NEW APPROACH FOR DECENTRALIZATION OF A*

Mohammed Chennoufi, Fatima Bendella, Maroua Bouzid

New Approach for Decentralization of A*

Mohammed Chennoufi
Computer Mathematics Faculty,
University of Science and Technology of Oran BP 1505, Oran 31000 Algeria
Mohammed.chennoufi@univ-usto.dz

Fatima Bendella
Computer Mathematics Faculty,
University of Science and Technology of Oran BP 1505, Oran 31000 Algeria.
Fatima.bendella@univ-usto.dz

Maroua Bouzid
University of Caen, Basse-Normandie Campus Cote de Nacre, BP 5186 - 14032 Caen France
bouzid@unicaen.fr

Abstract: *In this paper, we propose a new approach for the decentralization of the A* algorithm called DEPA* (decentralization parallel A*), which is a quick algorithm for finding the shortest path between two nodes in a large graph. The main disadvantage of the A* algorithm is that over the course instance to be processed is large and complex more resolution will be difficult in time of execution and space memory. For this, we propose an approach based on multi-agent systems that decomposes the graph into sub-graphs connecting (many agents). This connection is guaranteed thanks to the characteristics of intelligent agents all computing an A* at each sub-graph in a parallel way. The initial and final states of each agent will be chosen according to well-defined heuristics. A coordinator agent resolves the conflict in the case of many final states in a sub-graph. Then, the agents interact to achieve the goal. We illustrate this approach on a grid connected by square cells.*

Keywords*: Agent;A*;DEPA*;Grid;Node;Muli-Agent Systems.*

1 INTRODUCTION

The distributed approach is presented in our daily lives such as traffic for example going from city A to city B can escalate the latter. Another case is 0the Web Service. If you ask query <how to get from A to B>, the answer may be passed through several server. Booking in a travel agency is also regarded as a distributed approach (car, bus, avian ...).

Our objective is to design an intelligent and effective DEPA* able to accelerate the search of the shortest path between two nodes if it exists. Our algorithm is developed to find an optimal and complete solution. We illustrate our approach on a grid, this grid is divided into several agents: Some A* are run in parallel in accordance with the concept of depth (A* that do not meet certain conditions are not executed), each Agent calculate the shortest path as well as the successor to its final state. A coordinator agent solves the problem of conflict if it exists by calculating the minimum distance to the final state in the sub graph or global heuristics with adjacent agent.

This article is organized according to the following provisions. In the first section, we present previous works in the field of distributed planning and the variations of A*. In the second section, we discuss the modeling of the problem and the explanation of A*. The third section is devoted to the presentation of our approach based on the agents with a pseudo code. Results are given in the fourth section. Conclusion and perspectives are presented in the last section.

2 RELATED WORKS

The Djikstra algorithm (Research in width) (Hart,NJ. Nilsson PE. and B. Raphael, 1968) and its variations as A* so-called A START (search in depth) (Russell S., 1992) are well-known algorithms in artificial intelligence specifically in planning (the calculation of shortest path).

Our art state is divided into three major axes. The first is on the extensions of A* while the second on the distributed planning and the third on the decentralization of A*. (Koenig S. and M. Likhachev, May 2006a) Proposes an algorithm of shortest path based on A* algorithm, its main advantage is in the use of a bounded memory while the A* algorithm uses exponential memory. All other features are inherited from A*. It avoids repeated states as long as the related memory permits it. In (Koenig S., M. Likhachev and D. Furcy, 2004a), the author introduces the notion of time, in order to accelerate repeated heuristic A* research with the same state toward a goal. The idea is to place dates at the state level in its local search spaces in order to make the heuristics better informed after each A* heuristic search. The works of (Koenig S. and M. Likhachev, 2005b) are inspired from (*Koenig S. and M. Likhachev,* 2002b) Dynamic A* which behaves like A*, except that the costs of arc can vary as the algorithm works. Two other states are added namely Raise indicating that its cost is higher than the last time on the Open list and Lower indicating that the cost is less than the last time on the Open list. (Koenig S., M. Likhachev, Y. Liu and D. Furcy, 2004b) Proposes an incremental version of A* so-called D*. The idea is to take advantage of previous researches that they reuse after repair. These changes allow the graph of a well gained execution time better than rescheduling from zero. In (Sun X., S.Koenig and W. Yeoh, 2008a), the authors combine both incremental and heuristic searches. They reuse information from previous searches to speed up searches for similar sequences. GAA* (Generalized Adaptive A*) solves search problems potentially faster. The heuristic search often based on A* heuristics uses heuristics knowledge in the form of approximations of goal distances. GAA* is much faster than uninformed search algorithms. Recently, another algorithm (Sun X., W. Yeoh and S.Koenig,2009b) has been proposed. It is modeled as a grid it depends on a forward chaining search called FRA*(Efficient Incremental Search for Moving Target Search).Whenever the target moves, FRA* quickly adapts the search tree previously built to the new target position and recalls the function of A* search. This tree adaptation is largely dependent on environmental modeling. The changes in the environment are not treated. (Sun X., W. Yeoh and S.Koenig, 2010c) Developed a variant of this algorithm called GFRA*(Generalized Fringe-Retrieving A*) which enables FRA* to operate in environments modeled by arbitrary graph. In addition, the used heuristic function is ineligible. This is the case in the field of planning. (C. Hernández., X.Sun.,S. Koenig., P. Meseguer,2011) proposes a new incremental heuristic search algorithm called Tree Adaptive A* uses the update principle of Adaptive A* to make the h-values of the current A* search more informed.

In (David S., 2005), the author resolves collision; he added the concept of priority in the context of cooperation between agents in finding the shortest path. Each agent is assigned to a priority where agents will be executed in the order of this priority. In (Cáp M, P.Novák, J.Vokrínek and M. Pechoucek, 2012), the authors present an asynchronous variant of decentralized planning, exploiting the parallelism in distributed systems, which gives a speed in calculations. Unlike synchronized planning approaches, algorithm allows an agent to call his local planner of the spatial-temporal trajectory to find the best path. In (C.Hernández ,J.Baier,T.Uras and S.Koenig 2012) the authors introduce the game time model, where time is partitioned into uniform time intervals, an agent can execute one action during each time intervals (Standley1 T. and R. Korf, 2011) propose a complete algorithm that is fast enough for real-time applications based on MAS, at any time when an agent finds a solution. It uses the rest of the time to gradually improve the solution until it is optimal. Algorithm can solve problems at 80% with small grid but with the scale, the problem remains. In (Kamoun M., 2007) the cooperation between agents illustrated by the author is based on two algorithms: First, to find the interface agents that should cooperate to answer a query "how to get from A to B? ", and how to make these agents cooperate. The two algorithms are based on the Djikstra where each agent details the itinerary. The author did not unfortunately give a pseudo code that clarifies how this distribution is achieved.

Work on the A* decentralization is little although the areas covered are decentralized in nature such as road traffic, web service, game theory ...(EL Falou M.,M.Bouzid and M. Mouaddib,2012) proposed a distributed search algorithm DEC-A*(a Decentralized Multiagent Pathfinding) modeled as follows: when a problem is presented, each agent calculates its overall heuristic that estimates the cost of its shortest path to the goal intermediary to its neighboring gents. Then, the agent containing the initial state develops locally in A* by minimizing the cost until it reaches the border. After that, it stimulates on the other side a new execution of A*. These steps are repeated to reach the goal. So, the author has made an extension of the heuristic evaluation of the distance like the sum of two functions, a local, which evaluates the cost to reach the closest node, neighbor to the objective and the overall distance that estimates the cost of sub graph in the target through other graphics. His work decreases the time to find the shortest path and reduces complexity but it is not effective in a graph without hindrance.

In this domain, several studies have been carried out especially in the last 10 years, but we noticed that most of the researchers work on variations or extensions of A* applied in different areas such as game theory, robotics, traffic road.... But, little work on decentralization of A* especially on coordinating agents at boundaries nodes, that is the object of our work

3 MODELLING OF THE PROBLEM

3.1 Notation

Our approach is illustrated in a square grid that represents blocked cells in grey and unblocked in blond. The initial state is colored in green and red for the end state.

S is the finite set of unblocked states, ϵ S is the initial state (starting node) of our research, ϵ S represents the final state (arrival node), ϵ S (heuristics) is the approximation of a current state s to a state s' .In our illustration we work with the Manhattan distance (Pearl J., 1985).

h(s) = |xs-xs'|+|ys-ys'| , where x and y are the coordinates of the cell. C (s, s') = 1 is the cost of transition between adjacent cells in the 4 directions → ← ↓ ↑, C(s, s′) >0 is the cost of transition between the cell s ϵ S and s' ϵ S. G={ , } where is the nth sub graph, each is a sub graph of agent , the agents communicate using a border node Succ( )= , s ϵ and s' ϵ with i ≠j, let λ ={ , ,…. } a set of path to each sub graph starting from an initial border state of to a final boundary state .

A* uses the formula F (x) = g (x) + h (x) which is the current approximation of the shortest path to the goal, where g (x) is the total distance between the initial position to the current position and h(x) is the heuristic function which represents the approximate distance from the current location to the goal c

3.2 Standard A Algorithm Search*

A* pronounced A Star [4] is an algorithm search of artificial intelligence that performs a heuristic search [18] in an area to find an optimal path from the root node to the goal node. The algorithm search is based on a heuristic evaluation between two nodes in order to eliminate many paths of high costs. Two representations are possible namely tree or grid in Figure 1 and 2.

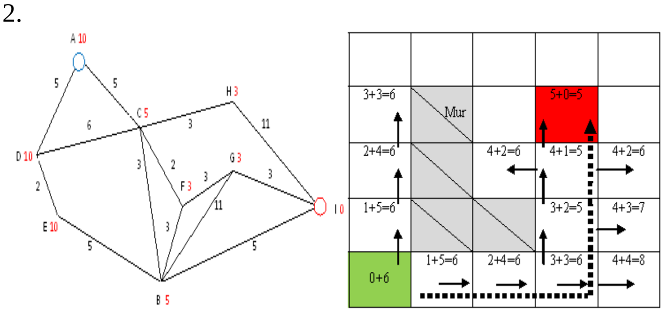

Figure 1: Representation in Greed and Tree

To find a path from one point to another, we must begin by heading to the destination. It is precisely this idea that the A* algorithm uses. The idea is very simple: at each iteration A* will try to get closer to the destination, it will therefore focus on possibilities directly closer to the destination, putting aside all others. All possibilities not allowing to get closer to the destination are set aside, but not suppressed. They are simply put in a list of possibilities to explore (open list) if ever the solution currently explored is poor. Indeed, we cannot know in advance whether a path will lead or be shorter. Enough for this path lead to a dead end that the solution becomes unusable. Algorithm will therefore first move towards more direct paths. And if these paths fail or prove wrong later, it will examine solutions being put aside. This going back to review the solutions that we set aside the algorithm ensures that we always find a solution (if it exists, of course). What makes this algorithm search complete, fast and optimal figure 2. Several distances can be used like the Manhattan distance, diagonal, Euclidean…

An illustrative example of A* running on a grid and a tree with the trace of open list and closed list is shown in Figure 2.

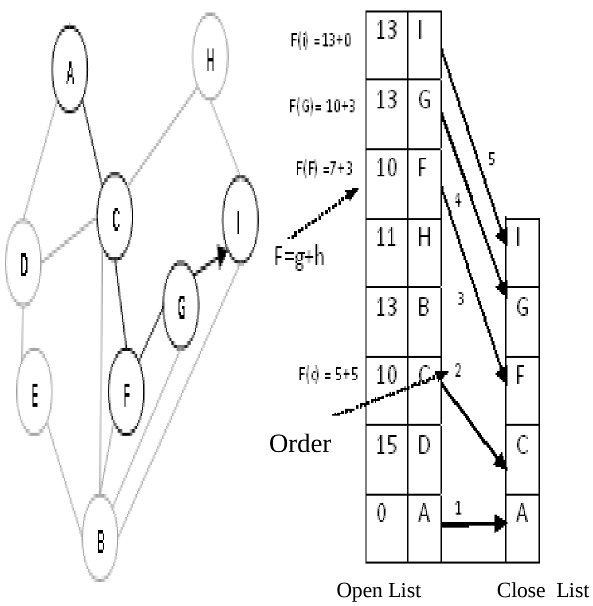

Figure 2: Illustrative Example of A*

4 PROPOSED APPROACH

Our idea is simple but effective. After the decomposition of the graph into a sub-graph (sub-matrix). Each agent calculates respectively the initial state which represents the minimum cost of the border states and the final state representing the minimum of the heuristics of the border states. Then, A* will execute in parallel with the sub-graphs provided that its initial state is a successor of the final status of another sub-graph (red crosses in Figure 3 for the case of untreated parallel). In case of a single processor, a multi thread is run. The agents communicate using a coordinator agent which calculates the overall heuristic [17] of each agent. It will be used in case of conflicts between the final states of each sub-graph and build the final path from s_{init} to s_{goal} .

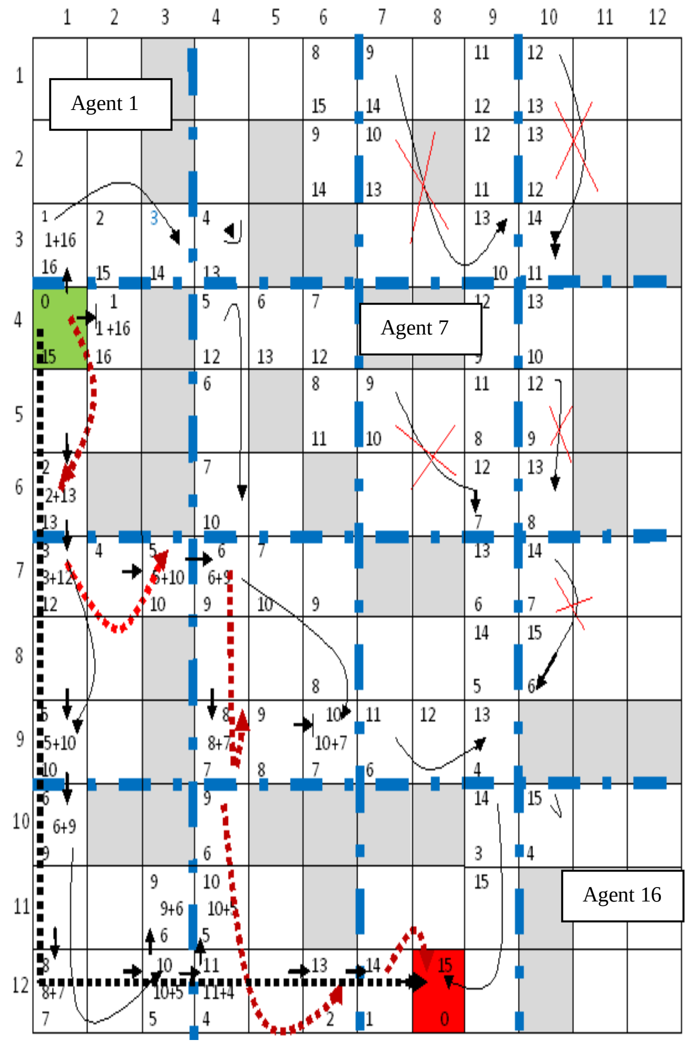

Figure 3: Illustration of an Example DEPA *

Here is a sample execution which explains the algorithm 1 and 2: Our grid is divided into 16 agents from .In each square, the value presented at the upper left corner represents the cost. The bottom left represents the heuristic and f = g+ h is in the middle of the cell. All initial and final states are stored in two vectors provided that the initial state is a successor of the final state.

The initial state of the A_9 agent is which is a successor of for agent. So will be considered contrary to the agent as its initial state is not a successor of of Then, we calculate the A* for agents well sorted and we appeal to the coordinate agent which regulates the conflict in case there are more final states in the same agent (two A* in the case of agent). The Figure 4 shows an example of cooperation between agents where each agent sends a message send () and receives message answer () of the coordination agent which contains the partial path λ_i

4.1 Pseudo code

Algorithm 1: DEPA*

1: **Begin**
 # Initialization
2: **Decomposition** *(grille, nbr noeud);*
3: **For each** *of A* **do**
 # Calculate the initial and final state of each sub graph
4: *= MIN (of node neighbor in);*
5: *= MIN (of node neighbor in);*
 # Condition for parallelism
6: **If** *S == Succ (S) i≠j* **then**
7: *Vector1 = ;*
8: *Vector2= ;*
9: **End of if**
10: **End of for**
11: **For** *i=1 to length[Vector1]* **do**
*# calculate of A * for each agent in parallel*
12: **Thread***[i].* **A*** *(Vector1[S)] , Vector2 [S)];*
13: **End of for**
 # Communication between agents
14: **Cordiagent** ( );
15: *Pathfinal= =*
16: **End.**

The line 2 of the algorithm 1 represents the decomposition of the grid in sub-grid depending on the desired number of agents. Take our example for a grid of 12*12. If we want to have agents with 09 nodes. We must split the rows and columns in 3.3, which gives us 16 agents (4*4). The 4 and 5 lines calculate the minimum of the costs and heuristics for boundary nodes for each agent. They will be stored in two vectors (lines 7 and 8). Much implementations is shown in line 12, which uses the standard function A * in parallel. Line 14 calls he coordinator agents (algorithm 2).

The Algorithm 2 guaranteed the communication between agents (boundary nodes). It is used to calculate the best paths f or each agent (line 12). Three tests are performed (lignes 3, 6.9) to solve the problem of multiple final states. Line 10 calls the heuristic function , which represents the minim um distance for to the final state , is calculated by achai nigback for = to the initial state = [17].

DEPA* is complete, unless there is an infinite number of nodes with $f \le f(G)$. Since h is admissible, best is optimal bec ause it is a simple A *. So is optimal.

Algorithm 2: Cordiagent ( )

1: **Begin**
 # Initialization;
2: **For each** *node of* **do**
3: **If** == **then**
4: = *MIN(c(,) / s'є)*
5: **End of if**
6: **If** *c(,)== c( )* **then**
7: = *MIN (Succ , Succ);*
 # s'є j≠i
8: **End of if**
9: **If** == **then**
10: *= MIN (Succ ,*
Succ); # s'є j≠i)
11: **End of if**
12: *Return best ;*
13: **End of for**
14: **End.**

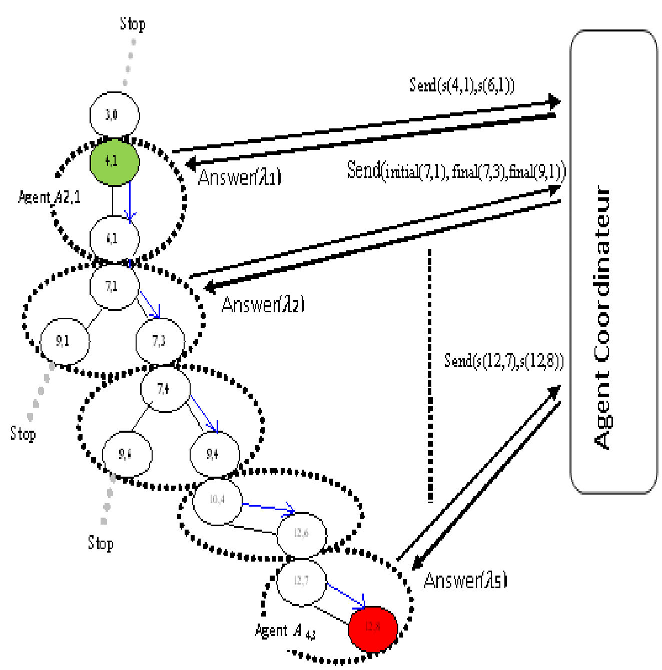

Figure 4: Coordination between Agents

5 EXPERIMENTATION

We have compared our DEPA* algorithm with A* and DECA* on a machine with an Intel (R) Core 2 Duo 3.16 GHz CPU and 2 GB of RAM. After several experiments, we obtained results that illustrate the execution time by varying the size of the problems with respect to the number of input nodes and the number of agents used for their resolution.

Obstacles in our grid that represent walls, rivers, mountains are programmed in a way random all depends on the problem. We used a Boolean function. When creating the grid. It is true for Adjace0nt nodes (not obstacle) if Math .random exceeds 0.1.

Grid size	A* en ms	DECA* en ms	DEPA* en ms
12*12	32	15	15
20*20	47	17	16
50*50	218	256	150
100*100	1000	410	300

TABLE 1: EXPERIENCE WITH DIFFERENT GRID

The first aspect we noted in figure 5 is the execution time of DEPA* which calculates the shortest path from node (0, 3) to (40, 40). This time with DEPA* is the least by contribution to A* and DECA*. Even if the grid is large (table 1: 10000, 25000 nodes).

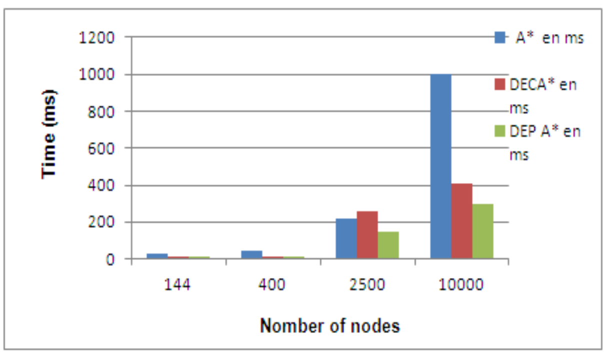

Figure 5: Comparaison between Variations of A*

Grid size	Number of agents	A* ms	DECA* ms	DEPA* ms
30*30= 900	10*10 agents of 3*3 nodes	153	145	16
	2*2 agents of 15*15 nodes	110	59	38

TABLE 2: EXPERIENCE WITH A GRID (30*30)

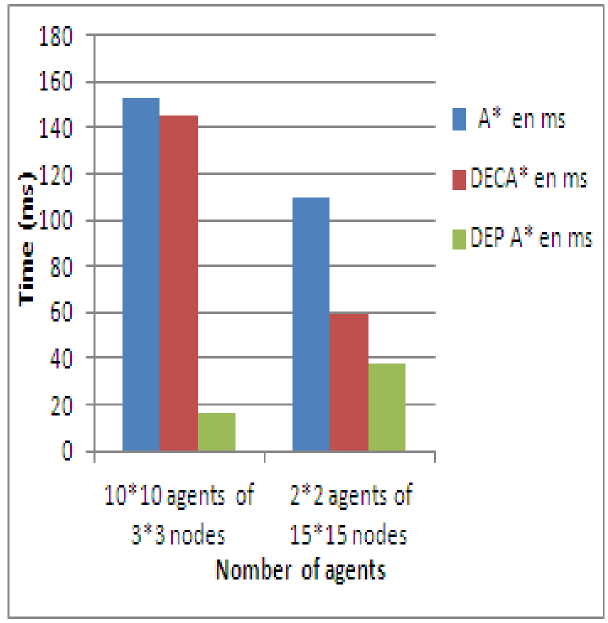

Figure 6: Grid of 30*30 Nodes

In Figures 6 and 7, which illustrate the tables 2 and 3, we notice the profit in the execution time in relation with the number of agents for 03 algorithm. DEPA* is the best in the time of executions. Consider the case of Figure 6, which shows a grid of 30 * 30 with initial and final states are (0.3) and (29.29), for a breakdown of 10 * 10 agents where each agent has 3 * 3 node. DEPA* runs in 16 milliseconds but in the decomposition of 2* 2 agents with 15 * 15 node, DEPA* runs in 38 ms which expresses the power of parallelism and coordination between agents.

Figure 8 shows the performance of the DEPA* on scale with initial and final states are at the end of the grid: (3.3) and (170.170). We notice that as the number of agents increases the time is decreased 983 ms for 4 agents and for 36 agents we have 109ms. This expresses a time saving through the parallel execution of A * at each agent.

Grid Size	Number of agents	A* ms	DECA* ms	DEPA* ms
50*50= 2500	10*10 agents of 5*5 nodes	218	57	15
	2*2 agents of 25*25 ndes	313	413	94

TABLE 3: EXPERIENCE WITH A GRID (50*50)

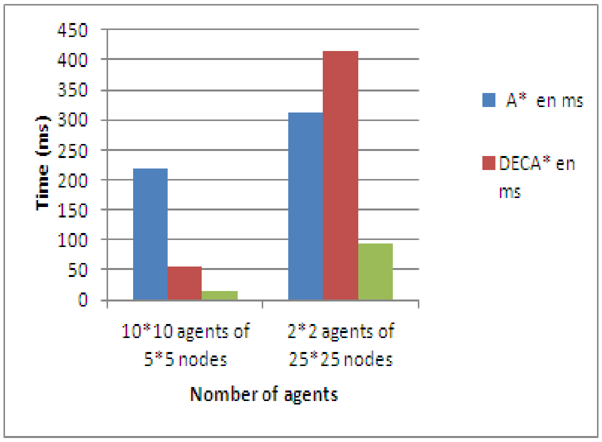

Figure 7 : Grid of 50*50 Nodes

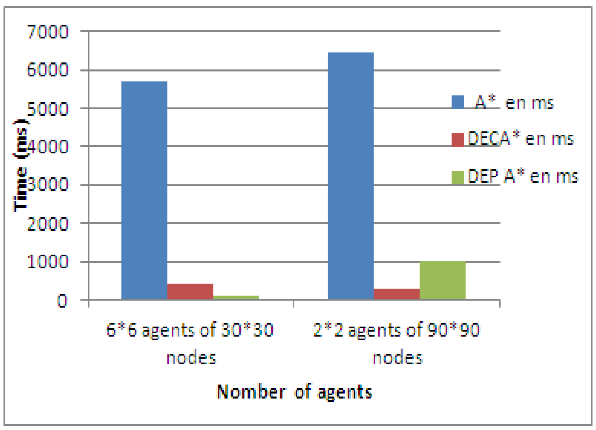

Figure 8: Grid of 180*180 Nodes

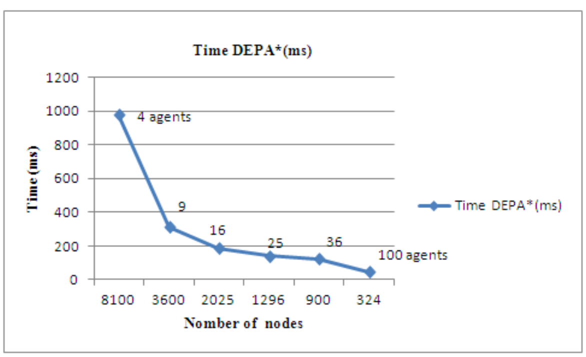

Figure 9: Grid of 180*180 Nodes

The figure 9 shows the relationship between the increase in the number of agents contribution to the reduction of time execution and the number of nodes for each agent on a grid of 180 * 180 nodes using the parallelism of A*.

We notice that, in our experiments, DEPA* finds a solution if it exists in at least a second with grids that can contain up to 32400 nodes.

From these results, we can say that decentralization is an appropriate approach for this type of problem when the graph is large.

6 CONCLUSION AND FUTURE WORKS

Networks grow continuously, which makes the system more complex. More recent works in artificial intelligence handle the problem of shortest path. The multi-agent systems are helping to solve this complexity with a decentralized manner through a "send and answer" communication and the coordination between agents in order to achieve the goal.

The proposed DEPA* algorithm is in keeping with this problem. We have illustrated our approach on a square grid like the game grid, which allows computing the shortest path from an initial state of an agent to a final state of another. It is based on the parallel A* that is run on agents of which their initial states are successors of the final states of other assistants agents. Much of the work is devoted to the coordination between agents to arrive at the final path in the case of several A* (several final states) in the sub grid (agent).

To determine the relevance of this approach, we undertook tests on different instances, by varying the size of the problems and the number of agents used for their resolution. The comparisons between A *, A * DEC and DEPA* show that on the whole if the results are not significant on small instances, they become much more visible as soon as you increase the size of the problems.

We obtained good results, especially in a scale of 25000 nodes and 32400 nodes where DEPA* finds the solution if it exists in some milliseconds. In future researches, we are interested in the extension of the field of application, such as the simple and dynamic graphs, when the final state changes in position.

REFERENCES

Hart,NJ. Nilsson PE. and B. Raphael, 1968. *A formal basis for the heuristic determination of minimum cost paths*, IEEE Transactions on Systems, Science, and Cybernetics, 100-107.

Russell S., 1992. *Efficient memory-bounded search methods*. In Neumann, B. Proceedings of the 10th European Conference on Artificial intelligence. Vienna, Austria: John Wiley & Sons, New York, NY. pp. 1.

Koenig S. and M. Likhachev, May 2006a. AAMAS," Hakodate, Japan

Koenig S., M. Likhachev and D. Furcy, 2004a. *Lifelong Planning A**. Artificial Intelligence Journal, 155, (1–2), 93–146.

Koenig S. and M. Likhachev, 2005. *Adaptive A**, In Proceedings of the International Conference on Autonomous Agent and Muti-Agent Systems, p. 1311–1312.

Koenig S. and M. Likhachev, 2002b. *D* lite*. In Proceedings of the Association for the Advancement of Artificial Intelligence, p. 476–483.

Koenig S., M. Likhachev, Y. Liu and D. Furcy, 2004b. *Incremental Heuristic Search in Artificial Intelligence*, Artificial Intelligence Magazine, 25(2), 99-112.

Sun X., S.Koenig and W. Yeoh, 2008a. *Generalized adaptive A**, In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, p. 469–476.

Sun X., W. Yeoh and S.Koenig,2009b. *Efficient incremental search for moving target search*, In Proceedings of the International Joint Conference on Artificial Intelligence, p. 615–620.

Sun X., W. Yeoh and S.Koenig, 2010c. *Generalized Fringe-Retrieving A**. Faster Moving Target Search on State Lattices". In Proceedings of the Internation Joint Conference on Autonous Agents and MultiAgent Systems, volume 1081–1088.

David S., 2005. *Cooperative Pathfinding*, Proceedings of the First Conference on Artificial Intelligence and Interactive Digital Entertainment: pp. 117-122.

Cáp M, P.Novák, J.Vokrínek and M. Pechoucek, 2012. *Asynchronous Decentralized Algorithm for Space-Time. Cooperative Pathfinding* , Workshop proceedings ECAI.

Standley1 T. and R. Korf, 2011. *Complete Algorithms for Cooperative Pathfinding Problems*, Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence.

Kamoun M., 2007. *Designing an information system using multimodal travel : A multi-agent approach to search and composition routes online* ", PHD thesis lile university.

EL Falou M.,M.Bouzid and M. Mouaddib,2012. *DECA*:a Decentralized Multiagent Pathfinding Algorithm*", AAAI.

Pearl J., 1985. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley.

Hernández,C., Baier,J.,Uras,T.,Koening,S., 2012. *Time – bounded adaptative A*, 11th International Conference on Autonomous Agents and Multiagent System, AAMAS:Innovative Application Track,1pp.224-231.

Hernández, C., Sun, X., Koenig, S., Meseguer, P., 2011. *Tree adaptive A**, 10th International Conference on Autonomous Agents and Multiagent Systems 2011, AAMAS 2011, 1, pp. 113-120.