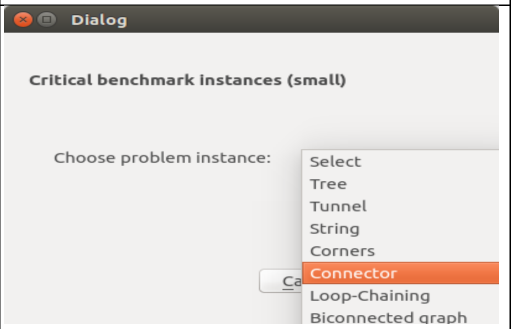
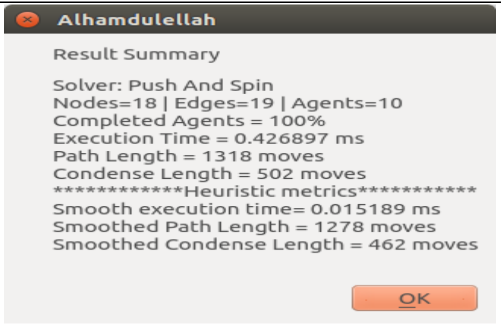
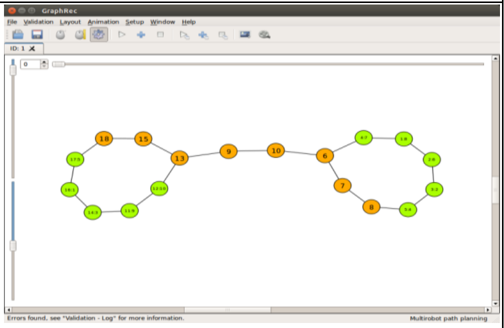
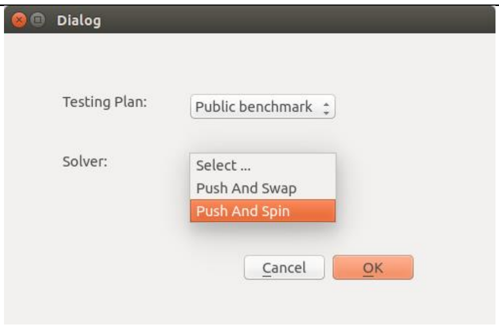
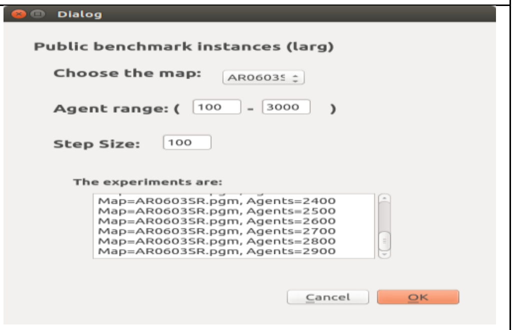
The animation results would be written in the format readable to GraphRec simulator [14]. On its completion, all these results would be stored in the experiment folder. The researchers can then track the movement and the execution of the algorithm by running the animation results in GraphRec.

IV. RESULTS

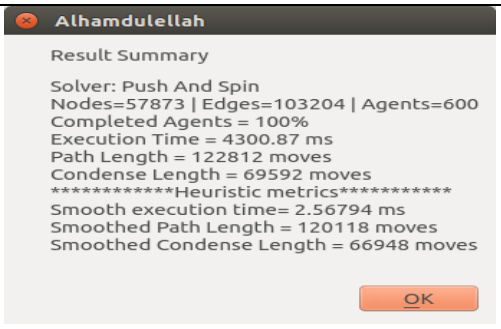
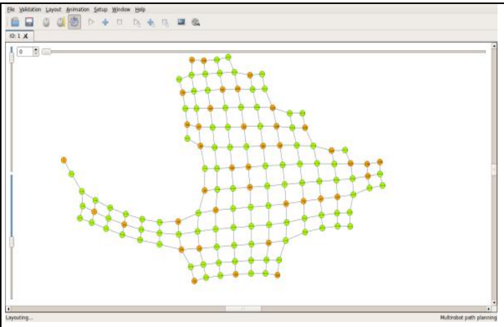
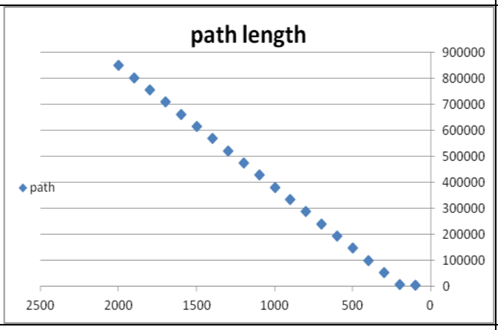
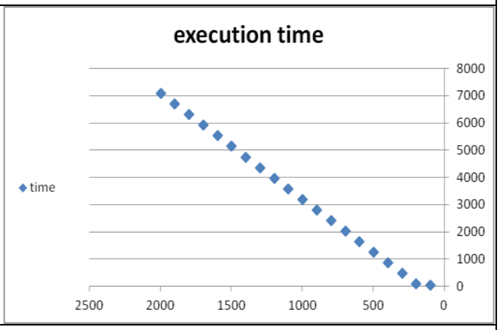
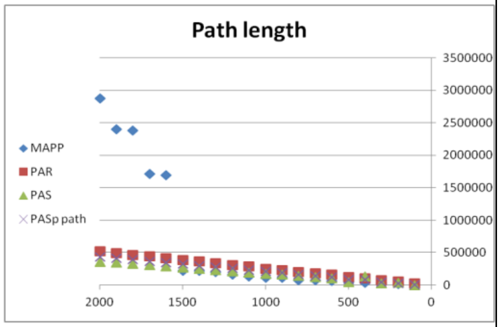
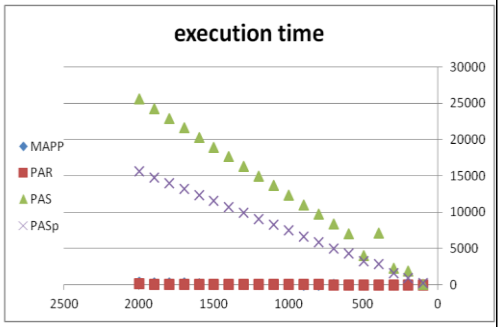
The results of this work are aimed to verify whether the objectives described earlier are satisfied or not. In this section, we will recall every objective and show the ability of MRPPSim to achieve it.

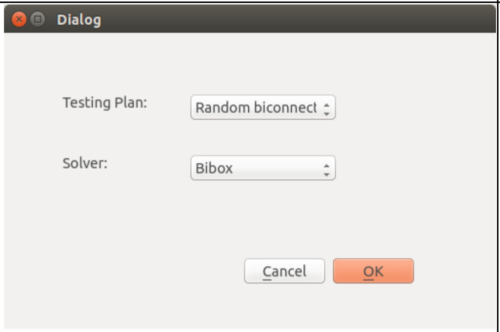
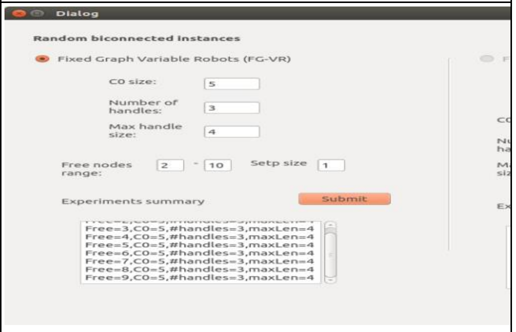
Objective.1: We will track the execution of Push and Swap algorithm[3] on Θ-shape graph with *G*(7,8) and two robots where the goal location of each one is the start location of the other *I*(2,(2,5),(5,2))	 Dialog Testing Plan: User Input Solver: Push And Swap	 Dialog Graph Information Enter nodes number: 7 Edge from: 0 1 2 3 4 5 6 5 To: 1 2 3 4 5 6 2 0 Robots Information Enter robots number: 2 ID: 0 1 Start 2 5 Goal: 5 2

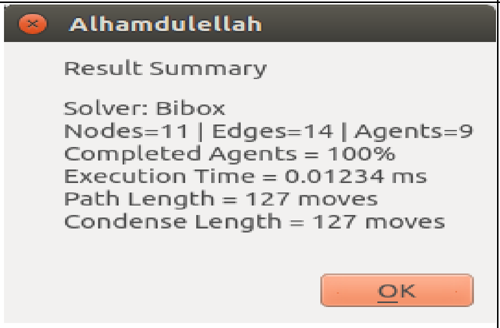
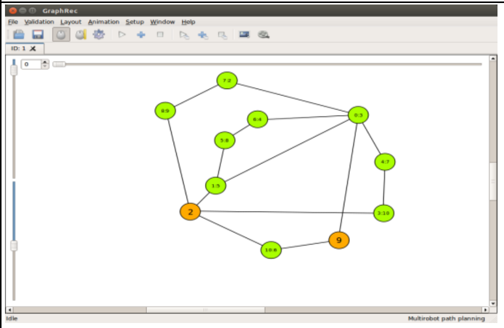
[3] One of the implemented algorithm in MRPPSim

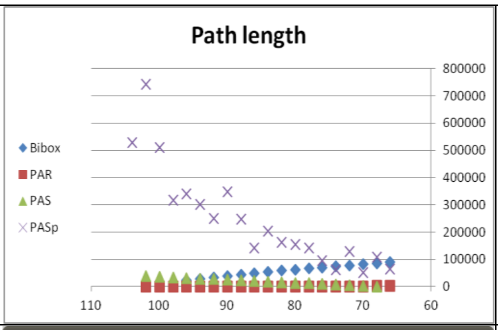
	 Alhamdulellah Result Summary Solver: Push And Swap Nodes=7 | Edges=8 | Agents=2 Completed Agents = 100% Execution Time = 0.000965 ms Path Length = 5 moves Condense Length = 5 moves ************Heuristic metrics*********** Smooth execution time= 0.001163 ms Smoothed Path Length = 4 moves OK	
Objective.2: We will track the execution of Push and Swap algorithm[4] on Connector problem.	 Dialog Testing Plan: Critical benchmark Solver: Push And Spin Cancel OK	 Dialog Critical benchmark instances (small) Choose problem instance: Select / Tree / Tunnel / String / Corners / [✓ Connector] / Loop-Chaining / Biconnected graph
	 Alhamdulellah Result Summary Solver: Push And Spin Nodes=18 | Edges=19 | Agents=10 Completed Agents = 100% Execution Time = 0.426897 ms Path Length = 1318 moves Condense Length = 502 moves ************Heuristic metrics*********** Smooth execution time= 0.015189 ms Smoothed Path Length = 1278 moves Smoothed Condense Length = 462 moves OK	 Errors found, see "Validation - Log" for more information.
Objective.3: We will evaluate the performance of Push and Spin algorithm on public Commercial game benchmark problem AR0603SR with 57873 vertices and number of robots ranging between 100 to 3000 with step size 100.	 Dialog Testing Plan: Public benchmark Solver: Select ... / Push And Swap / [✓ Push And Spin] Cancel OK	 Dialog Public benchmark instances (larg) Choose the map: AR0603S Agent range: (100 - 3000) Step Size: 100 The experiments are: Map=AR0603SR.pgm, Agents=2400 Map=AR0603SR.pgm, Agents=2500 Map=AR0603SR.pgm, Agents=2600 Map=AR0603SR.pgm, Agents=2700 Map=AR0603SR.pgm, Agents=2800 Map=AR0603SR.pgm, Agents=2900 Cancel OK

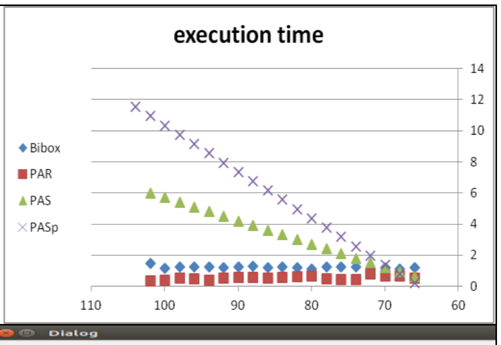
[4] One of the implemented algorithm in MRPPSim

	 Result Summary Solver: Push And Spin Nodes=57873 | Edges=103204 | Agents=600 Completed Agents = 100% Execution Time = 4300.87 ms Path Length = 122812 moves Condense Length = 69592 moves ************Heuristic metrics*********** Smooth execution time= 2.56794 ms Smoothed Path Length = 120118 moves Smoothed Condense Length = 66948 moves	
		
Objective.4: We will compare the performance of Push and Spin algorithm to the performance of Push and Swap, Push and Rotate and MAPP algorithms on public Commercial game benchmark problem AR0307SR with a number of robots ranging from 200 to 2000 with step size 100		

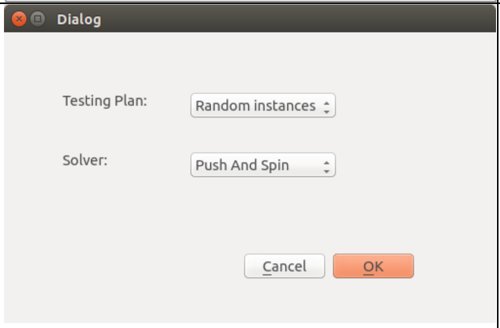
Objective.5,6: Bibox is the complete algorithm for biconnected graphs. We will track the behavior of Bibox algorithm on a biconnected graph with initial cycle=5, number of handles=4 and maximum handle length=3 when the number of free vertices ranging between 2 to 10 with a step of size 1. Then, we will compare that results to the results of Push and Swap, Push and Rotate and Push and Spin algorithms on the same experiments settings.	 Testing Plan: Random biconnect
	Solver: Bibox
		 Random biconnected instances
		Fixed Graph Variable Robots (FG-VR)
		C0 size: 5
		Number of handles: 3
		Max handle size: 4
		Free nodes range: 2 - 10 Setp size 1
		Experiments summary
		Free=3,C0=5,#handles=3,maxLen=4
		Free=4,C0=5,#handles=3,maxLen=4
		Free=5,C0=5,#handles=3,maxLen=4
		Free=6,C0=5,#handles=3,maxLen=4
		Free=7,C0=5,#handles=3,maxLen=4
		Free=8,C0=5,#handles=3,maxLen=4
		Free=9,C0=5,#handles=3,maxLen=4

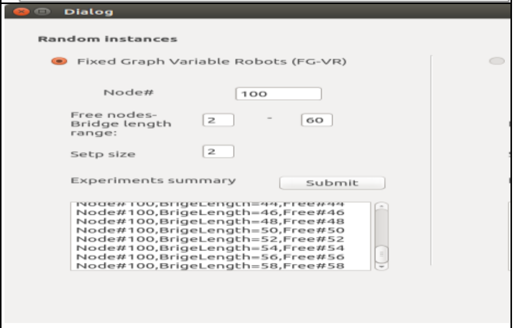
	 Result Summary Solver: Bibox Nodes=11 | Edges=14 | Agents=9 Completed Agents = 100% Execution Time = 0.01234 ms Path Length = 127 moves Condense Length = 127 moves	 GraphRec File Validation Layout Animation Setup Window Help 2 9 Multirobot path planning

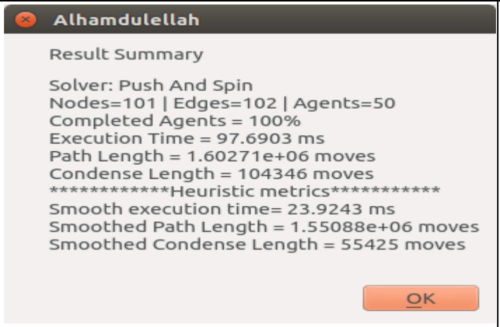


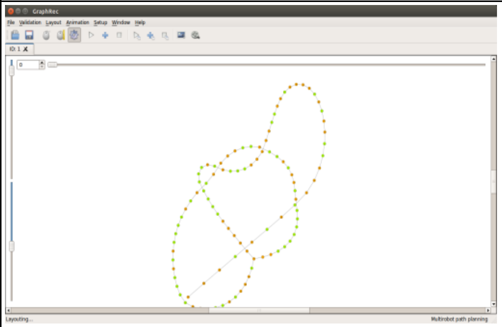


Objective.7,8: Push and Spin algorithm assumes to solve any random solvable graph, where the solvable graphs are any graph with a number of empty vertices equal the longest bridge. Since Push and Spin algorithm is the one which able to solve such instances, we will track its execution on a graph with 100 vertices and the number of free vertices ranging between 2 to 60 with step of size 2.

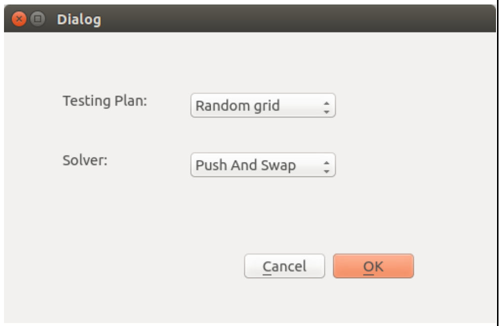

```
Dialog
Testing Plan:   Random instances
Solver:         Push And Spin
Cancel   OK
```

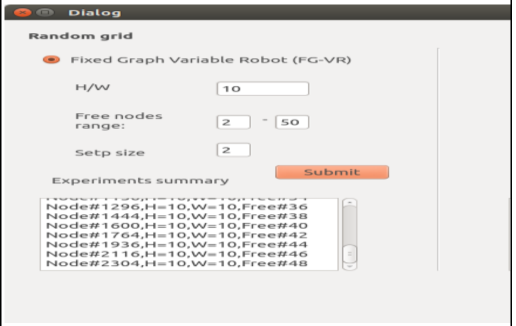

```
Dialog
Random instances
Fixed Graph Variable Robots (FG-VR)
Node#                         100
Free nodes-
Bridge length                 2   -   60
range:
Setp size                     2
Experiments summary           Submit
Node#100,BrigeLength=44,Free#44
Node#100,BrigeLength=46,Free#46
Node#100,BrigeLength=48,Free#48
Node#100,BrigeLength=50,Free#50
Node#100,BrigeLength=52,Free#52
Node#100,BrigeLength=54,Free#54
Node#100,BrigeLength=56,Free#56
Node#100,BrigeLength=58,Free#58
```

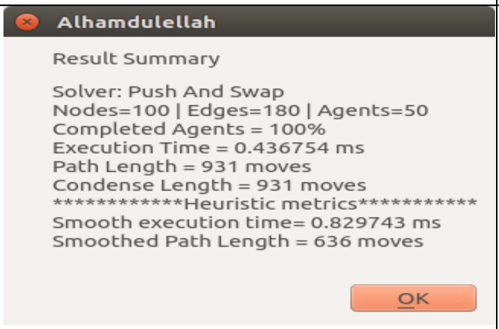

```
Alhamdulellah
Result Summary

Solver: Push And Spin
Nodes=101 | Edges=102 | Agents=50
Completed Agents = 100%
Execution Time = 97.6903 ms
Path Length = 1.60271e+06 moves
Condense Length = 104346 moves
************Heuristic metrics***********
Smooth execution time= 23.9243 ms
Smoothed Path Length = 1.55088e+06 moves
Smoothed Condense Length = 55425 moves
OK
```

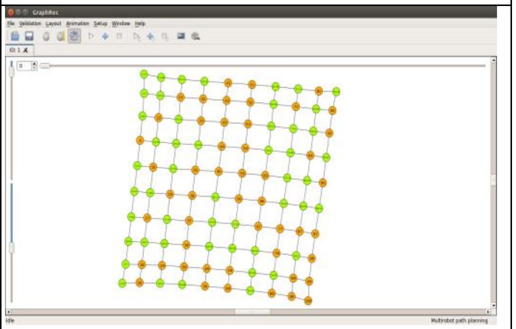


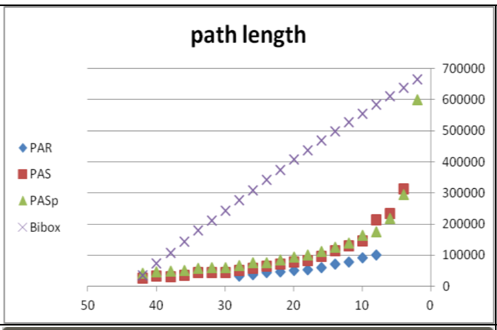
Objective.9,10: We will track Push and Swap algorithm execution on a graph of size 10x10 and number of free vertices ranging between 2 to 50 with a step of size 2. Then, we will compare that results to the results of Push and Rotate, Push and Spin and Bibox algorithms on the same experiments settings.

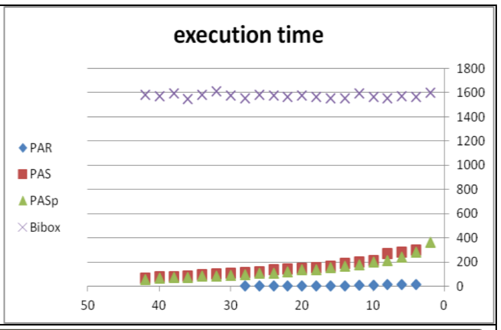

```
Dialog
Testing Plan:   Random grid
Solver:         Push And Swap
Cancel   OK
```


```
Dialog
Random grid
Fixed Graph Variable Robot (FG-VR)
H/W                 10
Free nodes
range:              2  -  50
Setp size           2
                             Submit
Experiments summary
Node#1296,H=10,W=10,Free#36
Node#1444,H=10,W=10,Free#38
Node#1600,H=10,W=10,Free#40
Node#1764,H=10,W=10,Free#42
Node#1936,H=10,W=10,Free#44
Node#2116,H=10,W=10,Free#46
Node#2304,H=10,W=10,Free#48
```

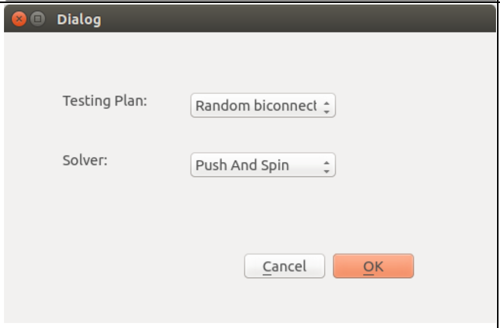

```
Alhamdulellah
Result Summary

Solver: Push And Swap
Nodes=100 | Edges=180 | Agents=50
Completed Agents = 100%
Execution Time = 0.436754 ms
Path Length = 931 moves
Condense Length = 931 moves
************Heuristic metrics***********
Smooth execution time= 0.829743 ms
Smoothed Path Length = 636 moves
OK
```



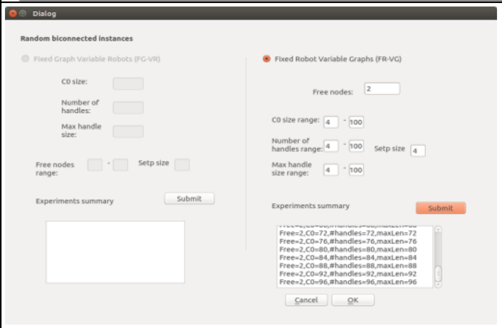




Objective.11: We will test Push and Spin algorithm’s on different sizes of biconnected graphs with a fixed occupying ratio 0.98. The graphs sizes are ranging between (initial cycle, number of handles, maximum handle length)=(4,4,4) to (100,100,100) with step of size 4.



Testing Plan: Random biconnect
Solver: Push And Spin

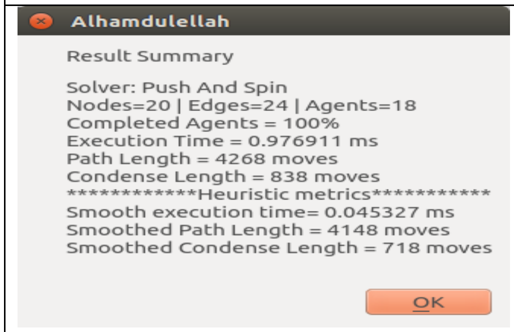


Random biconnected instances

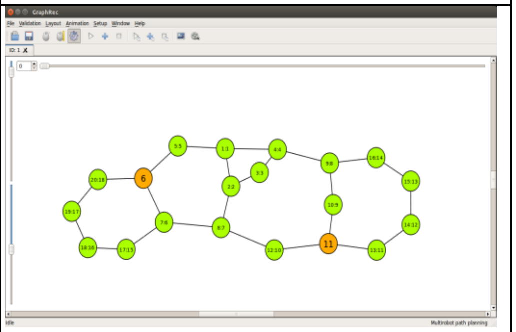
Fixed Graph Variable Robots (FG-VR)
C0 size:
Number of handles:
Max handle size:
Free nodes range: - Setp size
Experiments summary

[✓ Fixed Robot Variable Graphs (FR-VG)]
Free nodes: 2
C0 size range: 4 - 100
Number of handles range: 4 - 100 Setp size 4
Max handle size range: 4 - 100
Experiments summary

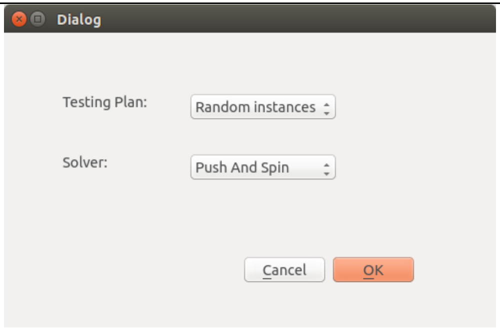
```
Free=2,C0=72,#handles=72,maxLen=72
Free=2,C0=76,#handles=76,maxLen=76
Free=2,C0=80,#handles=80,maxLen=80
Free=2,C0=84,#handles=84,maxLen=84
Free=2,C0=88,#handles=88,maxLen=88
Free=2,C0=92,#handles=92,maxLen=92
Free=2,C0=96,#handles=96,maxLen=96
```



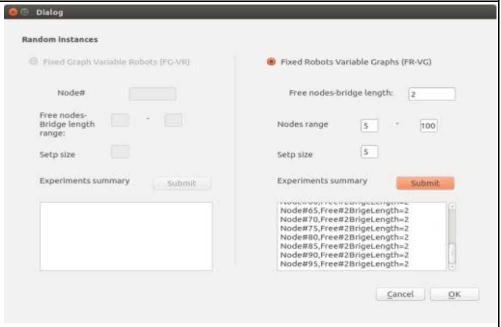
```
Result Summary

Solver: Push And Spin
Nodes=20 | Edges=24 | Agents=18
Completed Agents = 100%
Execution Time = 0.976911 ms
Path Length = 4268 moves
Condense Length = 838 moves
************Heuristic metrics***********
Smooth execution time= 0.045327 ms
Smoothed Path Length = 4148 moves
Smoothed Condense Length = 718 moves
```



Objective.12: We will track the execution of Push and Spin algorithm’s on different sizes of random graphs with a fixed occupying ratio, 0.98. The graphs sizes is ranging between 5 to 100 with step of size 5.



Testing Plan: Random instances
Solver: Push And Spin

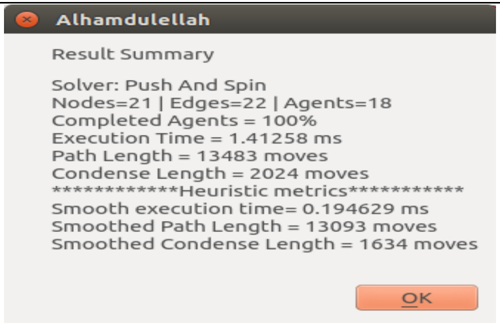


Random instances

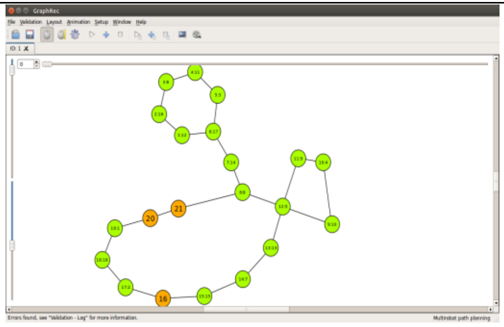
Fixed Graph Variable Robots (FG-VR)
Node#
Free nodes-Bridge length range: -
Setp size
Experiments summary

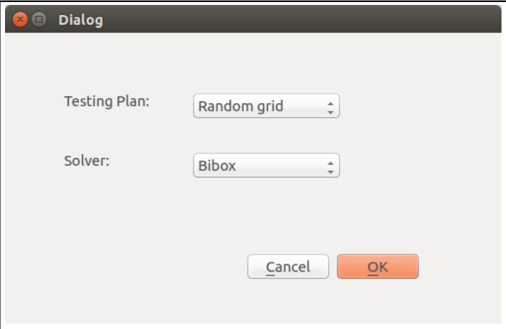
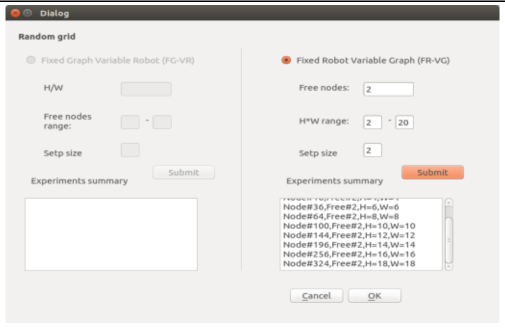
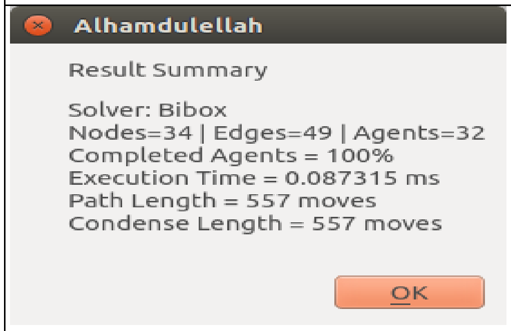
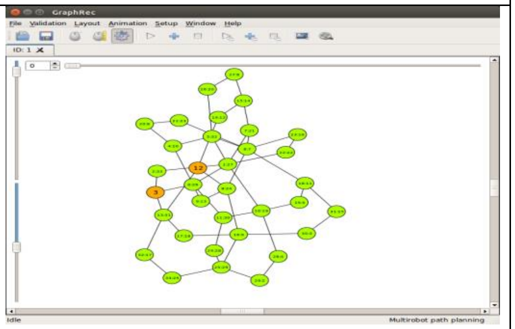
[✓ Fixed Robots Variable Graphs (FR-VG)]
Free nodes-bridge length: 2
Nodes range 5 - 100
Setp size 5
Experiments summary

```
Node#65,Free#2BrigeLength=2
Node#70,Free#2BrigeLength=2
Node#75,Free#2BrigeLength=2
Node#80,Free#2BrigeLength=2
Node#85,Free#2BrigeLength=2
Node#90,Free#2BrigeLength=2
Node#95,Free#2BrigeLength=2
```



```
Result Summary

Solver: Push And Spin
Nodes=21 | Edges=22 | Agents=18
Completed Agents = 100%
Execution Time = 1.41258 ms
Path Length = 13483 moves
Condense Length = 2024 moves
************Heuristic metrics***********
Smooth execution time= 0.194629 ms
Smoothed Path Length = 13093 moves
Smoothed Condense Length = 1634 moves
```


GraphRec
File Validation Layout Animation Setup Window Help
ID: 1
20
21
16
Errors found, see "Validation - Log" for more information.
Multirobot path planning

Objective.13: We will test Bibox algorithm's on different sizes of grid graphs with fixed occupying ratio 0.98. The grid sizes are ranging between 2x2 to 20x20 with step of size 2.	 Dialog Testing Plan: Random grid Solver: Bibox Cancel OK	 Dialog Random grid Fixed Graph Variable Robot (FG-VR) H/W Free nodes range: - Setp size Submit Experiments summary Fixed Robot Variable Graph (FR-VG) Free nodes: 2 H*W range: 2 - 20 Setp size 2 Submit Experiments summary Node#36,Free#2,H=6,W=6 Node#64,Free#2,H=8,W=8 Node#100,Free#2,H=10,W=10 Node#144,Free#2,H=12,W=12 Node#196,Free#2,H=14,W=14 Node#256,Free#2,H=16,W=16 Node#324,Free#2,H=18,W=18 Cancel OK
	 Alhamdulellah Result Summary Solver: Bibox Nodes=34 | Edges=49 | Agents=32 Completed Agents = 100% Execution Time = 0.087315 ms Path Length = 557 moves Condense Length = 557 moves OK	 GraphRec File Validation Layout Animation Setup Window Help ID: 1 Idle Multirobot path planning

V. Conclusion and Future Works

As can be observed, the MRPP simulator provides complete simulation/evaluation tool for offline multi-robot path planning. Its major strength is its ability to provide the algorithms imposed graph representation for its environment. MRPP allows the researchers to;

- Test and track the algorithm's behavior for specific cases in fully controlled instances.
- Track the algorithm's behavior and compare its performance with the algorithms that already evaluated on predefined small critical instances in [7].
- Test the algorithm behavior on very large graphs when the occupying ratio (robots number/vertices count) increases.
- Compare the algorithm performance to the algorithms already tested in public large problems [1] such as [7-9] .
- Compare the algorithm's performance to the performance of algorithms solves biconnected graphs [10], [7, 8].
- Test the algorithm behavior on fixed biconnected graph's size when the occupying ratio increases.
- Evaluate the algorithm performance against the algorithms solves random graphs.
- Track the algorithm behavior on fixed random graph's size when the occupying ratio increases.
- Compare the algorithm's performance to the performance of algorithms solves grid graphs [7, 8, 10].
- Test the algorithm behavior on fixed grid graph's size when the occupying ratio increases.
- Evaluate the algorithm performance on biconnected graphs of fixed occupying ratio when the graph's size increases.
- Evaluate the algorithm performance on random graphs of fixed occupying ratio when the graph's size increases.
- Evaluate the algorithm performance on grid graphs of fixed occupying ratio when the graph's size increases.
- Plug new simulation model, MRPP algorithm implementation or performance measurement calculation.

Future work may include designing secure database, reporting set of experiments using different types of diagrams and adding new evaluation algorithms.

References

[1] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," Computational Intelligence and AI in Games, IEEE Transactions on, vol. 4, pp. 144-148, 2012.

[2] E. Guizzo, "Three Engineers, Hundreds of Robots, One Warehouse," IEEE Spectr., vol. 45, pp. 26-34, 2008.

[3] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," Journal of Artificial Intelligence Research, pp. 591-656, 2008.

[4] J. Leitner, "Multi-robot cooperation in space: A survey," in Advanced Technologies for Enhanced Quality of Life, 2009. AT-EQUAL'09., 2009, pp. 144-151.

[5] J. M. Roberts, E. S. Duff, and P. I. Corke, "Reactive navigation and opportunistic localization for autonomous underground mining vehicles," Information Sciences, vol. 145, pp. 127-146, 2002.

[6] D. Nieuwenhuisen, A. Kamphuis, and M. H. Overmars, "High quality navigation in computer games," Science of Computer Programming, vol. 67, pp. 91-104, 2007.

[7] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in IJCAI, 2011, pp. 294-300.

[8] B. d. Wilde, A. W. ter Mors, and C. Witteveen, "Push and Rotate: a Complete Multi-agent Pathfinding Algorithm," Journal of Artificial Intelligence Research, pp. 443-492, 2014.

[9] K.-H. C. Wang and A. Botea, "Mapp: a scalable multi-agent path planning algorithm with tractability and completeness guarantees," Journal of Artificial Intelligence Research, pp. 55-90, 2011.

[10] P. Surynek, "A novel approach to path planning for multiple robots in bi-connected graphs," in Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, 2009, pp. 3613-3619.

[11] E. T. S. Alotaibi and H. Al-Rawi, "Multi-Robot Path-Planning Problem for a Heavy Traffic Control Application: A Survey International Journal of Advanced Computer Science and Applications(IJACSA), ," International Journal of Advanced Computer Science and Applications(IJACSA), vol. 7, p. 10, 1,7,2016 2016.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.

[13] D. Kornhauser, G. Miller, and P. Spirakis, Coordinating pebble motion on graphs, the diameter of permutation groups, and applications: IEEE, 1984.

[14] P. Koupý, "Visualization of problems of motion on a graph," BSc, Department of Theoretical Computer Science and Mathematical Logic, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2010.