2005 한국경영과학회/대한산업공학회 춘계공동학술대회
2005년 5월 13일 ~ 14일, 충북대학교

AGV의 분산제어를 위한 에이전트 기반의 제어시스템

Agent-based control system for distributed control of AGVs

오승진, 정무영

포항공과대학교 산업경영공학과 / 제품생산기술연구소

Abstract

This paper deals with a new automated guided vehicle (AGV) control system for distributed control. Proposed AGV control system adapts the multi-agent technology. The system is composed of two types of controller: routing and order. The order controller is in charge of assignment of orders to AGVs. Through the bidding-based negotiation with routing controllers, the order controller assigns a new order to the proper AGV. The order controller announces order information to the routing controllers. Then the routing controllers generate a routing schedule for the order and make a bid according to the routing schedule. If the routing schedule conflicts with other AGV's one, the routing controller makes an alternative through negotiation with other routing controllers. The order controller finally evaluates bids and selects one. Each controller consists of a set of agents: negotiation agent, decision making agent and communication agent. We focus on the agent architecture and negotiation-based AGV scheduling algorithm. Proposed system is validated through an exemplary scenario.

Keywords: AGV, agent negotiation, distributed control, AGV scheduling

1. Introduction

Automated Guided Vehicle(AGV)의 제어는 생산시스템 관리에서 매우 중요한 이슈이며 AGV의 제어성과가 전체 시스템의 효율에 많은 영향을 준다(Rajeeva Lochana Moorthya, 2003). 이에 따라 생산시스템의 특성에 맞는 적절한 AGV 제어시스템이 요구되고 있다. 기존의 계층적인 생산시스템의 구조로는 제품의 다양화와 생산 환경의 급격한 변화에 대응하기가 불가능하기 때문에 분산적이고 수평적인 구조의 생산시스템에 관한 관심이 높아졌다(Davis *et al.*, 1993). 이러한 분산생산시스템에 맞는 분산제어구조를 가진 AGV 제어시스템의 필요성이 제기되고 있다.

AGV의 분산제어에서 고려되어야할 문제들 중 하나는 order의 할당 문제이다. 즉 생산시스템 내에 다수의 AGV가 존재할 경우, 어떤 AGV에게 order를 할당할 것인가를 결정하는 문제가 제기된다. 분산생산시스템에서의 order 할당 문제를 해결하기 위하여 시스템내의 resource간의 negotiation을 이용한 방법이 제안되었다(Paulo Sousa *et al.*, 1999). 그러나 AGV는 다른 시스템 resource와는 달리 운행 중 서로 간의 물리적 충돌 문제가 발생할 수 있어 충돌의 감지와 해결이 중요한 문제로 부각되었다(Michiko Watanabe, 2001). 이러한 충돌의 감지와 해결을 위해서 order가 주어졌다는 가정 하에 order를 수행하는 과정에서 발생할 수 있는 AGV간의 충돌을 감지하고 해결할 수 있는 분산제어 알고리즘이 제안되었다(Seungjin Oh *et al.*, 2004). 그러나 제안된 논문에서는 order의 할당 문제는 다루어지 지지 않았다.

본 연구에서는 AGV의 분산제어를 위하여, multi-agent기술을 이용한 order 할당 알고리즘을 제안한다. 본 방법론은 AGV를 제어하는 controller와 order를 제어하는 controller간의 협상을 이용하여, AGV간의 충돌을 회피하면서 order의 capacity와 due date를 지킬 수 있는 AGV에게 order를 할당한다. 협상과정에서의 bid 생성을 위하여 AGV의 충돌 감지와 해결에서 사용되었던 충돌 회피 이동경로 생성 알고리즘(Seungjin Oh *et al.*, 2004)을 변형하여 사용하였다.

본 논문의 2절에서는 AGV control system의 전체 구조와 구성요소에 대해서 소개하고, 3절에서는 AGV control system의 각 구성요소의 구조와 작동원리에 관하여 기술한다. 4절에서는 논문에서 제안한 방법론을 적용하여 order를 할당하는 과정을 예를 통해 보여준다.

2. Architecture of AGV control system

논문에서 제안하는 AGV 제어를 위한 system은 routing controller, DB, order controller로 구성되어 있다. Routing controller는 각 AGV에 하나씩 존재하며, AGV의 운행에 관한 기능을 수행한다. Routing controller는 경로생성기능, 대화기능, 협상기능 을 수행한다. 이들에 대한 자세한 설명은 3절에서 다루고 있다. DB는 현재 shop layout 정보, 운행 중인 AGV정보 및 order정보 등을 관리한다. Order controller는 Order정보의 관리와 order의 할당에 관한 역할을 수행한다. AGV control system의 간략한 구조는 <Figure 1>과 같다. <Figure 1>에서 화살표는 정보의 흐름을 나타낸다.

AGV control system내에서 일어나는 event는 크게 4가지로 구분한다. 1)작업완료 event, 2)order의 추가 event, 3)order의 변화 event, 4)periodic event 이다. 이러한 event에 의해서 AGV는 작업을 할당받고, 할당 받은 작업을 수행하는 과정에서 대화와 협상을 한다.

작업완료 event는 시스템내의 AGV가 자신에게 할당된 작업을 완료하였을 때 발생한다. 작업완료event가 발생하였을 때만 order의 추가 event가 일어난다.

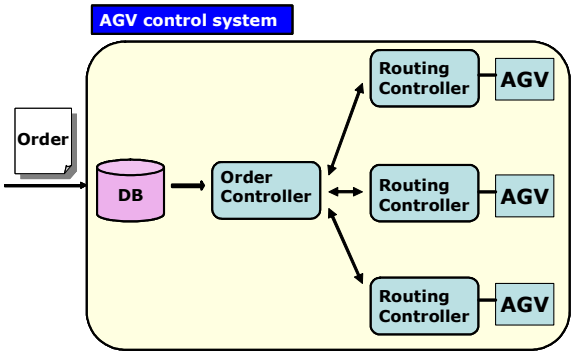

Figure 1. AGV control system architecture

Order의 추가 event는 작업완료 event가 발생한 상태에서 DB에 아직 할당되지 않은 새로운 order가 존재하면 발생한다. Order의 추가 event가 일어났을 때의 system 작동과정은 간략히 다음과 같다. DB에 할당되지 않은 order 정보가 존재하면 order controller가 DB에서 order 정보를 읽어 온다. 이 order의 capacity를 만족시키는 시스템 내에 있는 모든 AGV의 routing controller에게 새로운 order를 전달한다. 전달된 order를 바탕으로 routing controller는 새로운 order를 수행할 때의 routing path(initial routing information)를 결정한다. 그리고 그 결과를 order controller에게 보내준다. 이를 바탕으로 order controller는 order를 수행할 수 있는 가능성이 있는 AGV를 선택하고 선택된 AGV의 routing controller에게 다른 AGV의 initial routing information을 전달한다. 다른 AGV의 initial routing information을 이용하여 routing controller는 대안이동경로(bid)를 생성하고 이를 order controller에게 전달한다. Order controller는 bid를 판단하여 적절한 AGV에게 order를 할당한다.

Order정보의 변화 event가 일어났을 때의 system 작동과정은 order의 추가 event 일 때와 흡사하다. Order controller는 DB의 변화를 관찰하고 있다. 이러한 상황에서 order의 변화가 DB에 기록되면 order controller가 DB에서 변화된 order정보를 읽어 온다. 그 뒤의 과정은 order의 추가 event일 때와 동일하다.

Periodic event는 DB의 변화 없이 일정한 단위시간 간격으로 자동적으로 일어나는 event로서 이 event는 order가 할당되어진 후에 AGV의 운행 중에 발생할 수 있는 충돌을 방지하기 위해서 필요한 event이다.

3. Mechanisms of agent in AGV controller

Routing controller와 order controller는 mutli-agent로 구성되어 있다. 각 controller의 agent 역할은 시스템 내의 event에 따라서 다르다. <Figure 2>는 AGV controll system 내의 controller의 구조를 나타내고 있다.

각각의 routing controller는 다음과 같은 4개의 agent로 구성된다. 1)communication agent, 2)decision making agent, 3)negotiation agent, 4)routing agent.

Communication agent는 AGV와 외부와의 대화를 위한 정보교환의 역할을 수행한다. AGV는 communication agent를 통해서만 다른 routing controller나 order controller와 정보를 교환할 수 있다. 그리고 시스템 내부의 event도 감지한다. Negotiation agent는 order controller로부터 획득한 다른 AGV의 initial routing information을 바탕으로 대안이동경로(bid)를 산출하는 역할을 한다. 또한 충돌 회피를 위한 협상과정에서 사용될 bid를 생성하는 역할도 수행한다.

Decision making agent는 충돌 회피를 위한 협상과정

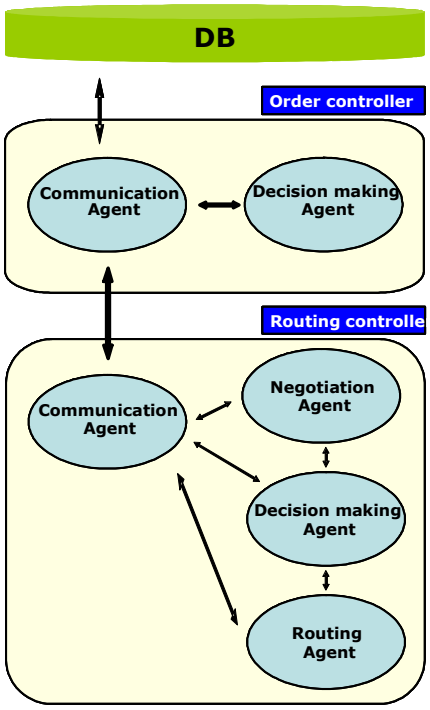

Figure 2. AGV controller architecture

에서 제시된 bid들을 판단하여 accept할 것인지 reject할 것인지를 결정하는 역할을 수행한다(Seungjin Oh *et al.*, 2004). Decision making agent는 Order의 할당에서는 담당하는 역할이 없다. Routing agent는 order controller에 의해서 주어진 order를 바탕으로 initial routing information을 생성하는 역할을 수행한다.

Order controller는 communication agent, decision making agent로 구성된다. Communication agent의 역할은 routing controller의 communication agent와 동일하다. Decision making agent는 order의 할당문제에서 각 routing controller에 의해 제시된 initial routing information을 평가하여 새로운 order의 실행가능성을 판단한다. 또한 routing controller에 의해서 생성된 대안이동경로(bid)를 판단하여 적절한 routing agent에게 order를 할당하는 역할을 맡는다.

3.1 Order allocation procedure

새로운 order를 할당하는 과정은 아래와 같다.

Step 1) Order controller의 communication agent가 order를 DB로부터 읽어온다. Order controller의 decision making agent는 order의 capacity를 만족할 수 있는 AGV가 있는지 판단한다. 만족하는 AGV가 존재하면 communication agent를 통해서 order정보를 routing controller에게 전달한다.

Step 2) Routing controller는 communication agent를 통해 획득한 order정보를 routing agent에게 전달하고 routing agent는 이를 이용해 order를 수행할 수 있는 routing path(initial routing information)을 생성한다. 이를 communication agent를 이용하여 order controller에게 전달한다.

Step 3) Order controller의 decision making agent는 전달받은 정보를 바탕으로 due date안에 order를 수행할 가능성이 있는 AGV가 있는지 판단한다. 가

능한 AGV가 존재하면 AGV의 initial routing information을 communication agent를 이용하여 각 AGV에게 전달한다.

Step 4) AGV의 routing controller는 communication agent를 통하여 획득한 다른 AGV의 initial routing information을 negotiation agent에게 전달한다. 이를 바탕으로 negotiation agent는 bid를 생성하고 이를 communication agent를 이용하여 order controller에게 넘겨준다.

Step 5) Order controller의 decision making agent는 bid간의 pair를 만들고 이를 평가한다. 평가한 결과에 따라서 order를 할당한다.

3.2 Routing agent

Routing agent는 AGV에게 주어진 order를 수행하기 위한 routing path(initial routing information)을 생성한다. 이때 AGV에게 주어지는 order 정보는 구체적인 AGV의 이동경로정보를 포함하고 있지 않고, 이동시킬 부품(P), 출발지(cp(i)), 목적지(cp(j)), due date(D), capacity(C) 정보만으로 이루어져 있다. Order 정보는 다음과 같이 주어진다.

Order : P; cp(i)→cp(j); D;C

<Figure 3>과 같은 shop layout이 있다고 가정하자. <Figure 3>에서 검은 점을 control point라고 부르고 숫자는 이 control point를 구별하기 위한 ID이다. 이때의 order 정보의 예는 아래와 같다.

Order : Part A; 11→1; 15; 16

Order controller는 이 정보를 communication agent를 통해 시스템 내에 있는 모든 AGV의 routing controller에게 전달한다. routing controller는 받은 order 정보를 먼저 routing agent에게 전달한다. routing agent는 받은 order 정보를 실제로 수행할 수 있는 구체적인 routing 정보로 이루어진 goal정보로 변환한다. 그리고 기존에 이미 수행 중이던 goal정보와 연결시킨다. 이를 decision making agent에게 전달한다. 이러한 goal정보를 initial routing information이라 하고 예는 아래와 같다.

Goal : 2→3→9→10→11→5→4→3→2→1

'→'는 한 단위시간당 AGV의 이동거리를 나타내는 것으로 '1→2'는 index가 1인 control point에서 index가 2인 control point로 이동하는데 1단위시간이 소요된다는 의미이다. 또한 밑줄이 쳐진 부분의 정보는 기존에 이미 수행 중이던 order의 goal정보이다. 이러한 goal정보를 산출하기 위해서 shortest path generation algorithm이 사용되었다.

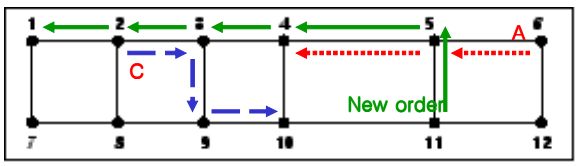

Figure 3. Shop layout

3.3 Communication agent

Routing controller와 order controller는 communication agent를 통하여 DB나 다른 controller와 통신할 수 있다. Communication agent를 통해서 교환되는 정보는 크게 2가지로 구분할 수 있다. 1)Order정보, 2)대안이동경로정보(bid)가 그것이다.

Order정보는 order의 추가 및 order정보 변화 event가 일어났을 때, DB와 order controller사이에 교환되는 정보이다. 또한 order controller와 routing controller사이에서도 교환된다. Order controller는 communication agent를 통해 DB의 변화를 관찰하고 있다. 이 상태에서 DB에 새로운 order가 관찰되며 동시에 작업완료 event가 일어나면 DB로부터 정보를 읽어오는 것이다. 이동경로정보는 routing controller와 order controller사이에 교환되는 정보와 routing controller들 사이에 교환되는 정보로 나누어진다. Order controller와 routing controller사이에 이동되는 정보는 order의 할당을 위해 교환되는 정보이고, routing controller사이에 교환되는 정보는 충돌 감지 및 해결을 위한 협상에 사용되는 정보이다.

3.4 Negotiation agent

Negotiation agent는 communication agent를 통해 얻은 다른 AGV의 initial routing information을 바탕으로 4가지 종류의 대안이동경로(bid)를 생성한다. Communication agent를 통해 얻은 initial routing information은 2가지 정보로 분리되는데, 첫 번째는 다른 AGV가 order controller로부터 제시된 새로운 order를 수행할 때의 routing 정보이고 두 번째는 새로운 order를 수행하지 않고 기존의 order만을 수행할 때의 routing 정보이다. 그 예는 아래와 같다.

AGV(C) :2→3→9→10→11→5→4→3→2→1
AGV(C) :2→3→9→10

Negotiation agent가 만드는 bid는 AGV가 새로운 order를 수행하는지의 여부, 기존 order의 수행을 위한 goal정보의 변경여부에 따라서 4가지로 구분된다. <Table 1>은 4가지 종류의 bid를 보여준다.

Table 1. Bid type

Bid type	Order 수행 여부	Goal정보 변경 여부
1	No	No
2	Yes	No
3	No	Yes
4	Yes	Yes

Bid type 1은 새로운 order를 수행하지 않으면서 기존의 order를 위한 자신의 goal정보를 변경하지 않는 bid로서 기존의 자신의 goal정보가 그대로 bid로 사용된다. Bid type 2는 새로운 order를 수행하면서 기존의 order를 위한 goal정보는 변경하지 않는 것이다. Bid type 2는 기존의 goal정보에 새로이 추가된 order의 goal정보의 연결로서 간단히 만들어 진다. AGV A의 기존 order에 관한 routing 정보와 새로운 order를 수행할 때의 routing 정보는 아래와 같다고 가정하자.

AGV(A) :6→5→4
AGV(A) :6→5→4→10→11→5→4→3→2→1

이 경우에 bid type1과 bid type2는 아래와 같다.

AGV(A) :6→5→4 (type1)
AGV(A) :6→5→4→10→11→5→4→3→2→1(type2)

Bid type 3은 다른 AGV가 order를 수행할 경우의 이동경로를 방해하지 않도록 자신의 이동경로를 변경하는 bid이다. Bid type 3을 생성하는 과정은 다음고 같다.

- Bid type 3 생성 과정

Step 1) Routing controller는 자신을 제외한 다른 AGV의 initial routing information을 획득한다. 자신을 제외한 다른 AGV의 initial routing information과 자신의 이동경로 정보의 pair를 만든다. AGV A가 AGV B로부터 initial routing information을 받아서 만든 pair의 예는 아래와 같다.

pair1;
AGV(A): 6→5→4
AGV(B) :2→3→9→10→11→5→4→3→2→1

Step 2) Step 1에서 만들어진 pair에서 자신의 이동경로와 다른 AGV의 이동경로가 충돌을 일으키지 않는다면 pair를 그래도 유지한다. 만약 충돌이 일어난다면 다른 AGV의 이동경로는 변화시키지 않고 충돌을 회피할 수 있는 모든 이동경로를 계산한다. 그 알고리즘은 <Figure 4>와 같고 사용되는 notation은 <Table 2>에 정리하였다.(Seungjin Oh *et al.*, 2004).

Table 2. Notation

k	*회피 AGV의 시작 control point*
c	*현재tree의 depth*
p	*충돌을 회피하는 AGV의 routing 시간 (단위시간으로 표현)*
A	*Shop layout 정보*
B	*기준AGV의 이동경로정보*
CP_i	*index가 i인 control point*
기준AGV	*충돌을 회피하지 않고 자신의 이동경로를 따라 행동하는 AGV*
회피AGV	*충돌을 회피하는 AGV*
is_conflict	*기준 AGV의 이동경로정보와 트리구조에서 추출한 이동경로정보 사이에 충돌이 있는지를 충돌여부를 확인하는 함수*

```
Make alternative-path{
  Root = new node(k)
  Node <- Root
  c <- 1
  Add-Child(Node, A, B, p, c)
  End
}

Add-Child(Node, A, B, p, c){
  if c = p
    then return
  for I <- 1 to number of control point
    if CPi is reachable to Node
      then if is_conflict(B, Node→CPi) = false
        then Child = new node(CPi)
             Insert(Node, Child)
             Add-child(Child, A, B, p, c+1)
}
```

Figure 4. 충돌을 회피하는 이동경로 계산 알고리즘

이 알고리즘은 트리구조를 이용하여 충돌 회피 이동경로를 생성하는 알고리즘이다. 트리의 root에 회피 AGV의 시작 control point를 입력한다. 이 control point로부터 1단위시간에 이동할 수 있는 모든 control point를 찾고 이를 reachable control point라 한다. 회피 AGV의 시작 control point와 reachable control point를 연결하여 이동경로를 생성한다. 이렇게 만들어진 이동경로가 기준 AGV의 이동경로와 충돌을 일으키지 않으면 reachable control point를 child node에 추가한다. 이와 같은 과정을 트리의 depth가 충돌을 회피하는 AGV의 routing시간과 같아 질 때까지 반복한다. 이 알고리즘을 통해 AGV B와 충돌을 일으키지 않는 AGV A의 모든 이동경로가 구해진다. 그 몇 가지 예는 아래와 같다.

AGV(A): 6→5→4
AGV(A): 6→5→11
AGV(A): 6→12→11

Step 3) Step 2에서 구해진 AGV A의 이동경로 의 마지막 control point를 시작점으로 하여 AGV A의 기존 order의 목적 control point까지의 shortest path를 구한다. 그 중 가장 짧은 routing path를 가지는 것을 bid type 3으로 정한다. 그 예는 아래와 같다.

AGV(A): 6→5→4
AGV(A): 6→5→11→5→4
AGV(A): 6→12→11→5→4

즉 이 예에서 bid type 3은 *AGV(A): 6→5→4* 이다.

Bid type 4는 자신이 order를 수행하지만 다른 AGV의 기존 order를 위한 goal의 이동경로를 방해하지 않고 자신의 이동경로를 변경하는 bid 이다. Bid type 4를 생성하는 과정은 bid type 3를 생성하는 과정과 매우 유사하다.

- Bid type 4 생성 과정

Step 1) Routing controller는 자신을 제외한 다른 AGV의 기존의 order 수행시의 이동경로정보를 획득한다. 자신을 제외한 AGV의 이동경로정보와 자신의 새로운 order를 위한 이동경로 정보(initial routing information)의 pair를 만든다. AGV A가 만든 pair의 예는 아래와 같다.

pair1;
AGV(A): 6→5→4→10→11→5→4→3→2→1
AGV(B) :2→3→9→10

Step 2) Step 1에서 만들어진 pair에서 자신의 이동경로와 다른 AGV의 이동경로가 충돌을 일으키지 않는다면 pair를 그래도 유지한다. 만약 충돌이 일어난다면 자신이 아닌 다른 AGV의 이동경로는 변화시키지 않고 충돌을 회피할 수 있는 모든 이동경로를 계산한다. 그 알고리즘은 type 3에서 사용했던 알고리즘과 동일하다. 한 가지 다른 점은 *P*가 충돌을 회피하지 않는 AGV의 routing시간이다.

이 알고리즘을 통해 AGV B와 충돌을 일으키지 않는 AGV A의 모든 이동경로가 구해진다. 그 몇 가지 예는 아래와 같다.

AGV(A): 6→5→4→5

AGV(A): 6→5→11→5
AGV(A): 6→12→11→5

Step 3) Step 2에서 구해진 AGV A의 이동경로 중에서 AGV A의 기존의 order의 목적 control point를 포함하는 이동경로가 있으면 이를 선택하고 없으면 AGV A의 이동경로의 마지막 control point에서 시작하여 AGV A의 기존 order의 목적 control point까지의 shortest path를 구한다. 그 중 가장 짧은 routing path를 선택한다. 선택한 routing path에 새로운 order를 수행할 수 있는 routing path 연결한다. 이것이 bid type 4이다. 즉 이 예에서는 아래의 routing path가 bid type 4이다..

AGV(A): 6→5→4→5→11→5→4→3→2→1

각 routing agent는 4가지의 bid를 생성하여 이를 order controller에게 전달한다.

3.5 Decision making agent

AGV의 scheduling과 관련하여서는 order controller의 decision making agent만이 역할을 수행하므로 이에 관하여서만 설명을 한다.

Decision making agent는 먼저 initial routing information을 평가하여 AGV의 order수행 가능성을 판단한다. Routing controller에 의해서 생성된 initial routing information은 order수행과정에서 발생할 수 있는 충돌문제는 배제하고 아무런 방해 없이 order를 수행하였을 시의 routing path이다. 이러한 상황에서 이 routing path가 order의 due date를
만족시키는지를 판단한다. 만족하는 AGV의 routing controller들에게만 다른 AGV의 initial routing information을 전달한다.

Decision making agent는 routing controller로부터 받은 bid를 평가하여 새로운 order를 할당받을 AGV를 선정하는 역할도 수행한다. 각 routing controller가 제시하는 bid는 4가지이다. Decision making agent는 이 내가지 bid들의 pair를 만든다. Pair를 만드는 방법은 한 AGV의 bid 중 order의 수행이 포함된 bid와 다른 AGV의 bid중 order의 수행이 포함되지 않은 bid를 pair로 만드는 것이다. 즉 한 AGV의 bid type 2,4번과 다른 AGV들의 bid type 1,3번과 pair를 만드는 방법을 사용한다. 이 과정을 bid를 제시한 모든 AGV에 관하여 수행한다. AGV A,B,C가 bid를 제시하였을 때 만들어 지는 pair의 예는 아래와 같다.

AGV A;
bid type 2(A); bid type 1(B); bid type 1(C)
bid type 2(A); bid type 1(B); bid type 3(C)
bid type 2(A); bid type 3(B); bid type 1(C)
bid type 2(A); bid type 3(B); bid type 3(C)
bid type 4(A); bid type 1(B); bid type 1(C)
bid type 4(A); bid type 1(B); bid type 3(C)
bid type 4(A); bid type 3(B); bid type 1(C)
bid type 4(A); bid type 3(B); bid type 3(C)

AGV B;
bid type 2(B); bid type 1(A); bid type 1(C)
bid type 2(B); bid type 1(A); bid type 3(C)
bid type 2(B); bid type 3(A); bid type 1(C)
bid type 2(B); bid type 3(A); bid type 3(C)
bid type 4(B); bid type 1(A); bid type 1(C)
bid type 4(B); bid type 1(A); bid type 3(C)
bid type 4(B); bid type 3(A); bid type 1(C)
bid type 4(B); bid type 3(A); bid type 3(C)

AGV C;
bid type 2(C); bid type 1(B); bid type 1(A)
bid type 2(C); bid type 1(B); bid type 3(A)
bid type 2(C); bid type 3(B); bid type 1(A)
bid type 2(C); bid type 3(B); bid type 3(A)
bid type 4(C); bid type 1(B); bid type 1(A)
bid type 4(C); bid type 1(B); bid type 3(A)
bid type 4(C); bid type 3(B); bid type 1(A)
bid type 4(C); bid type 3(B); bid type 3(A)

Decision making agent는 이렇게 만들어진 bid의 pair를 모두 평가한다. Pair에서 제시된 routing path에 따라서 AGV들이 움직일 때의 충돌발생여부가 첫 번째 평가기준이다. 즉 충돌을 일으키는 pair는 삭제된다. 두 번째 평가기준은 time capacity이다. 충돌이 일어나지 않는 pair들에 대해서는 각 AGV의 주어진 order들의 due date를 얼마나 준수하는 지를 판단하여 due date를 어기지 않는 pair를 선택하게 된다.

4. Example

Shop layout은 <Figure 3>과 같고 각 지점에서 지점으로 이동하는데 걸리는 단위시간은 1로서 같다고 가정한다. Shop내에는 3대의 AGV A,B,C가 존재한다. AGV B,C는 현재 order를 수행 중이고 AGV A는 order를 모두 마치고 control point 10에 정지하고 있다. AGV A,C의 capacity는 20이고 B의 capacity는 10이다. AGV B,C의 현재 goal과 due date는 다음과 같다.

AGV(B): 2→3→9→10; due date 6
AGV(C): 12→6→5 ; due date 3

이 상황에서 다음과 같은 새로운 order가 DB에 기록되었다.

Order : Part k; 5→1; 10; 16

이 새로운 order를 할당하는 과정은 아래와 같다.

Step 1) Order controller가 order를 DB로부터 읽어온다. 이 order의 capacity를 만족할 수 있는 AGV A,C에게 order정보를 넘겨준다.

Step 2) Routing controller는 communication agent를 통해 넘겨받은 order정보를 routing agent에게 넘겨주고 routing agent는 이를 이용해 order를 수행할 수 있는 routing path(initial routing information)을 생성한다.

AGV(A):10→11→5→4→3→2→1
AGV(C):12→6→5→4→3→2→1
이를 order controller에게 전달한다.

Step 3) Order controller는 넘겨받은 정보를 바탕으로 due date안에 order를 수행할 가능성이 있는지 판단하고 가능한 AGV에게 다른 AGV의 initial routing information을 넘겨준다.

AGV(A): completion time = 7<12

AGV(C): completion time = 7<12

본 예에서는 두 AGV 모두 가능하므로 각 AGV에게 다른 AGV의 initial routing information을 알려준다.

Step 4) 각 AGV의 routing controller는 획득한 다른 AGV의 initial routing information을 negotiation agent에게 넘겨준다. 이를 바탕으로 bid를 생성하고 이를 order controller에게 넘겨준다.

AGV A
10 (bid type 1)
10→11→5→4→3→2→1 (bid type 2)
10 (bid type 3)
10→4→4→5→4→3→2→1 (bid type 4)

AGV C
12→6→5 (bid type 1)
12→6→5→4→3→2→1 (bid type 2)
12→6→6→5(bid type 3)
12→6→5→4→3→2→1 (bid type 4)

Step 5) Order controller는 bid간의 pair를 만든다.

pair 1(bid type 2(A); bid type 1(C))
10→11→5→4→3→2→1; completion time 7
12→6→5; completion time 3

pair 2(bid type 2(A); bid type 3(C))
10→11→5→4→3→2→1; completion time 7
12→6→6→5; completion time 4

pair 3(bid type 4(A); bid type 1(C))
10→4→4→5→4→3→2→1; completion time 8
12→6→5; completion time 3

pair 4(bid type 4(A); bid type 3(C))
10→4→4→5→4→3→2→1; completion time 8
12→6→6→5; completion time 4

pair 5(bid type 2(C); bid type 1(A))
12→6→5→4→3→2→1; completion time 7
10; completion time 1

pair 6(bid type 2(C); bid type 3(A))
12→6→5→4→3→2→1; completion time 7
10; completion time 1

pair 7(bid type 4(C); bid type 1(A))
12→6→5→4→3→2→1; completion time 7
10; completion time 1

pair 8(bid type 4(C); bid type 3(A))
12→6→5→4→3→2→1; completion time 7
10; completion time 1

Negotiation agent는 각 pair를 평가한다. 평가한 결과는 <Table 3>과 같다.표에서 *는 order가 없는 경우를 말하며 due date violation에서 양수는 due date까지 남은 시간을 의미하며 음수는 due date에 늦은 시간을 의미한다. Negotiation agent는 conflict가 일어나지 않는 pair들 중에서 due date violation이 없는 pair중 양수의 합이 가장 큰 pair를 선택하게 된다.

Table 3. Pair 평가

Pair	Conflict	Due date violation-A (기존order/새로운order)	C
1	Yes	/	/
2	No	*/3	-1/*
3	No	*/2	0/*
4	Yes	/	/
5	No	*/*	0/3
6	No	*/*	0/3
7	No	*/*	0/3
8	No	*/*	0/3

양수의 합이 같은 pair가 다수 존재하면 무작위로 선택하게 된다. 예에서는 pair 5,6,7,8이 양수의 합이 3으로서 가장 크며 그 중 어떤 것을 택해도 AGV C에게 새로운 order가 할당된다.

5. Conclusion

본 연구에서는 분산제어방식에서의 AGV control을 위한 시스템을 제안하였으며, 시스템내에서의 AGV scheduling을 위한 알고리즘을 개발하였다. 본 연구에서 제시한 AGV control system은 order controller, routing controller, DB로 구성되어 있고, 각각의 controller는 multi-agent 구조를 가지고 있다. 제안한 시스템에서는 새로운 order의 할당 문제를 해결하기위해서 order controller와 routing controller내에 존재하는 agent간의 bid-based negotiation을 이용하였다. 이러한 분산 AGV 제어 시스템은 AGV의 고장, order의 변화와 같은 예측하기 힘든 상황에서도 유연하게 대처할 수 있다.

추후 연구로는 Negotiation시의 computational complexity에 영향을 주는 control point의 조밀도를 결정하는 연구가 진행되어야 한다. 또한 bid평가에 사용될 새로운 평가 방법의 개발도 필요하다.

Acknowledgement

이 논문은 2003년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2003-041-D00622).

References

Davis, W.J., Thompson, S. D. and White, L. R. (1990), The importance of decompositions in CIM control architectures, *Proc. CIMCON90 Conference, Washington, DC.*

Michiko Watanabe, Masashi Furukawa, Yukinori Kakazu, "Intelligent AGV driving toward an autonomous decentralized manufacturing system", *Robotics and Computer Integrated Manufacturing* vol.17, pp.57-64, 2001.

Pooya Farahvash, Thomas O. Boucher, "A multi-agent architecture for control of AGV systems", *Robotics and Computer-Integrated Manufacturing* vol. 20, pp. 473-483, 2004

Rajeeva Lochana Moorthya, Wee Hock-Guana, Ng

Wing-Cheongb, Teo Chung-Piawc, "Cyclic deadlock prediction and avoidance for zone-controlled AGV system", *International journal of Production Economics* vol.83, pp.309-324, 2003.

Seungjin Oh, Youngpil Cha, Mooyoung Jung(2004), Conflict detection and resolution for distributed control of AGVs based on agent technology, *Proceedings of 2004 KIIE Fall Conference, Seoul, Korea.*