Near Optimal Algorithm for Delivery Problem

KwangEui Lee

Department of Multimedia Engineering, Dongeui University, Busan, Korea

Summary
The delivery problem is that of minimizing the object delivery time from one place to another using n various speed robots. In this paper we propose two algorithms for the delivery problem. The first one is an optimal algorithm with some restriction in handover places. In this algorithm, we assume that the handover can be made at predefined spots called station. The second algorithm is a near optimal algorithm for general case delivery problem which is based on the previous algorithm. The second algorithm does not generate an optimal solution but we can make the result better than what we expect.

Key words:
Robot collaboration, Optimization, Delivery problem, Dijkstra's shortest path algorithm.

1. Introduction

The delivery problem is that of minimizing the object delivery time from one place (we call this place as source) to another (destination) in m-dimensional space [1]. There are n robot agents with various velocities and various initial positions. Initially the object is placed at the source position. 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 the simple example of the problem. Figure 1 shows an example configuration of the 2-dimensional delivery problem.

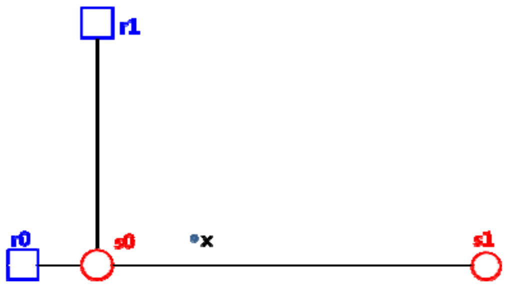

Fig. 1 a sample configuration of delivery problem

We assume that robot agent r1 is faster than r0 and the object is placed at s0 initially and should be delivered to s1. To minimize the delivery time, r0 pickup the object at s0 and carries to some place x, while r1 moves to x. At place x, r0 handovers the object to r1 and r1 carries the object to s1. If r0 placed at s0 initially, the x is on the Apollonian circle [2] defined by the position of r0 and r1 and the speed ratio of two robots. The exact position of x on Apollonian circle is the point that minimizing the path length composed by r1, x and s1. Figure 2 shows an Apollonian circle defined by two points and ratio b [1].

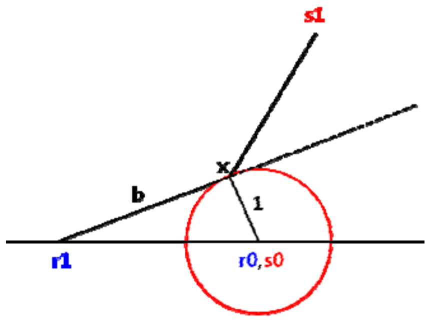

Fig. 2 Apollonian circle defined by two points and ratio b

Delivery problem can be considered as a path planning problem [3]. There are many results on the path planning problem with single robot results [4] and multi-agent results [5][6][7]. However, most researches focus on optimization of robot resources and average waiting time. This kind of path planning problem defined by Lee et al, and they suggested two algorithms for this problem. One is an optimal algorithm for 1-dimension delivery problem and the other is a genetic algorithm for 2-dimension delivery problem [1].
In this paper, we investigate the properties of the delivery problem and propose two algorithms for m-dimension delivery problem. The first algorithm produces optimal solution but, in this algorithm, handover can be made only at predefined spots. The second algorithm generates sub-optimal result but there is no restriction and we can make the result better than what we expect. The rest of the paper is organized as follows. We will give a formalization of the delivery problem in section 2. In section 3 present the optimal algorithm and in section 4 present the near optimal algorithm. Finally, section 5 draws the conclusion.

Manuscript received December 5, 2011
Manuscript revised December 20, 2011

2. Delivery Problem

The delivery problem concerns the delivery an object in m-dimensional space. There are n mobile robots each of them has different constant velocity. The robots collaborate to deliver an object from one place to another as fast as possible. Because of each robot has various initial position and velocity and we try to minimize the delivery time, handovers are occurred while delivering. The delivery problem is finding a sequence of robots that participate the delivery and the exact point where handovers are occurred.

For the sake of simplicity, we assume that no additional time is needed to pickup, to handover and to release the object. The problem can be formulated as follows. Basically, our formulation is same to that of lee et al [1], but we give slight modifications in notations.

For given:

$r(i), 0 \le i < n$: Robot agent, indexed i,

$r(i).p, 0 \le i < n$: Initial position of $r(i)$,

$r(i).s, 0 \le i < n$: Constant velocity of $r(i)$, each $r(i)$ moves at speed $r(i).s$ or stay still. Without loss of generality, we assume that $r(i).s < r(i+1).s$,

sp : Initial position of the only object,

dp : Destination position that the object should be delivered.

Decide the values of:

$rr[i], 0 \le i < k, k <= n$: A sequence of robot that participate the delivery.

$p[i], 0 \le i \le k$: A sequence of handover positions on the space. Robot $rr[i]$ carries the object from $p[i]$ to $p[i+1]$. So, $p[0] = sp$ and $p[k] = dp$.

To minimize the value of $c[k]$ where:

$$c[i] = \begin{cases} 0 & i = 0 \\ \max\{c[i-1], D(p[i], rr[i].p)/rr[i].s\} & \\ + D(p[i], p[i+1])/rr[i] & 1 < i <= k \end{cases}$$

$D(x, y)$ is the Euclidean distance between x and y.

Obviously, $rr[i].s < rr[i+1].s, 0 \le i < k-1$.

We will denote the minimum time delivery sequence as follows:

$sp = p[0], rr[0], p[1], rr[1], p[2], ..., p[k-1], rr[k-1], p[k] = dp$

3. Optimal Algorithm for the Restricted Case

In the this section, we restrict the problem to the case in which each $p[i]$ is the one of m predefined positions $s(x), 0 \le x < m$, called stations. Our algorithm borrows the idea of Dijkstra's shortest path algorithm [8]. First, we define 2 sets of nodes: Set A is consists of the stations that the minimum time delivery path from sp has computed. Initially, the set A contains the station sp only. Remaining stations compose set B . Similar to the Dijkstra's algorithm, our algorithm runs iteratively. In each round, we will compute the expected delivery time for all the station $s(x)$ of set B, and put the station with the minimum expected delivery time into the set A. Before going further, we will define some more notations:

$s(x), 0 \le x < m$: Station, indexed x,

$s(x).p, 0 \le x < m$: Position of $s(x)$

$s(x).b, 0 \le x < m$: The last station before $s(x)$ in the minimum time delivery path. Remember that for each of stations of set A, the minimum time delivery path from sp to the station $s(x)$ has computed already,

$s(x).r, 0 \le i < m$: Robot agent that delivers object from $s(x).b$ to $s(x)$,

$s(x).t, 0 \le i < m$: Time consumed to deliver the object from sp to $s(x)$ along the minimum time delivery path.

To explain our algorithm, we will give a sample. Figure 1 shows a configuration of the sample.

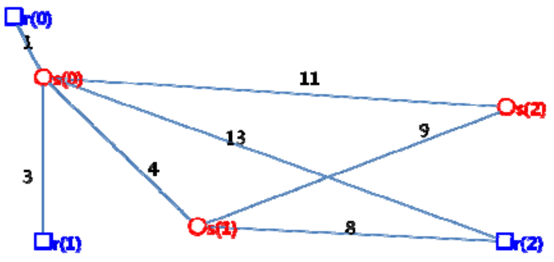

Fig. 3 sample of delivery problem

Table 1 and table 2 show the speeds of each robot agents and distances among robots and stations respectively.

robot	$r(0)$	$r(1)$	$r(2)$
speed	1 unit/sec	2 unit/sec	3 unit/sec

Table 1 Speeds of robots in Figure1

	$r(0)$	$r(1)$	$r(2)$	$s(0)$	$s(1)$	$s(2)$
$r(0)$	0	-	-	1	5	12
$r(1)$	-	0	-	3	3	12
$r(2)$	-	-	0	13	8	2
$s(0)$	1	3	13	0	4	11
$s(1)$	5	3	8	4	0	9
$s(2)$	12	12	2	11	9	0

Table 2 Unit distances of robots and stations in Figure 1

Without loss of generality, we assume that the object is at the station $s(0)$ and should be delivered to the station $s(2)$. Now, we will show first 3 rounds of the algorithm:

Round 0: In this round, we figure out the robot that comes first to the station. It will be the first deliverer. Each robot's arrival time to station s(0) is shown in table 3.

robot	$r(0)$	$r(1)$	$r(2)$
arrival time	1	1.5	$3\frac{1}{3}$

Table 3 Speeds of robots in Figure1

So, $s(0).r$ will be $r(0)$ and $s(0).t$ will be 1. $s(0).b$ will be $s(0)$ by assumption.

Round 1: Compute expected minimum delivery time for stations in set B, i.e. $s(1)$ and $s(2)$. Table 4 shows all possible delivery time.

object position		time to current	delivery robot	time to next
current	next			
$s(0)$	$s(1)$	1[(a)]	$r(0)$	1+4[(b)]
$s(0)$	$s(1)$	1	$r(1)$	1.5+2
$s(0)$	$s(1)$	1	$r(2)$	$4\frac{1}{3}+1\frac{1}{3}$
$s(0)$	$s(2)$	1	$r(0)$	1+11
$s(0)$	$s(2)$	1	$r(1)$	1.5+5.5
$s(0)$	$s(2)$	1	$r(2)$	$4\frac{1}{3}+4\frac{2}{3}$

(a) The time $r(0)$ is holding the object, this makes algorithm simple.

(b) Robot goes to station $s(0)$ first and then pick up the object and then moves to $s(1)$.

Table 4 Expected minimum delivery time to $s(1)$ and $s(2)$

The minimum time (3.5 sec) appears at row 2. That means robot $r(1)$ should deliver the object from $s(0)$ to $s(1)$.

Round 2: Compute the expected minimum delivery time for the station $s(2)$, the only station in Set B. Table 5 shows all possible delivery time.

object position		time to current	delivery robot	time to next
current	next			
$s(0)$	$s(2)$	1	$r(0)$	1+11
$s(0)$	$s(2)$	1	$r(1)$	1.5+5.5
$s(0)$	$s(2)$	1	$r(2)$	$4\frac{1}{3}+4\frac{2}{3}$
$s(1)$	$s(2)$	3.5	$r(0)$	-[(a)]
$s(1)$	$s(2)$	3.5	$r(1)$	3.5+4.5
$s(1)$	$s(2)$	3.5	$r(2)$	3.5+3[(b)]

(a) $r(0)$ is slower than $r(1)$. So, we don't have to compute this.

(b) $r(2)$ arrives at $s(1)$ at $2\frac{2}{3}$sec, but object is not there. So, $r(2)$ should wait until $r(1)$ arrives at $s(1)$ at time 3.5.

Table 5 Expected minimum delivery time to $s(2)$

The minimum time (6.5 sec) appears at row 6. That means, To deliver the object from $s(0)$ to $s(2)$ in minimum time, $r(1)$ delivers the object from $s(0)$ to $s(1)$ and handovers the object to $r(2)$ at station $s(1)$ and then $r(2)$ delivers the object from $s(1)$ to $s(2)$. Therefore, the minimum time delivery path will be $s(0)r(1)s(1)r(2)s(2)$.

Algorithm 1 shows the proposed algorithm. In the following description, we adopt the notational conventions of object oriented programming. That means, we use the notation $s(x).b.t$ to denote the time to deliver the object from sp to $s(x).b$.

```
Algorithm RestrictedDelivery {
  Set A = {ip};
  Set B = {all the stations except ip};
  for each Station s(x) in Set B {
    Call Function EstimatedTime(s(x));
    Let Station s(z) is the station has minimum s(x).et
      among stations s(x) in B.
    s(z).t = s(z).et;
    s(z).b = s(z).cb;
    s(z).r = s(z).cr;
    A = A + s(z);
    B = B – s(z);
  }
  Print dp;
  Call Function PrintDeliveryPath(dp);
}

Function EstimatedTime (Station s(x)) {
  for each Station s(y) in Set A {
  s(x).cr = s(y).r;
  s(x).cb = s(y);
  s(x).et = s(y).t + D(s(y).p, s(x).p)/s(y).r.s;
    for each Robot r(i) with r(i).s>s(y).r.s {
      Time t = D(s(y).p, r(i).p)/r(i).s;
      t += D(s(y).p, s(t).p)/r(i).s;
      if ((s(x).et > t) || (s(x).et==t &&s(x).cr.s<r(i).s) {
        s(x).cr = r(i);
        s(x).cb = s(y);
        s(x).et = t;
      }
    }
  }
}

Function PirntDeliveryPath (Station s(x)) {
  if (s(x).b=sp) return;
  Print s(x).p, s(x).r;
  PrintDeliveryPath(s(x).b);
}
```

Algorithm 1 Optimal algorithm for restricted case

Theorem 1. Algorithm RestrictedDelivery generates the minimum time delivery sequence.

Proof. The overall process of proof is similar to that of Dijkstra's algorithm. Here, we describe only the key part of the proof. Note that for every station s(x), if robot s(x).r delivers the object from s(x).b to s(x) in a minimum time along the minimum time delivery path, then the algorithm generates the minimum time delivery sequence. Now consider the time when the s(x) moves into set A. Let's assume that there is another delivery path to s(x) that is optimal then, this path only contains stations in set A. If not, it takes more time than the delivery path generated by the algorithm. Moreover, the delivery path generated by the algorithm is the minimum time delivery path among the delivery paths that constructed using station in set A. Therefore the algorithm generates the minimum time delivery sequence Q.E.D.
The time complexity of the algorithm is $O(nm^2)$ when the number of robots is n and the number of stations is m.

4. Near-optimal Algorithm for General Delivery Problem

This section presents a near optimal algorithm for delivery problem. The resulting delivery time is not optimal but we can control the bound of errors. In this section we will consider only 2-dimensional cases but the algorithm can be extended to m-dimension case easily. Proposed algorithm is as follows.

```
Algorithm LatticeDelivery {
   Draw a rectangle whose two corners are sp and dp;
   Draw a Lattice within the rectangle;
   Make a station on each lattice point;
   Call restrictedDelivery;
}
```

Algorithm 2 Near-optimal algorithm for general delivery problem

Figure 4 shows an example lattice for a delivery problem with 5 robot agents.

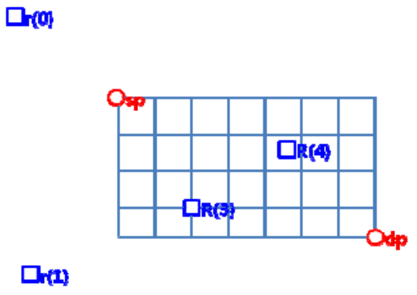

Fig. 4 Example lattice of the algorithm

The error of the algorithm is related to the size of the lattice and the number of handovers. Because of the maximum error occurs when the optimal handover point located at the center of the lattice, the maximum error in the length of path is bounded by $numberOfHandover \times 2 \times \sqrt{latticeSize}$. Almost every case, we can regard the number of handover is a constant. So, the error is directly proportional to the size of the lattice.

5. Conclusion

In this paper, we propose two algorithms for the delivery problem. The first one generates optimal result in $O(nm^2)$ time with restricted configuration. Here n is the number of robots is n and m is the number of stations. The second algorithm has controllable errors but, there is no restriction on the configuration. In the course of constructing the second algorithm, we use only lattice. In the further research, we will try various dividing method including triangulation.

References

[1] KwangEui Lee and JiHong Kim, "Genetic Algorithm for Delivery Problem," IJCSNS, V9, N2, February 2009, pp 248-251
[2] http://en.wikipedia.org/wiki/Circles_of_Apollonius
[3] http://en.wikipedia.org/wiki/Motion_planning
[4] S. M. Lavalle, Planning Algorithms. Cambridge University Press, 2006
[5] 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.
[6] 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.
[7] 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
[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 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 and their applications.