Genetic Algorithm for Delivery Problem

KwangEui Lee[†] **JiHong Kim**[††]

[†] Department of Multimedia Engineering, Dongeui University, Busan, Korea
[††] Department of Visual Information Engineering, Dongeui University, Busan, Korea

Summary
In this paper we introduce the delivery problem and propose a genetic algorithm solving the delivery problem. In the delivery problem, many heterogeneous robot agents collaborate to deliver an object to its destination aim to minimize the delivery time In this problem, each agent has different velocity So, the hand in sequence and places are critical to optimize the delivery time.
We analyze properties of the delivery problem, and propose an exact algorithm for 1-dimensional space delivery problem. And then we describe a genetic algorithm for 2-diemnsioal case. Finally, we compare the results of these two algorithms to show the effectiveness of given genetic algorithm.

Key words:
Genetic algorithm, Robot collaboration, Artificial agent, Optimization, Delivery problem

1. Introduction

The delivery problem is that of minimizing the delivery time in m-dimensional space. There are n robot agents with various velocity, an object and destination. In this problem, n robot agents collaborate to deliver the object to the destination as fast as possible. For the sake of clarity, we will show one of the simplest samples of 1-dimensional delivery problem with 2 agents. Figure 1 show three possible solutions for specific parameters. In this figure, o, a, b, d stand for object, agent a, agent b, destination, respectively. We assume that agent b is faster than agent a.

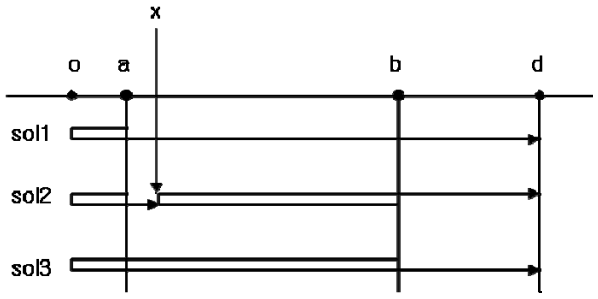

Fig. 1 3 possible solutions for simple delivery problem

We can easily figure out that the sol2 is the best solution and the position x will be the solution of the problem. In this paper, we will consider only the time to deliver, the solution will be (distance(a,o) + distance(o,x)) / speed(a) + distance(x,d) / speed(b) or (distance(b,x) + distance(x,d)) / speed(b).
Delivery problem can be considered as a kind of path planning problem [1]. There are many results on path planning, however most research focuses on single robot [2]. Some recent researches consider the multi agent case [3][4][5] but, within our knowledge, there is no previous result in the literature faces our definition
In this paper, we investigate the properties of the delivery problem and propose two algorithms for the problem. One is an optimal algorithm for 1-dimensional space delivery problem and the other is a genetic algorithm for 2-dimensional case. The rest of the paper is organized as follows. We introduce the delivery problem and its properties in section 2. In section 3 and 4, we present a polynomial time algorithm for 1-dimensioal delivery problem and a genetic algorithm for 2-dimensional delivery problem respectively. The analysis and experimental results are shown in section 5. Finally, section 6 draws the conclusion.

2. Delivery Problem

In this section, we will define the multi-agent single object delivery problem and show some mathematical aspect of the problem.

2.1 Problem Definition

The delivery problem is that of minimizing the delivery time in m-dimensional space. The problem is composed of n heterogeneous mobile robots and an initial position of and object and a final position where the object should be delivered. We assume that each robot has different constant velocity and no additional time is needed to pickup, hand in and release the object. In 2-dimnesional case, the problem can be formulated like follows:
For given

$rp[i], i = 1..n$: initial position of robot[i],

$rv[i], i = 1..n$: constant velocity of robot[i],

ip : initial position of the only object,

fp : final position that the object to be delivered

Manuscript received February 5, 2009
Manuscript revised February 20, 2009

Decide the values of

$s[i], i = 1..k, k <= n$: a permutation of a subset of robots,

$g[i], i = 1..k$: position on the space.

To minimize the value of the following cost function:

```
double cost(k) {
  return recur(k) + ED(rp[s[i]], fp) / rv[s[i]];
}
double recur(k) {
  if (k==1)
    return (ED(rp[s[1]], ip) + ED(ip, g[2])) / rv[s[1]];
  double costA= recur(k −1);
  costA+= ED(g[k −1], g[k]) / rv[s[k −1]];
  double costB= ED(rp[s[k]], g[k]) / rv[s[k]];
  return max{costA, costB}
}
```

Where,

$ED(x, y)$: Euclidean distance function,

$r[i], i = 1..n$: robot[i].

4.2 Mathematical Aspect of Delivery Problem

To show the difficulty of the problem, we will investigate a simple 2-dimensional delivery problem with two robot agents. rp[0], rv[0], rp[1], rv[1], ip, and fp are given and without loss of generality, we assume that r[1] is b times faster than r[0]. For the sake of easy explanation, we impose another constraint that rp[0] = ip (i.e., r[0] holds the object at the beginning).

Since r[1] and r[0] has velocity ratio b, the circle of Apollonius [6] defined by the set of points P that have a ratio of distances b to two points r[1].p and r[0].p is important. Figure 2 shows the Apollonian circle defined by rp[0], rp[1] and the ratio b.

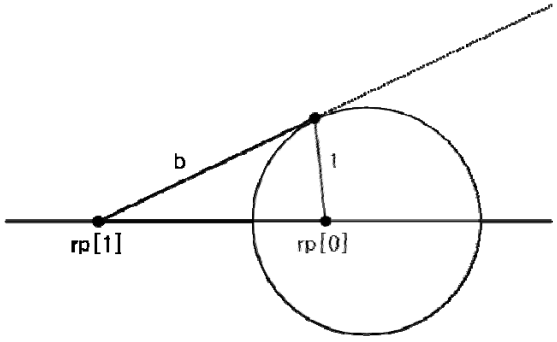

Fig. 2. Apollonian circle defined by two points and ratio b.

If fp is inside of the Apollonian circle, the solution will be ED(rp[0],fp)/rv[0]. Otherwise, the solution will be ED(rp[0],x)/rv[0]+ED(x,fp)/rv[1] or equivalently ED(rp[1],x)/rv[1]+ED(x,fp)/rv[1]. Where, x is a point on the circle that minimize ED(rp[1],x)+ED(x,fp). But, finding the point x is another challenging problem.

3. Optimal Algorithm for 1-Dimensional Case

Fortunately, 1-dimensional delivery problem can be solved by an exact algorithm in polynomial time. We will describe our algorithm with out the proof of correctness. But, the proof of correctness is not difficult.

Our algorithm runs iteratively. In each round the object is passed to another robot whose velocity is higher than the velocity of current holder. During the execution of the algorithm, the robot that holds the object moves to fp, and all the other robots move to the object.

Because each robot has different velocity, the robot that holds the object meets another robot as time passed. If the new robot has higher velocity, the object is passed from current holder to the new robot. This process is repeated until the object is delivered to fp. The overall algorithm is as follows:

```
// algorithm: 1-dimensional delivery algorithm

// first stage: every robot move to ip.
timeA = 0;
mini = min {i:ED(rp[i],ip)/rv[i]};
timeA += ED(rp[mini],ip)/rv[mini]);
For each r[i] { // update the robot's position.
   if (rp[i]>ip) rp[i] -= rv[i]*timeA;
   else rp[i] += rv[i]*timeA;
}
// now one of the robot holds the object
Do
   mini2 = min {i:ED(rp[i],rp[mini])/(rv[i]+rv[mini]),
                && rv[i]>rv[mini]}
   timeB = ED(rp[mini],rp[mini2])/(rv[mini]+rv[mini2]);
   If (timeB>ED(rp[mini],fp)/rv[mini]) {
      timeA += ED(rp[mini],fp)/rv[mini]);
      Break;
   } else {
      For each r[i] {
         if (rp[i]>rp[mini]) rp[i] -= rv[i]*timeB;
         else rp[i] += rv[i]*timeB;
      }
      timeA += timeB; mini = mini2;
   }
Until forever;
Output timeA;
```

Because of the object is passed to a robot with higher speed, the do loop repeated n times maximum. So, the time complexity of the algorithm will be $O(n^2)$ when the number of agents is n.

4. Genetic Algorithm for 2-Dimensioal Case

This section presents a genetic algorithm [7] for 2-dimensional delivery problem. Without loss of generality, we assume that the robots are sorted by velocity in increasing order. i.e., $rv[i] <= rv[i+1], i = 0..n-2$

Basically, robot r[j] moves specific place and meet robot r[i] (i<j) and take the object from r[i] and then moves some other specific place and meet robot r[k] (j<k) and hand in the object to robot r[k]

41 Gene Structure and Cost Function

A gene is a representative of a solution of the problem. Here, we describe the structure of gene and the solution that the gene represents. For the n-robot delivery problem, the structure of gene is as follows:

$gene[i], i = 1..n$: place in the space that robot[i] take the object if robot[i] participates the delivery.

With this meaning of gene, we can find the order of robots that contribute to the delivery and the minimum time needed to delivery the object by using the Dijkstra's shortest path algorithm [8]. Before going further we will give some more explanations about the symbols s[i] and g[i] in section 2

$s[i], i = 1..k, k <= n,$: the order of robots that contribute to the delivery for the given (optimal) gene. Only a subset of robots will be listed here.

$g[i] = gene[s[i]], i = 1..k$

The cost evaluation algorithm for a gene and robot sequence is as follows:

```
// algorithm: evaluation function

timeA = 0;
timeA += ED(rp[s[1]],ip) /rv[s[1]];
timeA += ED(ip,g [2]) /rv[s[1]];
For (int i=2; i<k; i++) {
   timeB = timeA + ED(g[i],g [i+1]) /rv[s[i]];
   timeC = ED(rp[s[i]],g[i]) /rv[s[i]];
   timeA = max {timeB, timeC}
}
timeA += ED(g[k],fp) /rv[s[k]];
Output timeA;
```

The algorithm computes the total time only, but someone can easily modify the algorithm to generate the delivery sequence.

4.2 Genetic Algorithm

Our algorithm stops when there is no improvement greater more than 0.0001 unit time during last 100 generations and we use only mutation that changes randomly selected g[i] to generate next generation. The overall algorithm is as follows.

```
// algorithm: 2-dimensional genetic algorithm

InitPopulation()
gen = 0;
While (gen<1000) {
   For all gene[i] call CostEvaluation(gene[i]);
   Call SortGeneByCost();
   Call NextGeneration();
   best[gen++] = CostEvaluation(gene[0]);
   if (best[gen-100]-best[gen]<0.0001) break;
}
Output best[gen-1];
```

In the function InitPopulation(), we generate initial gene randomly. Function CostEvaluation() use the modified Dijkstra's shortest path algorithm to evaluate the gene.

5. Experimental Results

This section presents a set of experiments designed to study the characteristics and performance of the proposed algorithm. We analyze our algorithm from two aspects: convergence speed and optimality

5.1 Experiments on 1D test suites

Table 1 shows optimal delivery time generated by 1-dimensional exact algorithm, delivery time generated by 2-dimensional genetic algorithm, number of rounds run by the genetic algorithm and the ratio of result of these two algorithms.

Test Suite#	1D algo. (time)	2D algo. (time)	# of rounds	1D/2D Ratio(%)
1	55.1308	55.1312	106	99.999
2	71.5989	71.7625	107	99.772
3	70.0768	70.0769	104	99.999
4	58.9000	58.9000	100	100.000
5	62.9090	62.9092	108	99.999
6	53.3018	53.3043	108	99.995
7	62.3621	62.8701	107	99.192
8	65.4839	65.4842	103	99.999
9	69.4422	69.4436	104	99.998
10	61.0709	62.7095	190	97.387
Avg.	63.0276	63.2592	113.7	99.634

Table 1 Comparison of 1D and 2D algorithm

As shown in the figure, almost all test run reaches the optimal solution within 0.01% error in small number of rounds Our stop condition of the genetic algorithm is

best[gen-100]-best[gen]<0.0001.

And this means, there is no meaningful improvement after 8th generation in 9 out of 10 test suites.

5.1 Experiments on 2D test suites

We generate a random 2D test suite with 10 agents and 10 random genes as initial population. The evaluated delivery time of 10 initial genes are distributed from 92.3248 to 95.0121 and after 160 generations we got a solution with 90.3861 unit time. Figure 3 shows the time gains in each generation.

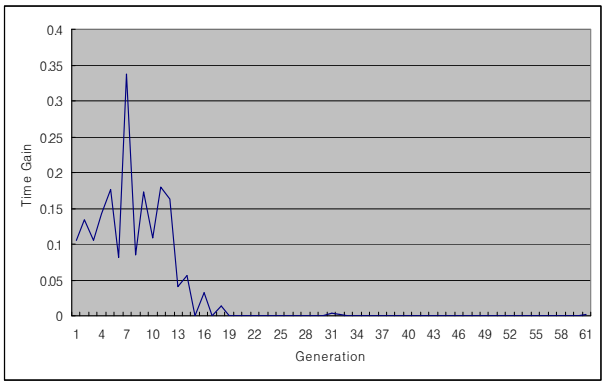

Fig. 3. time gains in each generation.

We also test 10 random 2D test suites and figure out that the average round is 173. As same as the 1D case the effective number of rounds is 73. It is still small but almost 10 times grater than 1D case. Table 2 shows the result

Test Suite#	2D algo. (time)	# of rounds
1	77.3686	171
2	78.2716	190
3	103.5188	103
4	86.5866	123
5	82.0702	100
6	903864	298
7	78.2225	101
8	77.0049	331
9	68.9628	163
10	67.8754	146
Avg.	81.0268	172.6

Table 2. The result of genetic algorithm on 2D test suite

6. Conclusion

In this paper, we introduce new type of delivery problem and propose two algorithms solve the problem. One is $O(n^2)$ time exact algorithm for 1-dimensional delivery problem and the other is genetic algorithm for 2-Dimensioal delivery problem. By comparing the results of these two algorithms, we show the effectiveness of the genetic algorithm.

Acknowledgments

This work was supported by Dong-eui University Grant (2006AA186)

References

[1] http://en.wikipedia.org/wiki/Motion_planning
[2] S. M. Lavalle, Planning Algorithms. Cambridge University Press, 2006
[3] D.K. Liu, D. Wang, G. Dissanayake, "A force field method based multi-robot collaboration," Proc. IEEE Int. Conf. on Robotics, Automation and Mechatronics, Bangkok, Thailand, April 2006, 662–667.
[4] Y. Guo, L.E. Parker, A distributed and optimal motion planning approach for multiple mobile robots, in: Proc. IEEE Int. Conf. on Robotics Automation, 2002, pp. 2612-2619.
[5] K. Azarm and G. Schmidt, "Conflict-Free Motion of Multiple Mobile Robots Based on Decentralized Motion Planning and Negotiation," IEEE Int. Conf. on Robotics and Automation, 1997, pp 3526-3533
[6] http://en.wikipedia.org/wiki/Circles_of_Apollonius
[7] Melanie Mitchell, An Introduction to Genetic Algorithms, MIT Press, 1996
[8] R. Neapolitan and K. Naimipour, Foundations of Algorithms Using C++ Pseudocode, 3rd Ed., Addison Wesley, 2003

KwangEui Lee received his B.S., M.S. and the Ph.D. degrees from Sogang University, Seoul, Korea in 1990, 1992, and 1997, respectively. From 1997 to 2001, he joined ETRI as a senior research member. Since 2001, He has been an associate professor of Dongeui University. His research interests include computation theory, artificial life, context awareness and their applications

JiHong Kim received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Korea, in 1986 and 1988, respectively, and the Ph.D. degree in electronic and electrical engineering from POSTECH, Korea, in 1996. He worked at ETRI from 1988-1997 and at Pusan University of Foreign Studies form 1997-2002. In 2002, he joined the Dongeui University, where he is an associate professor. His current research interests are in the area of digital image processing, computer graphics, and computer vision.