(0) 17 pages

A Review of Auto-Guided-Vehicles routing algorithms

Mahidzal Bin Dahari[1,a]， Liu Yang[2,b]

[1,2]Faculty of Engineering, University of Malaya, Malaysia
[a]mahidzal@um.edu.my
[b]adanos@sina.com

Keywords: AGV, routing, algorithm, collision-free

Abstract: In this paper, several principles of popular routing algorithms and models of Auto-Guided-Vehicles are presented. Tests were conducted and it was found that the Bezier Curve solution is the most unsteady algorithm because it is highly relative to the complexity of maps, particle density, curve dimension, numbers of AGVs, etc. Fuzzy logic is one of the easiest algorithms in programming, but the accuracy is highly dependent on the algorithm structure and parameters. In general hybrid methods are recommended in research and application. Meanwhile, Neural Network is an efficient assistant method in routing.

1. Introduction

As the number of AGV systems increases in part flow systems, routing problems have increased the complexity of AGV system management.

The AGV routing problems is one of the basic problem in AGV systems. Each AGV should be guided by efficient and collision-free routing algorithms to accomplish their tasks. The routing algorithm should be fast. That is, an AGV system may have tens of AGVs working simultaneously, if the algorithm calculation costs obvious long period, other AGVs may be blocked. The algorithms should be able to resolve dead-locks. For example, a fork AGV may have to drop a load in a position that might block other AGVs. This might be solved by enlarging the aisle which may be difficult to do if the system is running.

The vehicle's chosen path should be that it does not affect the existing active travel schedule. Maxwell et al. [22] developed a bi-directional AGV management system which computed the minimum number of AGV in a time-dependent environment to maximized efficiency with time ignored and collision problem unsolved. Gaskins and Tochoco[24] had developed a time-dependent model to solve the uni-directional AGV route optimization solution by minimizing the total travelling distance with weak robustness (the algorithm can easily run into dead-locks). Broadbent et al.[25] coined the concept of conflict-free shortest time route.

A matrix was generated by applying Dijkstra algorithm that describes the time occupied by a vehicle at a node. Conflicts at a node or catching-up conflicts can be resolved by slowing down the vehicle, which is yet to be scheduled. The procedure can be applied to any type of path, namely uni-directional and bi-directional models, and introduce the concept of virtual tunnel for the bi-directional path consisting of several segment of multi-uni-directional paths. This concept allows multiple AGVs simultaneously crossing at an intersection.

Egbelu [23] is one of the earliest researchers in bi-directional AGV path planning problems and

discussed different types of bi-directional AGV travelling problems. Egbelu[23] also compared uni-directional AGV systems and bi-directional AGV systems in an environment of the same layouts. The solution seemed efficient in simple models however in large scale maps it is easy to reach dead-locks.

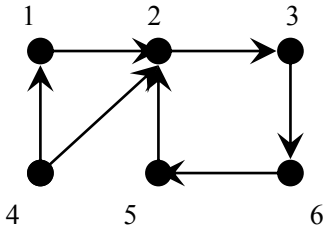
Dowsland[26] focused on the collision-free problems using petri-net to find possible collisions in a given node. The solution is composed of delays and deviations while Endo used the same net with genetic algorithm. Both of the solutions are only effective in small area problems but when dealing with large/complex maps, the algorithm is not capable of meeting real-time requests.

AGV routing algorithms can be classified into 2 categories: Wire-Guided AGV routing algorithms and Wireless-Guided AGVs routing algorithms. The former algorithms are normally modeled via graph theory and the latter ones are modeled via field analysis (or meshed maps) explained in section 3.

2. Wire-Guided AGV routing algorithms

The path-nets of Wire-Guided-AGVs are normally represented by graphs. The routes of an AGV can be described as either a set of edges or a set of nodes. In a given graph **G=<E, V>**, edges **E** and vertices **V** are the usual measurements of inputs and time-costs.

The connection relationship between vertices is normally represented via adjacency list or adjacency matrix. Such as a directed graph with 6 nodes:



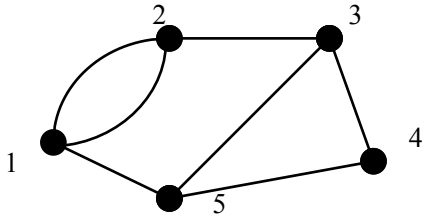
1	2	
2	3	
3	6	
4		2
5	2	
6	5	

$$\begin{pmatrix} 0&1&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&0&0&0&0&1 \\ 1&1&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&1&0 \end{pmatrix}$$

（a） A directed (b) adjacent list of (a) (c) adjacent matrix of (a)

Figure 2.1 adjacent list and adjacent matrix representation of a simple directed graph

The following picture shows the link and matrix for complex graph by multi-layers:



(a) A complex un-directed

Layer1	*Layer2*
1-2-5	1-2
2-1-3	2-1
3-2-4-5	
4-3-5	
5-1-3-4	

(b) Adjacent list of (a)

Layer 1 *Layer 2*

$$\begin{pmatrix} 0&1&0&0&1 \\ 1&0&1&0&0 \\ 0&1&0&1&1 \\ 0&0&1&0&1 \\ 1&0&1&1&0 \end{pmatrix} \qquad \begin{pmatrix} 0&1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix}$$

(c) Adjacent matrix of (a)

Figure 2.2 adjacent matrix and link for a complex graph

If we want to put the distance information into the adjacent matrix, we can use the following step:

i. Set the 0 in the adjacent matrix as ∞ which means there is no link between the 2 vertices;

ii. Set the 1 in the adjacent matrix as the real/scaled distance between the 2 vertices;

Then a distance matrix is formed. In the following algorithms we prefer distance matrix in order to save the computation costs.

2.1 Dijkstra Algorithm

One of the most popular algorithm used in shortest path problems is simple graph searching. The algorithm starts at the origin node in the graph, then search for the linked nodes and marked the nodes with values of distance from the original node. After that, the algorithm checks all the nodes in the graph until all the nodes are marked. The shortest path is formed by connecting the nodes with the lowest values until the end node.

The algorithm has 2 sets of vertices S and Q. Set S contains all vertices which we know the cost of the shortest path d[v]. Set Q contains the unknown ones. When the algorithm starts, Set S is empty. In each step one vertex is moved from Q to S. The vertex is chosen as the vertex with lowest value of d[u]. When a vertex u is moved to S, the algorithm relaxes every outgoing edge.

Dijkstra pseudo-code[1]:

```
1.  Function Dijkstra(G,w,s)
2.      For each vertex v in V[G] //initialization
3.          do d[v]:=infinity
4.             previous[v] := undefined
5.      d[s] = 0
6.    S := empty set
7.    Q:= set of all vertices
8.    while Q is not an empty set
9.        Do u := Extract – Min(Q)
10.           S := S union{u}
11.           for each edge (u,v) outgoing from u
12.              do if d[v] > d[u]+ w(u,v) //relax(u,v)
13.                 Then d[v] :=d[u] + w(u,v)
14.                      Previous[v] := u
```

The Dijkstra Algorithm is a useful algorithm but it has got high computation cost. In a computation cycle, the checking action is repeated n^2 times where n is the number of nodes on the graph. Therefore the algorithm is time consuming particularly when managing very large maps because the worst case performance is $O(n^3)$[1]. For example, in a 256*256 blocked map, the

algorithm might request 0.3s or more on personal computers (depends on the hardware performance). And this delay is always not acceptable in practical AGV systems.

2.2 Floyd-Washal Algorithm

The principle of Floyd-Washal Algorithm is to find the node that is deletable if the path from the original node to the end node does not have any influence. The time and volume complexity are the same as Dijkstra Algorithm[12].

The computation function is:

$$shortestPath(i,j,k) = \begin{cases} edgeCost(i,j) & if\ k = 0 \\ min(shortestPath(i,j,k-1), \\ \quad shortestPath(i,k,k-1) \\ +shortestPath(k,j,k-1)) & else \end{cases} \quad (1)$$

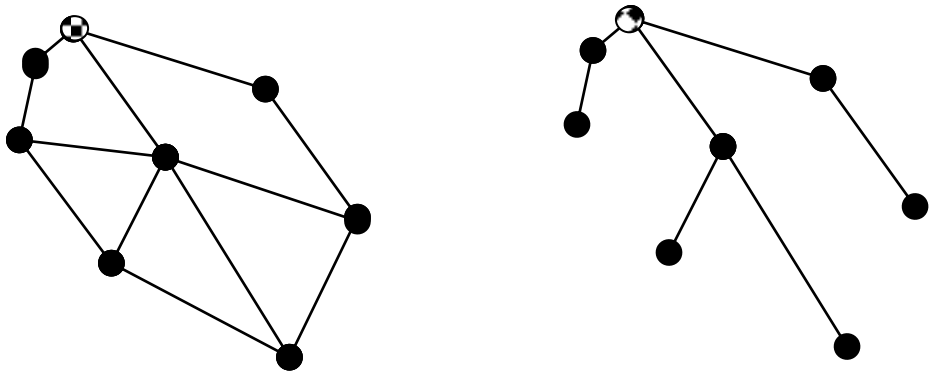
Floyd Algorithm is also a robust algorithm which is able to find the best solution. However it is also very time-costing in multi-AGV real-time routing problems.

2.3 Minimum-spanning-tree

In a connected, undirected graph **G = (V, E)** with the weight function w: E->**R**, if a minimum-spanning-tree is desired, we can use the following step (*Prim algorithm*) to find the shortest path of all the vertexes to the destiny vertex (*yellow point*):

a. *Find all the vertexes which are directly connected to destiny vertex, collect them (including the destiny vertex) into a new set V_1, collect other vertexes into set V_2;*
b. *From V_2, if a vertex $v \in V_2$ and have at least 2 edge connected with vertexes v_1', $v_2' \in V_1$, keep the shortest edge and delete the other edge, put v into V_1 and delete v from V_2;*
c. *Repeat step b until the number of $v \in V_2$ do not change;*
d. *Minimum-spanning-tree is found.*

A graph example is shown in figure 2.3:



(a) Original Graph (b) Minimum-spanning-tree

Figure 2.3 The development of a Minimum-Spanning-Tree

The Minimum-Spanning-Tree can be made to run in time O(E lgV) using ordinary binary heaps[1]. The principle is similar to Dijkstra algorithm.

2.4 Time-dependent algorithms

The time dependent algorithms had shown certain stability and even forecast the time-cost for

the tasks of AGV system. However, most of the time-dependent algorithms have the following assumptions:

a. Each AGV have the same maximum speed s_m;
b. Each AGV need the same time intervals t_m and distances d_m to reach the maximum speed;
c. Each AGV have the same speed in turning, s_t;
d. Each AGV need the same distance d_b and time to brake t_b;

Suppose we have the following path in which the lengths of the edges are known.

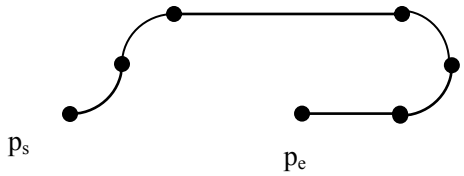


Figure 2.4 a path of AGV in graph

The time-cost of this path is easy to get by:

$$T = t_m + t_b + \sum(Es_i - d_b)/s_m + \sum(Et_i - d_b)/s_t \ , \qquad i = 1, 2, 3\ldots \tag{2}$$

Where Es_i means the length of straight edge i and Et_i means the length of bending edge i.

Then the position of AGV on the path can be forecasted. The path searching of other AGV can proceed by checking whether the edge is busy at a certain time.

We can find the adoption of time-dependent algorithms in several factories in which, most of the AGVs are not allowed to reach the motor/engine's maximum speed due to the algorithm's limitation. If the time dependent algorithms have more adaptability, the efficiency of the system will be significantly improved.

2.4 Genetic Algorithm in shortest path searching

Genetic Algorithm has been proved to be useful in various research fields. In wired-AGV path searching, Lee, etc.[6] developed an optimization model with various crossover methods checked on Genetic algorithm. They also proposed a way to combine Genetic algorithm with artificial potential algorithm.

In a typical path searching, we can adapt it by the following steps:

a. Initialize a path from $\boldsymbol{E \in G}$;
b. Set the path from step a as father chromosome c_1;
c. Repeat step a, generate several other father chromosomes, collect them into set $\boldsymbol{C}$;
d. Find 2 edge-chains ec_1 and ec_2 from 2 different c_1, $c_2 \in \boldsymbol{C}$ with the same start vertex and end vertex;
e. Find the shorter edge-chain ec from ec_1 and ec_2, delete the ec_1 and ec_2 in c_1 and c_2, insert g into the shorter c, the new c is set as a child chromosome cc (crossover);
f. Generate several child chromosome by repeating step e;
g. Compute the path length of each cc, set it as the fitness of cc;
h. Randomly select 2 none-adjacent vertex v_1 and v_2 in cc, if there is an edge $e \in \boldsymbol{E}$ between them, represent the edge chain from v_1 to v_2 by e (mutation);
i. *If generation < maximum generation, goto step d, else task over, the shortest cc is the shortest path.*

It can be seen that GA can find the shortest path. However, the path might not be the best path due to the random selection. This problem can be minimized by increasing the maximum generation.

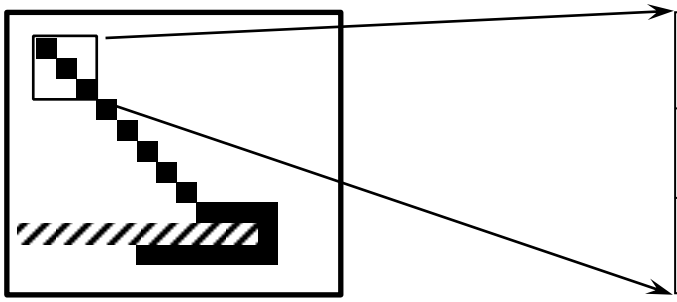
However, this measure will increase the computation time-costs.

3. Wireless-Guided AGV routing algorithms

One of the most common application of wireless-guided AGV routing methods is zone control. That is, manually divide the AGV's workspace into several squares (zone), each AGV working in different squares (zone) to prevent themselves from colliding. Zone control is easy to model and the mathematical calculation is efficient and robust. Hybrid methods are also easy to plug in zone control such as A* algorithm, hybrid control in task optimization, etc.

3.1 A* Algorithm

A* algorithm is a heuristic algorithm. It marked the current node in the search with heuristic variable H(s) to help the algorithm determine whether the current node is a first choice for later searching. By ignoring the irrelevant nodes the efficiency can be improved. A* algorithm is widely used in PC game developments.



105	110	110
110	100	110
110	110	120

(a) An application of A* algorithm (b) Depth value of the left-top 9 zones

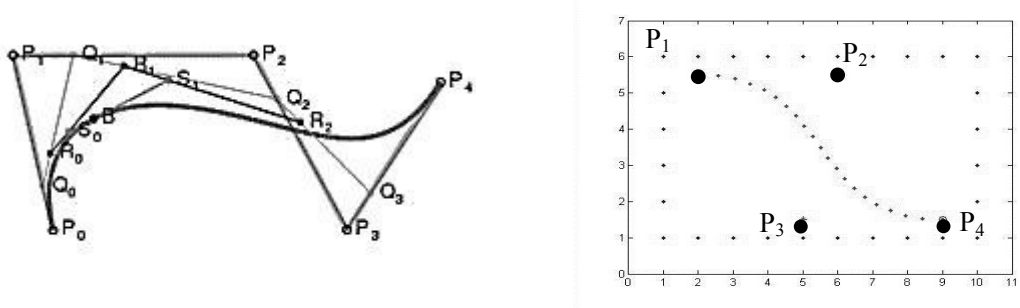
Figure 3.1 An example of A* algorithm

The depth values in Figure 3.1(b) are obtained by the heuristic function $h(x) \leqq d(x,y)+h(y)$, point 5 gets the best heuristic value among the points of 2, 4 and 5. An important potential advantage is that the point 3 and point 7 would not be computed in and by continuously computation the algorithm significantly reduced the total computation cost.

A* algorithm can be considered as a heuristic extension of Dijkstra algorithm. However, A* can reduce the computation complexity remarkably compared with Dijkstra algorithm and Floyd algorithm. In some case, it showed some inaccuracies if the heuristic function is not suitable for the map. If we want to apply A* algorithm into wire-less AGV routing, additional kinematic control methods should be added to the A* algorithm to prevent AGVs from local deadlocks.

3.2 Bezier Curve with Genetic Algorithms (BCGA) in AGV route optimization

The solution of BCGA is a time-based approach in mathematical structure which means the optimization is based on 3 or 4 dimensions in which one of the dimensions is time dimension. Due to the smooth curves and constant intervals between nodes, the Bezier Curve is a useful way to solve the wireless-guided routing problems by optimizing the control points (P_1, P_2, P_3 in figure 3.2.1).



(a) Construction of a quartic Bezier Curve (b) a Bezier Curve for a single AGV

Figure 3.2.1 The Bezier Curve and route of AGV

The mathematical structure of Bezier Curve is:

$$B(t)=\sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^{i}P_i=(1-t)^{n-1}tP_1+\cdots+\binom{n}{n-1}(1-t)t^{n-1}P_{n-1}+t^nP_n, t\in[0,1] \quad (3)$$

Parameter t is able to be considered as the time dimension of the AGV. Hence, we can get the time-dependent model of the path.

The AGV route of 3-degree Bezier Curve is shown in figure 3.2.1 (b):

P_1 is the start point and P_4 is the end point with the path in red dots. As can be seen, the curve is smooth for the AGV to make continuous kinematic control.

The path is given by:

$$X_i=P_{1x}(1-t)^3+3P_{2x}(1-t)^2t+3P_{3x}(1-t)t^2+P_{4x}t^3$$
$$Y_i=P_{1y}(1-t)^3+3P_{2y}t(1-t)^2+3P_{3y}(1-t)t^2+P_{4y}t^3 \quad (4)$$

Where P_{ix} is point P_i's x coordinate and P_{iy} is the y coordinate, i=1, 2, 3, 4 and $t\in[0, 1]$.

Respectively, the distance between 2 relative dots on the path is relative to the current curvature. Therefore, Bezier curve is an efficient way to measure the velocity at a certain point on the path.

As has can be seen in figure 3.2.1, the Bezier Curve can solve 2 main problems:

1. Parallel approaching problem: when the AGV is approaching the workstation, there should be a method to make the AGV's direction parallel to the workstation;
2. Turning radius limitation: Most of the AGVs have a minimum turning radius (MTR), and the turning radius is critical to the AGV's performance. Using Bezier Curve, it is very easy to find the minimum turning radius of the route and to optimize the route.

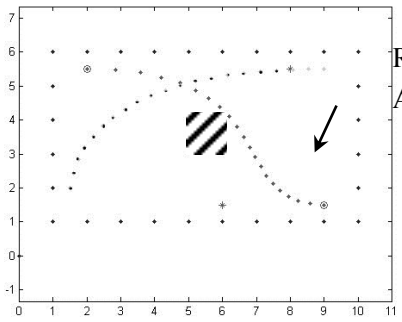
Therefore, it is easy to apply Genetic algorithm in the Bezier Curve by optimizing the position of the control points (P_2 and P_3).

An example of the rules is:

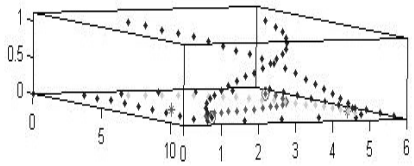
$$\begin{cases} fitness = \alpha \times l_c \times r_t; \\ if\ the\ curve\ collide\ with\ the\ workstation, \\ \quad fitness = fitness \times 1000; \\ if\ the\ curve's\ mimimum\ turning\ radius\ is \\ \quad less\ than\ the\ AGV's\ mimimum \\ turning\ radius, fitness = fitness \times 1000; \\ if\ the\ curve\ collides\ with\ other\ obstacles, \\ \quad fitnes = fitness \times 1000; \\ if\ the\ curve\ collides\ with\ other\ curves\ in\ time \\ \quad scale, fitness = fitness \times 1000; \end{cases} \quad (5)$$

Where, α is a scale constant

The fitness computation takes both the length of the curve and turning radius into consideration in order to optimize the route and kinematics. Meanwhile, the functions of LC and TR are able to be optimized due to the complexity of the maps.



Routes of AGV



Where the dashed rectangle represents the obstacle.

Figure 3.2.2 experimental result of a multi-AGV collision-free problem based on BCGA

It is a time-dependent model because the x and y coordinate represents the position on the map while the z dimension represents time. If a new curve is being designed, the minimum distance between 2 AGVs can be represented by the curves' distance. Then the optimization of the new curve is:

$$\forall\ t \in t_{nc}\ ,\forall\ c \in \boldsymbol{C},\ d(c_n,c)>d_s \quad (6)$$

Where t_{nc} is the time cost of new curve, $\boldsymbol{C}$ is the set of existing AGV path (curve), $d(c_n,c)$ is the distance between the new curve and an existing curve, d_s is the safe distance of AGVs

3.3 Artificial Potential Field Algorithm (APF) in AGV routing problem

The APF consider the 2-dimension environment of the AGV as a 3-dimension field with gravity from the target point and repulsion from the obstacles and other AGVs.

The APF is known as an efficient algorithm in omni-directional AGV routing problems (OARP) and able to give information of the obstacles by evaluating the resultant forces to the AGVs:

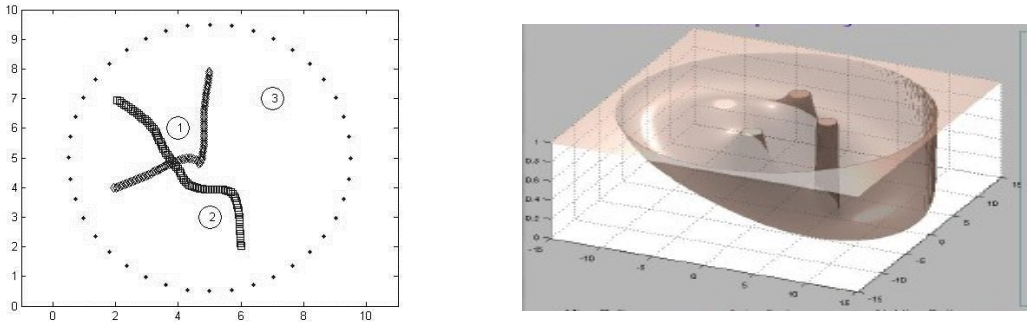
APF is believed to originate from physical potential field research for similarity. Nowadays various algorithms based on APF have been developed in AGV routing problems. Toshio[27] developed an APF model with adjustable temporal behavior. This model had certain ability to surpass the restriction of workspace[4]. In the collision-free model developed by Mark[5], it is also

applicable to optimize the navigation if we use Genetic algorithm as an assistance to APF.

A simple APF force rule is:

$$\begin{cases} F_i = f(D_i),\ i=1,2,\ldots n \\ F = \sum F_i + F_c,\ i=1,2,\ \ldots\ n \\ V = \Phi(F) < V_{max}, \end{cases} \tag{7}$$

Where F_i is the force between the AGV and an obstacle point/obstacle/AGV, F is the total force; F_c is a random force to prevent from the force balance point; V is the velocity of the AGV.



(a) the path of AGVs (b) three-dimension view of APF[3]

Figure 3.3.1 APF solution for AGV routing problem

For the current AGV, all the other AGVs are considered as obstacles. All the obstacles are gravitative (or repulsive) to the current AGV, and proportional to the distance. The target point of the current AGV is repulsive (or gravitative, contrary to the obstacles) to the current AGV. When the AGV encounters a region where F=0 (dead lock), the F_c will become none-0 value which gave the AGV a very small force to leave the region.

However, this solution is not a perfect solution because the selection of parameter is a critical problem. Another remarkable problem is the dead-lock which means if the field is too complex, in some places where the sum of force is 0 will cause self-lock.

A more effective method for fast mobile robots is uni-vector field approach. Jong-Hwan Kim[22] used 2 point named g and r to represent the target point and orientation. The uni-vector field is generated heuristically to find the optimized route for current AGV in figure 2.1.7(c).

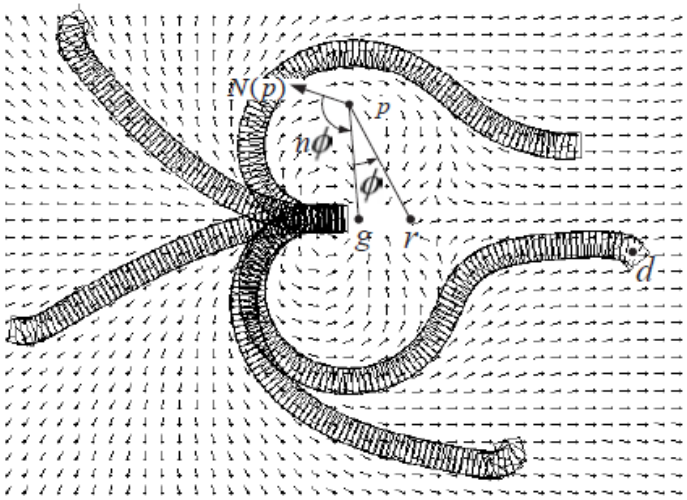


Figure 3.3.2 uni-vector field for final posture at a point

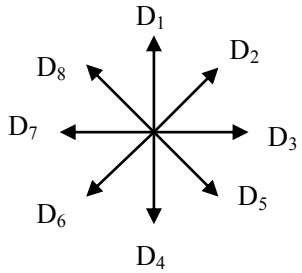
This method successfully solved the problem of the constraints of turning radius and wheel velocity. Simulation show it can prevent AGVs from oscillation

3.5 Fuzzy logic approach in wireless-AGV path finding

Fuzzy Logic algorithm is proved to be applicable in omni-directional wire-less guided AGVs, even when the working environment is unknown. Liao, etc. [2] developed the Fuzzy logic navigation system on AGV via several photosensitive resistances on the AGV. ZHANG YING, etc. [6] developed the model via rough set with genetic algorithm.

One of the obvious advantages of Fuzzy Logic algorithm is that the fuzzy set is able to control the speed and motor of AGV even without other additional control methods or accurate mathematical models.

In a typical fuzzy logic controlled AGV, 8 possible directions are put into the direction set C_d as condition attributes.



$D_i \in C_d$
i=1, 2, 3…

Figure 3.5.1 direction definition

And decision table gave the direction recommend for the controlling:

Table 3.5.1 fuzzy logic table for direction recommends

dir	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
d_1	-L	M	S	S	S	S	S	M
d_2	M	-L	M	S	S	S	S	S

d_3	S	M	-L	M	S	S	S	S
d_4	S	S	M	-L	M	S	S	S
d_5	S	S	S	M	-L	M	S	S
d_6	S	S	S	S	M	-L	M	S
d_7	S	S	S	S	S	M	-L	M
d_8	M	S	S	S	S	S	M	-L

(a) Direction discursion

Where d_i = 1, 2, 3… is the direction recommended in fuzzy logic
dir means direction

Worse

-L	*Strongly not recommended*
-M	*Medium not recommended*
-S	*Slightly not recommended*
0	*None suggestion state*
S	*Slightly recommended*
M	*Medium recommended*
L	*Strongly recommended*

Better

(b) State Definition

The discursion step is:
For i=1 to 8
If the direction state is not –L, then
If this direction is the direction just passed, set the state into a worse state;
If this direction reduced the distance to the goal, set the state into a better state until L;
Endif
Endif
Endif
Endfor

Then by discursion function we can get the direction:

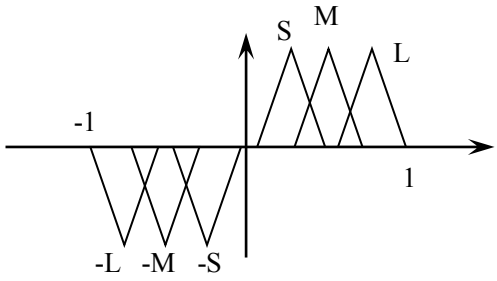


Figure 3.3.2 discursion function

The best recommendation can be obtained from the discursion function. Then AGV can decide the next direction. An example solution is shown below:

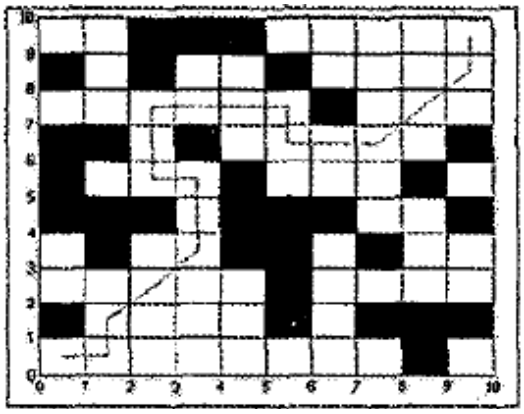


Figure 3.5.2 Fuzzy logic solution of AGV path [6]

The example in figure 3.5.2 gave the successful solution. However, if the heuristic functions are not suitable, the AGV might never reach the destination points.

3.6 Particle Swarm Optimization (PSO)

PSO is also an evolutionary algorithm. The initialization of the first generation can influence the performance and accuracy of the computation process [8]. In PSO, the solutions are represented by a set of particles. Then by optimizing the position of particles researchers can get better solutions.

The basic principle of PSO is:

1. *Generate a set of particles, including the random position and velocity ;*
2. *Find the fitness of each particle;*
3. *Compare the fitness of each particle with the Pbest-the best fitness that current particle had achieved. If the fitness of current particle is better, set it as the new Pbest;*
4. *Compare the Pbest of each particle with Gbest-global best fitness, if current Pbest is better, set it as new Gbest;*
5. *Change the velocity and position of particles by the following function[7]:*

$$v_{id} = w \cdot v_{id} + c_1 \cdot rand(m, n) \cdot (p_{id} - x_{id}) + c_2 \cdot rand(m, n) \cdot (p_{gd} - x_{id})$$

$$x_{id} = x_{id} + v_{id}$$

where the w, c_1,c_2 are constants and v_{id} is the velocity of the swarm, p_{id} and p_{gd} are the limitation of velocity

6. *If Gbest does not reach the end conditions, go to step 2.*

The process is:

Step1: develop the first generation of particles (figure 2.1.7.1):

The obstacle is marked as the circles and triangles, etc. The yellow dots are the tracks of particles;

Step2: to optimize the generation (solution) by GA:

The crossover function is:

$$p_i=(p_{i1}, p_{i2}, \cdots, p_{ik}, p_{i(k+1)}, \cdots, p_{in}) \quad \Rightarrow \quad p_i=(p_{i1}, p_{i2}, \cdots, p_{jk}, p_{i(k+1)}, \cdots, p_{in})$$
$$p_j=(p_{j1}, p_{j2}, \cdots, p_{jk}, p_{j(k+1)}, \cdots, p_{jn}) \qquad p_j=(p_{j1}, p_{j2}, \cdots, p_{ik}, p_{j(k+1)}, \cdots, p_{jn}) \qquad (8)$$

where p_i and p_j is the set of dots.

The mutation function is given by:

$$p_i' = p_i + kq_i, \quad \text{where } k \in [0,1] \qquad (9)$$

In basic (standard) PSO algorithm, the particles are learning from known global best particles. This process may cause the algorithm running into local minimum. Therefore, Kennedy,etc[9] gave the fully informed particle swarm(FIPS) to force the particle learn from local best particles. J.Liang, etc. developed a new learning strategy called comprehensive Learning Particle Swarm Optimization (CLPSO). CLPSO can search both in multi-dimension space and select different learning objects in different dimensions. The velocity refreshing function is:

$$v_{ij} = w*v_{ij}(t\text{-}1) + \varphi*r*(p_{f_i(j),j} - x_{ij}(t-1)) \quad (10)$$

where φ *is the acceleration vector, r is a random number in (0,1),* $f_i(j)$ *means the object to learning in j dimension for particle i.*

The method to determine f is:

generate a random number in (0,1);

if this number > learning rate Pc_i *, then the object is in its historical best position;*

else randomly select 2 individuals from the particles, find the best historical position of the better one, set it as the object.

After a period (refreshing gap), redo this process

Figure 3.6.1 showed the initialization of 1st generation of PSO:

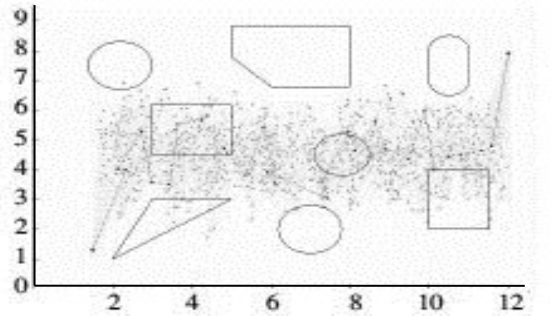
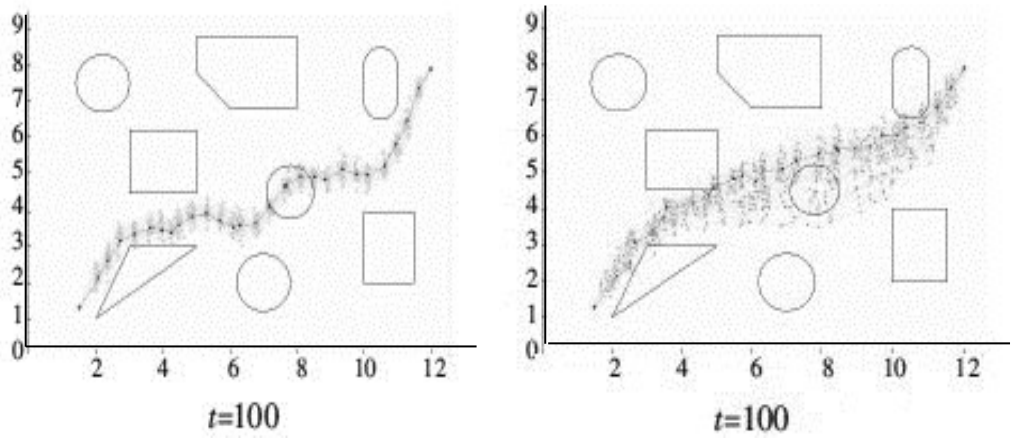


Figure 3.6.1 the 1st generation initialization of PSO [31]

Figure 3.6.1 showed that the more complex facility is the more particles is required to find a feasible path. Because the computation speed is positively relative to the number of particles N_p, it is important to balance N_p and algorithm efficiency.

Here is an example to compare the ability of standard PSO and improved PSO (IPSO):



(a) A failure of PSO (b) Success of IPSO in the same condition

Figure 3.6.2 comparing the results between PSO and IPSO

From the solution above we know that the solution from IPSO is able to keep the variety at the generation of 100 because the discrete distribution of the particles is able to give the effective individuals for further optimization while the solution of PSO is unacceptable.

3.7 Neural Network approach

Hao,etc.[28] developed a neural network model to solve free-ranging AGV routing problem by the elicitation of neural network in large scale TSP problems. The model can perform well in 10~20 job handling although it cannot solve complex working environments. Soylu et al.[29] developed a self-organizing neural network model for single AGV routing. Singh,etc [30] developed a dynamic model to solve the conflict-free route for AGV system. Most of the Neural Network algorithms in AGV routing problem combined the task handling and route searching problem together to minimize the total travel distance or to minimize the total time cost of the tasks.

A typical solution for the problem in the end of the paragraph above is:

(a) In a moment, find all the AGVs $A_i \in \mathbf{A}$, *where* ***A*** *is the set of AGVs, i = 1, 2, 3…*;

(b) Find all the idle AGVs $A_j \in \mathbf{A}$, *j = 1, 2, 3…*;

(c) Find all the un-assigned tasks $T_{u(k)}$, *k=1, 2, 3…*;

(d) Assign T_{ui} to suitable A_j;

Suppose in a moment t_o, we have the following task set and AGV set:

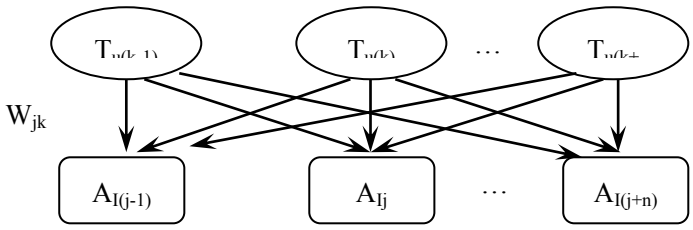


Figure 3.7.1 a typical structure in AGV task handling by neural network

Obviously, if the number of un-assigned tasks and the idle AGVs are not equal, there must be some of T_u or A_j left to be assigned for the next time (cycle). Therefore, this kind of solutions has to be rapid enough to meet real-time control (assigning). Because W_{jk} is the connect matrix between T_u and A_j, it is easy to transfer the network above into a neural network:

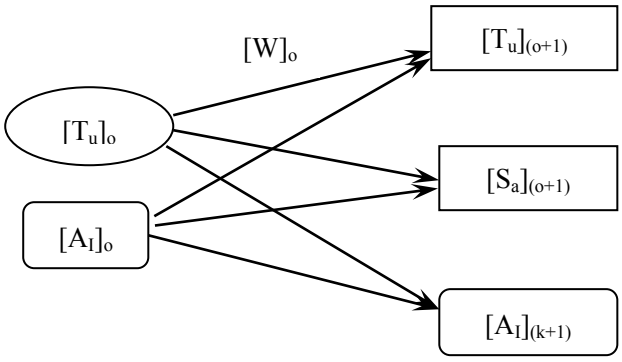


Figure 3.7.2 A typical Neural Network structure in the balancing of idle AGVs and un-assigned tasks

where

$[T_u]_o$ *the set of un-assigned task in the moment k*

$[W]_o$ *connect matrix*

$[A_I]_o$ *the set of idle AGVs*

$[S_a]_{(o+1)}$ *the set of successfully assigned tasks*

In the moment t_o, the importance X_{tk} of task T_{uk} may be different. Therefore, the fitness of the neural network can be defined as:

$$max([S_a]_{(o+1)}) \quad (11)$$

Then by heuristic functions or methods (such as Genetic algorithms), the optimized solution can be found.

4. Discussion and recommendation

Here we gave the performance comparison table of the algorithms listed in the paragraphs above:

Table 4.1 Performance comparison table of the algorithms listed in the paper

Algorithm	Dijkstra/ Floyd	Bezier Curve	Fuzzy Logic	APF	GA and GA relative	PSO	Neural Network
Design complexity	Low	Low	Low	Low	Medium	Medium	Medium
Algorithm Complexity	Low	Low	Low	Low	Medium	Medium	Medium
Map pre-treatment	No	No	No	No	No	No	No
Speed in Small maps	High	High	High	High	Low	Low	High
Speed in large maps	Low	Low	High	Low	Medium	Medium	High*
Programming complexity	Low	Medium	Low	Low	High	High	Low
Accuracy	100%	Medium*	Low	Low	High<100%	High<100%	High
Adaptability	Wire-guided	Free-route	Universal	Free-route	Universal	Universal	Universal

** depends on the map*

From tests of the algorithms, the Bezier Curve solution is the most unsteady algorithm because it is highly relative to the complexity of maps, particle density, curve dimension, numbers of AGVs, etc. It is not recommended for complex maps or multi-AGV systems, but it can be used as a sub-control of AGV positions (that is, if we solved the route via other efficient algorithms, we can use Bezier Curve as the accurate motion control method to guide the AGV).

Fuzzy logic is one of the easiest algorithms in programming, but the accuracy is highly dependent on the algorithm structure and parameters. Artificial Potential Field algorithm showed the adaptability in complex maps and multi-AGV systems. If suitable heuristic methods are used, it is able to jump out from the dead-locks. GA and PSO performed a little unsteadily in the test especially when the facilities are complex. The reason of this phenomenon is that the algorithms are not as accurate as Dijkstra algorithm. Meanwhile due to the different heuristic methods and initialization of the algorithms the results might be different.

5. Conclusion

Hybrid methods are recommended in research and application. Meanwhile, Neural Network is an efficient assistant method in routing.

Acknowledgement

Thank my supervisors for their patient and carefully help in my paper and research. I would extend my sincere thanks other professors and teachers for their help in my research.

References

[1] Edsger W. Dijksra: A note on two problems in connexion with graphs, Numer.Math., 1(1959). Pp 269-271

[2] LIAO Hua-li, ZHOU Xiang, DONG Feng, WANG Ting-qi**,** "AGV navigation alogorithm based on fuzzy control", Journal of Harbin institute of Technology, 2005-07

[3] Leng-Feng Lee, etc. "A standardized Testing-Ground for Artificial Potential-Field based Motion Planning for Robot Collectives", Performance Metrics for Intelligent Systems 2006

[4] Tong Heng Lee, etc. "Application of evolutionary artificial potential field in robot soccer system" IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th

[5] Mark A.C. Gill and Albert Y. Zomaya. "A parallel collision-avoidance algorithm for robot manipulators", Concurrency, IEEE, Jan-Mar, 1998

[6] YING ZHANG, CHENG-DONG WU, MENG-XIN LI. "Rough set and Genetic algorithms in path planning of robot" , International Journal of Automation and Computing, Volume 3, Number 1, 29-34

[7] Y.SHI and R.Eberhart, "A modified particle swam optimizer" [A] IEEE world Congr. Comput. Intell. [C] Jun2004 vol. 2 PP. 1390-1395

[8] Yuhuishi, etc. "Particle swarm optimization: developments, applications and resources" Proceedings of the 2001 congress on evolutionary, 2001

[9] J. Kennedy and R. Mendes. Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms [A]. in Proc. IEEE Int. Workshop Soft Computing in Industrial Applications[C], Jun 2003, pp.45-50.

[10] Chen Runwei, Gen Missuo, Yasuhiro Tsujimura. A Tutorial Survey of Job-shop Scheduling problems Using Genetic Algorithms -I[J]. Computers & Industry Engineering, 1996, 130(4):983-997

[11] Faraji m. ,etc. Forming cells to eliminate vehicle interference and system locking in an AGVS, International Journal of Production Research, v 32, Sep 1994, pp 2219-2241.

[12] Floyd algorithm Edsger –Wikipedia, the free encyclopedia

[13] Y. Tipsuwan adn M.Y.Chow, Gain Scheduling Middleware for Networked Mobile Robot Control, Proceeding of the 2004 American Control Conference, pp.4313-4318,Jun-July 2004.

[14] Chan F T S, Wong T C, Chan P L Y, Equal Size Lot Streaming to Job-shop Scheduling Problem Using genetic algorithms[C]//Proceeding of the IEEE International Symposium on Intelligent Control. Tai-pei, Taiwan, 2004:472-477;

[15] Guoyong S.A Genetic Algorithm Applied to a Class Job-shop Scheduling Problems[J]. Computers Industry Engineering, 1996, 30(4):983-997;

[16] Pawlak Z.Rough Sets. Theoretical Aspects of Reasoning about Data. Nowowiejska 15-19, 1990;

[17] Fan L H, Cui G C, the research in workshop scheduling by Genetic Algorithms[J], Journal Of Changchun University，2005（3）：11-15；

[18] Chen Runwei, Gen Missuo, Yasuhiro Tsujimura. A Tutorial Survey of Job-shop Scheduling problems Using Genetic Algorithms -I[J]. Computers & Industry Engineering, 1996, 130(4):983-997

[19] H. Andrew, S.Lai, H.Nelson, C.Yung. "Lane detection by orientation and length discrimination" IEEE transaction on Systems, Man and Cybernetics, pp 539-548,2000.

[20] Shirazi, B., Yih, S., 1988 Critical analysis of applying Hopfield neural net model to optimization problems. IEEE International Conference on Systems, Man and Cybernetics. Pp. 210-215.

[21] Sinriech, D., and Tanchoco, J.M.A., 1991, "Intersection Graph Method for AGV Flow Path Design" International Journal of Production Research 29(9) pp 1725-1732

[22] Maxwell, W.L. and Markstadt, l.A. "Design of Automate Guided Vehicle Systems- , III, Transactions, 14, 114-12 " i, 1982 .

[23] Egbelu, P.J., -Pull versus Push Strategy for Automated Guided Vehicle Load-Movement in a Batch Manufacturing System", Journal of Manufacturing Systems, 6, 209-220, 1986 .

[24] Gaskins, R. J., Tanchoco, J. M. A., "Flow Path Design for Automated Guided Vehicle Systems", International Journal Production Research, Vol. 25, No. 5. pp 667-676, 1987.

[25] Broadbent A.J., Besant C. B., Premi S. K., and Walker S. P., Free-ranging AGV Systems: Promises, Problems and Pathways. *Proc. of 2nd Int. Conf. on Automated Material Handling,* (IFS Publ. Ltd., UK), 1985, pp. 211-237.

[26] K. A. Dowsland and A. M. Greaves. Collision avoidance in bi-directional agv systems. J. Opl. Res. Soc., 45:817{826, 1994

[27] Toshio Emerging Technologies and Factory Automation, 1994. ETFA '94., IEEE Symposium Issue Date: 6-10 Nov 1994 PP. 472 - 477

[28] "A neural network model for the free-ranging AGV route-planning problem", Journal of Intelligent Manufacturing, Volume 7, Number 3, 217-227, DOI: 10.1007/BF00118081

[29] Mustafa Soylu, Nur E. Özdemirel , Sinan Kayaligil, "A self-organizing neural network approach for the single AGV routing problem**",** European Journal of Operational Research, Volume 121, Issue 1, 15 February 2000, Pages 124-137

[30] Dayal R. Parhi and Muskesh Kumar Singh, "Navigational strategies of mobile robots: a review", International Journal of Automation and Control, Issue Volume 3, Number 2-3/2009, pp. 114-134

[31] CAO You-hui, WANG Liang-xi, "Global path planning for Automated Guided Vehicles based on improved Particle Swarm Optimization" Computer Engineering and Application, 2009, 45(27):224-227