Applying DBR and MAS to Solving Container Loading Problem

Yuan LIU *1 • Yajie TIAN *2 • Tetsuo SAWARAGI *1

The container loading problem, a real hard problem, is usually difficult to obtain even a suboptimal solution because of not only multiple complicated restrictions but also of multiple objectives. In this paper, a heuristic algorithm is proposed for solving the container loading problem in the real-world. The algorithm is based on Drum-Buffer-Rope presented in the Theory of Constraints and the multi-agent cooperative negotiation strategy. A particular attention is focused on improving the constrained agent by striving for the trade-off of restrictions and cooperative negotiations, so that the final solution can arrive its biggest profit. Since many real-world problems are restricted by many complicated restriction that are difficult to be satisfied simultaneously, a method used by human experts called restriction relaxation is embedded in the proposed algorithm, which makes the algorithm have a high degree of flexibility. In this paper, the proposed algorithm is also compared with other two classical optimization algorithms based on Local Search and Tabu Search.

Key words : Container loading problem, Drum-Buffer-Rope, Multi-agent system.

I. INTRODUCTION

THERE are many kinds of loading patterns for the container loading problem (CLP), such as *layer building* [1], *tower building* [2], *block building* [3], and so on. The loading pattern is decided generally by the cargo characteristics and users' requests. In this paper, *two-row building* is discussed and cargoes are large in length and width, i.e., the length and width of the cargoes are close to the half width of the container.

CLP discussed in this paper is a multi-destination delivering and multi-container loading problem. It is divided into two processes; the first process is making modules, i.e., to allocate cargoes vertically with limited height and weight, and the second process is loading containers, i.e. to load those modules into containers. The first process was discussed in another paper [4] and only the second process is discussed in this paper.

CLP is a real hard problem because it is always accompanied with multiple restrictions and objectives. (1) Multiple restrictions are usually self-contradictory and difficult to be satisfied simultaneously. (2) It is to get the biggest profit of the whole loaded containers instead of parts of loaded containers. (3) The strength of restrictions and the priorities of sub-objectives in the problem are changing with the situations. Thus, it is a dynamic and diverse problem.

If each loaded container is considered as an agent, allocated modules are regarded as resources of the agents and loading principles are considered to be the action rules of the agents for managing the resources, thus CLP can be considered as a multi-agent system (MAS). The cooperative negotiation strategy (CNS) was proposed for solving conflicts among agents competing for limited resources [4]. In this paper a heuristic algorithm based on the Drum-Buffer-Rope (DBR) of the Theory of Constraints (TOC) and CNS, is proposed for solving CLP.

TOC is an idea from management science invented by Dr. Eliyahu M. Goldratt [5]. The techniques illustrated in TOC such as the DBR scheduling and the five-step focusing process for continuous improvement by focusing on the constraints can be applied to capacity constrained job shops with high product diversity. In the DBR scheduling, "*Drum*" refers to the pace followed, "*Buffer*" refers to the safeties used to exploit constraints, and "*Rope*" refers to interdependent events that trigger each other's activities. The five-step focusing process is described as follows: Step1, identify the system's constraint(s); Step2, decide how to exploit the system's constraint(s); Step3, subordinate/synchronize everything else to the above decisions; Step4, elevate the system's constraint(s); Step5, if in the above steps the constraint has shifted, go back to Step1, and do not allow inertia to become the system's constraint [6].

Based on DBR and the five-step focusing process, an approach called "Three-zone Buffer Management" (TZBM) was proposed [4]. TZBM is to divide all agents into three zones: Green-zone, Yellow-zone and Red-zone according to

*1 Graduate School of Engineering, Kyoto University

*2 Logistic System Group, Bigbang INC.

the situation of satisfying relaxable restrictions (soft restrictions). A particular attention of TZBM is focused on improving the situation of Red-zone, because it limits the system to reach its goal. The approach of TZBM is also improved in this paper for solving CLP. In order to reflect how much an agent violates the restrictions, each agent is evaluated through an evaluation function. The evaluation function is composed of several sub-evaluation functions and each of sub-evaluation functions is defined according to the rules of one kind of soft restriction. Furthermore, each of sub-evaluation functions is summed up with a weighted parameter that can be adjusted by the user according to real requirements. If an agent belongs to Red-zone and has the maximum (i.e., worst) value of sub-evaluation corresponding to unsatisfied soft restrictions, then it is called the constrained agent. This agent is improved by cooperative negotiations among agents and relaxations of soft restrictions.

MAS mentioned in this paper is an autonomous and co-operative system. Each agent makes competitive behavior for arriving its own objectives and concessive behavior for arriving the objectives of the whole system. However, it is difficult for an agent to make decision on when an agent needs to make competitive or concessive behavior, how much extent of concession needs to be done, which agent needs to make concessive, and so on. The heuristic algorithm will give answers to these questions.

This paper is organized as follows. In section II, details of CLP are introduced. Then, the heuristic algorithm is mentioned in details in section III. The computation experiment is shown in section IV, and finally the conclusion follows.

II. CONTAINER LOADING PROBLEM

CLP is a real-world problem coming from the centers of physical distributions, manufactories and warehouses. As mentioned above, there are two processes in *two-row building* which is shown in Fig.1.

Basically, all containers and cargoes are rectangular parallelepipeds of known sizes. Each cargo is positioned parallel to the side walls of the container. We assume origins of the containers and the cargoes are their innermost points respectively from the perspective shown in Fig.1. The length/width/height of the container is with respect to x -axis/ y -axis/ z -axis, respectively.

CLP is a multiple destination delivering problem. There are N_o ($N_o \geq 1$) packing orders to be allocated into modules then into containers, and those are delivered to the different destinations on a trip. Consequently all cargoes are divided into N_o groups. These packing orders should obey the rule of "first-in-last-out" (FILO) according to the delivering destinations and routes.

The container is defined as $C = \{L, W, H, CC\}$, where included parameters are length, width, height and carrying capacity of the container, respectively. There are N_m ($N_m \geq$

Fig.1 Making Modules and Loading Containers

1) modules in the container C , and the side ml_k of m_k is allocated along the W direction of the container.

Let $m_k = \{ml_k, mw_k, mh_k, mg_k, mx_k, my_k\}$ ($1 \leq k \leq N_m$) represents the k -th module in one container (ml_k : the length, mw_k : the width, mh_k : the height, mg_k : the weight, mx_k and my_k : the position along x -axis and y -axis in the container, respectively). The ml_k and mw_k are the maximum length and width of cargoes distributed in m_k . The mh_k and mg_k are calculated by summing the height and weight of each included cargo, respectively.

As known, real-world problems are usually restricted by many complicated restrictions and some restrictions are too strict to be satisfied simultaneously. In order to increase the flexibility and adaptability of the algorithm, restrictions are divided into hard and soft. Hard restrictions must be satisfied even though the loading efficiency is low and soft restrictions are allowed to be relaxed according to the different situations.

Hard Restrictions:

- H_1 : The packing orders must obey FILO.
- H_2 : The total weight of modules loaded in a container cannot exceed the carrying capacity of the container, i.e.

$$\sum_{k=1}^{N_m} mg_k \leq CC \quad (1)$$

- H_3 : Each module must be included completely within a container.

Soft Restrictions:

- S_1 : If the gravity center of the container floor is (X_{gc}, Y_{gc}) and θ is the angle between x -axis and a radial linking the geometry center and the gravity center (seeing Fig.2), (X_{gc}, Y_{gc}) should be satisfied as :

$$\begin{cases} (X_{gc} - \frac{L}{2})^2 + (Y_{gc} - \frac{W}{2})^2 \leq (X_c - \frac{L}{2})^2 + (Y_c - \frac{W}{2})^2 \\ X_c = L/2 + \varepsilon_c \times L \times \cos\theta \\ Y_c = W/2 + \varepsilon_c \times W \times \sin\theta \end{cases} \quad (2)$$

Fig.2 Soft restriction of gravity center (topview)

Fig.3 Inside six gaps (topview)

where ε_c is a relaxation parameter related to the gravity center and $\varepsilon_c \in [\varepsilon_{c0}, \varepsilon_{cmax}]$ ($\varepsilon_{c0} \leftarrow 0.125$ and $\varepsilon_{cmax} \leftarrow 0.25$ in this paper according to the user's request).

- **S₂:** The balance between the left and right row is important for CLP. The length of the left and right row as shown in Fig.3 should be satisfied as:

$$|L_{left} - L_{right}| \leq \varepsilon_b \times mw_{min} \quad (3)$$

where mw_{min} is the minimum width of all modules. ε_b is a relaxation parameter related to the balance and $\varepsilon_b \in [\varepsilon_{b0}, \varepsilon_{bmax}]$ ($\varepsilon_{b0} \leftarrow 0.1$ and $\varepsilon_{bmax} \leftarrow 0.3$ in this paper).

- **S₃:** CLP requires to leave a certain distance between the two rows, which is called the regular gap written as gw_{reg} ($gw_{reg} \leftarrow 20cm$ in this paper). The average wide of the gaps (seeing Fig.3) between the corresponding two modules belonging to different rows is defined as:

$$GW = \frac{\sum_{k=1}^{N_g} (|gw_k - gw_{reg}| \times gl_k)}{\sum_{k=1}^{N_g} gl_k} \quad (4)$$

where gl_k/gw_k is the length/width of the k -th gap, N_g is the total number of gaps in the current container. GW should be satisfied as:

$$GW \leq \varepsilon_g \times gw_{reg} \quad (5)$$

where ε_g is a relaxation parameter related to the gap wide and $\varepsilon_g \in [\varepsilon_{g0}, \varepsilon_{gmax}]$ ($\varepsilon_{g0} \leftarrow 0.25$ and $\varepsilon_{gmax} \leftarrow 0.5$ in this paper).

- **S₄:** It should be avoided that the top cargo falls off from the higher module such as shown in Fig.4. The

Fig.4 Unstable module (sideview)

average of height differences of neighbor modules is defined as:

$$HD = \frac{\sum_{rw=left}^{right} \sum_{k=2}^{N_m^{rm}} |mh_{k-1} - mh_k|}{N_m^{left} + N_m^{right} - 2} \quad (6)$$

where N_m^{left}/N_m^{right} is the module number of the left/right row ($N_m^{left} + N_m^{right} > 2$). HD should be satisfied as:

$$HD \leq \varepsilon_h \times h_{min} \quad (7)$$

where h_{min} is the minimum height of all cargoes, ε_h is a relaxation parameter related to the cargo stability and $\varepsilon_h \in [\varepsilon_{h0}, \varepsilon_{hmax}]$ ($\varepsilon_{h0} \leftarrow 0.1$ and $\varepsilon_{hmax} \leftarrow 0.3$ in this paper).

Evaluation Functions:

According to above discussions, the sub-evaluation functions are defined as follows:

- **E₁:** The sub-evaluation function for the gravity center is:

$$E_{s1}(C) = \sqrt{\frac{(X_{gc} - \frac{L}{2})^2 + (Y_{gc} - \frac{W}{2})^2}{(X_{cmax} - \frac{L}{2})^2 + (Y_{cmax} - \frac{W}{2})^2}} \quad (8)$$

(X_{cmax}, Y_{cmax}) is an intersection as shown in Fig.2 while ε_c being relaxed to the maximum.

- **E₂:** The sub-evaluation function for the balance of the left and right row is:

$$E_{s2}(C) = \frac{|L_{left} - L_{right}|}{\varepsilon_{bmax} \times mw_{min}} \quad (9)$$

- **E₃:** The sub-evaluation function for the gap width is:

$$E_{s3}(C) = \frac{GW}{\varepsilon_{gmax} \times gw_{reg}} \quad (10)$$

- **E₄:** The sub-evaluation function for the cargo stability is defined as :

$$E_{s4}(C) = \frac{HD}{\varepsilon_{hmax} \times h_{min}} \quad (11)$$

- **E₅:** The sub-evaluation function for the floor occupied ratio is:

$$E_{s5}(C) = 1 - \frac{\sum_{k=1}^{N_m} ml_k \times mw_k}{L \times W} \quad (12)$$

Based on these five sub-evaluation functions, the integrated evaluation function for a container can be described as:

$$E(C) = \sum_{i=1}^5 u_i \times E_{s_i}(C) \quad (13)$$

where $u_i (i = 1, 2, \dots, 5)$ are weighted parameters, which depend on how much the user prioritizes in relation with others.

Note that the bigger the E is the worse the loading situation is and the empty container does not join the negotiation.

III. HEURISTIC ALGORITHM

The loading process is divided into two parts, the first part is to generate the initial solution and the second part is to improve the initial solution.

A. Generating initial solution

The basic loading procedure is considered as a 2-dimension allocating problem and it is difficult to obtain the minimum container number because of the properties of modules and the complicated restrictions. To get the minimum container number is not the purpose in this paper, but the main purpose is to improve the constrained agents in Red-zone and to avoid the solving process from falling down the deadlock. So a concision but practical algorithm is utilized here.

The algorithm for generating the initial solution is as follows (Note: it is assumed that the length of a module is longer than its width initially and the ml_k side is allocated along the y -axis of the container):

- **Step1:** (Arrange packing orders) Arrange all given modules according to their packing orders and heights in a series $\mathbb{M} = \{m_1^1, m_2^1, \dots, m_{n_1}^1, \dots, m_1^j, \dots, m_i^j, \dots, m_{n_j}^j, \dots, m_1^{N_o}, \dots, m_{n_{N_o}}^{N_o}\}$, where N_o is the total number of packing orders (there are n_j modules in the j -th order group). Let $r \leftarrow 1$.
- **Step2:** (Choose a container) Choose an empty container $C_r(L_r, W_r, H_r, CC_r)$ and let $L_{left} \leftarrow 0, L_{right} \leftarrow 0$.
- **Step3:** (Choose a module in the case of $L_{left} = L_{right}$) If there is a module m_i^j satisfying *packing condition* (1)(2) in the front order group, place it into the left row. Let $L_{left} \leftarrow L_{left} + mw_i, C_r \leftarrow C_r + \{m_i^j\}, \mathbb{M} \leftarrow \mathbb{M} - \{m_i^j\}$ and go to *Step4*. Otherwise there is no module satisfying *packing condition* (1)(2) in the front order group, go to *Step5*.
- **Step4:** (Choose a module in the case of $L_{left} \neq L_{right}$) If there is a module m_i^j satisfying *packing condition* (1)-(3) in the front order group (because of the width restriction, the rotation station of m_i^j , i.e., its width is longer than its length, is considered), place it in the shorter row. Let $\min\{L_{left}, L_{right}\} \leftarrow \min\{L_{left}, L_{right}\} + mw_i, C_r \leftarrow C_r + \{m_i^j\}$ and $\mathbb{M} \leftarrow \mathbb{M} - \{m_i^j\}$, go back to

Step3; else if there is a set of modules satisfying *packing condition* (1)(2) but (3) in the front order group, find a module m_i^j satisfying $mw_i \leq L_r - \max\{L_{left}, L_{right}\}$ in the set, place it in the shorter row after a small gap being given up (seeing the third gap of Fig.3). Let $\min\{L_{left}, L_{right}\} \leftarrow \max\{L_{left}, L_{right}\} + mw_i, C_r \leftarrow C_r + \{m_i^j\}$ and $\mathbb{M} \leftarrow \mathbb{M} - \{m_i^j\}$, go back to *Step4*. Otherwise there is no module satisfying above conditions, go to *Step5*.

- **Step5:** (Judge the end condition) If $\mathbb{M} = \Phi$, the initial solution is generated and end the generating procedure. Otherwise, let $\mathbb{C} \leftarrow \mathbb{C} + \{C_r\}, r \leftarrow r + 1$ and go back to *Step2*.

packing condition:

- (1) There is a module m_i^j so that its weight should satisfy $mg_i \leq CC_r - \sum_{k=1}^{N_m} mg_k$.
- (2) There is a module m_i^j so that its width should satisfy $mw_i \leq L_r - \min\{L_{left}, L_{right}\}$.
- (3) There is a module m_i^j so that its length should satisfy $ml_i \leq W_r - ml'_i$, where ml'_i is the length of the module located at the opposite row.

According to the above generating procedure, the initial solution is obtained and described as $\mathbb{C} = \{C_1, C_2, \dots, C_{N_c}\}$.

B. Adjusting the heights of modules

Sometimes, because weights of the cargoes are too heavy, the occupied ratios of containers are quite low, which makes the stability of containers be worse. In this case, parts of modules need to be adjusted.

Generally, it is to decrease the heights of the modules by calling on the first process [4] and how much height to be decreased depends on the requirement of the floor occupied ratio. The criterion of the floor occupied ratio is given by the expert in the site.

C. Defining three zones

As mentioned above, loaded containers should satisfy many restrictions, which are usually interdependent and difficult to be satisfied simultaneously. In order to avoid the occurrence of violating situations as far as possible, an effective and feasible method called *restriction relaxation* is used in the real-world. For obtaining the optimal values of relaxed parameters and decreasing the useless relaxation, the approach TZBM is proposed.

In this *two-row building* problem, four relaxing parameters $\varepsilon_c, \varepsilon_b, \varepsilon_g$ and ε_h as shown in *soft restrictions* are defined to express the extent of relaxations. In the approach TZBM, all containers in $\mathbb{C}$ can be divided into three groups as follows:

- Green-zone (GZ): If an agent (container) satisfies all non-relaxed restriction equations, i.e., $\varepsilon_i = \varepsilon_{i_0}$ ($i = \{c, b, g, h\}$), it is assigned to GZ.
- Yellow-zone (YZ): If an agent (container) satisfies all relaxed restrictions equations, i.e., $\varepsilon_{i_0} < \varepsilon_i \leq \varepsilon_{i_{max}}$ ($i = \{c, b, g, h\}$), it is assigned to YZ.
- Red-zone (RZ): The rest agents (containers) violate even the relaxed restrictions and they are assigned to RZ.

A particular attention is focused on RZ. Because agents in this zone violate the relaxed restrictions, one purpose of this paper is to make RZ empty. The number of agents violating restrictions depends on the values of relaxed parameters. Usually, the minimum and maximum values of relaxed parameters are given by users. But the initial values of them can be set according to the computing time and the optimal values of them can be obtained by the algorithm. The large initial value can decrease the initial number of violating agents and decrease the computing time. In addition, the zones to be improved can be set to be RZ or RZUYZ, which also depends on the computing time. If there is no enough time, the zones to be improved is set to be RZ. In this paper, the zones to be improved is set to be RZ.

D. Improving initial solution

Firstly, how to decide the weighted parameters defining the evaluation function is very significant. In order to allow the negotiation to all containers, the last loaded container C_{N_c} is dealt with differently from others and its u_1 and u_5 (corresponding to the gravity center and the floor occupied ratio) are set to be zero by default.

Secondly, some key variables of the negotiation procedure are listed as follows:

- t : the current iteration number;
- t^* : the maximum iteration number (depending on the given computing time);
- C_γ^t : the constrained container, which is from RZ with the maximum violation and proposes the negotiation conditions;
- C_δ^t : the cooperative container, which responds to the negotiation conditions proposed by C_γ^t ;
- T_M : all modules having been tried in C_γ^t ;
- T_C : the container list whose element was identified as C_γ^t ever but could not be elevated.

After C_γ^t being identified, three kinds of negotiation operations can be attempted in order to improve C_γ^t .

“Self-adjusting”: As shown in the Fig.5, the constrained container can be improved by changing the positions of its resources (modules).

“Swapping”: As shown in the Fig.6, the constrained container can be improved by exchanging its resources (modules) with other cooperative containers.

“Inserting”: As shown in the Fig.7, the constrained con-

Fig.5 Improving solution by self-adjusting operation (topview)

Fig.6 Improving solution by swapping operation (topview)

Fig.7 Improving solution by inserting operation (topview)

tainer can be improved by inserting resources (modules) from other cooperative containers.

If C_γ^t cannot be improved by any of the negotiations, which means that C_γ^t cannot be improved in the current situation, then it is added into T_C . If there is a successful negotiation carried out between C_γ^t and C_δ^t , C_γ^t and C_δ^t are assigned to their suitable zone(s) again according to their characteristics.

Finally the core steps of the heuristic algorithm of applying DBR (**Algorithm HDBR**) are described as follows:

- **Step1:** (Initialize parameters) Let $\varepsilon_i \leftarrow \varepsilon_{i_0}$, $\Delta\varepsilon_i \leftarrow \frac{\varepsilon_{i_{max}} - \varepsilon_{i_0}}{10}$ ($i = \{c, b, g, h\}$); $t \leftarrow 0$, $T_C \leftarrow \Phi$ and $T_M \leftarrow \Phi$.
- **Step2:** (Divide into three zones) Divide all containers into three zones, GZ, YZ and RZ. If $RZ = \Phi$, go to Step9.

- **Step3:** (Identify the system's constraint) If $\mathbb{RZ} = \mathbb{T}_C$, go to *Step8*; else choose a container with the maximum value of sub-evaluation corresponding to unsatisfied soft restrictions from $\mathbb{RZ} - \mathbb{T}_C$ as the constraint to be improved.
- **Step4:** (Decide the negotiation operation) Decide one kind of unattempted negotiation operations randomly for the current C_γ^t . If it is "self-adjusting", go to *Step5*; else if it is "swapping", go to *Step6*; else it is "inserting", go to *Step7*. If these three kinds of negotiation operations have been attempted but C_γ^t can not be improved, let $\mathbb{T}_C \leftarrow \mathbb{T}_C + \{C_\gamma^t\}$ and go back to *Step3*.
- **Step5:** (Improve by "self-adjusting") If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step4*. Otherwise, choose a module m_p randomly from $C_\gamma^t - \mathbb{T}_M$ and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\gamma^t - \mathbb{T}_M$, $m_q \neq m_p$) satisfying *evaluation criteria* (1)(2) by exchanging the positions of m_p and m_q , update C_γ^t and put it into the suitable zone. If $t < t^*$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step3*; else go to *Step9*.
Otherwise in the case of that there does not exist such a module satisfying *evaluation criteria* (1)(2), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to *Step5*.
- **Step6:** (Improve by "swapping") If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step4*. Otherwise, choose a module m_p randomly from $C_\gamma^t - \mathbb{T}_M$ and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\delta^t$, $\delta \neq \gamma$, and belonging to the same group with m_p) satisfying *evaluation criteria* (1)(3)(4) by swapping m_p and m_q , let $C_\gamma^t \leftarrow C_\gamma^t + \{m_q\} - \{m_p\}$ and $C_\delta^t \leftarrow C_\delta^t + \{m_p\} - \{m_q\}$ and put C_γ^t and C_δ^t into the suitable zone(s) respectively. If $t < t^*$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step3*; else go to *Step9*.
Otherwise in the case of that there does not exist such a module satisfying *evaluation criteria* (1)(3)(4), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to *Step6*.
- **Step7:** (Improve by "inserting") If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step4*. Otherwise, choose a module m_p randomly from $C_\gamma^t - \mathbb{T}_M$ and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\delta^t$, $\delta \neq \gamma$, and belonging to the same group with m_p) satisfying *evaluation criteria* (1)(3)(4) by inserting m_q in the back position of m_p , let $C_\gamma^t \leftarrow C_\gamma^t + \{m_q\}$ and $C_\delta^t \leftarrow C_\delta^t - \{m_q\}$ and put C_γ^t and C_δ^t into the suitable zone(s) respectively. If $t < t^*$, let $\mathbb{T}_M \leftarrow \Phi$ and go to *Step3*; else go to *Step9*.
Otherwise in the case of that there does not exist such a module satisfying *evaluation criteria* (1)(3)(4), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to *Step7*.
- **Step8:** (Relax parameters) If $\varepsilon_c < \varepsilon_{c_{max}}$, let $\varepsilon_i \leftarrow \varepsilon_i + \Delta\varepsilon_i$ ($i = \{c, b, g, h\}$), $\mathbb{T}_C \leftarrow \Phi$ and go back to *Step2*. Otherwise, go to *Step9*.
- **Step9:** (Judge the end condition) In the case of no agent is in $\mathbb{RZ}$ or $\varepsilon_c \geq \varepsilon_{c_{max}}$ or $t \geq t^*$, end the negotiation

procedure.

evaluation criteria:

- (1) No hard restriction is violated;
- (2) $\max\{u_i \times E_{s_i}(C_\gamma^t)\} < \max\{u_i \times E_{s_i}(C_\gamma^{t-1})\}$; Or $\max\{u_i \times E_{s_i}(C_\gamma^t)\} = \max\{u_i \times E_{s_i}(C_\gamma^{t-1})\}$ and $E(C_\gamma^t) < E(C_\gamma^{t-1})$ ($i = 1, 2, \dots, 4$);
- (3) The size of $\mathbb{RZ}$ cannot be increased;
- (4) $\max\{u_i \times E_{s_i}(C_j^t)\} < \max\{u_i \times E_{s_i}(C_j^{t-1})\}$; Or $\max\{u_i \times E_{s_i}(C_j^t)\} = \max\{u_i \times E_{s_i}(C_j^{t-1})\}$ and $E(C_j^t) < E(C_j^{t-1})$ ($i = 1, 2, \dots, 4$, $j = \gamma, \delta$).

Finally, $\mathbb{GZ} \cup \mathbb{YZ} \cup \mathbb{RZ}$ is the final solution.

In analogizing DBR discussed in the first section, (1) the constrained container C_γ^t takes a "drum" and proposes the negotiation conditions; (2) all cooperative containers obey the negotiation conditions and try to join the negotiations with C_γ^t , so it is alike a "rope" tying from C_γ^t to them. (3) for CLP, the "buffer" is managed through controlling the parameters of relaxation and cooperative negotiations among agents. In particular, it is essential in CLP to elevate the constrained container by the cooperating behaviors in the relaxed situations. By doing so, the size of Red-zone is reduced and the buffer is curtailed.

Here we summarize the essential correspondences between DBR and our proposing algorithm for managing activities occurring in general MAS. As mentioned above, the core property of MAS might be its self-organized and/or distributed problem solving, but in reality purely bottom-up management would be difficult. Thus, if MAS works in a coordinated fashion, minimum management technique that can allow and preserve the agent's autonomous abilities will be required. For this purpose, we introduce DBR for our MAS.

The original idea of DBR in TOC was proposed as a technique for managing chained activities such as sequential manufacturing processes and more general supply chains. With respect to this, CLP formalized in MAS can be also regarded as an activity that consists of many negotiations among agents and those negotiations should be performed in some coordinated orders, i.e., not in a random order. Thus an idea of DBR can work for managing activities in MAS.

In the initial solution, the container C_γ^t violating the soft restrictions most strongly is the constraint in DBR and activities of the other cooperative containers should be synchronized with this constraint (i.e., a function of "drum"). In the proposed algorithm, the negotiation is carried out between the constrained agent and cooperative agents. In this sense, the difference of how much soft restrictions being satisfied between the best and the worst container in the initial solution can be regarded as a "rope" that restricts the activities of cooperative agents and lets them synchronized with the constraint by allowing them to negotiate with each other for the trade of modules. If a negotiation is successful, i.e., both agents accept this trade in referring to their own evaluation functions, the

Table I Statistic Properties

Problem	$N_o/N_{ct}/N_c/N_m$	$L_{av}/L_{sd}\{L_{min}/L_{max}\}$	$W_{av}/W_{sd}\{W_{min}/W_{max}\}$	$H_{av}/H_{sd}\{H_{min}/H_{max}\}$	$Wg_{av}/Wg_{sd}\{Wg_{min}/Wg_{max}\}$
1	1/28/916/296	1001/0.344{870/1070}	856/0.617{800/954}	672/2.781{280/700}	192/40.544{114/425}
2	1/142/705/206	1040/0.046{870/1144}	875/0.039{800/990}	598/1.333{166/700}	251/30.128{51/768}
3	3/33/1908/465	1103/0.041{970/1140}	927/0.457{860/984}	509/3.253{206/700}	279/16.167{202/550}
4	5/133/1533/456	1039/0.067{870/1140}	874/0.047{800/984}	621/1.865{166/700}	229/38.014{51/550}
5	1/34/2662/533	1025/0.140{870/1150}	889/0.196{800/1150}	437/0.676{206/900}	256/1.403{104/658}
6	1/37/4268/913	1085/0.229{980/1150}	979/0.274{800/1150}	449/0.621{230/700}	180/5.130{107/292}
7	3/61/649/209	1007/0.065{870/1140}	857/0.003{800/984}	672/0.515{206/700}	176/27.877{59/520}
8	2/37/8391/1015	1084/0.228{980/1150}	979/0.275{800/1150}	450/0.628{230/700}	179/4.954{107/292}
9	1/31/1536/385	1054/0.006{870/1150}	896/0.043{800/1150}	518/0.558{166/900}	242/13.998{51/658}

N_o :total number of packing orders N_{ct} :total number of cargo types N_c :total number of cargoes N_m :total number of modules

$L_{av}/L_{sd}\{L_{min}/L_{max}\}$ is the average, standard deviation, minimum and maximum of cargo lengths, etc.

dimension unit: "mm", weight unit: "kg".

violation degree of the constrained agent would be improved, while the change of the violation situation of the other is acceptable. As a result of this cooperative negotiation, MAS may find another solution satisfying a more global optimality.

Herein, the expected difficulty that those agents may encounter is to reach their consensus of exchanging the modules while sacrificing the preservation of their evaluation values. If all negotiations with the constrained agent are not successful, the problem solving will be stuck in the deadlock. In order to avoid the solving process falling down deadlock situations, a method called *restriction relaxation* used in the real-world is embedded in the proposed algorithm. By relaxing some a restriction, the solving process is changed to be flexible, more successful negotiations can be obtained and the constrained agents can be improved again. With the more restrictions being relaxed, deadlock situations are decreased in the solving process and the size of "buffer" is reduced gradually. However, the cooperative negotiation strategy can improve agents' violating states and reduce the size of "buffer" by adjusting constantly their evaluation.

IV. COMPUTATION EXPERIMENTS

The proposed heuristic algorithm has been applied to solving CLP from the real-world. Here some typical problems are listed in this paper for examining its validity.

Two kinds of optimization algorithms, **Algorithm LS** based on *Local Search* and **Algorithm TS** based on *Tabu Search*, are used to compare with **Algorithm HDBR**. For comparing easily, three algorithms use the same initial solution (generated by the algorithm for generating the initial solution), the same definition of the searching technology (all containers can participate in the negotiation if they satisfy negotiation conditions proposed by C_i^t), and the same judging criterion of successful negotiations (the first improved solution will be accepted) [7].

The core of **Algorithm LS** is to improve the current solution through searching a better solution from its neighborhood. The negotiation process can stop when the iteration reaches its maximum value (the detailed explanation is mentioned in

APPENDIX I). It is easy to run into a local optimum.

Algorithm TS is proposed to improve the disadvantage of **Algorithm LS**. A tabu list Γ_C storing searched solutions is defined to avoid repeating searches. When the iteration reaches its maximum value or T_C equals with C , the solving process is stopped (the detail is in APPENDIX II).

In this experiment, (1) t^* is initialized with 5×10^4 . (2) $u_i \leftarrow 1.0$ ($1 \leq i \leq 5$). (3) The CPU of the computer is Intel Pentium-4 2.8 GHz and its memory is 1.0 G. (4) The experiment is implemented by C++.

In Table I, some basic properties of these problems are listed. These problems have large differences in packing order, cargo type, cargo dimension and cargo weight.

Table II and Table III shows the number variations of containers in Green-zone and in Red-zone, respectively. The number is at different times during the problem solving process, (1) initial time, (2) middle time (time=70 sec) and (3) final time. It is found from Table II that in both the middle solution (at the middle time) and the final solution (at the final time), the number of containers in Green-zone for **Algorithm HDBR** is much more than those for **Algorithm LS** and **Algorithm TS** particularly in the final solution. Table III shows that in both the middle solution and the final solution, the number of containers in Red-zone for **Algorithm HDBR** is much less than those for other algorithms.

These results indicate that **Algorithm HDBR** can not only improve containers violating restrictions but also make more containers satisfy the original restrictions. So **Algorithm HDBR** is more effective for improving constrained containers violating restrictions and makes the whole system reach its goal faster than others.

Another important superiority of **Algorithm HDBR** to other two algorithms is its "anytimeness". This idea was originally developed by Dean and Bobby [8], who referred to this idea as anytime computations, and Horvitz [9], who referred to this as flexible computations. In the flexible computation, the object-related value of the result (i.e., the value of the evaluation function of the solution) returned by the algorithm is a continuous function of the time spent in computation. The

Table II Number of containers in Green - zone

Problem	Total number	Cont. number at initial time	Cont. number at time=70 sec			Cont. number at final time			
			HDBR	LS	TS	Time(sec)	HDBR	LS	TS
1	24	0	15	3	1	86	12	3	2
2	17	0	8	0	2	88	9	0	2
3	41	1	2	2	2	91	2	2	2
4	37	0	9	0	0	82	9	0	0
5	46	2	9	8	3	156	33	15	12
6	84	1	70	6	10	159	77	10	22
7	17	0	2	2	1	100	2	2	1
8	101	1	94	1	11	82	99	1	11
9	32	0	7	10	9	131	16	14	12

Cont. : container

Table III Number of containers in Red - zone

Problem	Total number	Cont. number at initial time	Cont. number at time=70 sec			Cont. number at final time				
			$(\varepsilon_i - \varepsilon_{i_0})/\Delta\varepsilon_i$	HDBR	LS	TS	$(\varepsilon_i - \varepsilon_{i_0})/\Delta\varepsilon_i$	HDBR	LS	TS
1	24	24	0	9	21	23	7	0	4	9
2	17	17	0	9	17	15	10	1	3	3
3	41	40	2	32	35	36	10	3	21	24
4	37	37	1	21	35	33	10	4	7	8
5	46	44	0	37	38	43	7	0	12	14
6	84	83	0	14	78	74	10	1	21	4
7	17	17	0	15	15	16	10	1	4	4
8	101	100	1	3	98	87	10	1	86	14
9	32	32	0	25	22	23	10	1	11	12

Table IV Maximum value and Difference of evaluation function

Problem	Initial max. evaluation	Final max. evaluation			Initial difference	Final difference		
		HDBR	LS	TS		HDBR	LS	TS
1	4.067	1.666	1.988	2.113	2.982	0.957	1.219	1.191
2	6.333	3.328	4.154	4.154	5.005	2.361	3.310	3.304
3	6.344	3.306	3.781	3.390	5.604	2.368	3.314	2.924
4	7.598	3.233	5.286	3.302	6.392	2.524	4.440	2.462
5	6.876	3.181	3.524	3.334	6.137	2.949	3.292	3.103
6	6.687	2.862	3.319	2.988	5.748	2.274	2.731	2.400
7	7.520	2.764	2.998	2.773	6.149	2.009	2.931	2.040
8	5.938	2.302	2.922	2.391	5.938	2.302	2.922	2.391
9	7.424	2.931	3.708	3.255	6.179	2.183	2.977	3.670

Maximum value of evaluation function means the one of the worst container

Difference of evaluation function means the one between the worst and best container

object-related value of a result is to be contrasted with the comprehensive value of a system's response to a given state. This means that the ideal algorithm should find a satisfactory solution as early as possible in computation, and should find an answer whose objective-related value increases as it is

allocated additional computation time. Computation can be interrupted at any point, hence we should take account both of how good the solution is good enough with respect to its object-related value and of when that solution is obtained. This is the definition of comprehensive value. For the logistics

Fig.8 Maximum value of evaluation function of containers in problem 3

Fig.9 Difference of evaluation function between the worst and best container in problem 3

problem, this characteristics of flexible computation is of great importance. The situation surrounding the real logistics problem may be changeable and dynamic due to the drifting market conditions, arrivals of unexpected express jobs, etc. Thus, CLP should be flexible. Even though the algorithm found the better solution with greater object-related value, that would not be “bounded rational” if it takes longer time to derive the solution. With respect to this, **Algorithm HDBR** is superior to the other two algorithms as explicated in the results shown in Table II and Table III.

Table IV shows the maximum value of evaluation functions of all containers and difference of evaluation functions between the worst and best containers. It is found from Table IV that by comparing the initial solution, these three algorithms have large improvement in the evaluation function of the worst container and the difference between the worst and best container. Particularly, **Algorithm HDBR** is better than others.

For more details, the variation of the evaluation function of the worst agent is shown in Fig.8 and the difference of evaluation functions between the worst and best agents is shown in Fig.9 for problem 3. It is found from both Fig.8 and Fig.9 that the results of **Algorithm HDBR** is better than others, which indicates that **Algorithm HDBR** does not only focus its attention on the constrained agent but also reduce the difference among agent so that all agents in the system

Fig.10 Variation of three zones in problem 3

cooperate each other to reach their goals, i.e., **Algorithm HDBR** is good at synchronizing agents.

The variation of zone sizes in problem 3 is shown in Fig.10. At the initial period, the agents in Red-zone are difficult to be improved under the strict restrictions. After parameters being relaxed step by step, since the restrictions are relaxed and the solving space is changed, more successful negotiations among agents are obtained and zone sizes vary a lot. The size of Red-zone decreases quickly, the size of Yellow-zone increases accordingly and Green-zone changes randomly.

As a matter of fact, **Algorithm HDBR** can reduce restriction violations greatly and synchronize all agents as much as possible. And **Algorithm HDBR** including the idea TZBM is of flexibility that a human being originally has.

V. CONCLUSIONS

The proposed algorithm in this paper is based on the Drum-Buffer-Rope and the five-step focusing process of the Theory of Constraints. It has been applied to dealing with *two-row building* of the container loading problem and the process of loading container has been discussed in this paper. The restrictions required by users have been divided into hard and soft, and the relaxing parameters have been defined for controlling the relaxing extent of the soft restrictions. Because it includes the cooperative negotiation strategy of autonomous agents and the restriction relaxation used by the human being, it can eliminate the real constraints of the system and has the flexibility like a human being.

As discussed in above sections, the proposed algorithm is good at improving the system's constraints and increasing the system efficiency because of the cooperating behaviors done by other autonomous agents. It is less frequent to trap in a deadlock because of the appropriating relaxation strategy. It can obtain a better solution at any stopping computing time because it focuses its attention on always solving the constrained agents. It has a high degree of flexibility and diversity based on its searching technology and negotiating strategy. It has been proved that the *two-row building* of the container loading problem can be solved well by the proposed algorithm.

In the real-world, there are many complicated problems whose characteristics are alike to the ones of the container loading problem. That is, they are of multiple objectives and of multiple restrictions. These restrictions are always self-contradictory and difficult to be satisfied simultaneously. And it is necessary to get the goal of the whole system. Our proposing **Algorithm HDBR** can be used to solve this kind of problems.

APPENDIX I LOCAL SEARCH ALGORITHM

Variables:

t : the current iteration number;
 t^* : the maximum iteration number;
 C_γ^t : the proposing container of the negotiation operation;
 C_δ^t : the cooperative container.

Algorithm LS is as follows:

- **Step1:** (Identify the proposing agent) Select a container randomly from $\mathbb{C}$ as C_γ^t .
- **Step2:** (Decide the negotiation operation) Decide one kind of negotiation operations randomly for C_γ^t . If it is "self-adjusting", go to *Step3*; else if it is "swapping", go to *Step4*; else if it is "inserting", go to *Step5*.
- **Step3:** (Improve by "self-adjusting") Choose a module m_p randomly from C_γ^t and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\gamma^t$ and $m_q \neq m_p$) satisfying *evaluation criteria* (1)(5) by exchanging the positions of m_p and m_q , update C_γ^t . If $t \geq t^*$, go to *Step6*; else go back to *Step1*.
- **Step4:** (Improve by "swapping") Choose a module m_p randomly from C_γ^t and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\delta^t$, $\delta \neq \eta$ and belonging to the same group with m_p) satisfying *evaluation criteria* (1)(6) by swapping m_p and m_q , let $C_\gamma^t \leftarrow C_\gamma^t + \{m_q\} - \{m_p\}$ and $C_\delta^t \leftarrow C_\delta^t + \{m_p\} - \{m_q\}$. If $t \geq t^*$, go to *Step6*; else go back to *Step1*.
- **Step5:** (Improve by "inserting") Choose a module m_p randomly from C_γ^t and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\delta^t$, $\delta \neq \eta$ and belonging to the same group with m_p) satisfying *evaluation criteria* (1)(6) by inserting m_q in the back position of m_p , let $C_\gamma^t \leftarrow C_\gamma^t + \{m_q\}$ and $C_\delta^t \leftarrow C_\delta^t - \{m_q\}$. If $t \geq t^*$, go to *Step6*; else go back to *Step1*.
- **Step6:** (Judge the end condition) In the case $t \geq t^*$, end the negotiation procedure.

evaluation criteria:

- (5) $E(C_\gamma^t) < E(C_\gamma^{t-1})$
- (6) $E(C_\gamma^t) + E(C_\delta^t) < E(C_\gamma^{t-1}) + E(C_\delta^{t-1})$

APPENDIX II TABU SEARCH ALGORITHM

Variables:

t : the current iteration number;
 t^* : the maximum iteration number;
 C_γ^t : the proposing container of the negotiation operation;
 C_δ^t : the cooperative container;
 $\mathbb{T}_M$: all modules having been tried in C_γ^t ;
 $\mathbb{T}_C$: the container list whose element was identified as C_γ^t ever but could not be elevated;
 Γ_C : the tabu list for saving last s moves ($s = 1 \times 10^3$ in this paper) ;
 ν : the experiential value to prevent too much deterioration ($\nu \leftarrow 0.03$ in this paper)

Algorithm TS is as follows:

- **Step1:** (Identify the proposing agent) If $\mathbb{C} = \mathbb{T}_C$, go to *Step6*; else select a container randomly from $\mathbb{C} - \mathbb{T}_C$ as C_γ^t .
- **Step2:** (Decide the negotiation operation.) Decide one kind of unattempted negotiation operations randomly for the current C_γ^t . If it is "self-adjusting", go to *Step3*; else if it is "swapping", go to *Step4*; else if it is "inserting", go to *Step5*. If these three kinds of negotiation operation have been attempted but C_γ^t can not be improved, let $\mathbb{T}_C \leftarrow \mathbb{T}_C + \{C_\gamma^t\}$ and go back to *Step1*.
- **Step3:** (Improve by "self-adjusting") If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step2*. Otherwise, choose a module m_p randomly from $C_\gamma^t - \mathbb{T}_M$ and let $t \leftarrow t + 1$. If there exists a module m_q ($m_q \in C_\gamma^t - \mathbb{T}_M$, $m_q \neq m_p$) satisfying *evaluation criteria* (1)(5)(7) by exchanging the positions of m_p and m_q , update C_γ^t ; Else if there is a set of modules satisfying *evaluation criteria* (1)(7)(8), choose m_q that can make C_γ^t having the minimum value of the evaluation function in the set and then update C_γ^t . In the case that there exists such a module satisfying *evaluation criteria* (1)(7)(8) at least, let $\mathbb{T}_M \leftarrow \Phi$ and $\Gamma_C \leftarrow \Gamma_C + \mathbb{C}$. If $t < t^*$, go back to *Step1*; else go to *Step6*. On the contrary (in the case of no module satisfying *evaluation criteria* (1)(7)(8)), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to *Step3*.
- **Step4:** (Improve by "swapping") If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to *Step2*. Otherwise, choose a module m_p randomly from $C_\gamma^t - \mathbb{T}_M$ and let $t \leftarrow t + 1$. If there exists a module m_q satisfying *evaluation criteria* (1)(6)(7) and by swapping m_p and m_q , update C_γ^t and C_δ^t ; Else if there is a set of modules satisfying *evaluation criteria* (1)(7)(9), choose m_q that can make C_γ^t and C_δ^t have the minimum sum value of evaluation functions in the set and then update C_γ^t and C_δ^t . In the case that there exists such a module satisfying *evaluation criteria* (1)(7)(9) at least, let $\mathbb{T}_M \leftarrow \Phi$ and $\Gamma_C \leftarrow \Gamma_C + \mathbb{C}$. If $t < t^*$, go back to *Step1*; else go

to Step6. On the contrary (in the case of no module satisfying *evaluation criteria* (1)(7)(9)), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to Step4.

- **Step5:** (Improve by “inserting”) If $C_\gamma^t = \mathbb{T}_M$, let $\mathbb{T}_M \leftarrow \Phi$ and go back to Step2. Otherwise, choose a module m_p randomly from $(C_\gamma^t - \mathbb{T}_M)$ and let $t \leftarrow t + 1$. If there exists a module m_q satisfying *evaluation criteria* (1)(6)(7) by inserting m_q in the back of m_p , update C_γ^t and C_δ^t ; Else if there is a set modules satisfying *evaluation criteria* (1)(7)(9), choose m_q that can make C_γ^t and C_δ^t have the minimum sum value of evaluation functions in the set and then update C_γ^t and C_δ^t .

In the case that there exists such a module satisfying *evaluation criteria* (1)(7)(9) at least, let $\mathbb{T}_M \leftarrow \Phi$ and $\Gamma_C \leftarrow \Gamma_C + C$. If $t < t^*$, go back to Step1; else go to Step6. On the contrary (in the case of no module satisfying *evaluation criteria* (1)(7)(9)), let $\mathbb{T}_M \leftarrow \mathbb{T}_M + \{m_p\}$ and go back to Step5.

- **Step6:** (Judge the end condition) In the case of $C = \mathbb{T}_C$ or $t \geq t^*$, the negotiation process ends.

evaluation criteria:

$$(7) C \notin \Gamma_C$$

$$(8) E(C_\gamma^t) - E(C_\gamma^{t-1}) < \nu \times E(C_\gamma^{t-1})$$

$$(9) E(C_\gamma^t) + E(C_\delta^t) - E(C_\gamma^{t-1}) - E(C_\delta^{t-1}) < \nu \times (E(C_\gamma^{t-1}) + E(C_\delta^{t-1}))$$

Finally, one solution in Γ_C , which is with the minimum sum value of evaluation functions of all containers, is the final solution.

ACKNOWLEDGMENT

The authors would like to thank an anonymous company for their giving us its real-world problems for the experiment.

References

- [1] J. Xue, K.K. Lai, *Effective Methods for a Container Packing Operation*, Mathematical and Computer Modelling, Vol. 25, No. 2, pp. 75-84 (1997)
- [2] H. Gehring, A. Bortfeldt, *A Genetic Algorithm for Solving the container Loading Problem*, International Transactions in Operational Research, Vol. 4, No. 5-6, pp. 401-418 (1999)
- [3] M. Eley, *Solving Container loading problems by block arrangement*, European Journal of Operational Research, Vol. 141, No. 2, pp. 393-409 (2002)
- [4] Y. Tian, Y. Liu, K. Shimohara, T. Sawaragi, *Cooperative Negotiation Strategy in Multi-agent System*, Proc. of the 42nd IEEE Conference on Decision and Control, pp. 2549-2554 (2003)
- [5] E.M. Goldratt, J. Cox, *The Goal: a process of ongoing improvement*, 2nd ed. The North River Press (1992)
- [6] D. Smith, *The Measurement Nightmare: How the Theory of Constraints Can Resolve Conflicting Strategies, Policies and Measures*, St. Lucie Press (2000)
- [7] T. Morton, D.W. Pentico, *Heuristic Scheduling System*, Wiley Inter science (1993)
- [8] T. Dean, M. Boddy, *An analysis of time-dependent planning*, Proc. of AAAI-88, pp. 49-54 (1988)
- [9] E.J. Horvitz, *Reasoning about beliefs and actions under computational resource constraints*, Proc. of the 1987 Workshop on Uncertainty in Artificial Intelligence (1987)

(Received May 8, 2006)

(Accepted June 23, 2006)

[Contact Address]

Graduate School of Engineering, Kyoto University
Sakyo-ku, Kyoto 606-8501, Japan

Yuan LIU

TEL : 075-753-5044

FAX : 075-753-5044

E-mail : liuyuan@eng.mbox.media.kyoto-u.ac.jp

Information about Author

Yuan LIU [non-member]

He received the M.S. degree from Dept. of Computer Science and Engineering, Beijing Institute of Technology, China, in 2003. He was a special research student of MEXT at Graduate School of Engineering of Kyoto University, Japan (Oct. 2003 - Mar. 2005). Now he is a Ph.D. student at Graduate School of Engineering of Kyoto University.

His research interests optimization algorithm, machine learning and logistics.

Yajie TIAN [non-member]

She received the M.S. degree (in Photoelectron Engineering) from Changchun Institute of Technology in 1989 and Ph.D. (in Information and Production Science) from Kyoto Institute of Technology in 1996. She was a researcher in Kyoto University from Oct. 1997 to July 2002. She was a senior researcher in Advanced Telecommunications Research Institute International (ATR). She is currently a general manager of Solution Development Dept. in Bigbang INC. She is engaged in research in modeling of systems, artificial intelligence, combinatorial optimization and their applications in the field of manufacture and physical distribution. She is a member of SICE, JORSJ and IEEE.

Tetsuo SAWARAGI [member]

Tetsuo Sawaragi was born in 1957 and is now a professor at the Dept. of Mechanical Engineering and Science, Graduate School of Engineering, Kyoto University, Japan. After obtaining B.S., M.S. degrees from Kyoto University, he obtained Ph.D. degrees from Kyoto University in 1988. He has been engaged in the researches on Systems Engineering, Cognitive Science and Artificial Intelligence, particularly in the development of human-machine collaborative systems, modeling the transfer of human cognitive skills into machines. He is a member of the Institute of Systems, Control and Information Engineers, Society of Instrumental and Control Engineers (SICE), Japanese Society for Artificial Intelligence, Human Interface Society, JSME, and IEEE.