Adaptive Action Selection without Explicit Communication for Multi-robot Box-pushing

Seiji Yamada Jun'ya Saito

CISS, IGSSE, Tokyo Institute of Technology
4259 Nagatsuta, Midori, Yokohama 226-8502, JAPAN

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. Though it is a significant problem to deal with adaptive action selection for multiple mobile-robots in a dynamic environment, few studies have been done. Decentralized control of robots without explicit communication is also practical and important for robustness. Thus we propose adaptive action selection without explicit communication for multi-robot box-pushing, which changes an available behavior set depending on a situation. First four situations are defined with two parameters: existence of other robots and task difficulty. Next we design a set of behaviors for each situations, and mobile robots are programmed to act with behavior-based approach. We fully implement our method on four real mobile robots, and make experiments in dynamic environments.

1 Introduction

For attacking a task which a single robot can not achieve, many studies on multiple mobile robots cooperation have been done. They are categorized into two classes: *centralized control* [11][6][12] and *decentralized control* [10][5][4][3][2][7]. In 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 stops when the central system is down. Thus the multi-robot system in decentralized control have also been investigated. However both of the two approach have the following significant problems.

1. *Explicit communication*: Most multi-robot systems [11][6][12][2][8] in centralized control need explicit communication using a transmitter and a receiver. Since such communication may be expensive and unstable depending on an environment, a multi-robot system without explicit communication is more robust and inexpensive.
2. *A dynamic environment* : It is practical that an environment changes due to a fault of a robot, introduce of new robots, task change, etc. However most multi-robot systems [10][5][11][6][12][2][7][4][3] does not have an effective mechanism to deal with a dynamic environment.

To cope with the problems above, we propose a novel action selection method for multiple mobile robots box-pushing in a dynamic environment. 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, first four situations are defined with two parameters: existence of other robots and task difficulty. Next we design a set of behaviors for each situations, and mobile robots are programmed to act by behavior-based approach. We fully implement our method on the four real mobile robots, and make experiments in dynamic environments.

2 Defining situations to describe a dynamic environment

2.1 A task and an 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 (110cm×90cm) enclosed with white plastic boards (Fig.1). A lamp is set beside the table, and the *goal* is the nearest wall to the lamp (Fig.1). The task is

achieved when a box touches the goal. In current experiments, there is no obstacle. A miniature mobile robot *Khepera*™ (Fig.2) is used for our research. As shown in Fig.3, it has two DC motors as actuators and eight Infra-Red 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. Since a box is made of clear plastic boards, a robot can sense the light through the box. A robot can sense the direction of the goal (lamp) at any place in an environment.

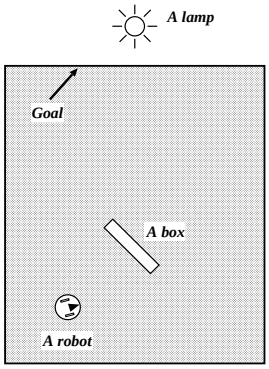

Figure 1 Environment

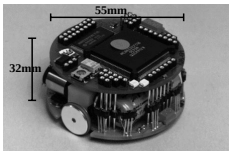

Figure 2 Khepera

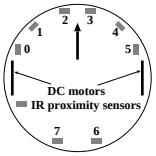

Figure 3 Sensors

We use the following assumptions for an environment, and these are actually held in all the experiments.

AS-1 There is no moving object except a robot.

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

2.2 Defining situations

For describing a dynamic environment in multi-robot box-pushing, we use two parameters: *the existence of other robots* and *task difficulty*. The existence of other robots means whether other robots exist in a environment, and task difficulty means whether there is a heavy box which a single robot can not push. 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 into or removed from the environment, too heavy boxes for single-robot pushing are added into or removed from the environment, etc. We describe the the existence of other robots and task difficulty with atomic formula M and T respectively. M means that another robot is observed, and $\neg M$ means that it is not observed. T means that a heavy box which a single robot can not push is not observed, and $\neg T$ means that there such a box is observed. Thus, using the conjunctions of the atoms, four classes $\{M \wedge T, \neg M \wedge T, M \wedge \neg T, \neg M \wedge \neg T\}$ of dynamic environments are described, and we call them *situation*s. The following explains the situations and suitable behaviors in them. 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.

- $S1 = \neg M \wedge T$ (*A single robot and easy task*) : Since a robot can push a box by itself, it achieves the task singly.
- $S2 = M \wedge T$ (*Multiple robots and easy task*) : Each robot pushes a box independently.
- $S3 = M \wedge \neg T$ (*Multiple robots and hard task*) : Since a robot can not push a box singly, robots push a box cooperatively.
- $S4 = \neg M \wedge \neg T$ (*A single robot and hard task*) : The task is not achieved as long as be in this situation $S4$. As mentioned above, other robots or a light box may globally exist. Thus a robot wonders to search for them. When a robot finds them, its situation changes to $S1 \sim S3$.

2.3 Architecture

Every robot is homogeneously designed using an architecture in Fig.4. The situation recognizer constantly monitors data from sensors, and determines the current situation. Then it activates a suitable *SBS* (*situated behavior set*, mentioned in the next section) to the current situation, and a robot acts using the activated *SBS*.

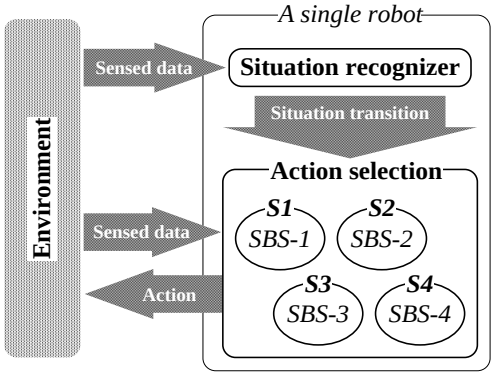

Figure 4 Architecture

2.4 Recognizing a situation and the situation transition

For adaptation to a dynamic environment, a robot recognizes a current situation and change 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 $\neg T$.

- 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 § 2.1.
- 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 wholes are rotated, there is a box which a single robot can push. Then T becomes true. This uses *AS-2* in § 2.1.
- Checking $\neg T$: When a robot continuously collides with objects, which it can not move, more than T_t times, $\neg T$ becomes true.

3 Situated behavior sets

We apply a behavior-based method [1] to control a mobile robot. Though a behavior-based method can not control a robot precisely, it does not need a strict model of a robot and an environment. A *behavior* is a rule: *If a state then an action*, where the state (not an internal state) is directly determined by sensed data, and the action is primitive. We design a set of behaviors for each situation, and such a set is called a *SBS* (*situated behavior set*). Each *SBS* is explained in the following, where *SBS-i* means a situated behavior set for a situation *Si*.

3.1 Describing states and actions

Directions used to describe states are defined. The *forward-sensors* and *back-sensors* are sensor 1∼4 and sensor 6, 7 in Fig.3, respectively. The *left-sensor* and *right-sensor* are sensor 0 and sensor 5 in Fig.3. The following states and actions are defined. Note that no explicit communication is utilized fro executing behaviors.

States

- *forward/back/left/right-object*: An object within 20 mm from a robot is sensed with forward/back /left/right-sensors.
- *forward/back/left/right-light*: The forward/back /left/right-sensors have the maximum light value.
- *no-light*: The light values in all the directions are almost same.
- *no-rotation*: Though motors are commanded to drive, they are sensed not to rotate by an encoder.

Actions

- *direction-change*: A robot turns 180°.
- *push-clockwise/counterclockwise*: A robot rotates a box clockwise/counterclockwise by pushing.
- *push-straight*: A robot pushes a box straight.
- *turn-left/right*: A robot turns left or right.
- *go-ahead*: A robot goes ahead.
- *stop*: A robot stops.

3.2 SBS-1: A single robot box-pushing

In *S1*, the following behaviors are used for a single robot to push a box to a goal[1] . Fig.5 shows the executions of B-3∼B-5.

B-1 If $\neg$ *forward-object* $\wedge$ $\neg$ *left-object* $\wedge$ $\neg$ *right-object* then *go-ahead*.

B-2 If *forward-object* $\wedge$ *no-rotation* then *direction-change*. (This is executed when a robot collides with a wall.)

B-3 If *forward-object* $\wedge$ *left-light* then *push-clockwise*. (Fig.5(a) shows the action.)

B-4 If *forward-object* $\wedge$ *forward-light* then *push-straight*. (Fig.5(b) shows the action.)

B-5 If *forward-object* $\wedge$ *right-light* then *push-counterclockwise*. (Fig.5(c) shows the action.)

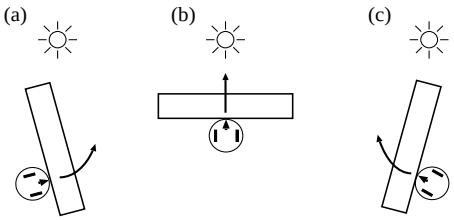

Figure 5 Execution of B-3 ∼ B-5

3.3 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.6. Fig.6(a) shows that two robots push the same box in opposite sides. Thus

[1] Though each *SBS*s are mutually independent, plural behaviors may conflict in the same *SBS*. Thus we uses the two conflict resolution criteria: (1) more specific (more conditions) behavior is preferred, (2) the behavior with a younger number is preferred.

both robots stop, consider the box a wall, and go away. Fig.6(b) shows that a robot pushes another robot. This case is less efficient than a case that both of them push a box. Fig.6(c) shows that two robots pushing a box touch together. This case often causes the Fig.6(b).

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

B-6 If *forward-object* ∧ *back-light* then *direction-change*.
B-7 If *forward-object* ∧ *back-object* then *stop*.
B-8 If *forward-light* ∧ *forward-object* ∧ *left-object* then *turn-right*.
B-9 If *forward-light* ∧ *forward-object* ∧ *right-object* then *turn-left*.

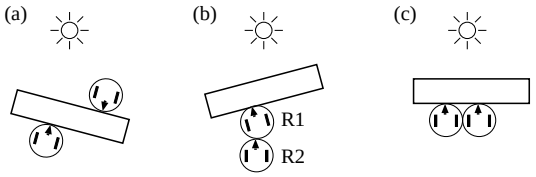

Figure 6 Harmful interaction

3.4 SBS-3: Box-pushing in swarms

In $S3$, since a single robot can not move a box, robots need to swarm for pushing a box cooperatively. A swarm has various shape: a line, a circle, a arrow, etc. We use a line so that a robot can avoid harmful interaction. Behaviors for swarming is 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 an action: *following*, *side&push*. The *forward/right/left/back-robot* means that another robot is sensed forward/right/left/back, and is determined by a procedure for checking M in § 2.4. The *forward/back-robot-leaving* means that another robot which was sensed forward/back becomes not to be sensed. The *following* means that a robot moves to the direction in which another robot was sensed or left. The *side&push* means that aligned robots move to side and pushes a box cooperatively like Fig.7. *SBS-3* consists of the following three subsets.

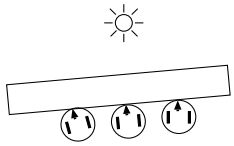

Figure 7 Cooperative formation

Swarming: Swam is constructed by wandering until a robot find other robots and following others. By adding a following behavior to *SBS-1*, a robot does such actions.

B-10 If *left-robot*∨*right-robot* ∨ then *following*.

Keeping a line: For keeping a line, suitable behaviors for a head and not-head robots are designed in the following. A head robot goes ahead when another robot is sensed in its back, and stops when no robot is sensed in its back. A not-head robot follows when a front robot disappears.

For a head robot

B-11 If ¬ *forward-robot* ∧ *back-robot* then *go-ahead*.
B-12 If ¬ *forward-robot* ∧ *back-robot-leaving* then *stop*.

For not-head robots

B-13 If *forward-robot-leaving* then *following*.
B-14 If *forward-robot* then *stop*.

Box-pushing in a swarm: When aligned robots find a box, they need to push the box cooperatively like Fig.7. They also 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* ∧ *back-robot* ∧ *forward-light* then *push-straight*.
B-16 If *forward-object* ∧ *back-robot-leaving* then *following*.

For not-head robots

B-17 If *forward-robot* ∧ *back-robot-leaving* then *following*.
B-18 If *forward-robot* ∧ ¬ *forward-light* then *direction-change*.
B-19 If *forward-robot* ∧ *forward-light* then *side&push*.

Finally *SBS-3* consists of *SBS-1* and B-10∼B-19.

3.5 SBS-4: Acting for transition

In *S-4*, since a robot recognizes neither other robots nor a box which it can move singly, 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. Thus a robot wanders using *SBS-1* until other robots or a light box is found.

4 Experiments

We implemented the adaptive action selection method on each of four *Khepera*s. The time parameters T_m and T_t in § 2.4 are set 300 sec. 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 m sec.

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 results, the probability that the robots achieves the task was more than 80% in each situation. We investigated 30 random initial positions for each situation except ones in which robots can not push boxes such as boxes touch with walls.

4.1 Results in each static situation

Experiments in S1 and S2: We set a light box and a single robot in an environment, and ran a robot. Fig.8 shows the trace 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, each robot independently pushed a box as well as in *S1* (Fig.9).

Experiments in S3 and S4: A heavy box and four robots are set for *S3*. Fig.10 shows the trace of the actions. The robots succeeded in swarming (Fig.10(a) ∼ (d)) and executing the *side&push* action (Fig.10(e) ∼ (h)). Next *S4* is set with a heavy box and a single robot. Then we observed that the robot wanders to search for other robots or a light box.

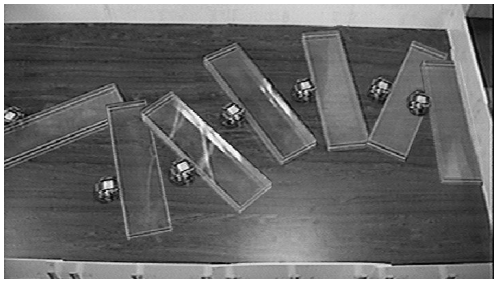

Figure 8 Trace of actions in *S1*

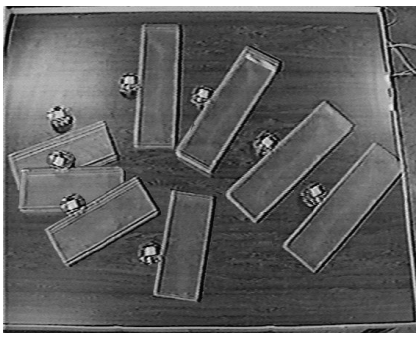

Figure 9 Trace of actions in *S2*

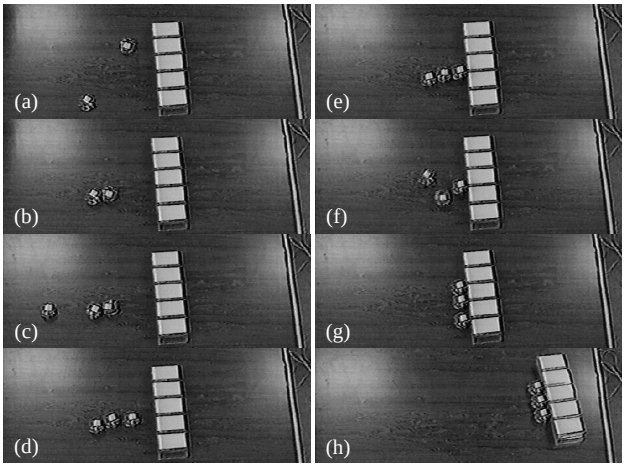

Figure 10 Box-pushing in *S3*

4.2 Adaptation to a dynamic environment

By adding and removing robots and heavy boxes, we changed the situation and observed behavior of robots. As results, for all the changes between arbitrary two situations in {*S1*, ···, *S4*}, we verified situation transition was independently done in each robot, and suitable *SBS* was activated. When multiple robots act in the same environment, each situation transitions in them occurred asynchronously, and all the robots presently converged to the same situation.

For example, Fig.11 shows actions after two robots encountered in an environment where no heavy box exists. They recognized that the current situation was

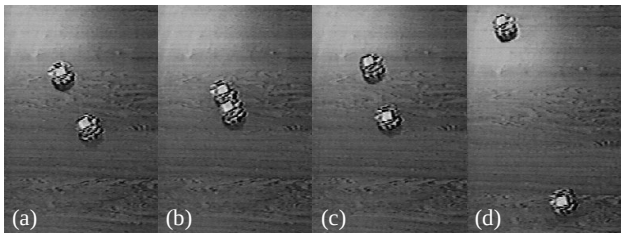

Figure 11 The actions after encounter in *S2*

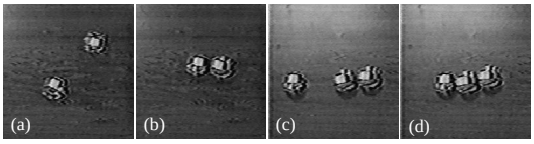

Figure 12 The actions after encounter in *S3*

S2, and *SBS-2* was activated. Thus they left mutually after they encountered. Then we added a heavy box into the environment. The robots observed the heavy box presently, and updated the current situation to *S3*. *SBS-3* was activated, and they acted in a swam. Fig.12 shows actions after the robots encountered in such situation. They swarmed after encounter, not leave mutually.

5 Discussion

Our approach have the following open problems.

Assumptions on an environment: We use some assumptions on an environment: *AS-1*, *AS-2* in § 2.1. If these assumptions are not held, our multi-robot 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 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.

Scalability: Due to physical constraints, we did not make the experiments using n robots ($n \geq 5$). We consider the *SBS*s 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.

6 Conclusion

We proposed adaptive action selection without explicit communication for dynamic multi-robot box-pushing. First, for describing dynamic environments, we defined situations with two parameters: existence of other robots and task difficulty. Next we designed behavior sets for each of the situations. We fully implemented our approach on four real mobile robots, and verified the utility experimentally.

References

[1] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Transaction on Robotics and Automation*, 2(1):14–23, 1986.

[2] K. Kosuge and T. Osumi. Decentralized control of multiple robots hnadling and object. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 318–323, 1996.

[3] C. Kube and H. Zhang. The use of perceptual cues in multi-robot box-pushing. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 2085–2090, 1996.

[4] M. J. Mataric. Learning in multi-robot systems. In G. Weißand S. Sen, editors, *Adaption and Learning in Multi-agent Systems*, pages 152–163. Springer, 1996.

[5] M. J. Mataric, M. Nilson, and K. T. Simsarian. Cooperative multi-robot box-pushing. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 556–561, 1995.

[6] N. Miyata, J. Ota, T. Arai, E. Yoshida, D. Kurabayashi, J. Sakaki, and Y. Aiyama. Cooperative transport with regrasping of torque-limited mobile robots. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 304–309, 1996.

[7] H. Osumi. Cooperative strategy for multiple mobile manipulators. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 554–559, 1996.

[8] L. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transaction on Robotics and Automation*, 14(2):220–240, 1998.

[9] C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *ACM Computer Graphics*, 21(4):25–34, 1987.

[10] D. J. Stilwell and J. S. Bay. Toward the development of a material transport system using swarms of ant-like robots. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pages 766–771, 1995.

[11] H. Sugie, Y. Inagaki, S. Ono, H. Aisu, and T. Unemi. Placing objects with multiple mobile robots – mutual help using intention inference. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, pages 2181–2186, 1995.

[12] Z. Wang, E. Nakano, and T. Matsukawa. Realizing cooperative object manipulation using multiple bebavior-based robots. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 310–317, 1996.