GPGPUを用いた連結センサカバーに対する蜂群最適化

(A bee colony optimization algorithm for a connected sensor cover on GPGPU)

Yukihide Sasamura and Akihiro Fujiwara
Graduate School of Computer Science and Systems Engineering,
Kyushu Institute of Technology
Iizuka, Fukuoka, 820-8502, Japan
Email: o676113y@mail.kyutech.jp, fujiwara@cse.kyutech.ac.jp

***Abstract*—In the sensor network, a set of connected sensors that covers all discrete targets is called the connected sensor cover (CSC). The CSC with a small number of sensors is desirable because the small CSC can reduce network energy and communication costs.**

In the present paper, we propose an algorithms for CSC using an artificial bee colony optimization, which is an optimization technique based on behaviors of honey bees, on GPGPU. The experimental result shows that the execution on GPGPU is 5 times faster than the execution on CPU in case that the number of bees in the optimization algorithm is enough large.

I. Introduction

Sensor network is a wireless communication network, which is composed of small autonomous sensors. Each sensor can sense given targets if the target is in the sensing area, and communicate with the other sensors if the sensor is in the communication area. In the sensor network, a set of sensors is called *connected* if each sensor in the set can communicate with any sensor in the set using the others sensors as relays. A set of connected sensors can obtain a vast range of information across the network by communicating messages with the other sensors. In addition, a set of sensors is called *cover* if all discrete targets are covered with sensors in the set. Thus, the connected sensor cover (CSC) is defined as a set of connected sensors that cover all discrete targets.

Although construction of the CSC with the minimum number of sensors is NP-hard [1], the CSC with a small number of sensors is desirable because construction of the small CSC can reduce network energy and communication costs. Therefore, a number of algorithms [1], [2], [3], [4], [5], [6], [7] have been proposed for constructing CSC with a small number of sensors. For example, Jaggi et al. [2] proposed centralized algorithms using linear programming and a greedy method. Cardei et al. [4] proposed another centralized approximation algorithms, and showed that their algorithms can be easily applied to a distributed environment.

In addition, some algorithms have been proposed for CSC using particle swarm optimization techniques. Begum et al. [3] proposed algorithms based on ant colony optimization, and Shimokawa and Fujiwara [7] proposed algorithms based on artificial bee colony optimization. Although the above algorithms with the swarm optimization obtain CSC with a small numbers of sensors, the algorithms are time-consuming, and huge computational powers are needed to execute the proposed algorithms.

In this paper, we propose an algorithms for CSC using an artificial bee colony optimization on GPGPU. GPU (Graphics Processing Unit) is a hardware device equipped with a number of small processors specialized in graphics processing, and GPGPU (General Purpose computation on GPU) is general parallel computation on GPU. Since recent GPU is developed according to CUDA (Compute Unified Device Architecture), we can implement proposed algorithms on GPGPU as programs for parallel processing.

We implement our proposed algorithm using CPU and GPU, and evaluate validity of the proposed algorithm. The experimental result shows that the execution on GPGPU is 5 times faster than the execution on CPU in case that the number of bees in the optimization algorithm is enough large.

II. Preliminaries

A. Sensor Model

In this paper, the sensor network $G = (V, E)$ is defined by a set of sensors $V = \{s_1, s_2, \ldots, s_n\}$, where n is the number of sensors, and the set of links E that is a set of communication links between sensors. The set of discrete targets is represented by $T = \{t_1, t_2, \ldots, t_m\}$, where m is the number of discrete targets. Each sensor s_i has an unique identification number ID_i. The sensors are deployed on two-dimensional plane R, and each sensor knows the geographical location of itself.

Figure 1 is the sensor model used in this paper. Each sensor s_i has communicating area C and can communicate with other sensors in the communication area. In case that sensors s_i and s_j can communicate each other, a communication link $e_{ij} \in E$ exists between the sensors s_i and s_j on a graph G, and the sensors are called adjacent. We assume that direct communication of messages is possible for only between adjacent sensors without collision. In addition, two sensors are called connected if there is a path of adjacent sensors between two sensors in G.

We also assume that each sensor s_i has a circular sensing area $S \subseteq C$, and can sense a target in the sensing area. We

call that a sensor s_i covers a target t if t is in sensing area of s_i.

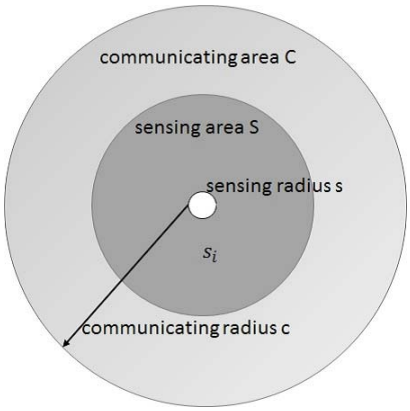


Fig. 1. A sensor model

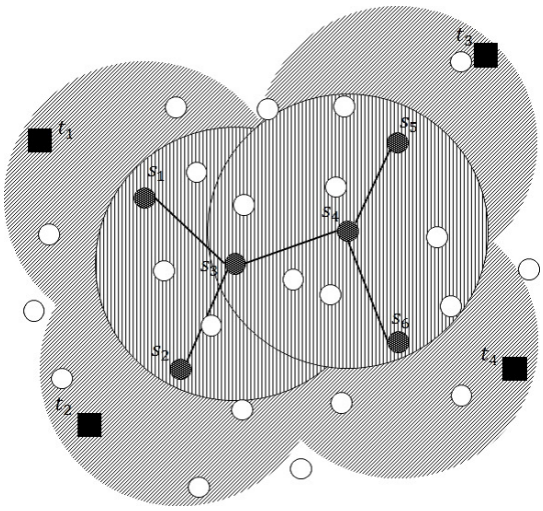


Fig. 2. An example of CSCDT

B. Connected Sensor Cover for discrete targets

Since each sensor equips a limited battery, coverage with a fewer number of sensors should be desired for discrete targets. However, connectivity of sensors is needed to communicate data of targets in the sensor network. As a set of sensors that satisfies these conditions, the connected sensor cover is defined as follows.

Definition 1 (Connected sensor cover for discrete targets): Given a sensor network $G = (V, E)$ and a set of discrete targets $T = \{t_1, t_2, \ldots, t_m\}$, the subset of sensors $M = \{s_{i_1}, s_{i_2}, \ldots, s_{i_n}\}$ $(M \subseteq V)$, which satisfies the following condition, is called the connected sensor cover for the discrete targets (CSCDT).

1) M is connected.
2) Any target in T is in at least one of sensing areas of sensors in M. □

The targets covering problem using line segments is known as NP-hard, and CSCDT with the minimum number of sensors is also known as NP-hard by reducing from the problem [4]. Therefore, a smaller number of sensors should be desired for CSCDT.

In this paper, we assume that the number of sensors n are enough large to construct CSCDT. In other words, CSCDT, which covers a set of input discrete targets, always exits for an input set of sensors.

Figure 2 is an example of CSCDT. In this example, the set of sensors $\{s_1, s_2, \ldots, s_6\}$ is CSCDT of the input sensors. The set of sensors $\{s_1, s_2, s_5, s_6\}$ covers a set of discrete targets $\{t_1, t_2, t_3, t_4\}$ with an union of sensing areas, and maintain the connectivity of sensors by including a set of sensors $\{s_3, s_4\}$ as relays.

III. A PROCEDURE FOR THE MINIMAL CSCDT

In this section, we explain a procedure for constructing a minimal CSCDT. We first define redundant sensors for CSCDT, and next propose a procedure, which deletes the redundant sensor, as a basic operation in the proposed algorithm.

Definition 2 (A redundant sensor for CSCDT): Let S_i be sensing area of sensor s_i in the sensor network $G = (V, E)$, and also let $M \subseteq V$ be CSCDT for the sensor network. We also assume that T is a set of discrete targets. Then, a sensor $s_d \in M$ is a redundant sensor for M if the sensor satisfies the following two conditions.

1) $M - \{s_d\}$ is connected.
2) Any target in T is in at least one of sensing areas of sensors in $M - \{s_d\}$. □

Using the above definition, we propose a procedure, which is called *Repeated deletion*, for deleting sensors from CSCDT.

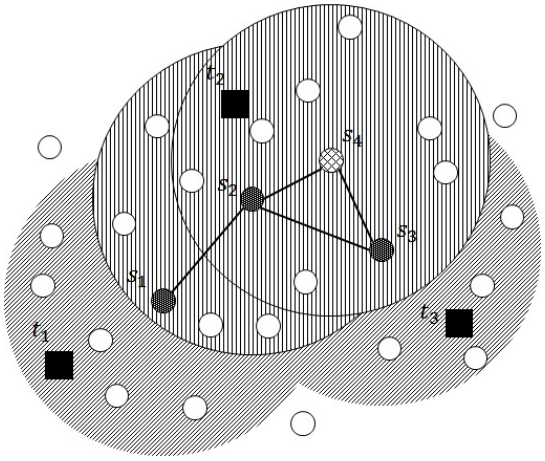
Repeated deletion

Repeat the following two steps for each sensor in an input set of sensors until no sensor is deleted in the second step.

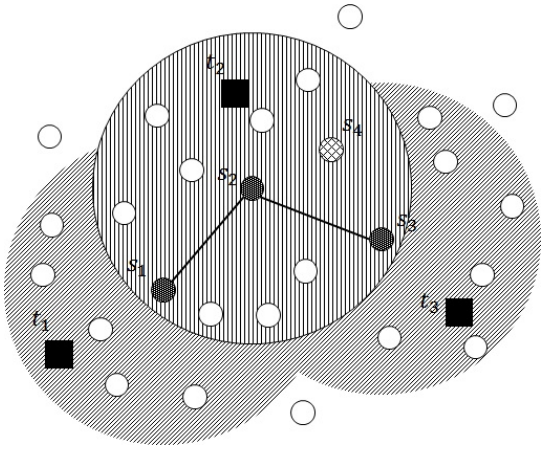
Step 1: Check whether the sensor is a redundant sensor or not for CSCDT.
Step 2: Delete the sensor in case that the sensor is redundant. □

The procedure, *Repeated deletion*, ensures connectivity and coverage of the sensor network. In addition, the obtained CSCDT is apparently minimal due to the end condition of the procedure.

Figure 3 shows an example for a redundant sensor and the procedure. Figure 3 (a) shows an input CSCDT, and the set of sensors $\{s_1, s_2, s_3, s_4\}$ is in CSCDT. An union of sensing areas of the set of sensors $\{s_1, s_3\}$ covers the set of discrete targets $\{t_1, t_2, t_3\}$, and the sensors are connected. In this case, sensor s_4 is redundant, and is deleted if two steps in *Repeated deletion* is executed for s_4. Then, we obtain CSCDT in Figure 3 (b) after deletion of s_4.



(a) An input CSCDT



(b) A CSCDT after deletion of a redundant sensor

Fig. 3. An example of *Sensor reduction*

IV. An algorithm using bee colony optimization on GPGPU

A. Artificial bee colony optimization for CSCDT

In this section, we explain an algorithm using artificial bee colony optimization algorithm for CSCDT. The algorithm is a revised version of an algorithm proposed in [7].

We first explain an outline of the artificial bee colony optimization. The artificial bee colony (ABC) optimization [8] is an optimization technique based on the following habit of honey bees. The bee gathers honey outside region, and shares information of gathered honey with other bees when bee arrives at comb. Then, each bee decides a next way of exploration using exchanged informations. As a result, the honey bees can collect high-quality honey.

In our algorithm for CSCDT, ABC optimization is used for reducing the number of sensors. We assume m bees, and B_j $(1 \leq j \leq m)$ denotes j-th bee used for optimizing CSCDT.

We next introduce an operation, which is called *Sensor reduction*. Let $G = (V, E)$ be an input sensor network, and $G' = (V', E')$ be an CSCDT for G. The following *Sensor reduction* is executed by each bee for reducing the number of sensors in CSCDT.

Sensor reduction

Step 1: Select NC sensors in V' randomly. (The NC is a parameter for optimization. and we assume V_c denotes a set of selected sensors.)

Step 2: For each sensor $s_i \in V_c$, execute the following sub-steps.

(2-1) For each adjacent sensor s_j of s_i, compute V'' such that $V'' = V' - \{s_i, s_j\}$, and then, check whether there exists a sensor $s_k \in V - V'$ such that $V'' \cup \{s_k\}$ is CSCDT. If the sensor s_k exists, the two sensors, s_i and s_j, are removed from V' and s_k is added to V'.

Using the above *Sensor reduction* and *Repeated deletion*, the algorithm for CSCDT using ABC optimization is given below.

An algorithm for CSCDT using ABC optimization

Step 1: Construct the minimal CSCDT V' for $G = (V, E)$ using *Repeated deletion*. The obtained CSCDT is copied to all m bees. (We assume that bee B_j stores the CSCDT as $CSCDT_j$.)

Step 2: The following 3 sub-steps are repeated by a given number of trials u, and output a CSCDT with the minimum number of sensors among all bees.

(2-1) Each bee B_j executes *Sensor reduction* for $CSCDT_j$, and set evaluation value O_j to the number of deleted sensors by *Sensor reduction*.

(2-2) Each bee B_j decides whether to become a recruiter or a follower according to the following probability F_j, where $O_{\max} = \max\{O_k \mid 1 \leq k \leq n\}$ and t is the number of repetition in Step 2.

$$F_i = e^{-\frac{O_{max} - O_i}{t}}$$

(2-3) Each bee B_j maintains $CSCDT_j$ in case of the recruiter. On the other hand, in case of the follower, B_j selects one of CSCDTs owned by recruiters according to following probability R_i, where R is a set of recruiters.

$$R_i = \frac{O_i}{\sum_{O_k \in R} O_k}$$

B. Implementation on GPGPU

In this section, we explain an implementation of our algorithm for CSCDT using ABC optimization on GPGPU. For parallel processing on GPGPU, sub-step (2-1) in the algorithm is executed in parallel on GPGPU, and the other steps are executed serially.

Figure 4 illustrates parallel execution of the proposed algorithm. Each bee in (2-1) is assigned a single thread and executed on each core. Informations of sensors and targets are stored in constant memory, and variables of the kernel function are stored in a register. Informations for the other steps are stored in the global memory, and all informations are exchanged throughout the global memory.

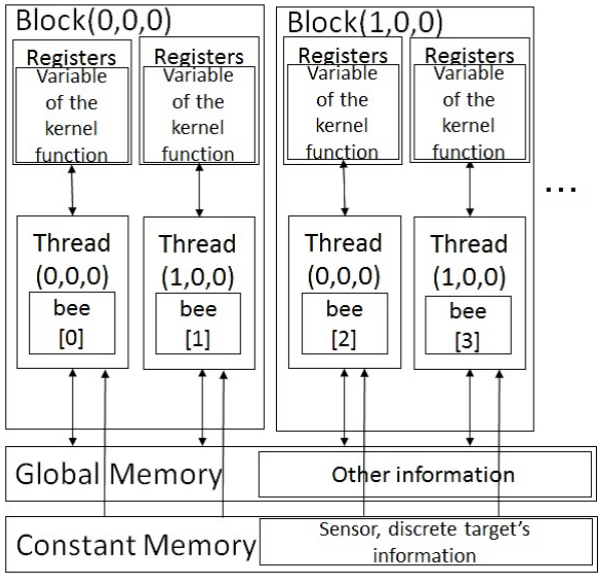


Fig. 4. a model of parallel processing of runtime

TABLE I
EXPERIMENTAL ENVIRONMENT

CPU	Intel Core i3-4130 (3.4GHz)
Main memory for CPU	8GB
GPU	NVIDIA GeForce GTX 760
The number of CUDA cores on GPU	1152
Memory on GPU	4GB
OS	CentOS release 6.5
Development Environment	CUDA 5.5

V. EXPERIMENTAL RESULTS

Our proposed algorithm is implemented on CPU on GPGPU, we compare execution times between the implementations.

We assume that an input region is a 100×100 square area, and sensors and targets are randomly located in the region. The numbers of sensors are 1000, and the number of targets is 20. Sensing radius and communication radius of each sensor are both 10.

In addition, the following parameters are set for proposed algorithms according to execution times.

- The number of bees used in ABC optimization: 32 to 65536
- The number of repetition in ABC optimization: $u = 16$
- The number of sensors that each bee selects randomly: $NC = 4$

We also show our experimental environment for CPU and GPU on Table V.

Figure 5 shows execution times for CPU and GPGPU in the simulation environment. The result shows that the execution on GPGPU is 5 times faster than the execution on CPU in case that the number of bees in the optimization algorithm is enough large.

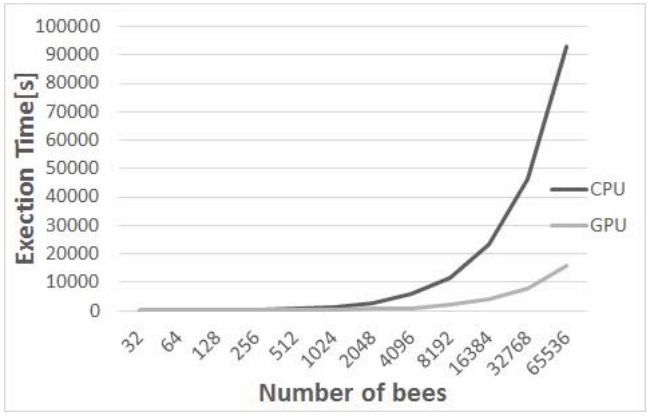


Fig. 5. Execution times on CPU and GPGPU

VI. CONCLUSIONS

In this paper, we proposed an algorithms for CSC using an artificial bee colony optimization on GPGPU. The experimental result shows that the execution on GPGPU is 5 times faster than the execution on CPU in case that the number of bees in the optimization algorithm is enough large.

In our future work, we are considering further improvement of the proposed algorithm using another optimization techniques, and we are also considering to simulate distributed algorithms on GPGPU.

REFERENCES

[1] H. Gupta, Z. Zhou, S. Das, and Q. Gu, "Connected sensor cover:self-organization of sensor networks for efficient query execution," *ACM/IEEE Transactions on Networking*, 14(1), 2006.
[2] N. Jaggi and A. Abouzeid, "Energy-efficient connected coverage in wireless sensor networks," *4th Asian International Mobile Computing Conference*, 2006.
[3] S. Begum, N. Tera, and S. Sultana, "Energy-efficient target coverage in wireless sensor networks based on modified ant colony," *International Journal of Ad hoc, Sensor & Ubiquitous Computing*, vol. 1, no. 4, 2010.
[4] I. Cardei and M. Cardei, "Energy-efficient connected-coverage in wireless sensor networks," *International Journal of Sensor Networks*, vol. 3, no. 3, 2008.
[5] M. Lu, J. Wu, M. Cardei, and M. Li, "Energy-efficient connected coverage of discrete targets in wireless sensor networks," *International Journal of Ad Hoc and Ubiquitous Computing*, 2009.
[6] K. Nakamoto and A. Fujiwara, "Distributed algorithms for 2-connected sensor cover in sensor network," in *Proceedings of the International Conference on Wireless Networks*, 2010.
[7] T. Shimokawa and A. Fujiwara, "Centralized algorithms for the connected target coverage in wireless sensor networks," in *Proceedings of 3rd International Workshop on Advances in Networking and Computing*, 2012, pp. 307–310.
[8] D. Karaboga, "An idea based on honey bee sarm for numerical optimization," Erciyes University, Tech. Rep. TR06, 2005.