Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Interception of a Moving Object in a FIFO Graph

M. M. Hizem [*] **E. Castelain** [**] **A. Toguyeni** [***]

[*] *LAGIS, Lille, France (e-mail: mohamed-mejdi.hizem@ec-lille.fr).*
[**] *LGI-L, Lille, France (e-mail: emmanuel.castelain@ec-lille.fr)*
[***] *LAGIS, Lille, France (e-mail: armand.toguyeni@ec-lille.fr)*

Abstract: This paper solves the problem of intercepting a moving object in FIFO graphs. In the graph, a first mobile (the target) is moving with a known fixed itinerary. The second mobile (the pursuer) aims to reach the first one with the minimum delay. So, our goal is to find the shortest path for the interception taking into account the dynamic nature of the graph. This interception problem can be applied to the case of an urban transport company that needs to send a rescue team to one of their buses that is still in movement. First, we propose a model to set the basic problem. Then we present an interception algorithm based on Dijkstra algorithm for shortest path computation in a graph. After, we prove that the result of the algorithm is optimal. Finally, we compute the complexity of the algorithm and we prove its efficiency.

1. INTRODUCTION

Graphs are used for modelling different types of systems. Indeed, they can be used to model road networks, computers network, amusement parks, etc. Graphs can give a static view of the system or a dynamic one. Static graphs have been studied for a long time and major problems have been solved. In opposition, dynamic graphs are an active research field. Indeed, problems solved for static graphs generally become harder to solve in the dynamic case. One of those problems is the shortest path computation between two nodes. Although several algorithms exist to solve the problem for static graphs, it is NP-hard for general dynamic graphs (Orda and Rom [1990]). Nevertheless, the problem can be solved for a subclass of dynamic graphs called FIFO graphs. One interesting problem is the computation of the best path to reach a moving target in a graph. This situation can be faced when we are trying to intercept a vehicle in a road network. For example, many urban transport companies using buses face the following situation : a bus driver announces that an incident occurred in his bus (mechanical, security, etc.). So, a rescue team must be sent to help it. If the incident is minor, the bus does not stop. Consequently, the rescue team must intercept the bus that is still in movement. The bus and the rescue team have GPS devices. So their exact positions are known. In this context, we have to determine where the bus can be intercepted and how to reach this point. Moreover, the road network is a dynamic system where the delay in the different roads changes. So we need an algorithm to compute the interception path for the rescue team that takes into account the varying delays in the different roads.
The aim of this paper is to present an algorithm that computes the best interception path in a FIFO graph. Indeed FIFO graphs are a subclass of dynamic graphs that have useful proprieties and can represent road networks. We will first propose a model to represent the interception problem for FIFO graphs. Then, we will propose an algorithm to solve it. After, we will prove the optimality of the result of the algorithm and we will compute its complexity. Finally we will present the results of different simulations conducted to evaluate the execution time.

2. LITERATURE REVIEW

Shortest path problem for static graphs was extensively studied in the past. Surveys concerning the subject can be found in Cherkassky et al. [1996] and Gallo and Pallottino [1988]. Several algorithms were developed and used in different fields. When treating transportation networks, another formulation of the problem rises : the *time-dependent shortest path problem* (Palma et al. [1993], Nachtigall [1995]). This problem treats the computation of a shortest path between two nodes of a dynamic graph. Dynamic graphs are graphs where edge weight is function of time. In the general case, the time-dependent shortest path problem is NP-hard (Orda and Rom [1990]). However, this problem can be solved for a subclass of dynamic graphs called FIFO graphs. It was proved that for FIFO graphs any static shortest path problem can be extended to solve the problem with the same complexity as in the static case (Ahn and Shin [1991], Kaufman and Smith [1993]).
Previous works treating the interception are mainly limited to the continuous case. Planar interception has been largely studied and has been frequently used in robotics. Some papers treat the case where the target trajectory is predictable. In this case we have to compute a unique interception point (Park and Lee [1992], Mikesell and Cipra [1994]) or a point that can vary if the target trajectory is modified (Croft et al. [1998]). Other papers treat the case where the target trajectory is unpredictable. This hypothesis, does not allow specifying an interception point but requesting to pursuit the target until the interception (Lei and Ghosh [1993], Papanikolopoulos et al. [1991], Belkhouche and Belkhouche [2004], Gans [1997]). The spatial interception has also been studied. Several authors investigated the mechanisms allowing humans to catch things like balls (McBeath et al. [1995], McLeod and

Dienes [1993]). Those works also have been used in robotics (Thomas et al. [2003], Borgstadt and Ferrier [2000], Suluh et al. [2001]). In opposition of the great number of works for the planar interception, according to our knowledge, there is no study about the interception problem in dynamic graphs.

3. THE INTERCEPTION PROBLEM MODEL

3.1 FIFO graphs

FIFO graphs are a subclass of dynamic graphs. A dynamic graph is a directed weighted graph $G = (N, A)$ with N the set of nodes and A the set of edges. The edge weight is time-dependant. So, a weight function $d(u, v, t)$ is associated to each edge $(u, v) \in A$. When dealing with transportation networks, an edge weight represents generally a delay. Consequently, $d(u, v, t)$ is called delay function. An edge $(u, v) \in A$ is said to be a First-In-First-Out edge if its delay function is defined such as leaving a node u for a node v at $t' \geq t$ implies to arrive to node v later than $t + d(u, v, t)$. More formally :

$$\forall t, t' \geq 0,\ t \leq t' \Longrightarrow t + d(u, v, t) \leq t' + d(u, v, t')$$

A graph is called FIFO graph if all its edges are FIFO edges. FIFO condition generally holds in road networks especially when they are congested. That is why such graphs can be used to represent road networks.

3.2 The model

In our model, a FIFO graph $G = (N, A)$ is considered. In the real world, G represents a road network : a node represents a bus stop or a crossroads and an edge represents a direct road linking two nodes. An edge weight represents the delay needed to go from the origin of the edge to its destination. The model developed in this paper is a discrete model. In consequence, the time is devised in equal periods of length T. During each period T, $d(u, v, t)$ is supposed to have a constant value. After M periods of time, $d(u, v, t)$ is assumed to take a constant value. In other words, $d(u, v, t) = constant\ \forall t > M.T$ and $(u, v) \in A$. Furthermore, $\forall (u, v) \in A$ and $\forall t \geq 0$, $d(u, v, t) \in \mathcal{N}^*$. In the general case, the target and the pursuer can move in different graphs. For example, buses and cars do not use necessarily the same roads and have not the same speed. Indeed, generally, there is bus lanes in road networks preventing buses to be delayed by traffic jams and even one-way roads dedicated to buses. In consequence, two graphs G_t and G_p are considered in our work. $G_t = (N_t, A_t)$ is the graph where the target moves and $G_p = (N_p, A_p)$ is the one where the pursuer moves. The delay function associated with each graph is respectively $d_t(u, v, t)$ and $d_p(u, v, t)$. In our model G_t and G_p must share some nodes but they do not share any edges. Even if the target and the pursuer are using the same road (for example there is no bus lane), this road is represented by two different edges, one in G_t and the other in G_p, that have the same delay. In consequence, $N_t \cap N_p \neq \emptyset$ and $A_t \cap A_p = \emptyset$.
In the following parts of the paper, TV designates the Target Vehicle (the bus in our example) and PV designates the Pursuer Vehicle (the rescue team in our example). $Init_{TV} \in N_t$ and $Init_{PV} \in N_p$ are special nodes that represent respectively the initial position of TV in G_t and PV in G_p. Fig. 1 depicts an example. G_t has nine nodes $N_t = \{1, 2, 4, 5, 6, 7, 8, 9, 10\}$ and G_p has nine nodes $N_p = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The set of shared nodes is $N_t \cap N_p = \{1, 2, 4, 5, 6, 7, 8, 9\}$. In the figure, edges belonging to A_t are represented by dashed lines and edges belonging to A_p are represented by continuous lines. The initial position of TV is the node 1 and the initial position of PV is the node 7.

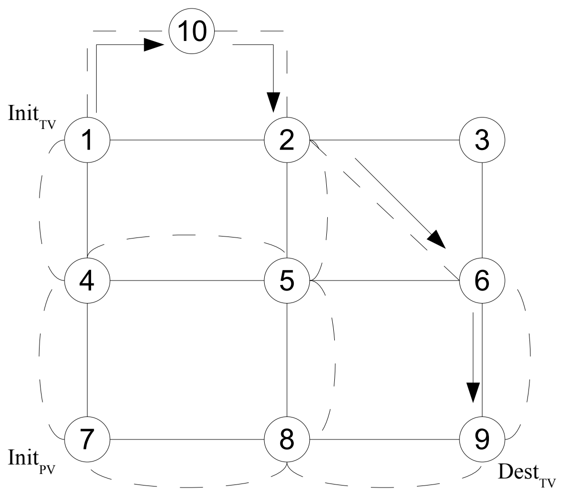


Fig. 1. Example of graphs G_t and G_p

Hypothesis 1. The functions d_t and d_p are supposed to be known in advance for the whole period of study.

Definition 1. The set of paths between $Init_{TV}$ and $w \in N_t$ is represented by $SPath_{TV}(w) = \{Path_{TV}(w),\ w \in N_t\}$ having $Path_{TV}(w) = (u_0, u_1, \ldots, u_l)$ with :

- $u_k \in N_t,\ \forall k \in [0, l]$
- $u_0 = Init_{TV},\ u_l = w$
- $\forall u_k, u_{k+1} \in Path_{TV}(w),\ (u_k, u_{k+1}) \in A_t$

In the same way, the set of paths between $Init_{PV}$ and $r \in N_p$ is represented by $SPath_{PV}(r) = \{Path_{PV}(r),\ r \in N_p\}$ having $Path_{PV}(r) = (v_0, v_1, \ldots, v_m)$ with :

- $v_k \in N_p,\ \forall k \in [0, m]$
- $v_0 = Init_{PV},\ v_m = r$
- $\forall v_k, v_{k+1} \in Path_{PV}(r),\ (v_k, v_{k+1}) \in A_p$

Hypothesis 2. TV can follow a unique path called $It(TV)$, connecting its initial position called $Init_{TV}$ to its final position called $Dest_{TV}$. $It(TV)$ is supposed to be known and fixed. Moreover, TV cannot pass through a node more than once.

The previous hypothesis is adapted to the context of bus interception because a bus has a known and fixed itinerary. Furthermore, a bus stops only once at every bus stop and, in general, it does not pass through a crossroad more than once. Even this special case can be easily modeled with respect to the previous hypothesis. Indeed, the node representing the crossroad is duplicated with duplicating all the edges linking it to the other nodes of G_t and G_p. Fig. 2 presents an example where the node 1 is duplicated.

Definition 2. Suppose that $It(TV) = (u_0, \ldots, u_q, \ldots, u_l)$, the partial itinerary to u_q is defined as the path $It(TV, u_q) = (u_0, \ldots, u_q)$.

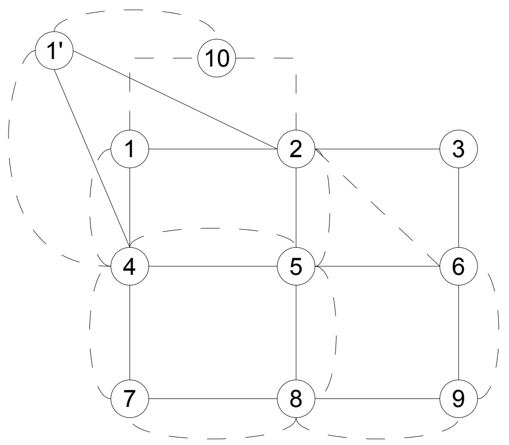


Fig. 2. Example of a node duplication

In the example of the Fig. 1, $Init_{TV} = 1$, $Dest_{TV} = 9$, $It(TV) = (1, 10, 2, 6, 9)$ and $It(TV, 2) = (1, 10, 2)$.

Definition 3. Let a_t be the arrival time function for G_t. The parameters of this function are the path in the graph and the departure time from the origin node. So, $a_t^{t_0}((u_0, u_1, \ldots, u_m))$ denotes the arrival time at node u_m when leaving u_0 at t_0 and following the path $(u_0, u_1, \ldots, u_m)$. The value of $a_t^{t_0}((u_0, \ldots, u_m))$ is defined as follows :

$$\begin{cases} a_t^{t_0}((u_0, \ldots, u_{m-1}))+ & \\ d_t(u_{m-1}, u_m, a_t^{t_0}((u_0, \ldots, u_{m-1}))) & \text{If } m > 0 \\ t_0 & \text{If } m = 0 \end{cases}$$

In the same way, a_p denotes the arrival time function G_p and the value of $a_p^{t_0}((v_0, \ldots, v_m))$ is defined as follows :

$$\begin{cases} a_p^{t_0}((v_0, \ldots, v_{m-1}))+ & \\ d_p(v_{m-1}, v_m, a_p^{t_0}((v_0, \ldots, v_{m-1}))) & \text{If } m > 0 \\ t_0 & \text{If } m = 0 \end{cases}$$

Definition 4. The shortest path from $Init_{PV}$ to $r \in N_p$ at t_0 is designated by $SP_{PV}^{t_0}(r)$ and it is the path allowing to reach r at the earliest time when leaving $Init_{PV}$ at t_0. More formally, $SP_{PV}^{t_0}(r) = Path_{PV}(r)$ with $a_p^{t_0}(Path_{PV}(r)) = min_{P \in SPath_{PV}(r)}\{a_p^{t_0}(P)\}$. In case of several paths with minimal arrival time, one is randomly chosen.

Definition 5. A node w is called interception node at departure time t_0, if $w \in N_t \cap N_p$, $w \in It(TV)$ and $a_p^{t_0}(SP_{PV}^{t_0}(w)) \leq a_t^{t_0}(It(TV, w))$. This means that w is a shared node between G_t and G_p, it belongs to the target itinerary and PV reaches w before TV when both leave their initial position at t_0.

Having all those definitions, our goal is the following

Goal : Find an interception node OPT at departure time t_0 such as $a_t^{t_0}(It(TV, OPT))$ is minimal. Such a node is called *optimal interception node at departure time* t_0.

4. SOLUTION FOR THE INTERCEPTION PROBLEM

4.1 The algorithm

Our solution for the dynamic interception problem in FIFO graphs consists on adapting the Dijkstra algorithm. The Dijkstra algorithm was developed for computing the shortest path between two nodes in static graphs. It is proved that any static shortest path algorithm can be extended to compute the shortest path between two nodes in FIFO graphs with the same complexity (Ahn and Shin [1991], Kaufman and Smith [1993]). Our idea is to compute the shortest path between $Init_{PV}$ and a destination node Obj. First Obj is initialized to the first node of $It(TV) \cap G_p$ starting from $Init_{TV}$. If, during the computation, $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) \leq a_t^{t_0}(It(TV, Obj))$ (t_0 is the departure time) then Obj is the optimal interception node. Otherwise the next node in $It(TV) \cap G_p$ is chosen as the new destination node Obj. The algorithm is the following :

```
1   //Compute the cost of the $TV$ itinerary nodes
2   Let $It(TV) = (u_0, u_1, u_2, \ldots, u_m)$
3   $\forall u_k \in It(TV)$ compute $a_t^{t_0}(It(TV, u_k))$
4   //Initialise the shortest path of each node of $G_p$
5   $SP_{PV}^{t_0}(Init_{PV}) = (Init_{PV})$
6   $\forall r \in N_p \ / \ r \neq Init_{PV},\ SP_{PV}^{t_0}(r) = \emptyset$
7   $//SP_{PV}^{t_0}(r) = \emptyset \Rightarrow a_p^{t_0}(SP_{PV}^{t_0}(r)) = \infty$
8   //Initialise the set $Q$ of eligible nodes
9   $Q = \{Init_{PV}\}$
10  //Initialise the set $F$ of nodes with final cost
11  $F = \emptyset$
12  //Initialise the destination node
13  $Obj = Init_{TV}$
14  If $Obj \notin It(TV) \cap G_p$ Then
15      $Obj = \text{next}(It(TV), Obj)$
16      //If there is no shared node then FAILURE
17      //FAILURE 1
18      If $Obj = \emptyset$ Then return FAILURE
19      End If
20  End If
21  While $Q \neq \emptyset$ do
22  Select $S_i$ from $Q$ with $a_p^{t_0}(SP_{PV}^{t_0}(S_i))$ minimal
23  $Q = Q \backslash \{S_i\}$ and $F = F \cup \{S_i\}$
24  //CONDITION 1
25  If $a_p^{t_0}(SP_{PV}^{t_0}(S_i)) \leq a_t^{t_0}(It(TV, Obj))$ Then
26      //CONDITION 2
27      If $S_i = Obj$ Then return SUCCESS
28      End If
29  Else
30      //NEXT 1
31      $Obj = \text{next}(It(TV), Obj)$
32      //FAILURE 2
33      If $Obj = \emptyset$ Then return FAILURE
34      End If
35      //CONDITION 3
36      While $Obj \neq \emptyset$ and $Obj \in F$ do
37          //CONDITION 4
38          If $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) \leq a_t^{t_0}(It(TV, Obj))$
39              Then return SUCCESS
40          Else
41              //NEXT 2
42              $Obj = \text{next}(It(TV), Obj)$
43          End If
44      End While
```

```
45         //**FAILURE 3**
46         If $Obj = \emptyset$ Then
47             return FAILURE
48         End If
49     End If
50     Develop successor $S_j$ of $S_i$
51     For each $S_j$ do
52         If $a_p^{t_0}(SP_{PV}^{t_0}(S_i)) + d_p(S_i, S_j, a_p^{t_0}(SP_{PV}^{t_0}(S_i)) <$
53         $a_p^{t_0}(SP_{PV}^{t_0}(S_j))$
54         Then $SP_{PV}^{t_0}(S_j) = SP_{PV}^{t_0}(S_i) \cup \{S_j\}$
55         $Q = Q \cup \{S_j\}$
56         End If
57     End For
58     End While
```

Suppose that $It(TV) = (u_0, \ldots, u_q, \ldots, u_m)$, the function 'next$(It(TV), u_q)$' is defined by

$$next(It(TV), u_q) = \begin{cases} u_k & \text{If } q < k \leq m \\ & \textit{and } u_k \in It(TV) \cap G_p \\ & \textit{and } \nexists\, l \in [q+1, k-1] \textit{ such as} \\ & u_l \in It(TV) \cap G_p \\ \emptyset & \text{Otherwise} \end{cases}$$

4.2 Algorithm result optimality

In this section the optimality of the solution given by our algorithm is proved. In the different proofs, many properties of the Dijkstra algorithm are used. Those properties are valid for FIFO graphs according to Ahn and Shin [1991] and Kaufman and Smith [1993].

Lemma 1. In any iteration of the algorithm, if $a_p^{t_0}(SP_{PV}^{t_0}(S_i)) > a_t^{t_0}(It(TV, Obj))$ then $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) > a_t^{t_0}(It(TV, Obj))$.

Proof. This lemma is a direct consequence of the Dijkstra algorithm. Indeed, in any iteration of the algorithm the processed node has a lower cost than the cost of all the nodes that will be processed in the next iterations. As Obj has not been selected yet, $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) \geq a_p^{t_0}(SP_{PV}^{t_0}(S_i)) > a_t^{t_0}(It(TV, Obj))$.

Lemma 2. If the algorithm is treating a node Obj then it does not exist $w \in It(TV) \cap G_p \backslash \{Obj\}$ such as $a_t^{t_0}(It(TV, w)) < a_t^{t_0}(It(TV, Obj))$ and $a_p^{t_0}(SP_{PV}^{t_0}(w)) \leq a_t^{t_0}(It(TV, w))$. In other words, it does not exist a node w that is an interception node at departure time t_0 and located before Obj in $It(TV)$.

Proof. If the function 'next$(It(TV), Obj)$' is called then the current Obj is not an interception node. Indeed, If the function 'next$(It(TV), Obj)$' is called, then $a_p^{t_0}(SP_{PV}^{t_0}(S_i)) > a_t^{t_0}(It(TV, Obj))$ (NEXT 1 or 2). Thus, according to lemma 1, Obj is not an interception node. In consequence, when treating a node of $It(TV)$, all node treated previously are not interception nodes.

Lemma 3. If the algorithm returns SUCCESS then the node Obj is an interception node at departure time t_0. This means that $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) \leq a_t^{t_0}(It(TV, Obj))$.

Proof. If the algorithm returns SUCCESS then $a_p^{t_0}(SP_{PV}^{t_0}(Obj)) \leq a_t^{t_0}(It(TV, Obj))$ according to CONDITION 1 and 2 or 3 and 4. Conclusion, Obj is an interception node.

Proposition 1. If the algorithm returns SUCCESS then Obj is the optimal interception node at departure time t_0. In other words, it does not exist an interception node $w \in It(TV) \cap G_p \backslash \{Obj\}$ such as $a_t^{t_0}(It(TV, w)) < a_t^{t_0}(It(TV, Obj))$.

Proof. According to lemma 3, if the algorithm returns SUCCESS then Obj is an interception node. In addition, according to lemma 2, it does not exist another interception node with a lower cost than $a_t^{t_0}(It(TV, Obj))$. Consequently, Obj is the optimal interception node at departure time t_0.

Proposition 2. If the algorithm returns FAILURE then there is no interception node at departure time t_0.

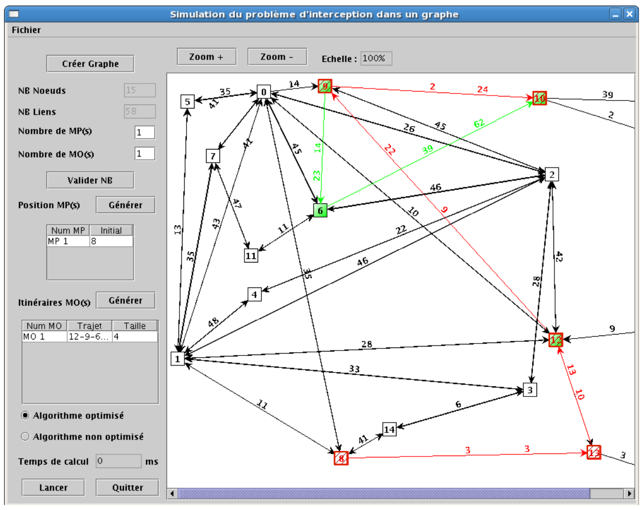
Proof. If the algorithm returns FAILURE then either there is no shared nodes between $It(TV)$ and G_p (FAILURE 1) or there is no shared nodes w such as $a_p^{t_0}(SP_{PV}^{t_0}(w)) \leq a_t^{t_0}(It(TV, w))$ (FAILURE 1 or 2 with lemma 2). In conclusion, there is no interception node at departure time t_0.

4.3 Algorithm complexity

The algorithm, developed in this section, results from adding some operations to the original Dijkstra algorithm. Those operations allow the initialisation of the destination node and its modification during the computation of the shortest paths. The added lines are from 1 to 3, from 12 to 20 and from 25 to 49. The added code contains essentially if statements that have no impact on the complexity. However, there are two loops. The first in line 3 and the second in lines 36 to 44. The loop in line 3 computes the arrival time for nodes in $It(TV)$. So, at most, there are $|N_t|$ iterations. The loop in lines 36 to 44 has no influence in the complexity of the main loop located from line 21 to 58. Indeed, this loop changes the destination node Obj thanks to the function '$next(It(TV), u_q)$'. This function allows getting the next node in $It(TV) \cap G_p$. Since, $It(TV) \cap G_p$ contains at most $|N_p|$ nodes, this function gives $\emptyset$ after $|N_p|$ calls at most. In consequence, the loop of lines 36 to 44 can make at most $|N_p|$ iterations for the *whole execution* of the algorithm independently of the main loop. The complexity of the main loop is $\bigcirc(|A_p| + |N_p|.ln(|N_p|))$ according to Barbehenn [1998]. In conclusion the complexity is $\bigcirc(|N_t| + |A_p| + |N_p|.ln(|N_p|))$. Supposing that $|N_t| \leq |N_p|$ the complexity becomes $\bigcirc(|A_p| + |N_p|.ln(|N_p|))$.

4.4 Algorithm implementation

The algorithm presented in section 4.1 has been implemented in JAVA and has been tested on different examples to show its efficiency. Fig. 3 represents a screenshot of this software. In the different tests, G_t and G_p are equals. This propriety was considered in the aim to maximize the number of shared nodes and, in consequence, to make the computation harder for the algorithm. For the tests, different FIFO graphs with different nodes number were generated randomly. The graph structure is generated using Waxman method (Waxman [1988]) with $\alpha = 0.15$ and $\beta = 0.2$. The delay function for each edge is generated randomly such as the FIFO condition is respected. For each graph, several tests are performed with a random selection of $Init_{PV}$ and $It(TV)$. The results of those tests are given in Fig. 4. All the tests were performed on a PC equipped with an Intel Core Duo processor of which the frequency is 1.83 GHz. The curve of Fig. 5 depicts the progression of execution time in function of nodes number. The results show that the algorithm is very efficient in term of execution time. For example it needs only 33 ms for a graph containing 5000 nodes. Moreover, the execution time increases proportionally with the nodes number of the graph.



Echelle : 100%

NB Noeuds 15

NB Liens 58

Nombre de MP(s) 1

Nombre de MO(s) 1

Position MP(s)

Num MP	Initial
MP 1	8

Itinéraires MO(s)

Num MO	Trajet	Taille
MO 1	12-9-6...	4

[● Algorithme optimisé]

○ Algorithme non optimisé

Temps de calcul 0 ms

Fig. 3. Software screenshot

Nodes number	Average Execution Time (ms)
500	5
1000	7
1500	9
2000	12
2500	14
3000	18
3500	20
4000	23
4500	26
5000	33

Fig. 4. Average execution time in function of nodes number

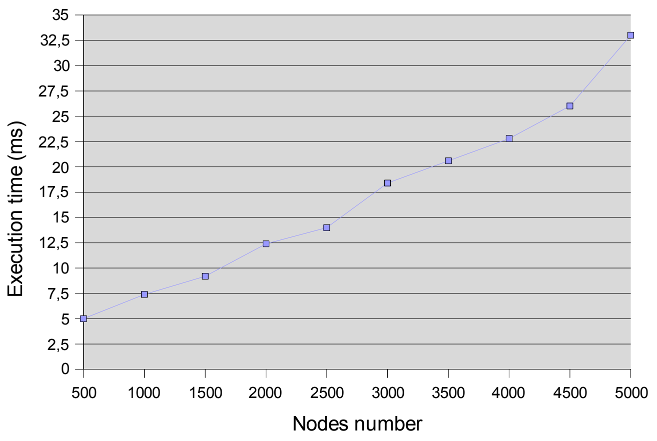


Fig. 5. Average execution time in function of nodes number

5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a model for the interception problem in FIFO graphs and an algorithm to solve it. Moreover, we have proved the efficiency of the algorithm and the optimality of its result.
In this work, we suppose that the delay function is known for the whole period of study. In the future, we will treat FIFO graphs of which the delay functions are not known or known with uncertainties. In consequence, many difficulties rise.
First, as the delay functions are not known, an optimal interception point cannot be computed off-line. So, we have to propose an on-line algorithm that takes into account the changing value of the delay functions.
Secondly, we have to predict the delay variation in time. The delay in a road depends on the traffic flow and several other events like accidents, traffic jams, sport events . . . In consequence, we have to investigate the different models for those variations.
Finally, we can suppose that different pursuers are presents in the graph and we have to choose one to intercept the target. When knowing the delay functions, it is easy. We compute the optimal interception path for every pursuit vehicle and then we take the best one. But, without the delay functions, it is more difficult. Indeed, an optimal interception path computed at t may differ from the optimal path computed at $t + \delta t$. So, we have to specify the criterion of choice. In addition, we have to decide if we authorize the substitution of the selected pursuit vehicle if there is an unexpected event. For instance, if the pursuit vehicle were blocked in its way by an accident, we would choose another one to intercept the target.

REFERENCES

B. H. Ahn and J. Y. Shin. Vehicle-routing with time windows and time-varying congestion. *The Journal of the Operational Research Society*, 1991.
M. Barbehenn. A note on the complexity of dijkstra algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 1998.
F. Belkhouche and B. Belkhouche. A control strategy for tracking-interception of moving objects using wheeled mobile robots. In *43rd IEEE Conference on Decision and Control*, pages 2129 – 2130, Atlantis, Bahamas, dec 2004.
J. A. Borgstadt and N. J. Ferrier. Interception of a projectile using a human vision-based strategy. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3189 – 3196, California, USA, 2000.
B. V. Cherkassky, A. V. Goldberg, and T. Radzik. Shortest paths algorithms : Theory and experimental evaluation. *Mathematical Programming*, 1996.
E. A. Croft, R. G. Fenton, and B. Benhabib. Optimal rendezvous-point selection for robotic interception of moving objects. *IEEE Transactions on Systems, Man and Cybernetics*, 28:192 – 204, apr 1998.
G. Gallo and S. Pallottino. Shortest path algorithms. *Annals of Operations Research*, 1988.
R. Gans. A control algorithm for automated pursuit. In *IEEE international conference on control applications*, pages 907 – 911, Connecticut, USA, 1997.

D. E. Kaufman and R. L. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1993.

M. Lei and B. K. Ghosh. Visually guided robotic tracking and grasping of a moving object. In *IEEE 32nd Conference on Decision and Control*, pages 1604 – 1609, Texas, USA, dec 1993.

M. K. McBeath, D. M. Shaiter, and M. K. Kaiser. How baseball outfielders determine where to run to catch fly balls. *Science*, 268(5210):569 – 573, 1995.

P. McLeod and Z. Dienes. Running to catch the ball. *Nature*, 362(6415), 1993.

M. D. Mikesell and R. J. Cipra. Development of a real-time intelligent robotic tracking system. In *ASME 23rd Mechanism Conference*, pages 213 – 222, Minnesota, USA, sep 1994.

K. Nachtigall. Time depending shortest-path problems with applications to railway networks. *European Journal of Operational Research*, 1995.

A. Orda and R. Rom. Shortest path and minimum delay algorithms in networks with time-dependent edge length. *Journal of the ACM 37*, 1990.

A. De Palma, P. Hansen, and M. Labbe. Commuters paths with penalties for early or late arrival times. *Transportation Science*, 1993.

N. Papanikolopoulos, P. K. Khosla, and T. Kanade. Vision and control techniques for robotic visual tracking. In *IEEE International Conference on Robotics and Automation*, pages 857 – 864, California, USA, apr 1991.

T. H. Park and B. H. Lee. An approach to robot motion analysis and planning for conveyor tracking. *IEEE Transactions on Systems, Man and Cybernetics*, 22:378 – 384, 1992.

A. Suluh, T. Sugar, and M. McBeath. Spatial navigational principles: Applications to mobile robotics. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1689 – 1694, 2001.

K. M. Thomas, G. Sugar, and M. K. McBeath. Perceptual navigation strategy: A unified approach to lnterception of ground balls and fly balls. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 3461 – 3466, Taipei, Taiwan, sep 2003.

B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 1988.