Suppressing Energy Consumption of Transportation Robots using Mobile Agents

Ryosuke Shibuya[1], Munehiro Takimoto[1] and Yasushi Kambayashi[2]

[1]*Department of Information Sciences, Tokyo University of Science, Chiba, Japan*
[2]*Department of Computer and Information Engineering, Nippon Institute of Technology, Saitama, Japan*

Keywords: Mobile Agent, Multi-robots, Energy Consumption, Intelligent Robot Control.

Abstract: This paper presents an application for controlling multiple robots connected by communication networks. Instead of making multiple robots pursue several tasks simultaneously, the framework makes mobile software agents migrate from one robot to another to perform the tasks. Since mobile software agents can migrate to arbitrary robots by wireless communication networks, they can find the most suitably equipped and/or the most suitably located robots to perform their task. We previously implemented an application of searching targets in our framework, and showed that it could suppress energy consumption of the entire system. In this paper, we propose a new mobile agent system that transports the found targets to a collection area. The approach is based on passing the targets among robots through throwing, and therefore, suppresses energy consumption as the same manner for searching targets. In order to show the effectiveness of our approach, we have implemented two strategies on a simulator and conducted numerical experiments. As a result, we show that our approach has more advantages than previous ones, and there is a remarkable difference between the two strategies in terms of energy saving.

1 INTRODUCTION

In the last decade, robot systems have made rapid progress not only in their behaviors but also in the way they are controlled. In particular, a control system based on multiple software agents can control robots efficiently (Takimoto et al., 2007).

On the other hand, excessive interactions among agents in the multi-agent system may cause problems in the multiple robot environments. In order to mitigate the problems of excessive communication, mobile agent methodologies have been developed for distributed environments (Kambayashi and Takimoto, 2005). Mobile agent systems are especially useful in an intermittently connected ad hoc network environment. In the minimal case, a mobile agent requires that the connection is established only when it performs migration (Binder et al., 2001).

The model of our system is a set of cooperative multiple mobile agents executing tasks by controlling a pool of multiple robots as shown by (Kambayashi and Takimoto, 2005). A mobile agent can migrate to the robot that is most conveniently located to a given task, e.g. closest robot to a physical object such as a soccer ball. Since the agent migration is much easier than the robot motion, the agent migration contributes to saving power consumption (Takimoto et al., 2007).

We have proposed our model in the previous paper (Takimoto et al., 2007) and have also shown the effectiveness of saving power consumption and the efficiency of our system for searching targets (Nagata et al., 2009; Abe et al., 2011). In this paper, we focus our attention on not only searching targets process but also transporting the found targets to a designated collection area. The sequence of behaviors for searching targets and transporting them is one of the most important tasks for autonomous robots. For example, the sampling missions on Moon and Mars are well-known. Also, it is important for transporting target things from the stricken area where the environment is too dangerous for human to work. Our approach enables robots saving energy consumption by passing targets among robots through throwing them. A software mobile agent controls a robot and makes it throw an object, and then the agent migrates to another robot that receives the target. Thus instead of mobile robots, the software mobile agent has the responsibly to transport the target object to the destination, and it achieves suppressing the number of rotations of robot-wheels. Furthermore, our transporta-

Shibuya R., Takimoto M. and Kambayashi Y..
Suppressing Energy Consumption of Transportation Robots using Mobile Agents.
DOI: 10.5220/0004240902190224
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (ICAART-2013), pages 219-224
ISBN: 978-989-8565-38-9

tion technique has a good property of holding the way in which robots are scattered, where it is more profitable for robots to be scattered in a work field for passing targets through throwing.

The structure of the balance of this paper is as follows. In the second section we describe the background. The third section describes an example of intelligent robot systems in which robots search multiple target objects and transport them cooperatively. In our robot control system, the mobility that the mobile agent system provides is the key feature that suppresses energy consumption; therefore the details of mobile agent based strategies are shown here. In the fourth section, we demonstrate how the properties of mobile agents can contribute to saving energy consumption through numerical experiments based on a simulator. Finally, we conclude in the fifth section.

2 BACKGROUND

Multi-agent robotic systems are recently becoming popular (Yasuda and Ohkura, 2011). In traditional multi-agent systems, robots communicate with each other to achieve cooperative behaviors. There are three major advantages of multi-robot systems over single robot systems (Stone and Veloso, 2000; Yasuda and Ohkura, 2005). The first is parallelism; a task can be achieved by autonomous and asynchronous robots in a system. The second is robustness; it is realized through redundancy. The system can have more robots than required for a certain task. The third is scalability; a robot can be added to or removed from the system easily.

Making multiple robots cooperatively carry and push common objects has been intensively studied and yet to established the standard way. Many research projects have dealt with this topic but few of them have demonstrated on physical multi-robot systems. One of the most demonstrated tasks involving cooperative transport is the pushing objects by multiple robot teams (Rus et al., 1995; Stilwell and Bay, 1993). This task is inherently easy to accomplish when comparing to carrying tasks, because a carrying task involves multiple robots' gripping a common object and navigating to a destination in a coordinated fashion (Khatib et al., 1996; Wang et al., 2000).

One of the most famous transportation problems is the box-pushing problem (Mataric et al., 1995). The problem is defined in (Gerkey and Mataric, 2002) and consists of cooperatively moving a box, which is relatively large when compared to the size of the multi-robots, from an initial position to a destination using robots that can only perform pushing movements. Then the fundamental research problem is to design the appropriate control mechanism in order to achieve the desired global behavior by the multiple robots' cooperation. This problem is known as NP-hard (Reif, 1979), and many research efforts are focused into planning. Our research objective is saving energy by much rougher behaviors of multiple robots but more intelligent behaviors through multiple mobile software agents.

3 ROBOT CONTROLLER AGENTS

The target recovery process consists of two tasks; one is the task for searching targets, and the other is for transporting them to collection areas. They are performed sequentially and exclusively. In our system, these tasks are allocated to different software mobile agents. They are *Searcher* agent and *Transporter* agent, respectively. In this section, we describe details of each mobile agent, and show how they contribute to suppressing energy consumption in a multi-robots system.

3.1 Mobile Agents for Searching

Considering how to make multi-robots search a target. The simplest solution would be to make all robots move in order to search for the target. It means, however, only one robot that ultimately finds the target do the job, and the other robots waste their energy in vain. In addition, even if one finds the target, the others may not be able to know the fact because of being out of the range where communications through wireless LAN are available. As the result, they may move around until they are exhausted to move.

We have proposed the mobile agent approach for searching targets and showed its effectiveness in the previous paper (Abe et al., 2011). The Searcher agent visits each robot through migration. When the agent reaches a robot, it checks the camera on the robot in order to find whether it is close to the target or not, where it would have to spin within 360 degrees to check all around it if the robot has no special camera such as an omnidirectional sensor. It repeats this process until it reaches the robot closest to the target, and finally, it drives that robot to capture the target. The searching strategy suppresses rotating wheels, and therefore, contributes to saving energy consumption. We have also adopted the same approach to the target recovery process. The only difference is that the recovery agent has a vector value that points to the collection area as an interior datum, which is

subsequently used to guide the agent for transporting the target. We assume that the Searcher agent know where the collection area is. The simplest implementation would be to make Searcher agents born at their own collection areas. Each Searcher agent has a vector to keep track of its collection area. In the searching process, each Searcher agent's vector value is dynamically modified so as to pointing to the destination. When the robot, which is driven by the Searcher agent, physically moves, the vector value is adjusted according to the physical motion. Also whenever the Searcher agent migrates to another robot, the vector value has to be modified with respect to the new position. The modification is achieved by synthesizing the vector value at the previous robot and the vector value from the source to destination in the migration. To achieve this synthesis, it is required that each robot can locate each other.

3.2 Mobile Agents for Transporting

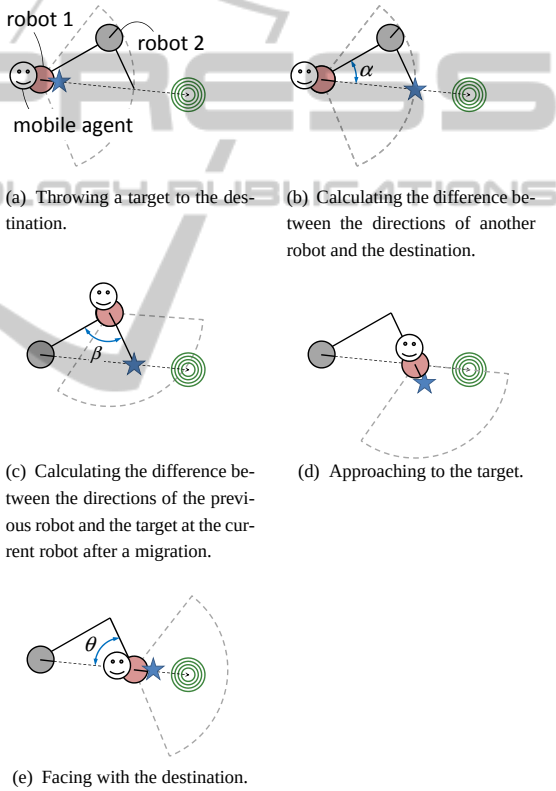
Once the Searcher agent finds the target object, the agent has to transport the object to the collection area. In this subsection we describe a new transporting algorithm that saves energy consumption using a software mobile agent. The basic idea is to make the robot throw the target in the direction of the destination instead of carrying it, where the direction of the destination is passed from Searcher agent as vector value. In the simplest case, the collection area would correspond to the location where the Searcher agent was generated as mentioned above. Assuming such vector value, all that robots should do is only to check the directional criteria shared with the other robots. Such criteria could be given by a simple device such as a compass. The algorithm for the Transporter agent is as follows: 1) Transporter makes the robot pick up the target, and then, 2) throws the target to the destination. 3) If there is not any other robot in the direction of the destination, Transporter agent makes the robot carry it by itself.

In the second step, there are two strategies of *ThrowStraight* and *DirectPass* as a manner where several robots transport the target to the collection area through cooperatively passing it.

3.2.1 ThrowStraight Strategy

The simplest strategy for passing a target is that the agent makes the robot throw the target to the destination as further as possible, and then, makes another robot that is closest to the thrown target pick it up. Once such a closest-to-the-target robot is found, ThrowStraight agent migrates to that robot to receive the target. This behavior makes the newly arrived robot approach to the target in order to pick up and throw the target again. The ThrowStraight agent can transport the target to the destination by repeating these steps.

On the other hand, if ThrowStraight agent cannot find any robot close to the target, the robot has to carry the target by itself. ThrowStraight agent throws the target to the destination regardless whether the closest robot is found or not. After that, if such a closest-to-the-target robot cannot be found, all what the agent should do is to make the robot approach to the target along thrown path. Notice here that the behaviors of throwing and approaching correspond to carrying the target with it to the location where the target is thrown.



(a) Throwing a target to the destination.

(b) Calculating the difference between the directions of another robot and the destination.

(c) Calculating the difference between the directions of the previous robot and the target at the current robot after a migration.

(d) Approaching to the target.

(e) Facing with the destination.

Figure 1: The migration steps of ThrowStraight.

These steps of the ThrowStraight strategy seem to work well if the direction to the destination, i.e. a collection area, is always known. Initially, the direction information is given to the ThrowStraight agent by Searcher agent. Notice here that though the Searcher agent has vector value to the destination, the ThrowStraight agent holds only the angle datum of the vector value. After that, the angle information is modified according to physical motions of the robot, and recalculated through every migration. The new angle value after the migration can be gotten by calculat-

ing each angle of a triangle consisting of the current robot, the next robot, and the target in order. Fig. 1 demonstrates the steps of the process, where a star, multi-circles, a circle with a face, and circles with a slit respectively represent a target, a destination, an agent, and robots. The slits of circles show the direction with which each robot faces. The steps are as follows:

1. The agent makes the robot throw the target to the destination as shown by Fig.1(a),
2. The agent calculates difference α between the directions of the other robot and the destination as shown by Fig.1(b).
3. The agent migrates to the next robot (robot 2).
4. The agent calculates difference β between the directions of the previous robot (robot 1) and the target as shown by Fig.1(c).
5. The agent makes the robot approach to the target and picks it up as shown by Fig.1(d).
6. The agent makes the robot face to the destination as shown by Fig.1(e), where the angle to be oriented, θ is given by the calculation $180-(\alpha+\beta)$.

Notice that the thrown point in the first step has to be within the view range of the robot. Otherwise, the target cannot be traced in the following process.

Summarizing a sequence of behaviors of ThrowStraight agent, an agent performs the following steps 1-5 until finding the specific destination, and then the agent makes the robot throw a target to the destination once it finds the destination: 1) The ThrowStraight agent drives the robot forward until finding out other robots within the angle of 120 degrees in the view range, 2) the agent makes the robot throw the target to the destination, 3) the agent migrates to the closest robot to the target, 4) the agent drives the robot, to which the agent has migrated, toward the target and makes the robot pick it up, and 5) the agent calculates the modified direction and makes the robot orient toward the destination.

3.2.2 DirectPass Strategy

More smart strategy, which we call DirectPass strategy, is to directly pass a target to another robot closer to a destination. In more detail, the thrower passes the target to the receiver only if the receiver is within 120 degrees range of the direction from thrower to the destination. In this strategy, the step for searching the next robot in ThrowStraight strategy is not necessary because the current robot throw the target to the next robot, instead of throwing to the direction of the destination. What DirectPass agent does is simply migrating to the receiver.

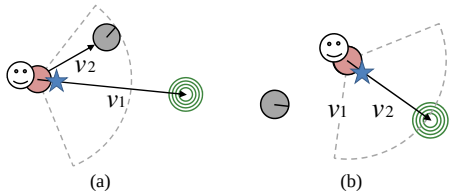


Figure 2: Passing process of DirectPass agent.

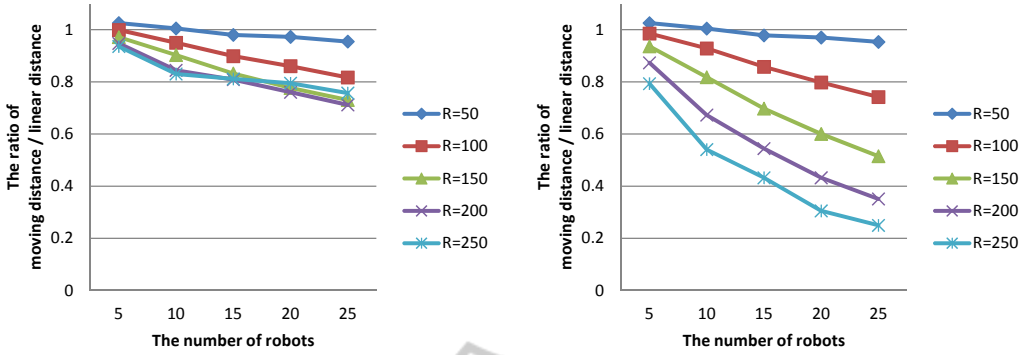
Thus, DirectPass strategy seems to be simpler and more direct than ThrowStraight strategy. In order to keep the direction to the destination, however, the agent has to have complex data such as a vector value, though the ThrowStrainght agent needs to have only the angle as an interior datum. The vector value has to be not only modified according to the physical motion of a robot, but also recalculated through migrations. Fig. 2 shows the process of calculating new vector value after a migration. The steps are as follows:

1. The agent calculates vector value v_2 from the current robot to the next robot as shown by Fig.2(a), where the initial vector value is passed by Searcher agent.
2. The agent modifies the interior datum v_1 with new vector value given by calculating $v_1 - v_2$.
3. The agent migrates to the next robot as shown in Fig.2(b).

Summarizing a sequence of behaviors of DirectPass agent, an agent performs the following steps 1-5 until finding the specific destination, and then the agent makes the robot throw a target to the destination once it finds the destination: 1) The DirectPass agent drives the robot forward until finding out other robots within angle of 120 degrees in the view range. 2) the agent makes the robot throw the target to the receiver, 3) the agent calculates new vector value as interior datum, 4) the agent migrates to the receiver robot, and 5) the agent makes the robot pick up the target and orient toward the destination.

4 EXPERIMENTAL RESULTS

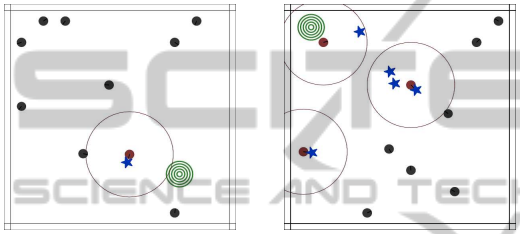
In order to demonstrate the effectiveness of our system, we have conducted numerical experiments on the example of target transportation that we have just discussed in the previous section. We have implemented the mobile agent system simulator as shown in Fig. 3. The simulator is based on hierarchical mobile agent (Kambayashi and Takimoto, 2005; Satoh, 2000), of which migration manner is based on the strong migration model (Cugola et al., 1997). We have evaluated our two strategies i.e. ThrowStraight and DirectPass,



(a) Results for ThrowStraightAgent

(b) Results for DirectPassAgent

Figure 4: The influences of the number of robots to moving distance.



(a) A snapshot of a running simulator with a single target.

(b) A snapshot of a running simulator with multi robots and multi targets.

Figure 3: A snapshot of a running simulator.

in terms of moving distance and how robots are located after the target is collected on the simulator. We assume that energy consumption of a robot is proportional to the moving distance because most energy is consumed for rotating wheels.

4.1 Energy Consumption

First, we have conducted several experiments on the sequential process of searching and transporting of a target. We have repeated a number of the same processes with varying the number of targets and the radius of the view range of each robot. Figs.4(a) and (b) show ratio of the total moving distance in the transportation process to the direct distance from the start point of the transportation to the destination. The five line charts represent the results for different radiuses of the view ranges: 50, 100, 150, 200 and 250, respectively. Each chart consists of results for 5, 10, 15, 20 and 25 robots. As shown by the shapes of the charts, the more the number of the robot and/or the radius of the view range increased, the more the total moving distance decreased in both cases of ThrowStraight and DirectPass strategies.

4.2 Arrangements of Robots

In both of our transporting approaches, the more widely robots are scattered, the more advantages they seem to have. In order to confirm this observation, we have evaluated the area size covered by the view ranges of all robots during the experiments as a criterion of the way in which robots are scattered. In the experiments, we assumed several targets, and sequential executions of searches and transportations such as SA_1(a search agent) $\rightarrow$ TA_1(a transport agent) $\rightarrow$ SA_2 $\rightarrow$ TA_2 $\rightarrow$ SA_3 $\rightarrow$ SA_4 $\rightarrow$ TA_3 $\rightarrow$ TA_4 $\rightarrow$ $\cdots$, TA_n is created after SA_n.

Table 1 shows the ratio of the area size covered by the view ranges in DirectPass strategy to that in ThrowStraightAgent, for 5, 15 and 25 robots, for 10, 30 and 50 targets and for 50, 150 and 250 view distance respectively. Though DirectPass strategy are more widely scattered than ThrowStraightAgent, the difference between them is negligible. In other words, the result shows that our approaches have little difference in influence on other tasks of the multi-robots.

5 CONCLUSIONS

We have implemented a multi-robot system that transports a target to a destination while suppressing energy consumption, by adding transport facility to the previous multi-robot system for searching (Abe et al., 2011). We have proposed two strategies ThrowStraight and DirectPass as transport facility. Both of them can suppress more energy consumption than carrying directly to destination, but DirectPass has remarkable advantages. DirectPass, however, requires more hardware equipment than ThrowStraight. DirectPass has to get both the directional angle and the distance toward other robots through robot's sensor

Table 1: Ratio sum of each robot's view area: DirectPass / ThrowStraight.

View distance	target = 10			target = 30			target = 50		
	The number of robots								
	5	15	25	5	15	25	5	15	25
R = 50	1.055	0.997	1.007	1.005	1.005	1.018	1.023	1.045	1.040
R = 150	1.056	1.041	1.016	1.064	1.094	1.049	1.094	1.114	1.094
R = 250	1.030	1.034	1.014	1.099	1.092	1.048	1.100	1.142	1.100

devices. ThrowStraight needs only the directional angle. Thus, when implementing the multi-robots system based on our approach, it would be necessary to decide which strategy should be adopted based on quality of the sensors equipped with the robots.

REFERENCES

Abe, T., Takimoto, M., and Kambayashi, Y. (2011). Searching targets using mobile agents in a large scale multi-robot environment. In *KES-AMSTA*, volume 6682 of *LNAI*, pages 211–220.

Binder, W., Hulaas, J. G., and Villaz, A. (2001). Portable resource control in the j-seal2 mobile agent system. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, pages 222–223. ACM.

Cugola, G., Ghezzi, C., Picco, G. P., and Vigna, G. (1997). Analyzing mobile code languages. In *Selected Presentations and Invited Papers Second International Workshop on Mobile Object Systems - Towards the Programmable Internet*, MOS '96, pages 93–110. Springer-Verlag.

Gerkey, B. P. and Mataric, M. J. (2002). Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination. In *Proceedings of the IEEE International Conference on Robotics and Automation 1*, pages 464–469.

Kambayashi, Y. and Takimoto, M. (2005). Higher-order mobile agents for controlling intelligent robots. *International Journal of Intelligent Information Technologies (IJIIT)*, 1(2):28–42.

Khatib, O., Yokoi, K., Chang, K., Ruspini, D., Holmberg, R., and Casal, A. (1996). Vehicle/arm coordination and mobile manipulator decentralized cooperation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 546–553.

Mataric, M. J., Nilsson, M., and Simsarian, K. T. (1995). Cooperative multi-robot box-pushing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 3*, pages 556–561.

Nagata, T., Takimoto, M., and Kambayashi, Y. (2009). Suppressing the total costs of executing tasks using mobile agents. In *Proceedings of Hawaii International Conference on System Sciences 42 CD-ROM*.

Reif, J. H. (1979). Complexity of the mover's problem and generalizations. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 421–427. IEEE Computer Society.

Rus, D., Donald, B., and Jennings, J. (1995). Moving furniture with teams of autonomous robots. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 235–242.

Satoh, I. (2000). Mobilespaces: A framework for building adaptive distributed applications using a hierarchical mobile agent system. In *Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, ICDCS '00, pages 161–168. IEEE Computer Society.

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

Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Takimoto, M., Mizuno, M., Kurio, M., and Kambayashi, Y. (2007). Saving energy consumption of multi-robots using higher-order mobile agents. In *KES-AMSTA*, volume 4496 of *LNAI*, pages 549–558.

Wang, Z. D., Kimura, Y., Takahashi, T., and Nakano, E. (2000). A control method of a multiple nonholonomic robot system for cooperative object transportation. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems on Distributed Autonomous Robotic Systems 4*, pages 447–456.

Yasuda, T. and Ohkura, K. (2005). Autonomous role assignment in a homogeneous multi-robot systems. *Journal of Robotics and Mechatronics*, 17(5):596–604.

Yasuda, T. and Ohkura, K., editors (2011). *Multi-Robot Systems, Trends and Development, InTech*.