

An Initial Task Assignment Method for Autonomous Distributed Vehicle Systems with Finite Buffer Capacity

Yusuke Morihiro
miyamoto@eei.eng.osaka-u.ac.jp

Toshiyuki Miyamoto
Graduate School of Engineering,
Osaka University
2-1 Yamadaoka, Suita, Osaka, Japan

Sadatoshi Kumagai
kumagai@eei.eng.osaka-u.ac.jp

Abstract

This paper discusses an on-line Tasks Assignment and Routing Problem(TARP) for Autonomous Transportation Systems(ATs) in manufacturing systems. The TARP results in a constrained version of the Pickup and Delivery Problem with Time Windows(PDPTW). As an approach to this problem, a cooperative algorithm with autonomous distributed agents has been proposed. The algorithm is able to plan deadlock-free routes even though the buffer capacity is less, but includes reformability at the point that computation time of that case increases drastically. This paper proposes an initial task assignment method to reduce computation time on planning routes. Results of computational experiments show effectiveness of the proposed method.

1 Introduction

Recently, consumers need more diverse and individual goods. In consequence, producers need to response to it and a high-mix low-volume manufacturing becomes the mainstream in manufacturing industries. Intelligent Manufacturing Systems(IMSs)[3], Flexible Manufacturing Systems(FMSs)[4] or cell manufacturing systems have received increasing attention in recent decades, and an effective control scheme for such systems is required. We can divide the control problem into two problems: the scheduling problem of process machines and the Tasks Assignment and Routing Problem(TARP) of Autonomous Transportation Systems(ATs) such as Automated Guided Vehicle(AGV)[5] systems, Overhead Hoist Vehicle(OHV) systems, etc.

Usually, these problems were solved by a central computer beforehand, otherwise dispatching rules are used without solving the problem. Due to its computational complexity, it is impossible to change or modify on-line the schedule in the former case. We, however, need more flexible and agile system to respond to a machine failure, an urgent work, and so on. The former method has

a drawback of the lack of agility. As for the dispatching rule method, solutions by the method are incomparably worse than optimal solutions in many cases, and this method may easily derive infeasible solutions, although this method is very fast, and capable of coping with changes. From these difficulties, it is natural to use agents to solve a local problem of each element in the manufacturing system, and to use cooperation among agents to resolve conflicts between elements[1, 2]. Autonomous Distributed Manufacturing Systems(ADMSs) have advantages such as improvement of fault tolerance, reduction of load, and so on, but it is difficult to achieve total optimization. How to obtain more efficient solutions is important in the framework.

This paper focuses on an on-line TARP for ATs along with a schedule made by a production scheduling system. We will show that the TARP results in a constrained version of the Pickup and Delivery Problem with Time Windows(PDPTW)[7]. The PDPTW targets on-demand transportation systems: demand-bus systems and courier services, and usually it is solved in a central manner: exact algorithms[8, 9, 10], tabu search[11, 12], genetic algorithm[13], etc. Our approach is different from existing studies at following points. (1) In the classical PDPTW, finite capacity of vehicles is considered, but finite capacity of pickup and delivery points is not considered. In case of manufacturing systems, a pickup or delivery point corresponds to a machine or its buffer. We need to consider finite capacity of buffers. (2) Our target systems are ADMSs, and therefore we need a distributed (cooperative) algorithm to solve the problem. The existing results cannot be applied directly. Dial introduced autonomous Dial-a-Ride transit in his paper[14]. Although its application area is a public transportation system, the concept of *autonomous* transit is close to our ADMSs. Unfortunately, just basic idea is described in the article. (3) In the classical PDPTW, only one kind of ordering relation among pickup and delivery points is given; a pickup point must be arrived at before the corresponding delivery point by the same vehicle. In addition to above relation, we need to consider ordering relation came from the man-

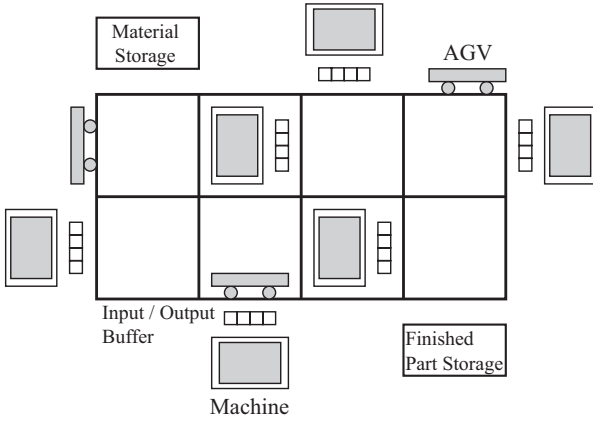


Figure 1. An example of target manufacturing system.

ufacturing schedule.

Due to the above new constraints, a possibility being a deadlock situation among vehicles arises. We have proposed a cooperative algorithm with autonomous distributed agents for task assignment and routing problems in ADMSs[19]. This algorithm is able to plan deadlock-free routes even though the buffer capacity is less, but includes reformability at the point that computation time increases drastically as the buffer capacity lessens. This paper proposes an initial task assignment method to reduce computation time on planning routes. This paper is organized as follows. A mathematical formulation of the problem is given and existing technique for it is summarizes in Sect.2. Section 3 describes the algorithm proposed in this paper to improve initial task assignment for reducing computation time. Then, computational evaluation and analysis are presented in Sect.4. Finally, we conclude in Sect.5.

2 Task Assignment and Routing Problem

2.1 Target Manufacturing System

An example of our target manufacturing system is shown in Fig.1. On the floor, several machines or cells are scatteringly located, and paths are led between locations. Autonomous transportation vehicles convey works along the paths. Several assumptions are listed as follows:

- Each machine can process only one operation at a time.
- Vehicle speeds are the same.
- Collisions between vehicles are not considered¹.
- The load/unload time is included in operational time.

¹In order to consider the collisions, we need more detail assumption about the target system and using technology. Because we want to generalize the problem as much as possible, we assumed this.

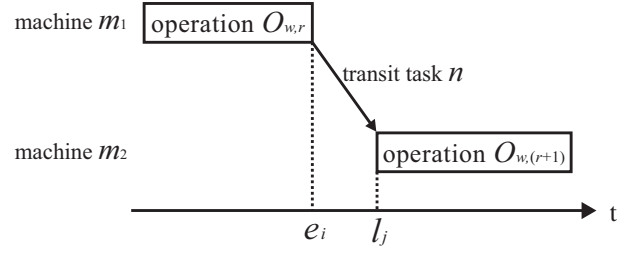


Figure 2. Transit task

- Each machine has the finite input/output buffer capacity.

Based on these assumptions, agents for a Production System(PS) and a ATS make a schedule cooperatively by using the rolling horizon scheme, where a PS makes a production schedule based on product requests. Therefore we do not need to consider failures on vehicles or machines. Because when a failure happens, agents will re-generate a schedule. This paper, however, focuses only on the ATS part, and coordination mechanism between the ATS part and the PS part is out of the scope of this paper. We assume that a manufacturing schedule is given by the PS part, and approximate travel time is considered on calculating the schedule.

2.2 Problem Formulation

For the mathematical formulation, the following notations are given.

We use the same notation $O_{w,r}$ for an operation and its work, where $O_{w,r}$ means the r -th operation of product w , and $O_{w,0}$ is a material of the product. Pickup point and delivery point of work $O_{w,r}$ is expressed as $O_{w,r}P$ and $O_{w,r}D$, respectively.

When consecutive operations are done at different machines, a transit task n occurs. A transit task is represented by a 4-tuple: (i, e_i, j, l_j) , where i is a pickup point ($O_{w,r}P$), e_i is the ready pickup time, j is a delivery point ($O_{w,r}D$), and l_j is the desired delivery time. e_i is given by the finish time of operation $O_{w,r}$, and l_j is given by the start time of operation $O_{w,(r+1)}$ and these times are obtained from the given schedule, see Fig.2. If i is the material storage, set e_i at 0 and if j is the products storage, set l_j at large number or its due date (when given).

Other notations are defined as follows:

W : the set of products,

R_w : the length of work sequences until product w ($w \in W$) is completed,

z_w : the totally-ordered set of work sequences of product w ($w \in W$), i.e.,
 $(z_w, \Rightarrow) = \{O_{w,1}, O_{w,2}, \dots, O_{w,R_w}\}$,

M : the set of machines,

U_m : the number of operations for machine m ($m \in M$) by the given schedule,

s_m : the totally-ordered set of production schedules for machine m ($m \in M$), i.e.,
 $(s_m, \rightarrow) = \{s_{m,1}, s_{m,2}, \dots, s_{m,U_m}\}$,

K : the set of vehicles,

N : the set of transit tasks,

N' : the set of transit tasks which have already picked up at the time the problem is being to be solved, where $N' \subset N$,

$G = (V, A)$: the complete graph, where

V : the set of points, $V = V^+ \cup V^- \cup P$, variance

V^+ : the set of pickup points, i.e.,
 $V^+ = \{i \mid \forall (i, e_i, j, l_j) \in N - N'\}$,

V^- : the set of delivery points, i.e.,
 $V^- = \{j \mid \forall (i, e_i, j, l_j) \in N\}$,

p_k : the current point of vehicle k ,

P : the set of current points of all vehicles.
 $P = \{p_k \mid \forall k \in K\}$,

A : the set of edges, where
 $A = \{(i, j) \mid \forall i, j \in V, i \neq j\}$,

q_i : the loaded quantity at pickup point or the unloaded quantity at delivery point,

$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels point } i \text{ to point } j, \\ 0 & \text{otherwise,} \end{cases}$

d_{ij} : the mileage from point i to point j ,

t_{ij} : the travel time from point i to point j ,

AT_i : the arrived time at point i ,

Q_k : the capacity of vehicle k ,

y_{ik} : the loaded capacity of vehicle k when arrived at point i ,

C : the set of machines and storage,

f : the mapping $f : V^+ \cup V^- \rightarrow C$,

UB_c : the buffer capacity of $c \in C$. If c is a storage, UB_c is infinite,

$AB_c(\tau)$: the consumed buffer capacity of c ($c \in C$) at time τ ,

D_j : the delay time at delivery point j ($j \in V^-$), where
 $D_j = \max(AT_j - l_j, 0)$,

$last_k$: the last point along a route of vehicle k ,

The TARP is a problem to find a task assignment and vehicle routes which minimize a summation of the delay time at delivery points as well as a summation of the mileage of all vehicles. The TARP is formulated as an integer programming problem as follows:

$$\text{minimize } Q = \alpha \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} x_{ijk} \cdot d_{ij} + \beta \sum_{j \in V^-} D_j \quad (1)$$

subject to

$$\forall (i, e_i, j, l_j) \in N - N', e_i \leq AT_i < AT_j \quad (2)$$

$$\forall i, j \in V, \forall k \in K,$$

$$x_{ijk} = 1 \Rightarrow AT_i + t_{ij} \leq AT_j \quad (3)$$

$$\forall p_k \in P, \sum_{j \in V^+ \cup V^-} \sum_{k \in K} x_{p_k j k} \leq 1 \quad (4)$$

$$\forall j \in V, \forall k \in K, \sum_{i \in V} x_{ijk} \leq 1 \quad (5)$$

$$\forall i \in V, \forall k \in K, \sum_{j \in V} x_{ijk} \leq 1 \quad (6)$$

$$\forall h \in V^+ \cup V^- - \{last_k\}, \forall k \in K,$$

$$\sum_{i \in V} x_{ihk} - \sum_{j \in V} x_{hjk} = 0 \quad (7)$$

$$\forall i \in V, \forall k \in K, y_{ik} \geq 0 \quad (8)$$

$$\forall j \in V, \forall k \in K, y_{jk} + q_j \leq \sum_{i \in V} x_{ijk} Q_k \quad (9)$$

$$\forall i \in V, \forall j \in V, \forall k \in K,$$

$$x_{ijk} = 1 \Rightarrow y_{ik} + q_i = y_{jk} \quad (10)$$

$$\forall i \in V^+ \cup V^-, \forall \tau \geq 0,$$

$$AB_{f(i)}(\tau) \leq UB_{f(i)} \quad (11)$$

$$\forall i, j \in V^+ \cup V^-, \forall k \in K, x_{ijk} = 1 \Rightarrow$$

$$AB_{f(i)}(AT_i) = AB_{f(i)}(AT_i - 1) - q_i,$$

$$AB_{f(j)}(AT_j) = AB_{f(j)}(AT_j - 1) - q_j \quad (12)$$

$$\forall w \in W, 1 \leq \forall r \leq R_w,$$

$$AT_{O_{w,(r-1)D}} < AT_{O_{w,rP}} \quad (13)$$

$$\forall m \in M, 1 \leq \forall u \leq U_m - 1, s_{m,u} = O_{x_1, y_1},$$

$$s_{m,u+1} = O_{x_2, y_2},$$

$$AT_{O_{x_1, (y_1-1)D}} < AT_{O_{x_2, y_2P}} \quad (14)$$

$$x_{ijk} \in \{0, 1\} \quad (15)$$

The objective function (1) is given by a weighted sum of a summation of mileage of all vehicles and a summation of delay time at delivery points. The time windows and precedence constraints are ensured (2) and (3). Constraint (4) specifies that there are maximum one routes going out of the current point for each vehicle. Constraints (5) and (6) ensure that a vehicle can visit a point at most once. Constraint (7) ensures that every point is entered and left by the same vehicle. Constraints (8)-(10) show the capacity constraints. Constraints (11) and (12) express the buffer capacity. Constraints (13) and (14) define the

constraint of work sequences and the production schedule, respectively. Constraint (15) ensures the integrity of the decision variables.

Different from classical PDPTW, constraints of the buffer capacity ((11), (12)) and the ordering relations ((13), (14)) are added. On this account, deadlock described in Sect. 2.3 may occur.

2.3 Deadlock Situations

Three types of deadlock situations can be listed up as follows:

- (a) The scheduler does not consider the buffer capacity.
- (b) A vehicle can not put a work due to buffer full.
- (c) Vehicles wait for each other.

(a) We assume that a task arises when the next operation is operated by another machine. If more than UB_c consecutive operations are assigned to the machine c , it occur a deadlock. In this case, we must change the schedule, otherwise we need to introduce inter-stage buffers. This kind of deadlocks is out of the scope of this paper.

(b) When the buffer of a machine is full, a vehicle k_1 can not put a work into the buffer. In this case, vehicle k_1 must wait for another vehicle k_2 to pickup one of works in the buffer. From the fourth assumption of the target system, vehicle k_1 can wait for vehicle k_2 at the buffer point without considering a collision with vehicle k_2 . However, if k_2 is waiting for k_1 's succeeding job, this situation becomes a deadlock.

(c) Two or more vehicles might wait according to the schedule each other, and the deadlock occurs in this case.

2.4 Cooperative Algorithm for TARP

The flow-chart of the existing algorithm [19] is shown in Fig. 3. At the Initial Task Assignment step, an initial task assignment procedure in [15] is used. Each vehicle agent plans a route autonomously by using a heuristic algorithm, which is called an insertion heuristic [19], based on decided task assignment.

Two inner loops: Resolve Cycle loop and Avoid Buffer Overflow loop, resolve deadlock situations (b) and (c) in Sect. 2.3. The outer most loop: Move Task loop, changes task assignment in order to improve evaluation value. A local search method, which finds a better solution in neighborhoods, is used for changing task assignment.

In three loops, agents avoid deadlock situations and change task assignment by communication and cooperation of agents, although these loops are shown like a centralized algorithm in Fig. 3. Details of cooperative mechanism are described in [19].

3 Initial Task Assignment Method

3.1 Problems of the Existing Cooperative Algorithm

Results of computational experiments by the cooperative algorithm are shown in Table 1. Table 1 shows evaluation value and computation time when buffer capacity is

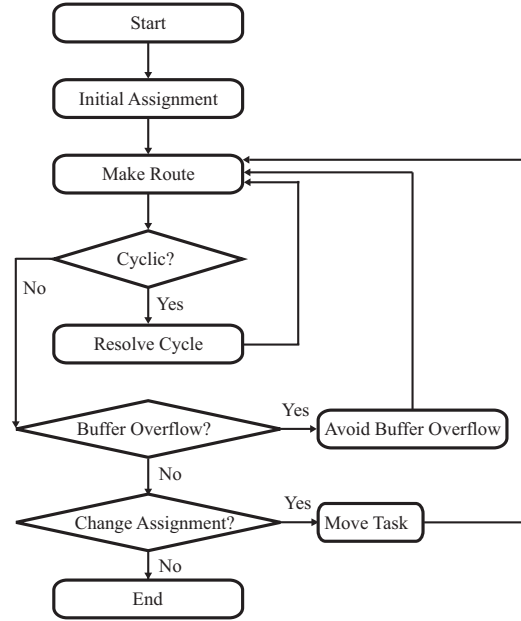


Figure 3. Flow-chart of the existing algorithm

changed in both cases where task reassignment procedure is performed and is not performed.

As the results show evaluation values are improved with the Move Task loop, but computation time increases drastically. We target an on-line planning problem, therefore computation time for determining routes must be reduced.

3.2 Basic Idea of Improve Initial Task Assignment

Computation time should be reduced by decreasing the number of executions of the Move Task loop in Fig. 3. We may be able to decrease executions of the Move Task loop if initial task assignment is close to the final task assignment. We tried to reduce computation time and maintain evaluation value by improving initial task assignment and removing the Move Task loop.

The flow-chart of the proposed algorithm is shown in Fig. 4. We improve initial task assignments by the Initial Move Task loop. Where the Initial Move Task loop is not able to consider the deadlock situations, a relaxed problem is solved. Sect. 3.3 describes the Initial Task Assignment loop, and the relaxed problem is described in Sect. 3.4.

3.3 Initial Task Assignment Loop

This procedure is based on greedy method, and improves task assignment. Evaluation value Q is computed by solving the relaxed problem in Sect. 3.4.

The following notations are given:

N_k The set of transit tasks assigned to vehicle k ($k \in K$).

LF The set of vehicle k where the set of undelivered tasks N'_k is not empty.

Table 1. Experimental results of existing method

Method	With Move Task Loop				
buffer capacity	2	3	4	5	6
evaluation value	2679.6	613.0	359.0	357.0	357.0
calculation time (second)	1418.5	162.2	75.8	61.2	61.5
Method	Without Move Task Loop				
buffer capacity	2	3	4	5	6
evaluation value	2819.2	780.3	359.2	359.2	359.2
calculation time(second)	35.6	3.5	2.1	2.0	1.9

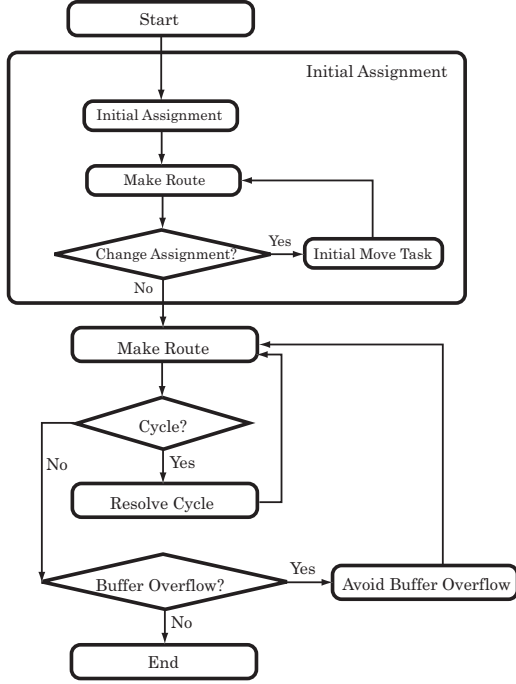


Figure 4. Flow-chart of the proposed algorithm

Cooperation is done as follows:

[Initial Task Assignment Loop]

Step1 Select a leader vehicle k' from L at random.

Step2 Vehicle k' select a task $worst$ which makes $Q_{k'}$ worst from $N'_{k'}$.

Step3 Let Q'_{leader} be an evaluation value when $worst$ is removed from $N'_{k'}$ and let Q_{leader} be an evaluation value before removing $worst$. Compute ΔQ_{leader} by (16).

$$\Delta Q_{leader} = Q'_{leader} - Q_{leader} \quad (16)$$

Step4 Vehicle k' sends the information of $worst$ to all other vehicles.

Step5 Each agent k ($\forall k \in K - \{k'\}$) adds $worst$ to N_k temporarily and solves the problem. Let Q'_{other} and Q_{other} be evaluation values after and before assigning $worst$ to k , respectively. Compute ΔQ_{other} by (17) and send it to vehicle k' .

$$\Delta Q_{other} = Q'_{other} - Q_{other} \quad (17)$$

Step6 Let $efficiency$ be a transfer efficiency computed by (18).

$$efficiency = -(\Delta Q_{leader} + \Delta Q_{other}) \quad (18)$$

If the maximum value of $efficiency$ is positive, $worst$ is re-assigned from k' to k_{max} which makes $efficiency$ maximum. Remove vehicle k' from L if N'_k becomes empty, and add vehicle k_{max} to L .

If the maximum value of $efficiency$ is negative, $worst$ is not re-assigned, and remove vehicle k' from L .

Step7 Iterate **Step1**–**Step6** till L becomes empty.

In **Step6**, re-assignment is performed only the case where $efficiency$ is positive. Therefore evaluation value of whole system must be smaller for any task reassignment. So, this loop terminates in finite steps, because evaluation values have a lower bound.

3.4 Problem Relaxation

Evaluation value Q in the Initial Task Assignment loop is calculated by solving TARP formulated in Sect. 2.2, but deadlock situations cannot be considered in the loop. Therefore each vehicle obtains the evaluation value Q by solving a problem which is relaxed in two points: considering only assigned tasks, and not considering the buffer capacity.

The relaxed TARP for each vehicle k is formulated as follows:

$$\text{minimize } Q_k = \alpha \sum_{i \in V_k} \sum_{j \in V_k} x_{ijk} \cdot d_{ij} + \beta \sum_{i \in V_k^-} D_i \quad (19)$$

subject to

$$\forall n \in N - N', e_i \leq AT_i < AT_j \quad (20)$$

$$\forall i, j \in V_k, x_{ijk} = 1 \Rightarrow AT_i + t_{ij} \leq AT_j \quad (21)$$

$$\forall j \in V_k, \sum_{i \in V_k} x_{ijk} \leq 1 \quad (22)$$

$$\forall i \in V_k, \sum_{j \in V_k} x_{ijk} \leq 1 \quad (23)$$

$$\forall h \in V_k - \{last_k\}, \sum_{i \in V_k} x_{ihk} - \sum_{j \in V_k} x_{hjk} = 0 \quad (24)$$

$$\forall i \in V_k, y_{ik} \geq 0 \quad (25)$$

$$\forall j \in V_k, y_{ik} + q_j \leq \sum_{i \in V_k} x_{ijk} Q_k \quad (26)$$

$$\forall i \in V_k, \forall j \in V_k, x_{ijk} = 1 \Rightarrow y_{ik} + q_i = y_{jk} \quad (27)$$

$$\forall w \in W, 1 \leq \forall r \leq R_w, AT_{f(O_{w,(r-1)D})} < AT_{f(O_{w,rP})} \quad (28)$$

$$\forall m \in M, 1 \leq \forall u \leq U_m - 1, s_{m,u} = O_{x_1,y_1}, s_{m,u+1} = O_{x_2,y_2}, \quad (29)$$

$$AT_{f(O_{x_1,(y_1-1)D})} < AT_{f(O_{x_2,y_2P})} \quad (29)$$

$$x_{ijk} \in \{0, 1\} \quad (30)$$

Constraint (4) about number of vehicle and constraints (11), (12) about buffer capacity are removed.

4 Computational Experiments

4.1 Experiment Conditions

Task sets are generated by using benchmark job-shop problems (la01, ..., la10 in [17]) and a famous production scheduler APSTMIZER[18]. Some characteristics of the problems are shown in table 2. Tasks during the first 400 unit times in the schedule are used. Because these problems does not consider travel times, we added travel time t_{ij} to the operation time of $O_{w,(r+1)}$, where $i = O_{w,rP}$ and $j = O_{w,rD}$ on calculating schedules.

The performance of the proposed algorithm was evaluated on the 10 problems using Pentium IV computer running at 2.4GHz. The program is implemented on our multi-agent environment[16], and written in Java language. Three vehicles are allocated at the material storage and return to the material storage after finished all tasks. A capacity of each vehicle is set at 20. Road network is a grid and a interval of one edge is 1. Each machines are located at grid point in the road network. Time is considered as discrete time. Vehicles move one edge in one unit time, so mileages is considered as traveling times. An evaluation value is calculated by equation (1), and to put emphasis on a delivery delay time more, the weighting factors of the objective function are set as $\alpha = 1, \beta = 3$, respectively. Buffer capacity is changed from 2 to 6.

Table 2. Job shop problem

	Type A	Type B
$ M $	5	5
$ W $	10	15
R_w	5	5
the number of problems	5	5

4.2 Experimental Results

We compared the proposed method with other methods to evaluate the effectivity. Compared methods are as follows:

Proposed : Computing according to flow-chart in Fig. 4.

Existing : Computing according to flow-chart in Fig. 3.

Combination : Computing with Initial Move Task loop and Move Task loop.

Without Move Task loop : Computing according to flow-chart in Fig. 3 without Move Task loop.

Table 3 shows the mean values of the ratio of total mileage, total delivery delay time, evaluation value and calculation time with a results of Existing and the variance (VAR) of the ratio of evaluation value over ten examples but Existing is experimental value. However only the total delivery time is the mean of experimental value (due to find the case that total delivery time is zero in Existing).

4.3 Discussion

The results of Proposed in Table 3 show that the proposed method was succeeded to reduce computation time and to improve evaluation value when buffer capacity is enough (UB_c : 4, 5, 6). This shows that initial task assignment was improved by the proposed method and proves efficiency of it. Regarding the results of Proposed and Without Move Task loop, the proposed method could improve evaluation value, too.

On the other hand, evaluation values of the Combination are smaller than these of the Existing. We can confirm that planning of a high quality route is possible by combining the Initial Move Task loop and the Move Task loop. But computation time tends to increase by combination of these two loops. There, however, exists a case that computation time is reduced like the case of buffer capacity is 3. In this case, we think that the proposed method is able to reduce the number of runs of the Move Task loop.

In addition, the variance of the proposed method is very large when buffer capacity is less (UB_c : 2,3). This shows that there are several problems whose evaluation value is enormously large. We analyzed the results of these examples in detail. In the result, it was found that a work whose next operation was not scheduled during target period made buffer busy. In order to avoid this situation, agents need to plan routes which improve total evaluation value even if the routes make own evaluation value worse. Proposing such a routing method is future issue.

We target an on-line operation, so reducing computation time is very important issue. In this point, the proposed method has a great advantage against the existing method. Using the proposed method gives huge benefits due to being able to get better results than the existing method when buffer capacity is enough.

Table 3. Experimental results

Method		Proposed					Combination				
buffer capacity		2	3	4	5	6	2	3	4	5	6
total mileage		0.935	0.913	0.915	0.915	0.921	0.948	.0921	0.921	0.927	0.916
delivery delay time		1007.8	213.1	3.0	3.1	3.0	507.3	86.2	2.1	1.5	1.9
evaluation value	mean	3.367	2.024	0.908	0.909	0.914	0.922	0.907	0.906	0.907	0.900
	VAR	21.8209	5.8048	0.0040	0.0040	0.0032	0.2447	0.0038	0.0044	0.0042	0.0049
calculation time		0.090	0.201	0.257	0.330	0.324	3.326	1.024	1.207	1.276	1.317
Method		Without Move Task loop					Existing				
buffer capacity		2	3	4	5	6	2	3	4	5	6
total mileage		0.994	0.995	0.997	1.003	1.003	357.0	349.0	347.0	345.0	345.0
delivery delay time		821.4	144.1	4.4	4.4	4.4	774.2	88.2	4.0	4.0	4.0
evaluation value	mean	1.289	1.487	1.000	1.007	1.007	2679.6	613.0	359.0	357.0	357.0
	VAR	0.6857	1.3771	0.00002	0.0003	0.0003					
calculation time		0.037	0.030	0.030	0.033	0.031	1418.5	162.2	75.8	61.2	61.5

5 Conclusions

This paper discussed initial task assignment method in an on-line routing problem, called the TARP, for vehicles in consideration of a production schedule and finite buffer capacity. The proposed algorithm in this paper improved initial task assignment with planning routes by relaxed constraints and moving tasks by cooperation.

The effectiveness of the proposed algorithm was shown by the results of computational experiments that computation time was reduced by improved initial task assignment when buffer capacity of machines was large. We could get more efficient routes by combining the Move Task loop and the Initial Move Task loop.

References

- [1] A.D. Baker, "A survey of factory control algorithms that can be implemented in a multi-agent heterarchy: Dispatching, scheduling, and pull", *Journal of Manufacturing Systems*, vol.17, no.4, pp.297–320, 1998.
- [2] A. Giret and V. Botti, "Holons and agents", *Journal of Intelligent Manufacturing*, vol.15, pp.645–659, 2004.
- [3] IMS Center, <http://www.ims.org/>
- [4] F. T. S. Chan and H. K. Chan, "A comprehensive survey and future trend of simulation study on FMS scheduling", *Journal of Intelligent Manufacturing*, vol.15, no.1, pp.87–102, 2004.
- [5] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems", *European Journal of Operational Research*, in press, available online.
- [6] W. Shen and D. H. Norrie, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey", *Knowledge and Information Systems*, vol.1, no.2, pp.129–156, 1999.
- [7] L. D. Bordin, A. Assad, and M. Ball, "Routing and scheduling of vehicles and crews: The state of the art", *Computers & Operations Research*, vol.10, no.2, pp.63–67, 1983.
- [8] H. Psaraftis, "An Exact Algorithm for the Single Vehicle Many-to-Many Dial-A-Ride Problem with Time Windows", *Transportation Science*, vol.17, no.3, pp.351–357, 1983.
- [9] J. Desrosiers, Y. Dumas, and F. Soumis, "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows", *American Journal of Mathematical and Management Sciences*, vol.6, no.3, pp.301–325, 1986.
- [10] K. Ruland and E. Rodyn, "The Pickup and Delivery Problem: Faces and Branch-and-Cut Algorithm", *Computers Math. Applic.*, vol.33, no.12, pp.1–13, 1997.
- [11] J. Cordeau and G. Laporte, "A tabu search heuristic for the static multi-vehicle dial-a-ride problem", *Transportation Research Part B*, vol.37, pp.579–594, 2003.
- [12] A. Landrieu, Y. Mati, and Z. Binder, "A tabu search heuristic for the single vehicle pickup and delivery problem with time windows", *Journal of Intelligent Manufacturing*, no.12, pp.497–508, 2001.
- [13] K. Uchimura, H. Takahashi and T. Saitoh, "Demand Responsive Services in Hierarchical public Transportation System", *IEEE Transactions on Vehicular Technology*, vol.51, no.4, pp.760–766, 2002.
- [14] R. Dial, "Autonomous Dial-a-Ride Transit Introductory Overview", *Transpn. Res.-C*, vol.3, no.5, pp.261–275, 1995.

- [15] T. Miyamoto, K. Nakatyou, S. Kumagai, “Agent-Based Planning Method for an On-Demand Transportation System”, *Proc. of the 2003 IEEE International Symposium on Intelligent Control*, pp.620–625, October 2003.
- [16] T. Miyamoto and S. Kumagai, “A Multi Agent Net Model and the Realization of Software Environment”, *Proc. of Workshop of Petri Nets to intelligent system development with 20th International Conference on Application and Theory of Petri Nets*, 83–92, 1999.
- [17] S. R. Lawrence, “Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques”, Working Paper, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, 1984.
- [18] Y. Nishioka, <http://www.img.k.hosei.ac.jp/pslib/>
- [19] T. Miyamoto, N. Tsujimoto, S. Kumagai, “A Cooperative Algorithm for Autonomous Distributed Vehicle Systems with Finite Buffer Capacity”, *IEICE Trans. Fundamentals*, vol.E88-A, NO.11, pp.3036–3044, 2005