REFERENCES

[1] A. Saffiotti, "The uses of fuzzy logic in autonomous robot navigation," *Soft Comput.*, vol. 1, no. 4, pp. 180–197, 1997.

[2] E. T. Lee, "Applying fuzzy logic to robot navigation," *Kybernetes*, vol. 24, no. 6, pp. 38–43, 1995.

[3] M. Toda *et al.*, "Navigation method for a mobile robot via sonar-based crop row mapping and fuzzy logic control," *J. Agric. Eng. Res.*, vol. 72, pp. 299–309, 1999.

[4] K. Althoefer *et al.*, "Fuzzy navigation for robotic manipulators," *Int. J. Uncertainty, Fuzziness, Knowl.-Based Syst.*, vol. 6, no. 2, pp. 179–188, 1998.

[5] J. de Lope, D. Maravall, and J. G. Zato, *Topological Modeling with Fuzzy Petri Nets for Autonomous Mobile Robots*, ser. No. 1416 in Lecture Notes in Artificial Intelligence. New York: Springer-Verlag, pp. 290–299.

[6] A. Elnagar and K. Gupta, "Motion prediction of moving objects based on autoregressive model," *IEEE Trans. Syst., Man, Cybern. A*, vol. 28, pp. 803–810, Nov. 1998.

[7] C. C. Chang and K. T. Song, "Environment prediction for a mobile robot in a dynamic environment," *IEEE Trans. Robot. Automat.*, vol. RA-13, pp. 862–872, Dec. 1997.

[8] R. Spence and S. Hutchinson, "An integrated architecture for robot motion planning and control in the presence of obstacles with unknown trajectories," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-25, pp. 100–110, Jan. 1995.

[9] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: A collision cone approach," *IEEE Trans. Syst., Man, Cybern. A*, vol. 28, pp. 562–574, Sept. 1998.

[10] J. F. Gil de Lamadrid and M. L. Gini, "Path tracking through uncharted moving obstacles," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-20, pp. 1408–1422, June 1990.

[11] A. Tsoularis and C. Kambahmpati, "On-line planning for collision avoidance on the nominal path," *J. Intell. Robot. Syst.*, vol. 21, no. 4, pp. 327–371, 1998.

[12] ——, "Avoiding moving obstacles by deviation from a mobile robot's nominal path," *Int. J. Robot. Res.*, vol. 18, no. 5, pp. 454–465, 1999.

[13] P. Garnier and T. Fraichard, "A fuzzy motion controller for a car-like vehicle," Institut National de Recherche en Informatique et en Automatique (INRIA), Tech. Rep. 3200, 1997.

[14] D. K. Pratihar, K. Deb, and A. Ghosh, "A genetic-fuzzy approach for mobile robot navigation among moving obstacles," *Int. J. Approx. Reasoning*, vol. 20, no. 2, pp. 145–172, 1999.

[15] Nomadic Technologies, Inc., "The Nomad 200 User's Guide," Nomadic Technologies, Inc., Mountain View, CA, 1997.

[16] M. Mucientes *et al.*, "Use of fuzzy temporal rules for avoidance of moving obstacles in mobile robotics," in *Proc. 1999 Eusflat-Estylf Joint Conf.*, Spain, 1999, pp. 167–170.

[17] P. Fiorini, "Robot motion planning among moving obstacles," Ph.D. dissertation, Dept. Mech., Aerospace, Nucl. Eng., Univ. California, Los Angeles, 1995.

[18] A. Bugarín *et al.*, *Fuzziness in Petri Nets*, ser. Vol. 22 of Studies in Fuzziness and Soft Computing. New York: Physica-Verlag, 1999, pp. 174–202.

Adaptive Action Selection Without Explicit Communication for Multirobot Box-Pushing

Seiji Yamada and Jun'ya Saito

***Abstract*—This paper describes a novel action selection method for multiple mobile robots box-pushing in a dynamic environment. The robots are designed to need no explicit communication and be adaptive to a dynamic environments by changing modules of behaviors. The various control methods for a multirobot system have been studied both in centralized and decentralized approaches, however, they needed explicit communication such as a radio though such communication is expensive and unstable. Furthermore, though it is a significant issue to develop adaptive action selection for a multirobot system to a dynamic environment, few studies have been done on it. Thus, we propose action selection without explicit communication for multirobot box-pushing which changes a suitable behavior set depending on a situation for adaptation to a dynamic environment. First, four situations are defined with two parameters: the existence of other robots and the task difficulty. Next, we propose an architecture of action selection which consists of a situation recognizer and sets of suitable behaviors to the situations and carefully design the suitable behaviors for each of the situations. Using the architecture, a mobile robot recognizes the current situation and activates the suitable behavior set to it. Then it acts with a behavior-based approach using the activated behaviors and can change the current situation when the environment changes. We fully implement our method on four real mobile robots and make various experiments in dynamic environments. As a result, we find out our approach is promising for designing adaptive multirobot box-pushing.**

***Index Terms*—Action selection, behavior-based robots, box-pushing, cooperation, multirobot systems.**

I. INTRODUCTION

For attacking a task which a single robot cannot achieve, many studies on multiple mobile robots cooperation have been done. They are categorized into two classes: *centralized control* [1]–[3] and *decentralized control* [4]–[11]. In the centralized control, a central system obtains global information on an environment including all the robots by sensing or communication and determines actions for all the robots. Then, the central system sends commands to all the robots and they act according to the commands. Though this approach has the advantage that robots act efficiently, it is less robust than decentralized control because all the robots stop when the central system is down. Thus, a multirobot system in decentralized control has also been investigated. However, both of the two approaches have the following significant issues.

1) *Explicit Communication*: Most multirobot systems [1]–[3], [9] using centralized control need explicit communication using a radio transmitter and a receiver. Even for decentralized control, some systems need explicit communication [12], [13]. Such explicit communication may be expensive and unstable depending on an environment. In contrast, a multirobot system without explicit communication is more robust and inexpensive.
2) *Dynamic Environment*: It is practical that an environment changes due to a fault of a robot, introduction of new robots, or task change, etc. However, most multirobot systems [1]–[11] do not have an effective mechanism to deal with a dynamic environment.

Manuscript received October 29, 1999; revised August 30, 2001. This paper was recommended by Associate Editor S. Lakshmivarahan.

The authors are with the CISS, IGSSE, Tokyo Institute of Technology, Yokohama 226-8502, Japan (e-mail: yamada@ymd.dis.titech.ac.jp).

Publisher Item Identifier S 1094-6977(01)10033-7.

To cope with the problems above, we propose a novel action selection method for multiple mobile robots box-pushing in a dynamic environment [14]. It does not need explicit communication and is adaptive to a dynamic environment in which the number of robots and task difficulty change. In this paper, at first, four situations are defined with two parameters: the existence of other robots and the task difficulty. Next we propose an action selection architecture consisting of a situation recognizer and sets of suitable behaviors to the situations. Then, we carefully design the suitable behaviors for each situation. Using the architecture, a mobile robot recognizes the current situation and activates the suitable behavior set to it. It acts under a behavior-based approach by executing the activated behaviors and can change the current situation adaptively when the environment changes. Finally, we fully implement our approach on four real mobile Khepera robots and make various experiments both in static and dynamic environments. As a result, we find out our approach is promising for designing adaptive multirobot box-pushing. Though the good results are obtained only in the multirobot box-pushing, our approach is considered applicable to other domains in a multirobot system.

In the artificial intelligence (AI) field, Sen applied reinforcement learning to multirobot box-pushing [15]. Unfortunately, they conducted experiments using only simulation and did not deal with adaptation to a dynamic environment. Controlling multiple robots has been studied in distributed AI. For example, Ohko proposed a framework in which multiple robots were controlled under contract net protocol and made a system more efficient by using case-based reasoning [16]. However, all the experiments in such studies were done by simulation.

Subsumption architecture [17] and behavior networks [18] also realized a mechanism by which a robot constantly observes an environment and selects appropriate actions depending on the environmental change. Our system defines concrete parameters describing a dynamic environment and provide concrete procedures to make a robot adaptive to the dynamic environment. Thus, we consider that our framework provides more concrete implementation than subsumption architecture and behavior networks for multirobot box-pushing.

ACTRESS [12] is a multirobot system in which a robot cannot push a box by itself; it cooperates with other robots. However, a robot requests help for other robots using explicit communication like radio. In contrast, our framework needs no explicit communication. Noreils *et al.* also realized cooperative box-pushing by multiple robots [19]. Unfortunately, their approach uses centralized control which divides a whole task into subtasks and assigns robots the subtasks.

Parker proposed an adaptive and fault-tolerant multirobot system, ALLIANCE [13]. In the ALLIANCE, each robot has plural behavior sets and activates one of them for action selection. A behavior set is activated depending on motivation which is computed by sensor input, activity of other behavior sets, and explicit communication among the robots. Since behavior sets inhibit each other, a single behavior set is activated at once. This approach is similar to our system. However, ALLIANCE allows robots to explicitly communicate on the current tasks which a robot tries to achieve. In our approach, a robot does not need to know other robots' tasks using explicit communication and realizes cooperative behavior.

II. Defining Situations to Describe a Dynamic Environment

A. Task and Environment

First of all, we describe a task and an environment. The *task of multiple mobile robots* is "to push boxes to a goal." The *environment* is a flat square table (110 cm × 90 cm) enclosed with plastic boards (Fig. 1). A light is set beside the table, and the *goal* is the nearest wall to the lamp (Fig. 1). The task is achieved when a pushed box touches the goal. In current experiments, there is no obstacle.[1]

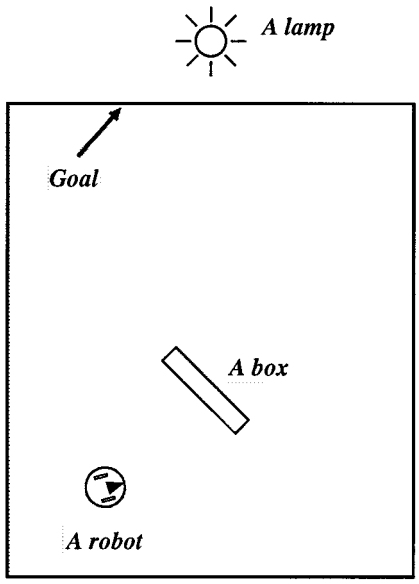


Fig. 1. Environment.

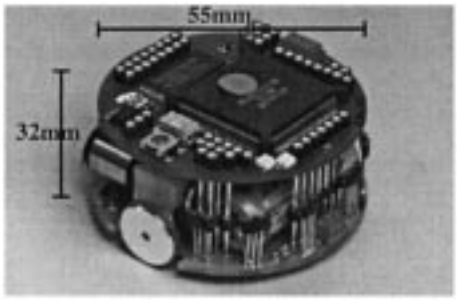


Fig. 2. Khepera.

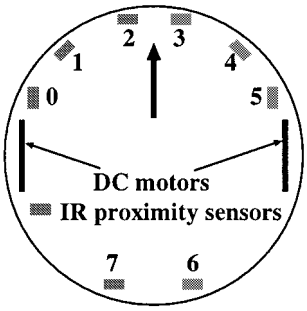


Fig. 3. Sensors.

A miniature mobile robot *Khepera*™ (Fig. 2) is used for our research. *Khepera* has a Motorola 68 331 (16 MHz) microprocessor, RAM 256 Kb, ROM 512 Kb, and is programmable by using C. As shown in Fig. 3, it has two dc motors as actuators and eight infrared proximity sensors which measure both distance to an obstacle and light strength. It also has an encoder for investigating the rotation of wheels. However, the sensor data is imprecise and local. Actually, the

[1]This fact does not restrict the results of our research because our architecture is considered independent of the existence of obstacles. To extend our framework to environments including obstacles, we have only to design suitable behaviors for obstacle avoidance.

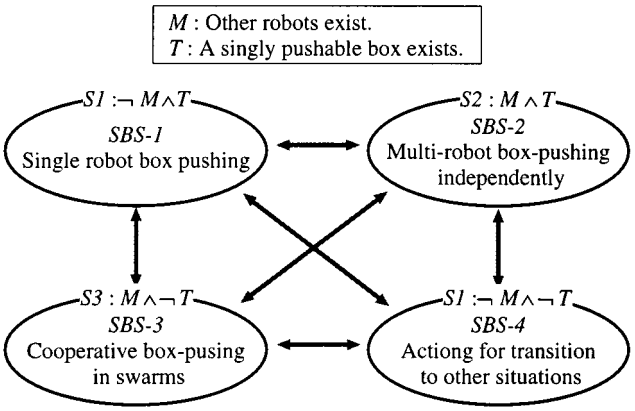


Fig. 4. Four situations and the transitions.

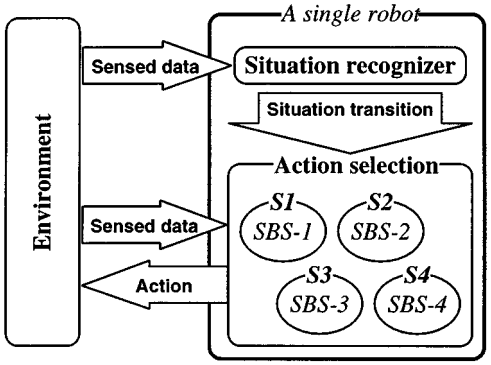


Fig. 5. Adaptive action selection architecture.

infrared proximity sensors measure distance within about 20 mm of a robot. Since a box is made of clear plastic boards, a robot can sense the light through the box. Thus, a robot can sense the direction of the goal (light) at any place in an environment.

B. Assumptions on an Environment

For realizing appropriate action selection without explicit communication and with local information, we need assumptions on an environment. We use the following assumptions, and these are actually held in all the experiments. These assumptions are used to recognize the current situation of a robot.

AS-1: There is no moving object except robots.

AS-2: When a robot tries to push an object (like a wall or a heavy box) which can not be moved, its wheels do not rotate. In other words, a robot does not skid.

C. Defining Situations

For describing a dynamic environment in multirobot box-pushing, we introduce two parameters: *the existence of other robots* and *the task difficulty*. The existence of other robots means whether other robots exist in a environment or not, and the task difficulty means whether there is a heavy box which a robot cannot push by itself or not. Using the parameters, we can describe a large part of the change in general dynamic environments, e.g., some robots stop by breakdown, some robots are added to or removed from an environment, boxes which are too heavy for single-robot pushing are added to or removed from an environment, etc.

We describe the existence of other robots and task difficulty with proposition M and T, respectively. M means that another robot is recognized and $\neg M$ means that it is not recognized. T means that a heavy box which a single robot cannot push is not recognized and $\neg T$ means that such a box is recognized. Thus, using the conjunctions of the propositions, four classes $M \wedge T$, $\neg M \wedge T$, $M \wedge \neg T$, and $\neg M \wedge \neg T$ of dynamic environments are described and we call them *situations*. Next, we describe suitable behaviors for each situation.

In the following, we explain the situations and the suitable behaviors for each of them. The suitable behaviors will be concretely described by IF–THEN rule representation later. Note that each robot determines its own situation by itself without explicit communication on a situation with other robots. Thus, the determined situation may be globally incorrect. Fig. 4 also shows the four situations, transitions between them, and suitable behaviors for each situation (*SBS*) which will be explained later.

- $S1 = \neg M \wedge T$ (*A single robot and easy task*): Since a robot can push a box by itself it achieves the task by itself.
- $S2 = M \wedge T$ (*Multiple robots and easy task*): Each robot pushes a box independently. When there are multiple boxes, independent box-pushing by each robot is efficient.
- $S3 = M \wedge \neg T$ (*Multiple robots and hard task*): Since a robot cannot push a box by itself, multiple robots swarm and push a box cooperatively.
- $S4 = \neg M \wedge \neg T$ (*A single robot and hard task*): The task is not achieved as long as it is in this situation. As mentioned above, though other robots or a light box exist, the robot may not find it yet. Also, as time passes, the current situation $S4$ may change into another situation. Thus, a robot wonders whether to search for them or not. When a robot finds them, its situation changes to one of $S1$, $S2$, or $S3$.

D. Action Selection Architecture

Every robot is homogeneously designed using an architecture in Fig. 5. The situation recognizer constantly monitors data from sensors and determines the current situation. Then, it activates a suitable SBS to the current situation and a robot acts using the activated SBS.

E. Recognizing a Situation

For adaptation to a dynamic environment, a robot recognizes a current situation and selects a suitable SBS by itself. Thus, the situation recognizer of a robot constantly monitors the following conditions for determining M or $\neg M$ and T or $\neg T$. A situation recognizer is implemented using procedural programming, not IF–THEN rules, which will be used for describing behaviors.

- Checking M: A situation recognizer investigates the change of sensor data when a robot stops. If the change occurs, other robots exist in the environment and M becomes true. This uses *AS-1* in Section II-A.
- Checking $\neg M$: $\neg M$ becomes true if M does not become true within a certain time t_M after M became true last.
- Checking T: When a robot tries to push an object and its wheels are rotated, there is a box which a robot can push by itself. Then T becomes true. This uses *AS-2* in Section II-A.
- Checking $\neg T$: When a robot continuously collides with objects, which it cannot move more than t_T times, $\neg T$ becomes true.

III. Situated Behavior Sets

We apply a behavior-based approach [17] to control a mobile robot. Though a behavior-based method cannot control a robot precisely, it does not need a strict model of a robot and an environment. Actually we have verified the utility experimentally in our other research [20]. A *behavior* is described as a rule: IF a *state* THEN an *action*, where the state is directly determined by sensed data, not an internal state, and the action consists of executable primitive motor commands. We

design a set of behaviors for each situation, and such a set is called a SBS. Each SBS is explained in the following, where SBS-i means a situated behavior set for a situation Si.

A. Describing States and Actions

We describe a state as a proposition, thus logical calculus is applied to it. For describing states, the sensors are classified into some classes depending on directions. The *forward-sensors* and *back-sensors* stand for sensor 1 $\sim$ 4 and sensor 6 and 7 in Fig. 3, respectively. The *left-sensor* and *right-sensor* indicate sensor 0 and sensor 5 in Fig. 3. The following states and actions are defined using the classified sensors. Note that no explicit communication is utilized for executing behaviors.

1) States:

a) *Forward/back/left/right-object*: An object within 20 mm of a robot is sensed with forward/back/left/right-sensors.[2]
b) *Forward/back/left/right-light*: The forward/back/left/right-sensors have the maximum light value.
c) *No-light*: The light values in all the directions are almost the same.
d) *No-rotation*: Though motors are commanded to drive, no rotation is sensed by an encoder.

2) Actions:

a) *Direction-change*: A robot turns 180°.
b) *Push-clockwise/counterclockwise*: A robot rotates a box clockwise/counterclockwise by pushing.
c) *Push-straight*: A robot pushes a box straight.
d) *Turn-left/right*: A robot turns left or right on the spot.
e) *Go-ahead*: A robot goes ahead.
f) *Stop*: A robot stops.

Using conjunction of states, we can describe a IF part of a behavior, and the THEN part consists of a single action. In the following subsections, we will describe the suitable SBSs for each situations.

Though each of the SBSs is mutually independent, plural behaviors in the same SBS may be applicable to the current state and the behaviors may conflict with one another. Using the following criterion, we resolve the conflict and select a single behavior to execute.

1) Prefer an applicable behavior with a larger number of propositions in the IF part.
2) Prefer an applicable behavior with a younger ID number behavior.

Note there is no guarantee that the SBSs cover all the states in an environment. When no behavior is applicable, a robot just goes straight. This behavior is applied to all the situations.

B. SBS-1: Single Robot Box-Pushing

In $S1$, the following behaviors are used for a single robot to push a box to a goal. Fig. 6 shows the executions of `B-3`$\sim$`B-5`.

`B-1` IF $\neg$*forward-object* $\wedge$ $\neg$*left-object* $\wedge$ $\neg$*right-object* THEN *go-ahead*. (Go forward when no object in front, left and right of a robot.)

`B-2` IF *forward-object*$\wedge$*no-rotation* THEN *direction-change*. (When an object exists in front of a robot and no wheel rotation, change the direction by turning 180°. This behavior is executed when a robot collides with a wall.)

`B-3` IF *forward-object*$\wedge$*left-light* THEN *push-clockwise*. [When an object is in front of a robot and a light is in left, pushes the object to turn clockwise. Fig. 6(a) shows the action.]

[2]Khepera's infrared rays (IR) proximity sensors can sense an object within about 20 mm of it.

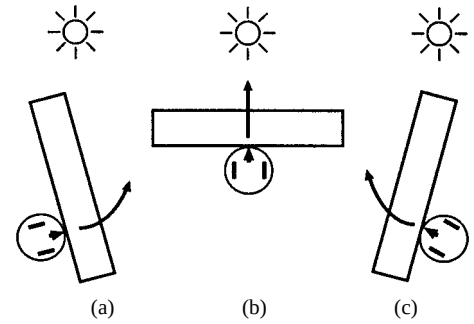


Fig. 6. Executions of `B-3`$\sim$`B-5`.

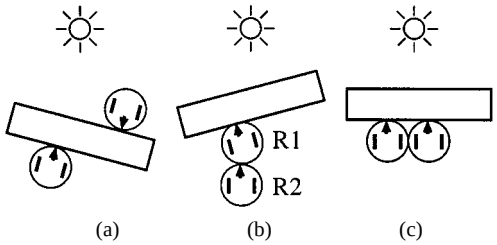


Fig. 7. Harmful interactions.

`B-4` IF *forward-object*$\wedge$*forward-light* THEN *push-straight*. [When an object and a light are in front of a robot, pushes the object straight. Fig. 6(b) shows the action.]

`B-5` IF *forward-object*$\wedge$*right-light* THEN *push-counterclockwise*. [When an object is in front of a robot and a light is in right, pushes the object to turn counterclockwise. Fig. 6(c) shows the action.]

C. SBS-2: Distributed Box-Pushing

SBS-2 for $S2$ is almost similar to *SBS-1*. However, we need to deal with interaction among robots. Through experiments in which *SBS-1* is straightforward applied to $S2$, we found harmful interaction between robots shown in Fig. 7. Fig. 7(a) shows that two robots push the same box in opposite sides. Thus both robots stop, recognize the box as a wall, and go away by executing `B-2`. Fig. 7(b) shows that a robot R2 pushes another robot R1. This case is less efficient than a case that both of them push a box directly. Fig. 7(c) shows that two robots pushing a box touch together. This case often causes the interaction in Fig. 7(b).

For avoiding such harmful interactions, we add the following behaviors to *SBS-1*, and construct *SBS-2* with `B-1`$\sim$`B-9`. Using `B-6` for a case in Fig. 7(a), a robot with its back to a goal changes its direction and go away, thus another robot facing a goal can push a box. Using `B-7` for a case in Fig. 7(b), a robot R1 stops when an object is sensed in its back, and a robot R2 goes away by executing `B-2` because it recognizes the robot R1 as a wall. Using `B-8` and `B-9` for a case in Fig. 7(c), a robot turns to the opposite direction a little and separates from another robot when an object is sensed in the left or right side. These behaviors are inspired by behaviors for simulating a flock of birds [21].

`B-6` IF *forward-object* $\wedge$ *back-light* THEN *direction-change*. (A robot turns 180° when an object exists in the front of it and a light exists in the rear of it.)

`B-7` IF *forward-object* $\wedge$ *back-object* THEN *stop*. (A robot stops when objects exist both in the front and the rear of it.)

`B-8` IF *forward-object* $\wedge$ *forward-light* $\wedge$ *left-object* THEN *turn-right*. (A robot turns right when an object and a light exist in the front of it and an object exists in the left side.)

`B-9` IF *forward-object* $\wedge$ *forward-light* $\wedge$ *right-object* THEN *turn-left*. (A robot turns left when an object and a light exist in the front of it and an object exists in the right side.)

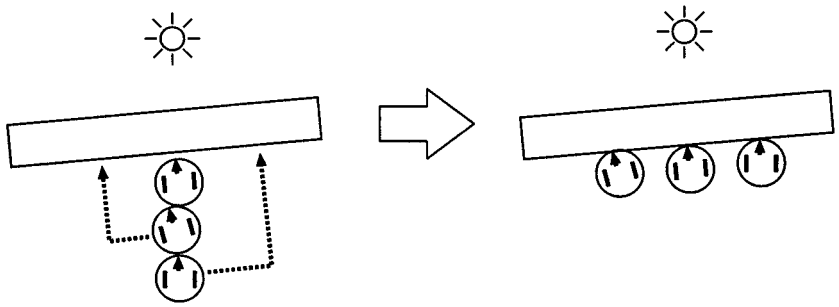


Fig. 8. Cooperative box-pushing.

In $S2$, boxes which are pushed by robots may collide with each other. In that case, the robots continue to push boxes, if possible. If impossible, the robots change the direction and go away by executing B-2. However, the robots which have collided with an object that they can not push less than t_T times does not negate a proposition T and goes away to find a box which is pushable by a single robot.

D. SBS-3: Box-Pushing in Swarms

In $S3$, since robots cannot move a box by itself, they need to swarm for pushing a box cooperatively. In general, a swarm forms various shapes: a line, a circle, a arrow, etc. We use a line so that a robot can avoid harmful interaction. Behaviors for swarming are somewhat complex because a robot needs to recognize other robots. Hence we introduce additional states: *forward/right/left/back-robot*, *forward/back-robot-leaving*, and actions: *following*, *side&push*.

1) States:

a) *Forward/right/left/back-robot*: Another robot is sensed forward/right/left/back, and is determined by a procedure for checking M in Section II-E.

b) *Forward/back-robot-leaving*: A robot does not sense another robot which has been sensed forward/back of it.

2) Actions:

a) *Following*: A robot moves to the direction in which another robot was sensed or left.

b) *Side&push*: Aligned robots move to side and pushes a box cooperatively like Fig. 8.

By using above additional states and actions, *SBS-3* consists of the following three parts.

Swarming: For swarming, a robot wanders until it finds other robots and follows others. By adding the following behavior to *SBS-1*, a robot does such actions.

B-10 IF *left-robot* $\vee$ *right-robot* THEN *following*. (A robot follows an object which is sensed in the left or the right of it.)

Keeping a Line: For keeping linear shape, suitable behaviors for a head and not-head robots of the line are designed in the following. Basically, a head robot goes ahead when the back robot is sensed in the back, and stops when no robot is sensed in its back. When no robot is sensed in the back of a head robot, the back robot often misses the head robot, and the head robot needs to wait for the back robot to approach. A not-head robot follows when a head robot disappears and stops when the head robot is sensed.

For a Head Robot:

B-11 IF $\neg$*forward-robot* $\wedge$ *back-robot* THEN *go-ahead*. (Go forward when no robot is sensed in the front and a robot is sensed in the rear.)

B-12 IF $\neg$*forward-robot* $\wedge$ *back-robot-leaving* THEN *stop*. (Stop when no robot is sensed in the front and the back robot becomes imperceptible.)

For Not-Head Robots:

B-13 IF *forward-robot-leaving* THEN *following*. (When the front robot becomes imperceptible, a robot follows it.)

B-14 IF *forward-robot* THEN *stop*. (Stop when a robot is sensed in the front.)

3) Box-Pushing in a Swarm: When aligned robots find a box, they need to push the box cooperatively as seen in Fig. 8. Furthermore, they need to leave in a swarm when they encounter a wall. These actions are done using the following behaviors.

For a Head Robot:

B-15 IF *forward-object* $\wedge$ *back-robot* $\wedge$ *forward-light* THEN *push-straight*. (Push a box straight when an object and a light exist in the front and another robot is sensed in the rear.)

B-16 IF *forward-object* $\wedge$ *back-robot-leaving* THEN *following*. (Follow the robot when an object exists in the front and the back robot becomes imperceptible.)

For Not-Head Robots:

B-17 IF *forward-robot* $\wedge$ *back-robot-leaving* THEN *following*. (When another robot is sensed in the front and the back robot becomes imperceptible, follow the back robot.)

B-18 IF *forward-robot* $\wedge$ $\neg$*forward-light* THEN *direction-change*. (Turn 180° when another robot is sensed in the front and no light is sensed in the front.)

B-19 IF *forward-robot* $\wedge$ *forward-light* THEN *side&push*. (Do side&push when a robot and a light are sensed in the front.)

Finally *SBS-3* consists of *SBS-1* and B-10$\sim$B-19 Note that, in $S3$, a robot does not need to recognize weather it is head or not in a line, and just executes an applicable behavior in *SBS-3*. Though we designed behaviors for a head robot and a not-head robot above, they are not distinguished in *SBS-3*.

E. SBS-4: Acting for Transition

In *S-4*, since a robot finds neither other robots nor a box which it can move by itself, a box-pushing task can not be achieved. However, though there are multiple robots or a box which a single robot can move, the robot may only fail to find them. Also, as time passes, environment change like adding robots or light boxes may transit the situation *S-4* into other situations. Hence we make a robot wander using *SBS-1* until other robots or a light box is found.

IV. Experiments With Multiple Mobile Robots

We fully implemented the adaptive action selection method on each of four *Khepera*s using C programming language. The programming was done in a host computer PC and the program was downloaded into each Khepera through RS-232C. After downloading, a mobile robot autonomously acts without communication with a host computer and other robots.

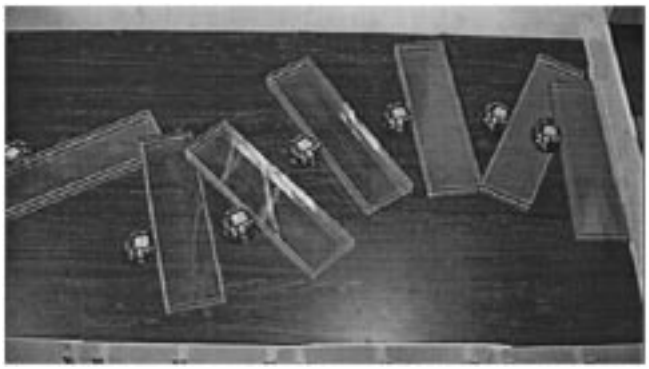


Fig. 9. Trajectory of actions in $S1$.

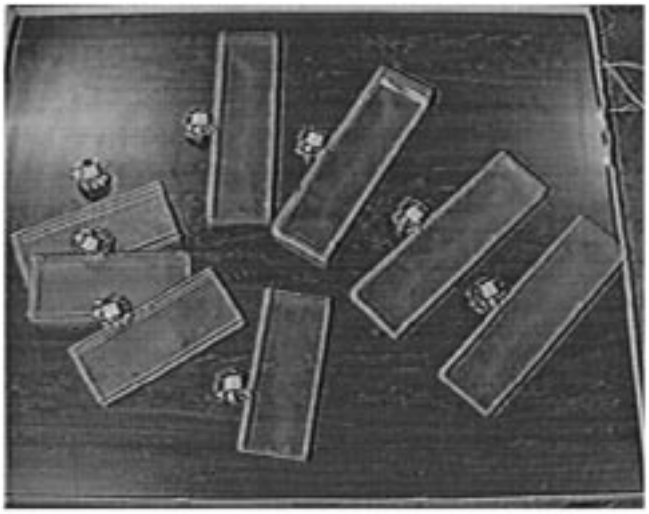


Fig. 10. Trajectory of actions in $S2$.

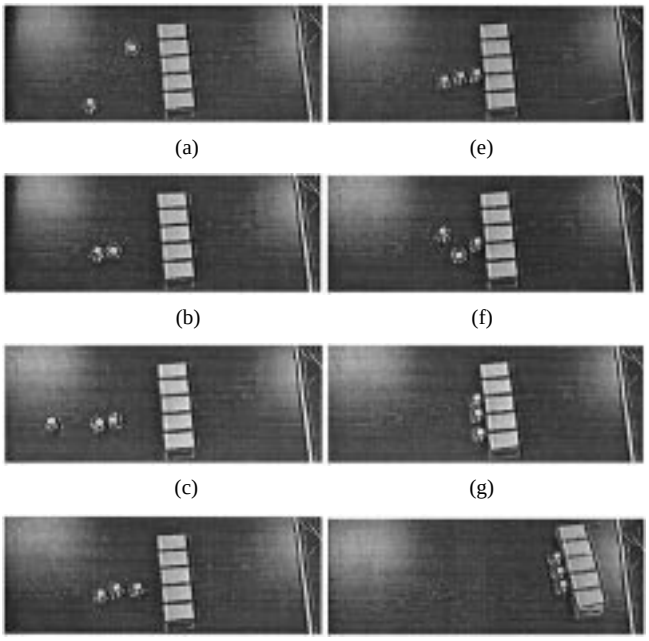


Fig. 11. Box-pushing in $S3$.

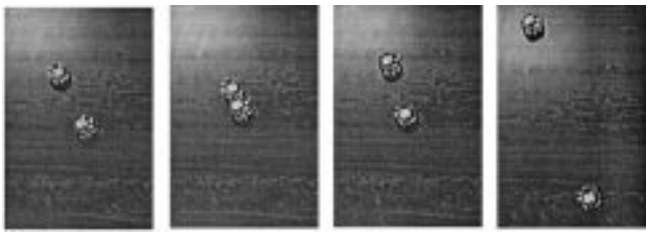


Fig. 12. Actions after encounter in $S2$.

The parameters t_M and t_T in Section II-E are set 300 s and 10 times, respectively. In all experiments, the goal is the right wall. Thus a robot tries to move a box to the right wall. The cycle of action selection including time for executing an action is 100 ms.

For investigating the utility of our approach, we made experiments in various environments. First the experiments were made in static environments without the change of situations. Next we made experiments in environments where a situation changed.

As a result, the probability that the robots achieve the task was about 80% in each situation. We investigated 30 random initial positions for each situation.

A. Results in Static Situations

1) Experiments in $S1$ and $S2$: We set a box which a robot can push by itself and a single robot in an environment, and ran a robot. This environment corresponds to $S1$ globally. However a robot initially recognized the current situation as $S4$ and wandered. Then it found a box, and the current situation was successfully changed to $S1$. By using *SBS-1*, a robot pushed a box to a goal. Fig. 9 shows the trajectory of the actions. From seeing this figure, we verified that a robot worked well in $S1$.

In $S2$ where two robots and two light boxes were set, we observed that each robot independently pushed a box as well as in $S1$ (Fig. 10).

2) Experiments in $S3$ and $S4$: We set a box which a robot cannot push by itself and four robots for $S3$. As time passes, the four robots independently recognized the current situation was $S3$ and tried to swarm by using *SBS-3*. Fig. 11 shows the trajectory of such actions. The robots succeeded in swarming [Fig. 11(a)–(d)] and executing the *side&push* action [Fig. 11(e)–(h)].

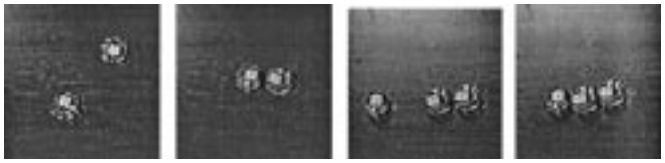
Next $S4$ was set with a heavy box which a robot cannot push by itself and a single robot. Then we observed that the robot wanders to search for other robots or a light box.

B. Adaptation to a Dynamic Environment

By adding and removing robots and heavy boxes which a robot cannot push by itself, we changed the situation and observed actions of robots. As a result, for all the changes between arbitrary two situations in $S1, \ldots, S4$, we verified situation transition was independently done in each robot, and suitable SBSs were eventually activated. When multiple robots act in the same environment, each of the situation transitions in them occurred asynchronously, and all the robots presently converged to the same situation.

For example, Fig. 12 shows actions before and after two robots encountered in an environment where no heavy box exists. They recognized that the current situation was $S2$, and SBS-2 was activated. Thus they left mutually after they encountered.

Then we added a heavy box into the environment. The robots found the heavy box eventually, and changed the current situation to $S3$. SBS-3 was activated in the two robots, and they acted in a swam. Fig. 13 shows actions after the robots encountered in such a situation. They swarmed after the encounter and did not leave mutually.



(a) (b) (c) (d)

Fig. 13. Actions after encounter in $S3$.

Through all the experiments, we did not observe that a robot applied no behavior. Thus we consider our design of situations and behaviors as valid.

V. Discussion

Though we experimentally verified the utility of our approach, there are still the following open problems. In this section, we discuss the problems.

A. Assumptions on an Environment

We use some assumptions on an environment: *AS-1*, and *AS-2* in Section II-A. If these assumptions are not held, our multirobot system may not work well. Furthermore, when an environment is very large or is not closed, multiple robots may not swarm because they hardly encounter only by wandering in such an environment. Currently we assume there is no obstacle in an environment. We consider our system can deal with obstacles by modifying behaviors.

B. Scalability

Due to physical constraints, we did not make the experiments using n robots ($n \geq 5$). We consider the SBSs and the behaviors defined above are easily applied to such environments. However if the number of robots increases more than several tens, our approach may not be applied straightforward.

C. Task Achievement

As mentioned earlier, our method does not guarantee that the box-pushing is achieved in all environments. The possibility of achievement depends on various environmental conditions such as the initial positions of robots and boxes and the ability of a robot. We show typical cases in which the box-pushing cannot be achieved in the following.

1) *A box in the corner*: Since a robot cannot pull a box in our experimental environments, it cannot handle a box in the corner.
2) *Incomplete search for other robots and light boxes*: When a robot recognizes the current situation as $S4$ and the global situation is one of $S1B!(BS4B!(BS3$, it needs to accidentally find other robots or light boxes for recognizing the correct situation. However, since the search is incomplete, we have no guarantee that a robot recognizes the global situation.

VI. Conclusion

To overcome the disadvantages of traditional multirobot systems,we proposed novel action selection without explicit communication for multirobot box-pushing. First, for describing dynamic environments, we defined situations with two parameters: existence of other robots and task difficulty. Next we designed architecture consisting of a situation recognizer and sets of suitable behaviors for each of the situations. Using the architecture, a robot asynchronously recognizes the current situation, activates suitable behaviors, and executes them. Then a robot is able to select and execute valid actions even in a dynamic environment without explicit communication with other robots. We fully implemented our approach on four real mobile robots and verified the utility experimentally.

References

[1] H. Sugie *et al.*, "Placing objects with multiple mobile robots—Mutual help using intention inference," in *Proc IEEE Int. Conf. Robot. Automat.*, 1995, pp. 2181–2186.
[2] N. Miyata *et al.*, "Cooperative transport with regrasping of torque-limited mobile robots," in *Proc IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, pp. 304–309.
[3] Z. Wang, E. Nakano, and T. Matsukawa, "Realizing cooperative object manipulation using multiple behavior-based robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, pp. 310–317.
[4] D. J. Stilwell and J. S. Bay, "Toward the development of a material transport system using swarms of ant-like robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1995, pp. 766–771.
[5] R. Beckers, O. E. Holland, and J. L. Deneubourg, "From local actions to global tasks: Stigmergy and collective robotics," in *Artificial Life IV*, 1994, pp. 181–189.
[6] M. J. Mataric, M. Nilson, and K. T. Simsarian, "Cooperative multi-robot box-pushing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1995, pp. 556–561.
[7] M. J. Mataric, "Learning in multi-robot systems," in *Adaption and Learning in Multi-Agent Systems*, G. Weiband and S. Sen, Eds. New York: Springer-Verlag, 1996, pp. 152–163.
[8] C. R. Kube and H. Zhang, "The use of perceptual cues in multi-robot box-pushing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1996, pp. 2085–2090.
[9] K. Kosuge and T. Osumi, "Decentralized control of multiple robots handling and object," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, pp. 318–323.
[10] H. Osumi, "Cooperative strategy for multiple mobile manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1996, pp. 554–559.
[11] Y. Aiyama *et al.*, "Cooperative transportation by two 4-legged robots with implicit communication," in *Proc. Fourth Int. Symp. Distributed Autonom. Robot. Syst.*, 1998.
[12] H. Asama, A. Matsumoto, and T. Ishida, "Design of an autonomous and distributed robot system," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots Syst.*, 1989, pp. 283–290.
[13] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 220–240, Apr. 1998.
[14] S. Yamada and J. Saito, "Adaptive action selection without explicit communication for multi-robot box-pushing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1999, pp. 1444–1449.
[15] S. Sen, M. Sekaran, and J. Hale, "Learning to coordinate without sharing information," in *Proc. 12th Nat. Conf. Artif. Intell.*, 1994, pp. 426–431.
[16] T. Ohko, K. Hikaki, and Y. Anzai, "Learning to reduce communication cost on task negotiation among multiple autonomous mobile robots," in *Adaption and Learning in Multi-Agent Systems*. New York: Springer-Verlag, 1996, pp. 177–190.
[17] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Trans. Robot. Automat.*, vol. 2, pp. 14–23, Feb. 1986.
[18] P. Maes, "Learning behavior networks from experience," in *Proc. First Euro. Conf. Artif. Life*, 1991, pp. 48–57.
[19] F. R. Noreils and R. de Nozay, "An architecture for cooperation and autonomous mobile robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1992, pp. 2703–2710.
[20] S. Yamada and M. Murota, "Unsupervised learning to recognize environments from behavior sequences in a mobile robot," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1998, pp. 1871–1876.
[21] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," *ACM Comput. Graph.*, vol. 21, no. 4, pp. 25–34, 1987.