A Distributed Approach to Design Satellite Tracking System

Saeid Samadidana,Student member,IEEE
Tandy School of Computer Science
The University of Tulsa
Tulsa, Oklahoma, USA
Email: saeid-samadidana@utulsa.edu
Supervisor: Dr. Roger Mailler

***Abstract*—Tracking the satellites is a costly and difficult to manage task. To do this task many telescopes around the world are used. Each telescope is responsible for a few satellites during the tracking time. The task of tracking a satellite is to assign it to a telescope at proper time to capture the satellite movement. Therefore, this task can be seen as a distributed problem in which different nodes collaborate with each other to satisfy the goal. The satellite tracking problem can be seen as a Distributed Constraint Satisfaction Problem and to solve that many distributed algorithms can be applied. We apply the Distributed Probabilistic Protocol to accomplish this task. To have a better vision of the problem, an application is required to process the captured images from sky to track the moving objects. Hence, we develop an application called Sky Simulator to tackle the tracking problem. In addition, we show some experiments and problems associated with tracking the satellites. Some of these problems are limited time slots and light pollution. This project also provides a test environment for image processing projects with the ability to add additional features. Many processes can be done with this application before doing the tracking of satellites that reduces the cost associated with capturing and tracking the real objects.**

I. Introduction

Tracking the satellites in the sky is a difficult and costly task that needs to be planned frequently. Different telescopes must be applied to do this task as the number of satellites is increasing. In addition, the number of telescopes that are used for this task is limited, therefore a very precise model must be developed to optimally schedule tracking of the satellites. It must be considered that due to some limitations each telescope can monitor a few satellites during the 24 hours period. Problems like light pollution, bad weather conditions, telescope size and etc. can make this task even more difficult. Therefore, scheduling the tracking time slot needs to be fast and be able to handle the immediate changes in the system. In this study, we use the DPP protocol[3]) to tackle this problem. In addition, we developed a simulator to simulate the moving objects in the sky.

In the first years of developing computers, many scientist were designing the centralized algorithms to solve a problem. On that time, mostly inputs of a problem were given to a system and the output was produced on the same system and problems were mostly centralized. However, due to some limitations like change in the problems' sizes, memory and security issues [17] the need for developing the Distributed Algorithms raised. One of the practical example of distributed problems is tracking the satellites by telescopes that are developed on earth. It is clear that this problem is distributed by time and location and is very complex as the problem is constantly changing during the time. Indeed, the discussing problem is a type of problems known as Dynamic Distributed Constraint Satisfaction Problems(DynDCSP). Usually, these problems are measured by the change rate. In our discussing problem, the change can be seen as a new satellites is being visible to one telescope or a satellites is being invisible from the view of the telescope. One can see that the change rate in this problem is very high and in just few seconds a problem might change. Algorithms like MGM [1], DSA [2], DPP [3], and DBA [4] are just a very few examples of algorithms that are designed to solve these problems. In addition to this problem, it is very easy to see different problems as distributed algorithms like distributed resource allocation [15], distributed scheduling [16], distributed planning [14], and distributed routing [6].

As we explained the time is a very important factor in the satellites scheduling problem. Therefore, we used the DPP in this project because it considers the problem based on time and it uses *a priori* knowledge to solve the problem.

The structure of this study is as follows: In Section II we describe the DCSP problems. In Section III we explain the DPP algorithm. In Section IV we talk about the Object tracking system including the hardware design and its detail. In Section V we talk about the Sky Simulator an image processing software that is developed to simulate the sky. Then we provide an explanation of how tracking the satellites can be considered as an DCSP problem and we use DPP algorithm to solve some problems in Section VI. Finally, in Section VII we conclude the study and provide some approaches for future work.

II. Problem description

Following is the formal definition of a Distributed Constraint Satisfaction Problem (DCSP) [9]:

- a set of n variables: $V = \{v_1, \ldots, v_n\}$,

- a set of g agents: $A = \{a_1, \ldots, a_g\}$,
- discrete, finite domains for each variable: $D = \{D_1, \ldots, D_n\}$,
- a set of constraints $C = \{c_1, \ldots, c_m\}$ where each $c_i(d_{i,1}, \ldots, d_{i,j})$ is a function $c_i : D_{i,1} \times \cdots \times D_{i,j} \rightarrow \{true, false\}$.

The objective of the problem is to find an assignment $S = \{d_1, \ldots, d_n\}$ of domain values such that all constraints of C are satisfied or to conclude that no such assignment exists. CSP and DCSP problems are known to be NP-Complete. Therefore, in many cases an approximate solution is desired. One can see that in a problem like tracking the satellites finding the optimal solution in a way that all satellites could be monitored all the time is almost impossible. In DCSP problems, usually we use the term agent that is used to manage and handle the variables. Without the loss of generality, it is often assumed that each agent is responsible for only one variable. Each agent then tries to solve its assigned variables constraints while collaborating with other agents to solve the whole problem. This collaboration is done by communicating some type of messages. In our problem, the constraints can be defined as the relation between a satellite and a telescope. Therefore, to satisfy the constraint agents collaborate with other agents to find the best time for a telescope to track the satellite.

Two important factors *density* (p_1) and *tightness* (p_2) are defined in DCSP problems to categorize them. These parameters are directly related to the difficulty of the problem and are defined as:

- $p_1 = 2c/n(n-1)$
- $p_2 = \#false\ assignments/\#total\ assignments$

In which n is the number of variables and c is the number of constraints. In addition to these two parameters sometimes it is useful to use another parameter average *degree* that is defined as $d = c/n$.

III. Distributed Scheduling Algorithm

In this section we briefly explain the Distributed Probabilistic Protocol (DPP) [3] that is designed to rapidly solve the DCSP problems. DPP is based on using fewer messages as much as possible. The key point of DPP is that agents do not need always to communicate with their neighbors. Indeed in DPP, agents predict the behavior of their neighbors even if they have not communicated with them for a while. Furthermore, agents do not wait for messages from their neighbors if they believe that they have the highest improve value amongst their neighbors.

To accomplish this goal, agents use a function called probability distribution function(PDF) of improve values based on their constraints. Protocol starts with each agent sending its DPP value plus its initial value to its neighbors. Each agent then in every cycle calculates two values called $improve$ and $change$ probabilities. The first one is used to determine when an agent should send the improve value to neighbors. This kind of message is called $improve?$ message. This value is calculated based on the error in neighbors predication. To calculate this value, each agent must receive the improve probabilities from its neighbors. Beside, it needs to use the most recent improve value it has sent to its neighbors. In addition it needs to use PDF values.

PDF values must be calculated once when an agent knows about its neighbors and constraint tables. Different methods can be applied to calculate the PDF values. One method is to calculate the PDF values for each agent based on the constraint tables. It is obvious that by increasing the number of neighbors of an agent the probability of having higher improve values is very small and close to zero. Based on the empirical experiments done to calculate the PDF with this method is it known that this is a very time consuming task. The second method is to use a uniform probability value for all PDF values. One drawback of this method is that the probabilities are not distributed correctly. For example, the probability of having one improve often is higher than having 10 improve values. Therefore, same PDF values might decrease the solution quality. The third method is using an approximate values based on empirical tests on different problems and getting the mean value of probabilities of test cases. This method is quite fast and can be used instead of exact values. In addition, one may use the graph coloring PDF values for a random problem as it is easier to calculate. These methods are compared in test result section. Equation 1 shows the formula for calculating the improve value.

$$P(improve_X^t = j | X = j \wedge improve_X^{t-1} = i) = |P(Y \leq j) - P(Y \leq i)| \quad (1)$$

On the other side, the change probability is used to calculate the probability of changing the current value of an agents variable. To do this, an agent multiplies the probabilities that it has the highest improve value among all of its neighbors. equation 2 shows this formula.

The fully explanation of DPP protocol can be found in [3]. DPP has shown to be very effective on Graph Coloring and DCSP problems. Figure 1, Figure 2, and Figure 3 show the different procedures used in the DPP. A simple DCSP problem is shown in figure 4.

$$\prod_{Y \in neighs(X)} P(Y \leq x | improve_Y^t = i) \quad (2)$$

IV. Object Tracking System

In this section we explain the object tracking System built to track the satellites. following is the list of hardware used in this project.

- Telescope
  For this project we used the 8 inches $LX200 - ACF^{®}$ telescope. This telescope has the ability to install a camera on it to capture the visible location as a video or image file.
- Camera

```
procedure main
  initialize;
  while (not terminated) do
    update agent_view with ok? (x_j, d_j) messages;
    update agent_view with improve (x_j, improve_j)
messages;
    calculate_improve;
    send_ok;
    send_improve;
  end do;
end main;
procedure initialize
  pdf_i ← calc_improve_pdf;
  send ((init, (x_i, d_i, pdf_i)) to all x_j ∈ neighbors;
  while (not received init from x_j ∈ neighbors) do
    update agent_view with incoming init (x_j, d_j, pdf_j)
messages;
end initialize;
```

Fig. 1. The main and initialize procedures of the DPP algorithm.

```
procedure send_ok
  if new_value ≠ d_i do
  p ← calc_change_probability;
    if  random < p
      d_i ← new_value;
      send ((ok?, (x_i, d_i)) to all x_j ∈ neighbors;
      improve_i ← 0;
    end if;
  end if;
end send_ok;
procedure send_improve
  for all x_j ∈ neighbors do
    p ← calc_improve_probability(x_j);
    if  random < p
      send ((improve, (x_i, improve_i)) to x_j;
    end if;
  end do;
end send_improve;
```

Fig. 3. The send_ok, and send_improve procedures of the DPP algorithm.

```
procedure calculate_improve
  v ← the value with the least conflict (v ≠ d_i);
  if  d_i has no conflicts or v has more conflicts than d_i
do
    new_value = d_i;
    improve_i ← 0;
  else
    new_value = v;
    improve_i ← difference in conflicts between d_i and v;
  end if;
end calculate_improve;
```

Fig. 2. The calculate_improve procedure of the DPP algorithm.

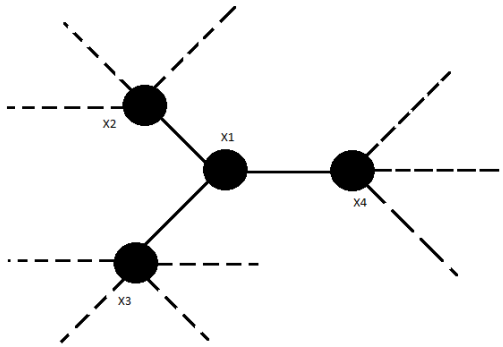

Fig. 4. A simple DCSP problem

A $WZO^{®}$ camera is installed on the telescope to capture images and videos of the sky.

- Filters
  Different filter used to have a better vision of the sky during the night.
- High capacity storage device
  It is used to store the video or image files. It is required to capture the pictures with very high resolution therefore, the size of files are extremely large.
- Satellites list
  From each location on the earth just some satellites are visible. Based on the longitude and latitude of the location it is possible to get the list of passing satellites in a specified time.
  Figure 5 shows the telescope and its supplementary equipment. The photo had been taken in Fall 2017. The figure shows the camera installed on telescope and the camera is connected to a laptop to transfer the images to the computer. The process of adjusting the camera must be started a few minutes before sunset.

A. Tracking Limitations

There was some limitations for tracking the satellites. Following is the list of problems for tracking the satellites.

- light pollution
  The first issue was the light pollution. Satellites mostly are visible in dark sky. Therefore, finding a place with minimum light pollution was difficult. This place must be far from the cities.
- Telescope Adjusting
  Before starting the tracking, telescope needs to be adjusted. Based on the experiments between 30 minutes to one hour is required to adjust the telescope. The adjustment of the telescope is based on locating

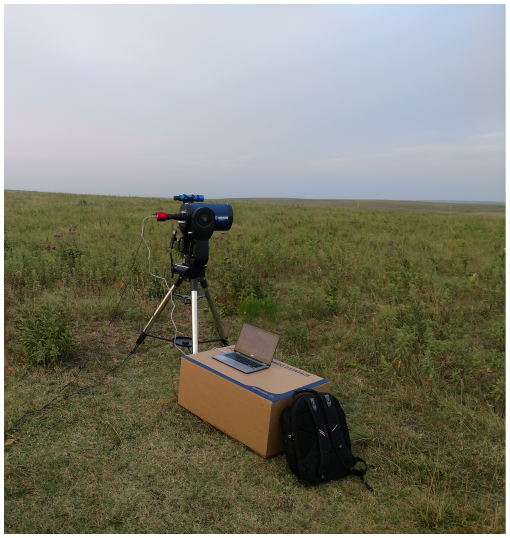

Fig. 5. An experiment done in Fall 2017

some famous stars like polar star. Therefore, the sky must be dark to adjust the telescope. Of course, adjusting the telescope might be different in different telescopes. In the telescope used in this project the process for adjusting the telescope was almost automatic. The process starts with finding the exact location of the telescope using GPS that exists in the telescope. Then, as mentioned some starts are used to adjust the telescope.

- Weather Condition
  Satellites are visible just in a clear sky with minimum clouds. Therefore, having a nice weather is a need.
- File Storage
  Capturing the video of moving objects in sky needs a very high resolution. Therefore, a storage equipment like external hard drive was required to store the video files. A free software is used to capture the video. The camera is installed on the telescope. Some parameters like exposure time, resolution must be adjusted to have a clear screen.
- Tracking Time Slots Limitation
  Interestingly, there is just a limited time slot to track the satellites. If the sky is not dark enough then satellites are not visible due to lightness and if it is too dark they are not visible too. Therefore, a time slot of about one hour after sunset was the proper time to track the satellites.

V. Sky Simulator

In this section we talk about an image processing application that has been developed to track the moving objects in sky and simulates the sky for image processing purposes. To track the objects the Otsu method[13] is used. The algorithm for detecting objects in filtered image is as follow. The filtered image, is examined pixel by pixel to detect the objects. Whenever a pixel with different color with background is found

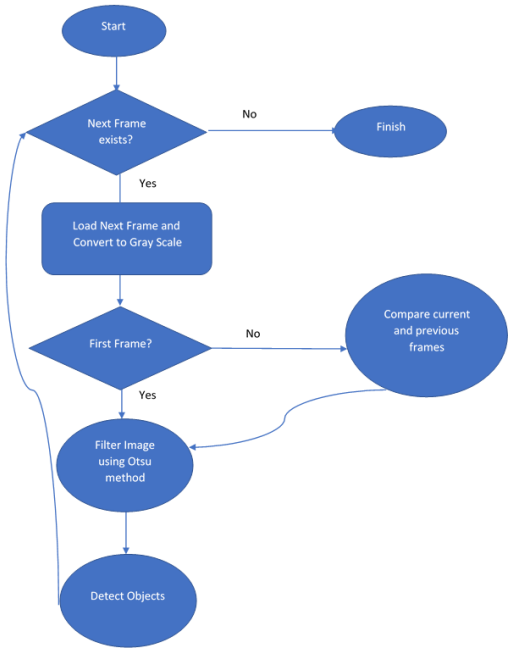

Fig. 6. Object Tracking flow chart

then algorithm tries to find a diagonal to where objects shape is continued. Then, Algorithm tries to check the boundary of the image to make sure the object is surrounded by the large enough rectangle. When no different color is found then algorithm creates a rectangle and sets all pixels in the rectangle as checked pixel. This prevents from the problem of finding an object several times. Then, algorithm finishes detecting objects and draws the rectangles around the found objects. This process repeats until all frames are processed. Figure 6 shows the simple flow chart of the algorithm.

The sky simulator consists of three setting sections and one video panel. The first section is about general setting including speed of earth, refresh time and the darkness of the sky. The Second section is about creating constant objects in the sky with the ability to randomly put the objects in the

A. Tracking Algorithm

In this section the general approach used for tracking the objects is explained then the implemented software is described. The algorithm implemented in this study is as follow. First the video is loaded, then the first frame is read and is converted to the gray scale. To convert an RGB image to a gray scale different method can be used that are

- Average
- Lightness
- Luminosity PAL
- Luminosity HD

The difference between these methods are the ratio of using red, green and blue colors in the gray image. The table I shows how the gray image is obtained by using different methods.

The converted images by all methods for several test images were almost same therefore the average method is used in

TABLE I
RGB TO GRAY SCALE METHODS

Method Name	Formula for RGB to Gray
Average	(red + green + blue) / 3
Lightness	Max(blue, Max(red, green)) + Min(blue, Min(red, green)) / 2
Luminosity HD	(0.2126 * red + 0.7152 * green + 0.0722 * blue)
Luminosity PAL	(0.299 * red + 0.587 * green + 0.114 * blue)

this study. Then Otsu method is used to find a threshold for separating the background and foreground of the image. Then the algorithm for finding the objects on the filtered image find the objects and stores them. Although this step is not required for the first image however, it can make the differentiating the images faster as the approximate location of objects are already determined. After reading the first frame the next frame is read and is converted to gray. Then it is compared with the previous frame and Otsu method again find the threshold and the algorithm for finding the objects detects the moving object and draws a rectangle around them. The flowchart of the Object Tracking algorithm is showed in the following chart.

The algorithm for detecting objects in filtered image is as follow. The filtered image, is examined pixel by pixel to detect the objects. Whenever a pixel with different color with background is found then algorithm tries to find a diagonal to where objects shape is continued. Then, Algorithm tries to check the boundary of the image to make sure the object is surrounded by the large enough rectangle. When no different color is found then algorithm creates a rectangle and sets all pixels in the rectangle as checked pixel. This prevents from the problem of finding an object several times. Then, algorithm finishes detecting objects and draws the rectangles around the found objects.

The discussed algorithm implemented in the Sky Simulator software. In the next section the implemented software and its features is explained.

B. Sky Simulator Software

In this section the implemented software is explained. This software includes the setting sections for creating the simulated video of sky and the moving objects and processing the simulated video. Following are the sections of the software:

- Settings
  - General Setting
    This section includes the speed of earth, refresh time of sky screen panel and the darkness of the sky. Table II shows the range of values for the settings. The refresh rate indicates the speed of updating the panel. The higher speed refreshes the screen faster. In addition this value is used to determine the frame rate of the video that will be saved. The ratio is defined as $1000/refresh\ rate$. The darkness ranges from 0 to 255 and a lower value indicates the darker background and a larger value shows a lighted background. The $Start$ and $Stop$ buttons are for starting and stopping the capturing the video from the video panel. Whenever, user clicks the $Start$ buttons in every determined time the panel will be refreshed and the last image of the panel will be saved as a video frame. When user clicks the $Stop$ button, the video will be end and can be saved. It is possible to add additional frames to the existing video by clicking the $Start$ button again. The figure 7 shows the $General\ Setting$ panel.
  - Moving Object Setting
    This section is for creating moving objects in the sky. User can determine the size of the object, its speed and radius and create the object by both dragging the shape in the panel to the video panel or by clicking the $create\ object$ button, provided that the $Random\ Position$ check box is selected. This option, randomly places the object in the video panel. In addition, user can remove all constant objects by clicking the $Clear\ Objects$ button. The figure 8 shows the $Moving\ Objects\ Setting$ panel.
  - Constant Object Setting
    This section is for creating constant objects in the sky. User can determine the size of the object and creates the object by both dragging the shape in the panel to the video panel or by clicking the $create\ object$ button provided that the Random position check box is selected. This option, randomly places the object in the video panel. In addition, user can

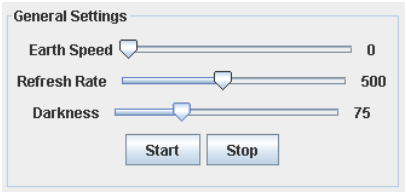

Fig. 7. Sky Simulator General Setting

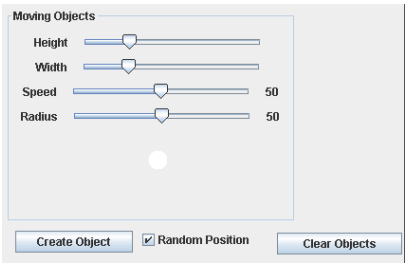

Fig. 8. Sky Simulator Moving Object Setting

TABLE II
RGB TO GRAY SCALE METHODS

Setting name	Min value	Max value
Earth Speed	0 (No move)	5
Refresh Rate	0 ms	1000 ms
Darkness	0 (Black)	255 (White)

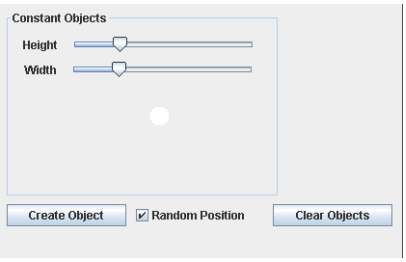

Fig. 9. Sky Simulator Moving Object Setting

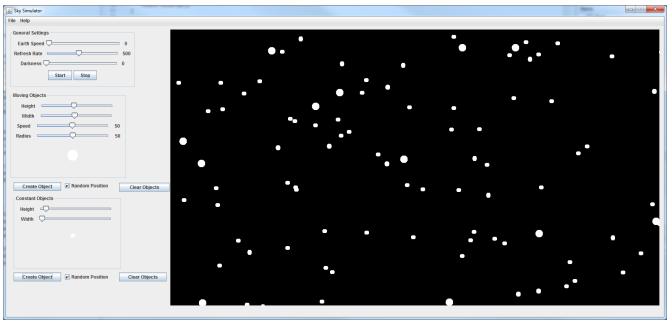

Fig. 10. A Sample test case of Sky Simulator

remove all constant objects by clicking the $clear\ objects$ button. The figure 9 shows the $Constant\ Objects\ Setting$ panel.

- Video Panel
  This panel is designed to both show the video and therefore processing the video and creating the simulated video for processing. Whenever, user creates a moving or constant object, then it is showed in this panel. General settings, changes the darkness of the panel and the speed of the earth.
  - Save settings
    It is possible to save all settings defined by the user. By choosing this item, a save form will be appeared that allows user to save the setting as a sss file. The settings that will be saved are the general settings including the earth speed, refresh rate and darkness, the moving objects that is a list of them with the information about their sizes and speeds, radius and their locations, and the constant objects as a list of them with the location and size information.
  - Load settings
    To load the settings user selects the $Load\ Setting$ and chooses a file with sss extension to load the settings. After settings are loaded the screen will be filled with defined objects if any exists. As mentioned above the settings include the general setting about the video and panel and a list of moving objects and their locations and a list of constant objects and their locations.
  - Save Video
    By selecting this menu, user can save the sky screen panels images as video. The video starts by clicking the $Start$ button and finishes by clicking the $Stop$ button. The video will be saved as $.wmv$ file. The frame rate of the video is determined by the $refresh\ rate$ value. After refreshing the panel, the last frame will be added to the video.
  - Load Video
    By selecting this menu, user can load the saved video files with $.wmv$ extension. After loading the video the moving objects will be tracked automatically with a rectangle around moving objects. If for any reason, the video cannot be load an error message will be shown to the user.

Figure 10 shows the Sky Simulator Software main page.

VI. TEST RESULTS

In this section, we show the result of running DPP algorithm on DCSP problems. We created some random problems to simulate the real problems of satellites tracking system. To do the test we used the Farmx software developed in our lab as a simulator to create and solve different DCSP problems. Then, we used the following parameters for testing the problems.

- $n = \{100, 200, 400\}$ where n is the number of variables/agents,
- $density = \{low, medium, high\}$,
- $p_2 = \{0.33\}$.
- $|D_i| = 3$ where D_i is the domain for variable i

First we compared the different ways of calculating the PDF values. To do this, we created random problems and compared the total number of messages that agents send during the execution of the algorithm. In addition, we compared the number of conflicts based on each PDF calculation method. Figures 11 and 12 show the comparing among number of conflicts and number of messages in different methods of calculating the PDF values. As we expected the result of methods $Normal$ and $Perfect$ is better than the others. The worst result belongs to $Uniform$ method as the number of conflicts is much higher than the others. The $Graph\ Coloring$ method also shows a very good result. Both $Graph\ Coloring$ and $Normal$ method are faster than $Perfect$ method therefore, it is better to use these methods instead.

TABLE III
CONVERGENCE RATE

Density	Low			Medium			High		
Variable Count	100	200	400	100	200	400	100	200	400
DPP	14.68	15.02	14.65	16.64	18.23	17.71	20.64	20.60	21.68

TABLE IV
CONVERGENCE POINT

Density	Low			Medium			High		
Variable Count	100	200	400	100	200	400	100	200	400
DPP	5.58	11.53	22.96	9.17	18.49	37.16	15.35	29.39	59.81

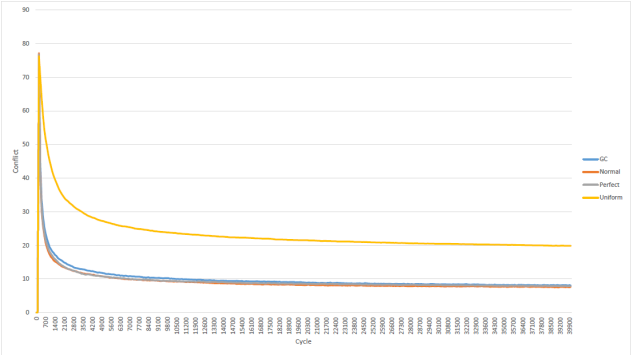

Fig. 11. Number of Conflicts based on different methods of calculating the PDF

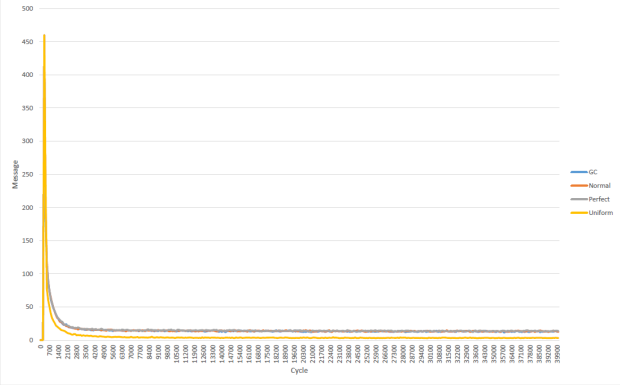

Fig. 12. Number of messages based on different methods of calculating the PDF

We used three different degrees (2.0, 2.3, and 2.7) to evaluate the DPP. These degrees represent low, medium, and high densities respectively. For example, a low density problem with 200 variable problems had an average degree of 2.0, 400 constraints, and a p_1 of 0.0201005. A problem with 400 variables, 800 constraints were generated and the p_1 was 0.010025.

Following procedures is used to randomly generate a problem. First of all, variables are generated and assigned domains. Then the constraints are generated by randomly selecting two variables that do not have a constraint already between them then randomly selecting combinations of values to false. Finally, to each agent one variable is assigned and each agent is known of its assigned variable constraints.

To measure the performance of the algorithms and be able to compare different problems together we measure three things: how fast an algorithm converges on a solution, the overall quality of the solution, and total the number of messages exchanged among agents. These values give an insight to better understand the behaviour of the protocols and be able to predict the algorithms performance on different problems. In a paper by Mailler and Zheng [11] they showed that the convergence rate is an important parameter in evaluating and analyzing the DCSP problems. A very good result about DCSP problem is that the convergence rate remains almost unchanged as the problem size increases. This is shown in the work of Ridgway and Mailler [12].

The tables III and IV show the convergence rate and competence point for random problems with different densities. When we talk about random problem it means that the constraint table among variables are created by random procedure. As results show by increase in the density of the problems the convergence points becomes worse. This completely makes sense as a problem with higher density means more constraints and we expect to have more conflicts in a problem with higher density. In addition, as we see in the result the number of conflicts (convergence point) in problems with 200 variables are almost twice the convergence point of the problem with 100 variables. Furthermore, this pattern follows in the problems with 400 variables and convergence point is about twice the convergence point of the problems with 200 variable.

VII. CONCLUSIONS

In this study, we introduced a new system for tracking the satellites by defining the problem as a Distributed Constraint Satisfaction Problem. Then we applied the DPP algorithm to solve the problem and tested DPP on some examples. In addition, we developed a software to track the object in the sky that is able to simulate the moving objects in the sky with some sky features like the darkness and speed of the earth and satellites and size of stars etc. The software showed to be working very well with different test cases and can be used in real environment to track the satellites. In addition we explained the Satellites tracking system structure and the

required hardware and the process of tracking the satellites. We also mentioned some limitations associated with this system.

ACKNOWLEDGMENT

The author would like to thank Dr. Roger Mailler for his support. This work is sponsored by the National Science Foundation under Grant No. IIS-1350671 and partially supported by the IEEE 2017 Graduate Student Research Grant.

REFERENCES

[1] R. T. Maheswaran, J. P. Pearce and M. Tambe, "Distributed algorithms for DCOP: A graphical-game-based approach," in Parallel and Distributed Computing Systems (PDCS), 2004.

[2] W. Zhang, G. Wang and L. Wittenberg, "Distributed stochastic search for constraint satisfaction and optimization:Parallelism, phase transitions and performance," in AAAI workshop on probabilistic Approaches in Search, 2002.

[3] R. Mailler, "Using Prior Knowledge to Improve Distributed Hill Climbing," in IEEE/WIC/ACM international conference on Intelligent Agent Technology, 2006.

[4] M. Yokoo and K. Hirayama, "Distributed Breakout algorithm for solving distributed constraint satisfaction problems," in International Conference on Multi-Agent Systems(ICMAS), 1996.

[5] R. Newton and W. Thomas, "Design of school bus routes by computer," Socio-Economic Planning Sciences, vol. 3, no. 1, p. 7585, 1969.

[6] S. Samadidana, M. Paydar and J. Jouzdani, "A simulated annealing solution method for robust school bus routing," International Journal of Operational Research, vol. 28, no. 3, pp. 307-326, 2017.

[7] J. Park and B. Kim, "The school bus routing problem: a review," European Journal of Operational Research, vol. 202, no. 2, pp. 311-319, 2010.

[8] J. Braca, J. Bramel, B. Posner and Simchi-Levi, "A computerized approach to the New York City school bus routing problem," IIE Transactions, vol. 29, no. 8, p. 693702, 1997.

[9] M. Yokoo, E. Durfee, T. Ishida and K. Kuwabara, "Distributed constraint satisfaction for formalizing distributed problem solving," in International Conference on Distributed Computing Systems, 1992.

[10] A. R. Leite, F. Enembreck and J. P. A. Barthes, "Distributed constraint optimization problems: Review and perspectives," Expert Systems with Applications, pp. 5139-5157, 2014.

[11] R. Mailler and H. Zheng, "A new analysis method for dynamic, distributed constraint satisfaction," in In Proceedings of the 2014 International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2014.

[12] A. Ridgway and R. Mailler, "Dynamic Theoretical Analysis of the Distributed Stochastic and Distributed Breakout Algorithms," in 14th International Conference on Autonomous Agents and Multiagent System, 2015.

[13] Otsu N., 1979, A threshold selection method from gray level histograms. , IEEE Trans. Syst. Man Cybern (1979). SMC-9, 6266.

[14] Oscar Sapena and Eva Onaindia and Antonio Garrido and Marlene Arangu, "A distributed {CSP} approach for collaborative planning systems",Engineering Applications of Artificial Intelligence, vol 21, no 5, 698-709,2008

[15] S. E. Conry and K. Kuwabara and V. R. Lesser and R. A. Meyer,"Multistage Negotiation for Distributed Constraint Satisfaction",IEEE Transactions on Systems, Man, and Cybernetics,Vol 21, No 6,1991

[16] K. Sycara and S. Roth and N. Sadeh and M. Fox,"Distributed Constrained Heuristic Search",IEEE Transactions on Systems, Man, and Cybernetics, Vol 21, No 6, 1446-1461, 1991

[17] Saeid Samadidana And Roger Mailler,"Solving DCSP problems in highly degraded communication environments", WI '17 Proceedings of the International Conference on Web Intelligence, pages 348-355,2017