Proceedings of KFIS Fall Conference, 2003. 5. 3

퍼지 처리시간을 갖는 다중 에이전트 시스템의 유전자 알고리즘 기반 작업 조정

Genetic Algorithm-based Coordination of Multiagent Systems with Fuzzy Processing Time

이 건 명
충북대학교 전기전자컴퓨터공학부, 첨단정보기술연구센터(AITrc)

Keon Myung Lee
School of Electric and Computer Engineering, Chungbuk National University, AITrc
E-mail : kmlee@cbucc.chungbuk.ac.kr

요 약

에이전트에서 수행할 수 있는 작업들에 대한 처리시간이 실제 작업 전에는 퍼지값으로만 주어지고, 실제 작업이 수행될 때야 작업 시간이 결정되는 다중 에이전트 시스템에 대해서 작업을 조정하는 방법을 제안한다. 제안한 방법은 두 단계의 유전자 알고리즘으로 구성되는데, 상위 단계의 유전자 알고리즘에서는 작업들을 적합한 에이전트에 할당하는 역할을 하고, 하위 단계의 유전자 알고리즘은 첫 번째 유전자 알고리즘의 제시하는 작업 할당 방법에 가장 적합한 작업 스케쥴을 탐색하는 역할을 한다. 이 논문에서는 제안한 유전자 알고리즘 기반 작업 조정 방법을 소개하고, 제안한 방법을 구현하여 실험한 결과를 보인다.

1. 서론*

실세계에서 많은 태스크(task)들이 일련의 구성 작업(operation)으로 구성된다. 이들 구성 작업들은 자원(resource) 제약, 생산성 제고 등으로 인해서 여러 에이전트들에 나누어져서 처리되기도 한다. 이러한 예로는 어떤 프로젝트를 세부 프로젝트로 나누어 외부 업체에 위탁을 하는 것, 제조 생산 시스템에서 태스크들이 여러 워크스테이션을 거쳐서 처리되는 것 등이 있다. 이러한 분야에서는 작업들을 에이전트들에 효과적으로 할당하고, 각 에이전트에 대해 할당된 작업들을 효율적으로 스케쥴링하는 것이 주요 관심사 중의 하나이다.[6,7] 에이전트들은 일반적으로 서로 다른 처리비용을 요구하는 여러 종류의 작업을 처리할 수 있는 능력을 가진다. 경우에 따라서는 이러한 처리 비용이 미리 정확히 알려지지 않은 경우도 있다. 이와 같이 퍼지한 값으로 비용이 주어지는 상황에서 태스크들의 작업 할당과 스케쥴링을 해야 하는 경우가 있다. 이 논문에서는 다중 에이전트 시스템에서 에이전트 작업시간이 퍼지값으로 주어이지는 상황에서의 태스크 할당과 스케쥴링 방법을 다룬다.

이 논문에서 대상으로 하는 다중 에이전트 시스템 환경은 다음과 같다. 에이전트들은 여러 가지 작업을 처리할 수 있고, 동일 작업이라도 에이전트에 따라 작업 처리시간이 다를 수 있다. 에이전트에서 각 작업의 처리시간은 퍼지숫자로 표현된 값을 가지면, 이 값은 작업의 비용으로 간주된다. 태스크들은 일련의 작업으로 구성되고, 태스크들은 다중 에이전트 시스템에 일괄적으로 주어진다고 전제한다. 에이전트는 작업 처리 중 장애가 발생해서 작동하지 못할 수 있다. 이 논문에서 대상으로 하는 문제는, 전체 태스크를 수행하는데 걸리는 시간(즉, makespan)이 최소화되도록, 태스크의 구성 작업들을 적합한 에이전트에 할당하고, 각 에이전트에 대해서 할당된 작업들에 대한 적합한 스케쥴을 찾는 것이다.

이 문제는 다음과 같이 두개의 세부문제로 구성된다. 하나는 태스크들의 작업을 효과적으로 에이전트들에 할당하는 것이고, 다른 하나는 각 에이전트에 대해 할당된 작업의 처리순서를 결정하는 태스크 스케쥴링을 하는 것이다. 작업할당에 대해서는 다중 에이전트 분야에서 contract net, acquaintence network 등 방법[6]이 제안되어 있으나, 시스템 전체의 성능을 고려하는데는 어려움이 있는 방법들이다. 태스크 스케쥴링은 NP-complete로 알려진 문제로, 다수의 휴리스틱 알고리즘들과, 신경회로망, 시뮬레이티드 어닐링(simulated annealing), 유전자 알고리즘, 타부(tabu) 검색 등을 이용하는 다양한 방법에 개발되어 왔다.[2,3,4,8,9] 이 논문은 유전자 알고리즘에 기반하여, 태스크 할당과 스케쥴링을 수행하는 방법을 제안한다.

2. 태스크 할당과 스케쥴링

대상이 되는 다중 에이전트 시스템에서는, 에이전트

* 이 연구는 첨단정보기술연구센터를 통해 과학재단으로부터 일부 지원을 받아 수행된 것임.

들이 여러 가지 작업을 처리할 수 있는 기능을 가지고 있다. 따라서 특정 작업을 처리할 수 있는 에이전트가 다수 존재할 수 있다. 태스크 할당 문제는 태스크들을 구성하는 각 작업에 대해, 해당 작업을 처리할 수 있는 에이전트 중에서, 다중 에이전트 시스템 전체의 성능을 최적이 되게 하는 가장 적합한 에이전트를 선택하는 것이다.

태스크의 할당이 끝나고 나면, 각 에이전트는 처리해야 할 작업들을 갖게 된다. 태스크 스케쥴링에서는 각 에이전트에 대해서 전체 시스템의 성능이 최적이 되도록 작업의 처리 순서를 결정한다. 이러한 태스크 스케쥴링은 Job shop 스케쥴링과 유사하기 때문에, Job shop 스케쥴링 등에서 이용되는 방법을 태스크 스케쥴링에 이용할 수 있다.[8] 태스크 할당이 이루어진 다음에 이에 대응하는 태스크 스케쥴링이 이루어지기 때문에, 태스크 할당이 어떻게 이루어지는가 하는 것이 전체 시스템의 성능에 크게 영향을 미치게 된다.

3. 유전자 알고리즘에 의한 작업조정

이 절에서는 다중 에이전트 시스템에 대한 태스크 할당과 스케쥴링을 하는 2단계 유전자 알고리즘을 제안한다. 제안한 유전자 알고리즘에서 상위 단계는 효율적인 태스크 할당 방법을 찾고, 하위 단계는 상위 단계에서 제시하는 태스크 할당 방법에 대응하는 효율적인 태스크 스케쥴을 찾는 역할을 한다.

procedure 상위 태스크 할당 유전자 알고리즘

step 1. 실행가능한 태스크 할당 방법들로 모집단을 초기화한다.

step 2. 하위 태스크 스케쥴링 유전자 알고리즘을 사용하여 각 태스크 할당 방법의 적합도를 평가한다.

step 3. 재생산, 교차, 돌연변이 등을 이용하여 다음 세대를 생성한다.

step 4. 하위 태스크 스케쥴링 유전자 알고리즘을 사용하여 각 태스크 할당 방법의 적합도를 평가한다.

step 5. 종료조건이 만족되면, 가장 적합도가 높은 태스크 할당 방법과 이에 대응하는 스케쥴을 출력하고, 그렇지 않으면 step 3으로 돌아간다.

procedure 하위 태스크 스케쥴링 유전자 알고리즘

step 1. 상위 태스크 할당 유전자 알고리즘이 제시한 태스크 할당 방법에 대응하는 실행가능한 태스크 스케쥴로 모집단을 초기화한다.

step 2. 각 스케쥴에 대한 적합도 값으로 makespan을 계산한다.

step 3. 다음 세대를 재생산, 교차, 돌연변이 등을 이용하여 생성한다.

step 4. 각 후보 스케쥴의 makespan을 계산한다.

step 5. 지정된 세대교체 수가 경과하면, 가장 작은 makespan 값을 갖는 스케쥴을 반환하고, 그렇지 않으면 step 3으로 돌아간다.

논문 기술의 편의상 다음 표기법을 사용한다.

$A = A_1, A_2, \ldots, A_m$: 에이전트의 집합

$T = T_1, T_2, \ldots, T_n$: 태스크의 집합

t_{ij} : 태스크 T_i의 j번째 작업

$O = O_1, O_2, \ldots, O_l$: 작업 종류의 전체 집합

$PT_i(O_j)$: 에이전트 A_i에서 작업O_j의 처리시간

$CA(O_i)$: 작업O_i를 처리할 수 있는 에이전트의 집합

$PR_i(O_j)$: 태스크 T_i에서 O_j의 직전 작업

N: 모든 태스크에 있는 전체 작업의 개수

3.1 상위 작업할당 유전자 알고리즘

여기에서는 상위 작업 할당 유전자 알고리즘의 기본 구성요소인 후보해 코딩방법, 모집단 초기화 방법, 사용하는 유전 연산자 및 후보해 적합도 평가 방법에 대해서 소개한다.

후보해 코딩 방법

각 후보해는 각 작업을 에이전트에 할당하는 방법을 나타낸다. 여기에서는 후보해를 크기 N의 배열로 표현하는데, 각 원소는 하나의 작업 t_{ij}에 대응하고, 원소값은 해당 작업을 처리할 에이전트의 식별자이다.

모집단 초기화 방법

초기 모집단을 구성할 후보해는 각 작업 t_{ij}에 대해 이를 처리할 수 있는 에이전트들 $CA(t_{ij})$ 중에서 무작위로 하나를 선택하여 할당하는 방법으로 생성한다.

유전 연산자

교차 연산자는 두개의 기존 후보해 C_1, C_2를 선택하여 다음과 같이 두개의 후보해 P_1, P_2를 생성한다. 우선 크기 N의 원소값이 0 또는 1로 무작위로 초기화된 배열 $MASK$를 만든다. 새로운 후보해 C_1을 만들 때, $MASK[i]=0$이면 $P_1[i]$를 $C_1[i]$에 복사하고, 그렇지 않으면 $P_2[i]$를 $C_1[i]$에 복사한다. 마찬가지 방법으로 P_1과 P_2의 역할을 바꿔 C_2를 생성한다.

돌연변이 연산자는 기존 후보해 P로부터 새로운 후보해 C를 다음과 같이 생성한다. 먼저 P를 C의 배열에 그대로 복사한 다음, 각 i번째 원소에 대해서, 지정된 돌연변이 확률로 $C[i]$에 대응하는 작업을 할 수 있는 에이전트들 하나를 무작위로 선택하여, 이것의 식별자를 $C[i]$에 지정한다.

적합도 평가

후보해의 적합도는 하위 태스크 스케쥴링 유전자 알고리즘에서 찾은 가장 좋은 스케쥴의 makespan 값이 된다. 논문에서는 에이전트에서 각 작업에 대해 미리 알고 있는 처리시간이 사다리꼴 퍼지숫자(Trapezoidal fuzzy number) $Trap(a,b,c,d)$로 주어진다고 가정한다. 따라서 하위 유전자 알고리즘에서 찾은 스케쥴의 makespan 값도 퍼지 사다리꼴 숫자가 된다.

상위 태스크 할당 유전자는 유전 연산자에서 사용될 후보해를 선택하기 위해 순위기반 선택(rank-based selection) 방법[1]을 이용한다. 이 선택방법을 이용하기 위해서는 후보해들을 적합도에 따라 정렬해야 하는데, 여기에서는 적합도가 사다리꼴 퍼지숫자이다. 제안한 방법에서는 정렬을 할 때, $d_1 < d_2$이거나, $d_1 = d_2$이면서 $a_1 < a_2$이면, 사다리꼴 퍼지숫자$Trap(a_1, b_1, c_1, d_1)$

이 $Trap(a_2, b_2, c_2, d_2)d_2$)보다 작다고 간주한다.

3.2 하위 태스크 스케쥴링 유전자 알고리즘

여기에서는 하위 작업 스케쥴링 유전자 알고리즘에서 사용되는 기본 구성요소에 대해 기술한다.

후보해 코딩 방법

후보해는 상위 태스크 할당 유전자 알고리즘에서 제시한 태스크 할당 방법에 부합되는 실행가능한 스케쥴이다. 후보해는 작업들의 순서화된 리스트(ordered list)로 표현된다. 각 태스크에서의 작업 처리 순서를 선행관계 제약조건으로 간주할 때, 이 순서화된 작업 리스트는, 선행관계에 의해 부분순서(partial order)화된 작업들에 대한 하나의 위상정렬(topological sort)에 해당한다. 순서화된 작업 리스트로 부터, 이에 대응하는 각 에이전트 별 스케쥴은 다음과 같이 구성할 수 있다. 리스트를 처음부터 읽어가면서, 각 작업이 할당된 에이전트를 확인하고, 해당 에이전트의 작업 리스트에 작업을 순차적으로 추가하면, 에이전트별 작업 순서가 찾아진다.

모집단 초기화

초기 모집단의 실행 가능한 후보 스케쥴을 생성하기 위하여, 다음과 같은 위상정렬 방법을 이용한다. 편의상 작업들에 대해서 1부터 N까지 정수의 인덱스가 부여되어 있다고 가정한다.

procedure 후보해 생성

step 1. 비어있는 배열을 만든다.

step 2. 1에서 N사이에서 임의로 하나의 숫자 c를 선택한다.

step 3. c가 아직 배열에 포함되지 않았고, c에 대응하는 작업의 이전에 처리되어야 할 작업들에 대한 인덱스가 이미 배열에 포함되어 있다면, c를 배열 끝에 추가한다.

step 4. 1부터 N까지 모든 숫자가 배열에 포함되어 있으면, 배열을 후보해로 반환한다. 그렇지 않으면 step 2로 돌아간다.

유전 연산자

교차 연산자는 먼저 순위기반 선택 방법에 따라 기존 모집단으로부터 두개의 순서화된 리스트 P_1, P_2를 선택한 다음, P_1으로부터 우선 하나의 작업 O_i를 무작위로 선택한다. O_i와 동일한 태스크에 속하는 작업들을 표시한 다음, P_1에서 표시된 작업들을 새로운 배열의 동일 위치에 복사한다. 나머지 작업들은 P_2에 나타나는 순서대로 배열에 채워넣는다.

돌연변이 연산자는 모집단으로 선택한 순서화된 리스트 $(o_1, o_2, \ldots, o_N)$를 새로운 리스트에 복사한 다음, 리스트에서 무작위로 하나의 위치 i를 선택한다. 선택한 위치의 작업을 포함하는 태스크의 작업중에서 i로부터 왼쪽편으로 가장 가까운 곳에 있는 작업의 위치 lmp를 찾고, 동일한 태스크에 속하는 작업중에서 i의 오른쪽에 위치한 작업들 중에서 가장 가까운 작업의 위치 rmp를 찾는다. $lmp+1$과 $rmp-1$ 범위 내에서 하나의 위치 p를 무작위로 선택해서, o_i와 o_p를 교환한다.

이들 교차 연산자와 돌연변이 연산자는 부여된 선행관계를 유지하는 후보해를 생성한다고 증명되어 있다.[5] 태스크 스케쥴에서 후보해는 작업의 선행관계를 만족하는 위상정렬로 볼 수 있기 때문에, 이들 연산자는 유효한 스케쥴을 생성한다.

적합도 평가

후보해의 적합도는 해당 스케쥴의 makespan 값이다. 에이전트에서 작업 O_i의 시작 시각 $O_i.start\text{-}time$은 다음과 같이 결정된다.

$O_i.start\text{-}time$ = max\{O_i를 처리하는 에이전트에서 O_i 바로 이전에 처리한 작업의 종료시각, O_i를 포함한 태스크에서 O_i 바로 이전에 처리되어야 하는 작업의 종료시각\}

작업 O_i의 종료 시각 $O_i.finish\text{-}time$은 다음과 같이 결정된다.

$O_i.finish\text{-}time$ = $O_i.start\text{-}time + PT_i(O_i)$

4. 실험

제안한 방법의 유용성을 검증하기 위해, 제안한 태스크 할당 및 스케쥴링을 하는 2단계 유전자 알고리즘과, 이를 통해서 찾아 스케쥴을 실행해 보는 시뮬레이터를 구현하였다. 대상으로 하는 다중 에이전트 시스템에서는 에이전트의 작업 처리 시간 정보가 사다리꼴 퍼지숫자로 주어지기 때문에, 생성된 스케쥴에서는 각 작업의 시작시각과 종료시각이 사다리꼴 퍼지숫자로 나타난다. 이러한 스케쥴을 사용하여 태스크를 처리하더라도, 작업이 처리되면서는 결국 일반값(crisp value)으로 시간이 나타나야 한다. 시뮬레이션에서는 퍼지숫자의 분포를 확률분포로 간주하고, 분포에 따라 확률적으로 하나의 일반값을 선택하여 사용하였다.

다음은 실험에서 사용한 예제 문제이다.

태스크별 작업 순서

$T_1 : O_2O_4O_6O_7O_1O_5$ $T_2 : O_6O_9O_1O_{10}O_8$

$T_3 : O_2O_4O_5O_7O_9O_8$ $T_4 : O_1O_4O_3O_8O_7O_2$

에이전트별 처리가능 작업 및 처리시간

$CO(A_1) = \{(O_1, Trap(1,2,3,4)), (O_2, Trap(2,2,3,3)),$
$(O_4, Trap(4,5,7,7)), (O_6, Trap(3,4,6,8)),$
$(O_7, Trap(2,4,5,6)), (O_8, Trap(7,9,9,10)),$
$(O_9, Trap(5,7,8,10)), (O_{10}, Trap(4,5,7,7))\}$

$CO(A_2) = \{(O_1, Trap(2,3,4,5)), (O_2, Trap(3,5,6,6)),$
$(O_3, Trap(5,7,8,10)), (O_4, Trap(2,3,6,7)),$
$(O_5, Trap(4,6,7,9)), (O_6, Trap(4,8,9,9)),$
$(O_7, Trap(3,7,8,10))\}$

$CO(A_3) = \{(O_4, Trap(5,6,7,8)), (O_5, Trap(2,2,5,5)),$
$(O_6, Trap(1,3,5,6)), (O_7, Trap(8,9,9,12)),$
$(O_8, Trap(6,8,9,11)), (O_9, Trap(3,5,6,7)),$
$(O_{10}, Trap(7,8,9,10))\}$

$CO(A_4) = \{(O_1, Trap(4,6,7,7)), (O_3, Trap(5,5,7,9)),$
$(O_5, Trap(6,7,7,8)), (O_7, Trap(2,4,5,6)),$
$(O_9, Trap(3,5,5,6)), (O_{10}, Trap(4,5,7,8))\}$

$CO(A_5) = \{O_2, Trap(4,5,6,7)), (O_3, Trap(4,5,8,8)),$
$(O_4, Trap(1,2,4.5)), (O_6, Trap(6,7.8,9)),$
$(O_8, Trap(4,5,7.8)), (O_{10}, Trap(3.6.8,9))\}$

다음은 제안한 유전자 알고리즘 기반 방법에 의해서 구한 태스크 할당과 이에 대응하는 스케쥴이다. 여기

에서 $(T_i, O_j) : Trap(a, b, c, d)$은 태스크 T_i의 작업 O_j가 $Trap(a, b, c, d)$의 종료시각을 갖는다는 것을 나타낸다.

$A_1[(T_1, O_2) : Trap(2,2,3,3)-(T_1, O_6) : Trap$
$(8,13,18,19)-(T_1, O_1) : Trap(16,24,33,37)-(T_3, O_9)$:
$Trap(12,18,26,30)-(T_3, O_8)$:$Trap(20,29,40,45)-$
$(T_4, O_2) : Trap(22,31,43,49)]$
$A_2[(T_1, O_4) : Trap(4,5,9,10)-(T_4, O_1) : Trap$
$(5,7,12,14)-(T_3, O_7) : Trap(9,13,20,23)-$
$(T_2, O_{10}) : Trap(16,22,32,36)]$
$A_3[(T_3, O_5)$:$Trap(7,9,15,17)-(T_2, O_9)$:$Trap$
$(10,14,21,24)-(T_1, O_7) : Trap(12,18,26,30)$
(T_1, O_5):$Trap(22,31,40,45)]$
$A_4[(T_2, O_6)$:$Trap(1,3,5,6)-(T_3, O_4)$:$Trap(5,7,10,12)-$
(T_2, O_1):$Trap(12,17,25,29)-(T_2, O_8)$:
$Trap(20,27,39,44)]$
$A_5[(T_3, O_2)$:$Trap(4,5,6,7)-(T_4, O_4)$:$Trap(6,9,16,19)-$
(T_4, O_3):$Trap(11,16,24,29)-(T_4, O_8)$:
$Trap(17,24,33,40)-(T_4, O_7)$:$Trap(19,28,38,46)]$

다음은 시뮬레이터를 이용하여 위의 스케쥴에 따라 작업을 수행한 예이다. 여기에서 (T_i, O_j):(t_s, t_f)는 태스크 T_i의 작업 O_j가 t_s에 시작해서 t_f에 종료되는 것을 나타낸다. 이 실행 예는 makespan 값 $Trap(22,31,43,49)$를 갖는 원래 퍼지 스케쥴을 시뮬레이터에 적용하여 얻은 makespan값이 36.8인 일반 스케쥴이다.

$A_1[(T_1, O_2)$:$(0, 2.3)-(T_1, O_6)$:$(5.99, 12.7)-(T_1, O_1)$:
$(18.8, 24.4)-(T_3, O_9)$:$(24.4, 29.1)-(T_3, O_8)$:$(29.1, 34.5)-$
(T_4, O_2):$(34.5, 36.8)]$
$A_2[(T_1, O_4)$:$(2.3, 5.99)-(T_4, O_1)$:$(5.99, 8.08)-(T_3, O_7)$:
$(10.4, 14.1)-(T_2, O_{10})$:$(18.2, 23.4)]$
$A_3[(T_3, O_5)$:$(7.49, 10.4)-(T_2, O_9)$:$(10.4, 15.1)-(T_1, O_7)$:
$(15.1, 18.8)-(T_1, O_5)$:$(24.4, 31.1)]$
$A_4[(T_2, O_6)$:$(0, 3.04)-(T_3, O_4)$:$(5.1, 7.49)-(T_2, O_1)$:
$(15.1, 18.2)-(T_2, O_8)$:$(23.4, 28.8)]$
$A_5[(T_3, O_2)$:$(0, 5.1)-(T_4, O_4)$:$(8.08, 10.5)-(T_4, O_3)$:
$(10.5, 17.4)-(T_4, O_8)$:$(17.4, 25.3)-(T_4, O_7)$:$(25.3, 29)]$

5. 결론

이 논문에서는 작업 처리시간을 퍼지숫자로 가지고 있는 에이전트들로 구성된 다중 에이전트 시스템에 대해서 태스크 할당과 태스크 스케쥴링을 수행하는 방법을 제안하였다. 제안한 방법에서는 상위 단계 유전자 알고리즘에서 유효한 태스크 할당을 찾도록 하고, 이에 대응하는 태스크 스케쥴을 하위 단계 유전자 알고리즘에서 찾도록 한다. 태스크 할당와 태스크 스케쥴링 문제는 NP-complete인 조합 최적화 문제이며, 태스크 할당 결과에 의해 최적의 태스크 스케쥴이 결정된다. 따라서 어떻게 태스크가 할당되는가에 따라, 전체 시스템의 성능이 크게 영향을 받는다. 이러한 관점에서 제안한 2단계 유전자 알고리즘은 효과적인 태스크 할당과 태스크 스케쥴을 찾는데 효과적인 방법이다. 제안한 방법의 적용가능성을 확인하기 위해 제안한 방법을 구현하고, 이를 이용하여 실제 적용해 보기 위한 시뮬레이터를 구현하였다. 실험을 통해서 제안한 방법이 효과적인 태스크 할당 및 스케쥴을 찾을 수 있다는 것을 확인할 수 있었다. 제안한 방법에서는 태스크들이 미리 주어지고 이에 대해 태스크 할당과 태스크 스케쥴링을 한다는 전제하고 있지만, 실제 태스크가 온라인으로 동적으로 시스템에 주어지는 경우에도 제안한 방법을 다음과 같이 사용한다. 일정 시간 간격으로 동적으로 주어지는 태스크를 모아서, 이에 대해서 주기적으로 태스크 할당과 스케쥴링을 하도록 하여, 이에 따라 에이전트들이 작업을 처리하도록 하면, 제안한 방법을 이와 같은 동적 환경에도 적용할 수 있다.

6. 참고문헌

[1] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1998.

[2] P. Bruker, B. Jurisch, B. Sievers. A branch and bound algorithm for the job shop scheduling problem. In *OsnabruckerSchriften zur Mathematik*, Universitat Osnabruck, 1992.

[3] P. J. M. V. Laarhoven, E. H. L. Aarts, J. K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research.* Vol.40, No.1., 1992, pp.113-125.

[4] I. Ono, M. Yamamura, S. Kobayashi. A genetic algorithm for job shop scheduling problems using job-based order crossover. In *Proc. of IEEE conf. on Evol. Comp.*, 1996.

[5] K.-M. Lee, T. Yamakawa, K.M. Lee. Genetic algorithm approaches to job shop scheduling problems: An Overview. *Int. J. of Knowledge -based Intelligent Engineering Systems*, Vol.4, No.2, 2000, pp.72-85.

[6] J. Ferber, Multi-Agent Systems: *An Introduction to Distributed Artificial Intelligence*, Addison Wesley Longman, 1999.

[7] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* (eds.) MIT Press, 1999.

[8] J. Blazewicz, K. Ecker, G. Schmit, J. Weglarz, *Scheduling in Computer and Manufacturing Systems*, Springer-Verlag, 1993.

[9] S. Kobayashi, I. Ono, M. Yamamura. An Efficient Genetic Algorithm for Job Shop Scheduling Problems. In *Proc. of the 6-th Int. Conf. on Genetic Algorithms*, 1995, pp.506-511.