**MEGHNAD SAHA INSTITUTE OF TECHNOLOGY**

**COMPUTER SCIENCE AND ENGINEERING**

**PROJECT STATUS REPORT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reporting period:** | **From Beginning of Assigning the project to 9th April 2020** | **Project title:** | SOLVING PUZZLE USING SWARM INTELLIGENCE TECHNIQUE | |
| **Date of report:** | 13/04/2020 | **Project Guide Name:** | DR. SUTIRTHA KUMAR GUHA | |
| **Project members name:** | SAYAK HALDAR, SOURAV DAS, SOUMYA DEEP SAMANTA, PRITAM KUMAR RAY | | **Name of the member submitting this report** | **SAYAK HALDAR** |

**PROJECT MILESTONES STATUS REVIEW**

[NOTE: THIS PART WILL BE FILLED UP IN CONSULTATION WITH ALL MEMBERS IN THE PROJECT GROUP]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PROJECT MILESTONES** | **100 % DONE** | **66 % DONE** | **33 % DONE** | **Not Yet Started** |
| Put **YES** in the designated cell provide for each milestone as per your work status | | | |
| **LITERATURE SURVEY ON YOUR PROJECT**  [Existing Software/Hardware/Research work] | YES |  |  |  |
| **PROBLEM STATEMENT DEFINITION** [Identifying and defining the area/problem/specific topic you want to address] | YES |  |  |  |
| **CONCEPT DEVELOPMENT**  [Knowledge gathering on the topic and learning software languages/tools or hardware tools] | YES |  |  |  |
| **DESIGN PLANNING** [ Defining Design strategy Module-wise /Function-wise breaking of the project , Data Flow Diagram/ Usecase diagram/Object Diagram/Sequence Diagram ] | YES |  |  |  |
| **ALGORITHM DEVELOPMENT** [ Flow chart and Pseudocode or stepwise algorithms ( can be module-wise or for the whole project)] | YES |  |  |  |
| **IMPLEMENTATION** [coding and testing (unit or integration) ] | YES |  |  |  |
| **ACCEPTANCE TESTING** [comparison against previous works or benchmark]- Comparative Studies |  | YES |  |  |
| **RESULT ANALYSIS** [efficiency testing and complexity analysis] |  |  | YES |  |

**PROJECT -MEMBER-WISE PROJECT WORK INVOLVEMENT**

[NOTE: THIS PART WILL BE FILLED UP IN CONSULTATION WITH ALL MEMBERS IN THE PROJECT GROUP]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PROJECT MILESTONES** | *SOURAV*  *DAS* | *SAYAK*  *HALDAR* | SOUMYA DEEP SAMANTA | *PRITAM*  *KUMAR*  *RAY* | *<member name 5>* |
| Put **YES** in the designated cell provide for each milestone as per your work status | | | | |
| **LITERATURE SURVEY** | YES | YES | YES | YES |  |
| **PROBLEM STATEMENT** | YES | YES | YES | YES |  |
| **CONCEPT DEVELOPMENT** | YES | YES | YES | YES |  |
| **DESIGN PLANNING** | YES | YES | YES | YES |  |
| **ALGORITHM DEVELOPMENT** | YES | YES | YES | YES |  |
| **IMPLEMENTATION** | YES | YES |  |  |  |
| **ACCEPTANCE TESTING** | YES | YES |  |  |  |
| **RESULT ANALYSIS** |  |  |  |  |  |

**PROJECT DETAILED SUMMARYOF WHAT YOU HAVE DONE TILL THIS STATUS REPORT DATE**

[NOTE:THIS PART WILL BE SAME FOR ALL MEMBERS IN THE PROJECT GROUP]

**1. WHAT LITERATURE SURVEY ON YOUR PROJECT YOU HAVE DONE?**  [Existing Software/Hardware/Research work] 1000 words

Ans. Sudoku puzzles belong to a set of hard problems called NP-Complete problems. A Sudoku puzzle is a logic-based combinatorial puzzle with rules that are relatively simple. Various algorithms have been applied to solve this combinatorial problem. A relatively new algorithm called Artificial Ant Colony algorithm was developed in 2004. The algorithm mimics the way ants forage for food and has been successfully applied to a wide array of NP-Complete problems.

In this paper we present a new algorithm for the well-known and computationally-challenging Sudoku puzzle game. It explores the possibility of using an improved variant of the Artificial Ant Colony algorithm in solving Sudoku Puzzles. Our Ant Colony Optimization-based method significantly out-performs the state-of-the-art algorithm on the hardest, large instances of Sudoku. The obtained results support the conclusion that the algorithm can be used to solve Sudoku Puzzles efficiently and effectively. We provide evidence that compared to traditional backtracking methods, our algorithm offers a much more efficient search of the solution space, and demonstrate the utility of a novel anti-stagnation operator.

**2. STATE THE PROBLEM STATEMENT DEFINITION** [Identifying and defining the area/problem/specific topic you want to address] 200 words

Ans: Sudoku is a well-known logic-based puzzle game that was ﬁrst published in 1979 under the name of

“Number Place”. It was popularized in Japan in 1984 by the puzzle company Nikoli, and later named

“Sudoku”, which roughly translates to “single digits”.

The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3

Sub-grids that compose the grid (also called “boxes”, “blocks”, or “regions”) contain all of the digits from

1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single

solution.

**3. WHAT CONCEPT YOU HAVE DEVELOPED**  [Knowledge gathering on the topic and learning software languages/tools or hardware tools] 600 words

Ans: A Sudoku problem is made up of a grid of cells (or squares), arranged into 3\*3 sub grids known as boxes. A unit is row, column or box, each containing nine cells. A problem is solved when each unit (that is, every row, every column and box) contains a permutation of the digits 1...9. Any given cell has exactly three units and 20 peers ; the row , the column and box in which the cell resides, and the set of peers is made up of the other cells in those units(that is , 2\*8= 16 neighbours in the relevant row and column plus 4 other cell occupying the same box).

Our algorithm is based on Ant Colony System (ACS) which is a variant Ant Colony Optimization (ACO).

At each iteration, each ant starts with a “fresh” copy of the puzzle and the aim of each ant is fix as many cell values as possible. Each ant starts on a different, randomly selected cell, and then iterate over all cells on the board. Whenever it leaves a cell that does not have a fixed value (that is a cell with a number of possible values), an ant must make a decision on which element of the cell’s value set to choose (thus setting the cell to that value). Importantly, as soon as ant set the value of a cell, the constraints that it introduces are propagated across the board.

**4. WHAT YOU HAVE DONE REGARDING DESIGN PLANNING** [ Defining Design strategy Module-wise /Function-wise breaking of the project , Data Flow Diagram/ Usecase diagram/Object Diagram/Sequence Diagram ]

Ans: The main thing in the design of the algorithm is how an ant will choose which number has to put in a particular cell of the puzzle. Decisions on which value to choose are based on relative pheromone level, which are assigned to each possible value. These are stored in a pheromone  matrix , which keeps track of a single pheromone amount for each possible value in each cell. this is for an order-3 (9\*9) Sudoku puzzle, matrix with 81\*9 values, with each cell corresponding to the pheromone level for each possible value (1..9) in a cell (indexed 1....81). Depending on the greediness of the selection, either the value with the highest pheromone value is chosen, or a weighted selection is made. After the cell’s value is set , the standard ACS local pheromone operator is applied , which reduces the probability of that value being selected by the following ant(thus preventing early convergence).once all ants have covered every square of the board we then perform the global pheromone  update, which rewards only the best solution found.

**5.** **WHAT ALGORITHM YOU HAVE DEVELOPED?** [ Flow chart and Pseudo code or step wise algorithms ( can be module-wise or for the whole project)] 5000 words

|  |
| --- |
| Algorithm 1: Pheromone\_Elimination (ph[9], p) |
| Input:  Total Pheromone of Ant Agent  Pheromone found in a cell |
| Output:  Pheromone available in the Ant Agent after traversing the allotted path |
| Data structures used:  ph[9]:= It is a 1-D array, used for storing the 9 types of pheromone representing digits from 1 to 9;  p:= It is a variable, it stores the pheromone found in the corresponding cell of the Sudoku matrix;  n:= Number of rows of the Sudoku matrix=Number of columns of the Sudoku matrix |
| Step 1: Start  Step 2: Repeat Step 3 to Step 4 for i:=1 to n.  Step 3: Repeat for j:=1 to 9  if (ph[i]=p)  Step 4: i:=i+1, go to Step 2.  [End of Step 2 loop]  Step 5: for i:=0 to (N-1)  Display B[i].  i:=i+1  Step 6: Exit. |

**6.** **WHETHER YOU HAVE IMPLEMENTED THE ALGORITHM ?** [ Explain what are the constraints and assumption you have taken. What are the risks for this project or exception handling you have imposed? ] 500 words

Ans:

**Constraints and Assumptions:**

1. Sudoku board must be 9\*9
2. Minimum number of clues strictly greater equals 17
3. We assumed that the given input is solvable

**Risks:**

1. Linking the puzzle array with the Pheromone matrix is very time consuming and requires lot of hard works to make it efficient.
2. It was just a thought before we implemented the algorithm. We were not sure that it will reduce the time complexity than the classical approach (like Backtracking).
3. Testing is not completely done. Yet it gives more efficient result than other existing approaches.

**7. WHAT COMPARATIVE STUDIES YOU HAVE MADE ?** [comparison against previous works or benchmark]

1500 words

Ans: In previous brute force approach we randomly fill up all the numbers 1-9 in the empty cells and try to solve the 9\*9 Sudoku puzzle and check if all the constraints are being satisfied or not. So there for a single empty cell there are a lot of 9\*9 boards generated in this approach. So there exist so many possibilities in this approach. and in backtracking approach we recursive put the numbers and then check if after putting that number it is satisfying the properties of a Sudoku board, if yes then again filling the next cell, if no then try putting another number and checking the same. So in the backtracking there also exist total of 9 numbers for a single empty cell. In our algorithmic approach we, at first eliminating the numbers which are not suitable for a particular cell after checking the row, column as well as the 3\*3 sub grid in which the empty cell is lying. So the cell now have only the possible candidates those will not violate the constraints. So now if we recursively try to fill the board, we don’t have to try with 9 choices, the choices are less. So it will take less time to solve the board. Again if any of the cell contain only one choice we fill that cell with that choice and again try to eliminate more choices from the cell candidates. In this approach many empty cells can be filled. This takes less time to solve the puzzle than the brute force or backtracking approach. Eliminating most of the choices help us to solve the puzzle in less time.

**8. WHAT RESULT ANALYSIS** **YOU HAVE DONE?** [efficiency testing and complexity analysis]

Ans: We are trying to compare the time complexity of our algorithm with existing classical approaches to solve sudoku. It’s not in the stage to give a conclusive answer.

[NOTE : Save the doc document with this convention as,

**<member name\_Project Group no.\_Date >**

before uploading into Google Classroom and also sending it to respective Project guides. ]