A Static Packet Scheduling Approach for Fast Collective Communication by Using PSO

Takashi YOKOTA†, Kanemitsu OOTSU†, and Takeshi OHKAWA†, Members

SUMMARY Interconnection network is one of the inevitable components in parallel computers, since it is responsible to communication capabilities of the systems. It affects the system-level performance as well as the physical and logical structure of the systems. Although many studies are reported to enhance the interconnection network technology, we have to discuss many issues remaining. One of the most important issues is congestion management. In an interconnection network, many packets are transferred simultaneously and the packets interfere to each other in the network. Congestion arises as a result of the interferences. Its fast spreading seriously degrades communication performance and it continues for long time. Thus, we should appropriately control the network to suppress the congested situation for maintaining the maximum performance. Many studies address the problem and present effective methods, however, the maximal performance in an ideal situation is not sufficiently clarified. Solving the ideal performance is, in general, an NP-hard problem. This paper introduces particle swarm optimization (PSO) methodology to overcome the problem. In this paper, we first formalize the optimization problem suitable for the PSO method and present a simple PSO application as naïve models. Then, we discuss reduction of the size of search space and introduce three practical variations of the PSO computation models as repetitive model, expansion model, and coding model. We furthermore introduce some non-PSO methods for comparison. Our evaluation results reveal high potentials of the PSO method. The repetitive and expansion models achieve significant acceleration of collective communication performance at most 1.72 times faster than that in the bursty communication condition.

key words: parallel computers, interconnection networks, collective communication, communication performance, packet scheduling

1. Introduction

An interconnection network is one of the essential components for building parallel computers. It is responsible to communication performance and also its structure defines physical and logical structure of the whole system. Literature shows various discussions on the interconnection network technologies[1], [2].

One of the major research issues on interconnection networks is congestion control. Packets, which are traversing across the routers, interfere to each other in an interconnection network. When two or more packets conflict on a common router resource, only one of them goes through the router and all others are blocked to stay at the same router. These conflicts arise congestion, where the system’s communication performance is seriously degraded. The congested situation has critical characteristics that it spreads quite rapidly and that it continues for a long period of time. On the other hand, in a sparse communication situation, the network is not fully utilized and it has room in performance improvement. Thus, we should keep the network at the edge state of congestion.

With respect to congestion control methods, many research results are reported. In the steady communication situation, some convergence methods are effective[3] and some of them can follow sudden and temporal changes in communication load. On the other hand, we still have no effective ways in unsteady (collective) communication situations due to high-levels of difficulties in congestion control. Thus, we are now in the way to effective congestion control methods[4], [5].

We have difficulty in comparing between new and existing congestion control methods in terms of collective communication performance, since we do not have any knowledge on the theoretical (best) performance. Thus, we can not clearly know whether a proposed method achieves the ideal performance or not.

The ideal performance can be solved theoretically, since all of the conditions are clearly given. However, the problem is NP-hard in general. This study is motivated from the current situation, and it aims at approximate solutions by metaheuristics methods/algorithms. This paper addresses duration times of collective communication by means of coordinated packet scheduling. We first formalize the optimization problem as packet scheduling at every computing node. In our assumptions, each node has its own packet schedule, and the node starts injection of packets according to the predefined schedule. The goal of the optimization problem is to minimize the duration time of collective communication.

Packet conflict occurs when multiple packets simultaneously require the same resource in a router. Only one of the packet goes through the router and all others are blocked. The blocked packets should stay in the same buffer until the conflict situation is resolved. Difficulty in the problem comes from temporal behaviors of routers in transferring packets. Any two packets that go through the same router do not always cause conflict when their arrival times at the router differ and no simultaneous request on the shared resource is detected. This implies that only simple gradient methods of optimization algorithms will not be sufficient for finding appropriate solutions. Thus, we introduce a particle
swarm optimization (PSO) method that has strong capability in search problems.

The rest of this paper is organized as follows. Section 2 shows related work of this study to clarify the unique points of this paper. Section 3 introduces the general PSO method to apply to our packet scheduling problem and proposes naive models. The naive application of PSO to the problem has vast search space. So, in Sect. 4, we discuss reduction of search space to enhance the practical capability of searching and propose three variant models. Section 5 shows our evaluation results and Sect. 6 discusses the search problem from various viewpoints. Finally, Sect. 7 summarizes this paper.

2. Related Work

Optimal use of interconnection network is one of the most crucial issues especially in large-scale parallel systems. In terms of the optimal network control, many discussions are reported from wide spectrum of viewpoints based on various characteristics and requirements of target systems. They are categorized into several classes.

One of the typical dynamic control solutions is throttling, which temporarily suspends new packet injection from computing nodes according to the network status so that the congested situation is controlled. Literature [3], [6]–[8] shows representative methods and the authors have also proposed effective throttling methods [4], [5], [9]. The dynamic control solutions are effective in stable situations of communication, however, they do not guarantee optimal behaviors in unsteady communications.

Many of parallel applications employ MPI (message-passing interface) that offers the core communication solutions. Morie et al. discuss optimization in rank assignment in MPI communication [10], and Soga et al. improve performance in MPI broadcast operations [11]. As a recent result, Gong et al. discuss low-cost MPI communication in cloud environments [12]. These methods aim at offering practical solutions in the context of MPI communication, however, they do not discuss ideal or optimal solutions.

Parallel programs also use collective communications. Reduction and broadcasting are typical operations and their performance issues are discussed. For example, Touzene presents an optimal one-to-all communication method on a certain network topology [13]. However, sufficient discussions on general traffic patterns are still not presented.

Literature [4], [5] suggests that appropriate intervals between injection of packets possibly increase performance. It is a natural consequence since consecutive (i.e., zero-delay) packet injection causes unnecessary congestion. Shibamura et al. present pacing methods for optimal communication [14], [15]. The basic idea of pacing is closed to the throttling concept described by Dally et al. [16], the former intends static solution whereas the latter aims at dynamic (feedback) control of packet flow.

With respect to optimization methods that employ metaheuristics in the parallel computation areas, many studies are reported. Task allocation is one of the hard problems in that area and various metaheuristics algorithms and methods are applied. As recent reports, Guo et al. show fault-tolerant task allocation by using PSO method [17], Xu et al. present application of the chemical reaction optimization (CRO) method to the task allocation problems in directed asynchronous graphs (DAGs) [18]. From a packet communication point of view, these task allocation approaches stand on spacial arrangement of packet communication routes. On the other hand, this paper aims at temporal coordination, where the method keeps routes of packets unchanged. Furthermore, no other research reports introduce the metaheuristics (PSO) to solve the optimization problem so far.

3. Introducing Particle Swarm Optimization

Particle swarm optimization (PSO, [19]) is widely recognized as one of the most effective metaheuristics algorithms that find (quasi-)optimal solutions for NP-hard problems. The optimization methodology has remarkable advantages in searching in multi-dimensional space where many local-minima exit. This section overviews the general PSO method to discuss its application to the packet scheduling problem.

3.1 Particle Swarm Optimization

PSO searches for the optimal solution that is represented in an $M$-dimensional vector that consists of $M$ parameters. PSO is formalized as exploration of the points that show appropriately small score of an evaluation function $F(x)$ that is a positive definite function $(F(x) \geq 0)$.

PSO employs an appropriate number of particles that are placed in the $M$-dimensional space. The $i$-th particle has position: $x^i = (x^i_1, \ldots, x^i_M)$ and velocity: $v^i = (v^i_1, \ldots, v^i_M)$. (1)

The fundamental idea of PSO is to search the optimal point by appropriately moving the particles. PSO maintains the best solutions according to the different scopes; Pbest, Lbest and Gbest. Pbest that stands for the personal best shows the best position where the individual particle has traversed. Lbest (local best) is the best point among neighboring particles, and Gbest (global best) shows the best point among the Lbests. Positions of Pbest, Lbest, and Gbest are represented as

\[ p^i = (p^i_1, \ldots, p^i_M), \]
\[ l^i = (l^i_1, \ldots, l^i_M), \quad \text{and} \]
\[ g = (g_1, \ldots, g_M), \quad \text{respectively.} \]

(2)

PSO moves particle positions step-by-step. Let the position of the $i$-th particle at $n$-th PSO step

\[ x'(n) = (x'_1(n), \ldots, x'_M(n)). \]

The next position and velocity of the particle at the $n+1$-th
step are updated as follows:

\[
v'_j(n+1) = w v'_j(n) + \rho_1(p'_j - x'_j(n)) + \rho_2(t'_j - x'_j(n)),
\]

(4)

\[
x'_j(n+1) = x'_j(n) + v'_j(n),
\]

(5)

where \(i = 1, \ldots, N_p\) and \(j = 1, \ldots, M\). \(N_p\) shows the number of particles. Furthermore, \(w, \rho_1,\) and \(\rho_2\) are random parameters whose ranges are \(0 \leq w, \rho_1, \rho_2 < 1\). Equation (4) shows that velocity of each particle is updated towards the local and global-best point at every step. Each trajectory of particle is not perfectly randomized, but it is guided to the best points, i.e., Gbest and Lbest. Thus, we can expect to find approximated solution after a certain steps of PSO operations.

3.2 PSO Application to Packet Scheduling

We assume two dimensional torus network that consists of \(N \times N\) nodes. Besides the 2D-torus network, actual parallel systems employ various topologies for their interconnection network. However, most of the networks consist of routers which are connected to each other, where they have an essential common characteristic that packets are transferred among the routers to reach their destinations. Thus, the 2D-torus can represent the network topology without loss of generality.

We assume that each node divides a long message into multiple packets in a collective communication session. Each node sends \(S\) (\(S \geq 1\)) packets in one collective communication. Performance is evaluated as a duration time from the beginning of the communication to the completion of receiving \(S\) packets at every node\(^1\).

Every packet has pre-determined schedule time when it will be injected into the network from its corresponding node. We denote the scheduled time of the \(s\)-th packet at the node whose address is \((x, y)\) as \(t_{(x,y,s)}\), where \(0 \leq s < S\) and \(0 \leq x, y < N\). Obviously, \(t_{(x,y,s)}\) forms an \(N^2S\)-dimensional vector that has the only constraint\(^{11}\)

\[
t_{(x,y,p)} < t_{(x,y,q)}, \quad \text{where} \quad p < q.
\]

(6)

We can approximately consider

\[
t = \{t_{(x,y,s)} \mid 0 \leq x, y < N, \quad 0 \leq s < S\}
\]

(7)

as the search space in the packet scheduling problem in which the optimal schedule offers the shortest duration time. This discussion leads us to the simple application of the PSO method into the packet scheduling problem.

Figure 1 intuitively illustrates the concept of the packet scheduling problem. The vertical axis shows positions of nodes that are unrolled from the original two-dimensional coordinate \((x, y)\). The horizontal axis shows the injection time of the head flit of packet in red color with following body flits in green color\(^{111}\). Each member element in a particle position vector specifies the injection time of the corresponding packet. Precisely, the vector element \(t_{(x,y,s)}\) shows when the node \((x, y)\) will start injecting its \(s\)-th packet. Thus, when a packet is generated before its scheduled time, injection of the packet should be suspended until the scheduled time. Note that we cannot expect immediate delivery of injected packets. Capacity of buffers in router is strictly limited, thus, when the buffer is fully occupied, the injected packet should wait until the buffer has sufficient room.

Evaluation function \(F(t)\), introduced in the previous section, cannot be expressed as a mathematical formula in our problem. We have no other methods than simulation of packet transfer, thus, to evaluate the particle \(t\), we should simulate the network according to the given schedule.

Many PSO textbooks show the termination condition of iterative PSO method as \(F(t) < C_1\) where \(C_1\) is an appropriate threshold value. However, in this paper, we consider the solution of our PSO method as the Gbest vector after the certain PSO steps, since we cannot know the appropriate threshold beforehand.

4. Search Space Reduction

As the previous section states, a naive application of PSO to the packet scheduling problem imposes large search space of \(N \times N \times S\) dimensions. For a concrete discussion, the search space extends to 2048 dimensions when \(N = 16\) and \(S = 8\). Although PSO methods have strong search capabilities in combinatorial problems, too large search space will prevent from conducting satisfactory solutions. We should discuss efficient methods for reducing the search space.

4.1 Naive Model

The fundamental method is naive application of PSO to the packet schedule problem as described in Sect. 3.2. We call the fundamental method naive model.

4.2 Repetitive Model

This paper assumes that each node sends \(S\) packets in one collective communication. The optimization problem is to coordinate the injection schedule of \(N \times N \times S\) packets.

\(^1\)Discussions on different packet sizes are left for our future work.

\(^{11}\)Note that this constraint is applied to each node and each schedule set is independent of each other.

\(^{111}\)The length of the red- and green-colored bar of time corresponds to the packet length.
From the viewpoint of individual computing node, each node sends $S$ packets with reasonable intervals. With rough approximation, we can consider that an $S$-packet communication consists of $S$ sets of simple 1-packet sessions. Suppose that we can find optimal packet schedule for 1-packet session, $S$ repetition of the optimal 1-packet sessions will perform considerably efficient communication.

Whereas we cannot expect strictly optimal solutions by the repetitive method, search space dimension can be drastically reduced. In one PSO operation, a 1-packet session is firstly scheduled and, then, $S$ copies of the scheduled sessions are placed with a fixed interval. We call the method repetitive model.

The repetitive model coordinates only the first schedule time at each node $t_{i,j,0}$, and each succeeding packet is scheduled after a fixed interval $\delta$. Thus, packet schedule at node $(i,j)$ is represented as $(t_{i,j,0}, t_{i,j,0} + \delta, \ldots, t_{i,j,0} + (S - 2)\delta, t_{i,j,0} + (S - 1)\delta)$. The search space is reduced from $N \times N \times S$ dimensions to $N \times N + 1$. Note that “$+1$” corresponds the fixed interval $\delta$.

### 4.3 Expansion Model

If the repetitive model succeeds in enhancing collective communication performance, as we can expect, we will discuss further approximation for reducing search space. As discussed in the previous subsection, we can regard an $S$-packet communication session as a composition of $S$ sets of 1-packet sessions. When the 1-packet session is sufficiently optimized, duration time of the session should be sufficiently short, which shortens the whole duration time of the $S$-packet session.

We name the method as expansion model where a 1-packet session is optimized by PSO and then the optimized schedule is repetitively placed with a fixed interval. This two-step method further reduces search space from $N \times N + 1$ (repetitive model) to $N \times N^3$.

### 4.4 Coding Model

In the packet scheduling problem, each scheduled time is represented as an integer number, since we suppose that the network operates at a synchronous clock signal. Suppose that the $S$-packet collective communication has an upper bound duration time of $t_{\text{max}}$, the search space includes roughly $D^{t_{\text{max}}}$ candidates of packet schedule (where $D$ is the number of dimensions of search space, $D = N \times N \times S$).

The coding model employs a bit-mapped representation of packet schedule at each computing node. The model restricts schedule times of packet injection in an appropriate discrete system. Every schedule time corresponds to a specific bit in a $k$-bit integer. The $i$-th bit in the bit-mapped representation represents the time $\delta \cdot i$ where $i = 0, 1, \ldots, k - 1$. Suppose that $\delta$ is discrete factor given as $\delta = t_{\text{max}}/k$, where

Fig. 2 Bit-mapped representation of packet schedule in coding model.

$t_{\text{max}}$ is the upper-bound duration time.

Schedule times of $S$ packets at each node is represented by the $k$-bit integer. Each node has $k \cdot S$ possible schedule times and the PSO search space is reduced to $N \times N^3$ dimensions. Figure 2 shows a coding example of $S = 8$ and $k = 18$. In this figure, shaded circles correspond to the scheduled times.

In this model, prior to the PSO operations, a bitmap table, which corresponds to Fig. 2, is once generated. Each node has one of the IDs in the bitmap table and actual injection schedule times on the node are determined by the bitmap table. Thus, the search space of the PSO operation consists of the IDs of each computing node. The number of possible candidates in the search space is reduced roughly from $D^{t_{\text{max}}}$ to $(k \cdot S)^{N^3}$.

### 5. Evaluation

#### 5.1 Simulation Environment and Common Conditions

We use two-dimensional torus network. We select 16 × 16 and 8 × 8 configurations as the sizes of the network through a practical limitation of complexity (i.e., computation time in evaluation)\(^{1}\). Packet length is 8 [flits]. The number of virtual channels (VC) is three and each VC has four-flit buffer in each input port of router. Routing algorithm is dimension-order routing (DOR), where every packet firstly goes long $x$-axis and then goes along $y$-axis\(^{1}\). Each computing node injects packets into the 0-th VC in the dedicated port for CPU. Deadlock prevention method is so-called date-line method\(^{1}\). A packet goes through the VC that is assigned by routers after injected at the 0-th one. We set two date-lines in each of $x$- and $y$-axis; $x = 0$ ($y = 0$) and $x = N/2$ ($y = N/2$). When the packet goes across the date-line, its VC is changed to the next lane by increasing by one. For example, when a packet in 0-th VC at (15, 3) router goes to the adjacent router (0, 3) via a wrap-around link, it goes across the date-line at $x = 0$, thus, its VC is changed from 0

\(^{1}\)Finding optimal interval is a simple task as compared to the complex PSO operations.

\(^{1\dagger}\)A 32×32 system requires tremendous evaluation time and it is not realistic at this time.
to 1.

We use following six traffic patterns as typical ones: transpose (trns), perfect shuffle (shfl), bit complement (bcmp), bit reversal (brev), bit rotation (brot), and tornado (torn). Note that random traffic, which is used for many research evaluations, is not suitable for packet scheduling discussions. Thus, we do not use the traffic pattern in this paper.

Table 1 briefly explains the traffic patterns used in this paper. In the table, \((X, Y)\) is two dimensional representation of router address. \(N\) is the number of routers in each of \(x\) and \(y\) dimension and \(N = 2^n\) \((n\) is a positive integer). \(X\) \((Y)\) satisfies \(0 \leq X(Y) < N\) and each of \(X\) and \(Y\) is represented by \(n\)-column binary number: \(Y = w_{2n-1}w_{2n-2} \cdots w_1w_0\) and \(X = w_{n-1}w_{n-2} \cdots w_1w_0\). Furthermore, \(W\) is a concatenated representation of \(Y\) and \(X\) as \(W = w_{2n-1}w_{2n-2} \cdots w_{n+1}w_nw_{n-1}w_{n-2} \cdots w_1w_0\) \((0 \leq W < N^2)\).

5.1.1 Buffer Capacity

Each router has buffers to maintain incoming packets. This paper assumes that each input port in a router has independent buffers for virtual channels. We assume that no packet should be dropped in the network. When two or more packets conflict at a router for using any common resource, only one of them can be transferred to the next router and the others are blocked in the corresponding packet buffers until the contention situation is resolved.

If each buffer has small capacity, a blocked packet saturates the corresponding buffer and the situation spreads rapidly\(^1\). On the other hand, large capacity of buffer absorbs packet conflict. This means that buffer capacity is critical to the evaluation function \(F(t)\); we can expect that large capacity of buffer is insensitive to packet scheduling.

Thus, we should select the minimal buffer capacity that keeps performance. We have measured duration times, varying buffer sizes in several traffic patterns. As Fig. 3 shows the results on \(16 \times 16\) 2D-torus network with dimension-order routing (DOR) and wormhole flow-control, we use four [flits] for buffer capacity in this paper.

5.1.2 Upper-Bound Duration Time

Before we start the PSO evaluations, we should discuss the range of scheduled times. As described in the previous section, our PSO method searches in an \(N^2S\)-dimensional space. This tremendously large search space drives us to discuss the practical range of the schedule time to reduce unnecessary search operations.

As far as the routing algorithm guarantees deadlock-freedom and the number of packets in a collective communication session is limited, the duration time should have upper bound for each operating condition. If we know the practical upper-bound of the duration time, we can eliminate unnecessary search operations in the PSO method, since any of schedule time should not succeed the upper-bound time.

A simple but practical condition is bursty injection without any control mechanisms. We firstly evaluated duration times for the bursty injection. Table 2 includes the results.

5.2 Details of PSO Models

5.2.1 Naive Model (denoted as \(\text{pso}\))

We firstly show the discussion results in the naive PSO method applied to the packet scheduling problem [20]. The basic part of the naive method is given in Sect. 3.2 with \(w = 1.0\) in Eq. (5), but we should add necessary extensions for avoiding local-minima behaviors which PSO methods inherently involve. As the extension from the primitive PSO method, we use following three methods as native models, whose re-initialization procedures for saturated particles differ.

\(^1\)It is known as tree saturation.

---

| abbrevi-ation | description |
|---------------|-------------|
| bcmp          | bit-complement. \(w_{2n-1}w_{2n-2} \cdots w_1w_0 \rightarrow w_{2n-1}w_{2n-2} \cdots w_0\) |
| brev          | bit-reverse. \(w_{2n-1}w_{2n-2} \cdots w_1w_0 \rightarrow w_0 \cdots w_{2n-2}w_{2n-1}\) |
| brot          | bit-rotation. \(w_0w_1 \cdots w_{2n-1} \rightarrow w_0w_{2n-1} \cdots w_1\) |
| shfl          | perfect shuffle. \(w_{2n-1}w_{2n-2} \cdots w_1w_0 \rightarrow w_{2n-1}w_0 \cdots w_{2n-2}\) |
| trns          | transpose. \((X, Y) \rightarrow (Y, X)\) |

Table 1 Traffic patterns used.

Table 2 Duration times of bursty injection and constant intervals \((N = 16, S = 8)\).

| traffic pattern | bursty injection | constant intervals (interval) |
|-----------------|------------------|--------------------------------|
| bcmp            | 420              | 365 (37)                      |
| brev            | 661              | 604 (72)                      |
| brot            | 736              | 664 (70)                      |
| shfl            | 905              | 626 (72)                      |
| torn            | 484              | 464 (34)                      |
| trns            | 525              | 525 (8-64) (unit: cycles)     |

---

![Fig. 3 Buffer size per virtual channel and duration times \((N = 16, S = 8)\).](image-url)
(1) **pso1**
Particles that do not update their personal best (Pbest) scores in ten consecutive steps are re-initialized, since we cannot expect further improvement in their PSO scores. Schedule and velocity vectors are re-initialized.

(2) **pso2**
Particles that do not update their Pbest scores in ten or fifteen consecutive steps are forced to move to their Pbest position and their velocities are re-initialized. This re-initialization intends to increase the search capability of each particle by expanding from the Pbest location. Furthermore, particles that do not update their Pbest in twenty consecutive steps are totally re-initialized, since we cannot expect the particles for further improvements.

(3) **pso3**
pso1 and pso2 models use a uniform random function to select initial schedule time. The resulting schedule times distribute uniformly, however, they hardly show extremely unbalanced cases. For example of the extreme case, some groups of nodes do burst injection of packets while other nodes start injection after large delays. To increase the search capability of the PSO method, wide spectrum of cases should be evaluated. Thus, we further extend the PSO method in initialization of particle position as follows:

- uniform random placement within the upper-bound time,
- many packets are injected at early stages of communication,
- many packets are injected at later stages of communication, and
- packets are injected with a fixed interval.

The optimization problem in this paper employs no local (or neighboring) concept of particles. Thus, Lbest is randomly selected in the Pbests of current particles, since update operation only with Gbest will easily fall into local-minima from where the particle will hardly escape.

To avoid the local-minima, particles that do not update their Pbest for more than ten steps and whose Pbest value exceeds 1.5 times of Gbest are forced to re-initialized. Furthermore, any particle that does not update Pbest for more than twenty steps is forced to re-initialized.

5.2.2 **Repetitive Model (rep)**

In our preceding results [20], the fixed interval with first packet randomly scheduled (rde, Sect. 5.3.6) model achieves preferably good performance. In this evaluation, we inherit the rde principle into the repetitive model and extend as an alternative PSO method.

Actual optimization processes inherit rde except that the 1-packet session is optimized by PSO, not random-selection. We further extend the PSO operation so that it increases local-search capability. As stated in Sect. 3.1, in ordinal PSO methods, particle position (in the search space) and velocity of individual particle is updated at every PSO operation. Particle velocity is updated at every operation so that it will forward to the local or global best particle.

In our modified PSO operation, each particle moves five steps according to its own velocity only. If the score is updated in the five steps, the particle moves further five steps and updates its personal best score. Then, each particle shifts injection schedule of the 1-packet session, as the rde model does (see Sect. 5.3.6). We will discuss effects of the local-search in Sect. 6.6.

5.2.3 **Expansion Model (exp)**

In this model, optimization process employs the same one from the rep model with the only difference of \( S = 1 \) (and \( \delta \) discarded). Throughout the PSO operation steps, every time a new global-best score is found, its packet schedule is recorded into a log-file. After the whole PSO operations completed, all of the recorded Gbest schedules are evaluated according to the expansion model.

In the second-step evaluation, fixed interval value is varied from the length of packet \( L \times N \times 2 \). The smallest duration time among that variation is the resulting score of the expansion model.

5.2.4 **Coding Model (cod)**

The coding model has two parameters that defines resolution and size of the discrete search space, i.e., the number of bits \( k \) used in the bit-mapped representation and the quantization factor \( \delta \). Packet scheduling time is defined within the range of \( 0 \) to \((k-1)\delta \) denoted as \([0 : k\delta)\).

\( k \) corresponds the resolution in the search space. Fine resolution, where \( \delta \) is small, will conduct fine solution, however, it requires a large size of search space. In this paper, we use \( k = 18 \) and the number of possible schedule times is \( 2^{18} = 43758 \), where \( S = 8 \). This coding situation is depicted in Fig. 2.

The resolution \( \delta \) is determined by the upper-bound duration time of bursty injection model \( t_{1pkt} \), the duration time of 1-packet communication \( t_{1pkt} \), and \( k \).

\[
\delta = \left\lfloor (t_{1pkt} - t_{1pkt})/(k-1) \right\rfloor \tag{8}
\]

Table 3 shows the actual \( \delta \) values for each traffic pattern with \( t_{1pkt} \) and \( t_{1pkt} \) that are used in our evaluation.

| Traffic | \( t_{bst} \) | \( t_{1pkt} \) | \( \delta \) |
|---------|--------------|--------------|---------|
| bcmp    | 420          | 61           | 22      |
| brev    | 661          | 80           | 35      |
| brot    | 736          | 85           | 39      |
| shfl    | 905          | 81           | 48      |
| torn    | 484          | 68           | 25      |
| trns    | 525          | 76           | 27      |

(unit: cycles)

Table 3 Resolution in coding model, \( N = 16, S = 8 \).
5.3 Comparison Models

5.3.1 Bursty Model (bst)

This model refuses any packet injection control. Every node starts packet injection at the beginning of collective communication in unison, and remaining packets are successively injected regardless to the congestion situation of the interconnection network.

5.3.2 Throttling Model (eth)

For comparison purpose, we use our dynamic control method named Entropy Throttling [4], [5]. This method suspends packet injection (i.e., throttling), when the network detects congested situation. The method also introduces guard time that inhibits new packet injection. After new packet injection, every node suspends the succeeding packet injection for the given guard time. Even the guard time is passed, each node continues the suspension until it detects uncongested situation of the network. This control method can be regarded as a dynamic packet scheduling according to the degree of congestion.

5.3.3 Fixed Interval (evn)

What is the simplest scheduling is constant interval, where each node sends packets with a certain interval. As the simple scheduling policy works with a single parameter, i.e., interval time, we can regard that the degree of freedom is one.

In this evaluation, every node begins packet injection of its first packet at $t = 0$ in unison with remaining $S - 1$ packets injected by a given interval. Note that all of the nodes inject their packets in unison and the degree of freedom is only one (i.e., interval time). Packet scheduling in this case is formalized as follows. For $Y(x,y)$ nodes, $s$-th packet will be injected at

$$t_{(x,y,s)} = t_{\text{int}} \cdot s,$$  

where $t_{\text{int}}$ is the given interval.

5.3.4 Random-Only Model (rdo)

Every particle is re-initialized at every step. Schedule times are randomly selected. This method intends to show the search capability of the PSO methods.

$$dw = t_{\text{int}} \cdot S$$

$$t_{(x,y,s)} = \text{sort}(dw \cdot \text{rand}())$$

$t_{\text{int}}$ intends an appropriate interval time that is selected experimentally and rand() is a random function with the range $[0 : 1]$.

5.3.5 Random Fluctuation from Fixed Interval Model (frm)

This method extends the degree of freedom from the fixed interval case given in Sect. 5.3.3. Every scheduling time is once placed according to Eq. (9) and then, it is slightly moved forward/backward randomly. Fluctuation of movement has the upper-bound $dw$

$$dw = t_{\text{int}} \cdot 2 \cdot \text{rand}()$$

and the actual schedule time is given by

$$t_{(i,j,s)} = t_{\text{int}} \cdot s + dw \cdot (\text{rand}() - 0.5).$$

Similar to Sect. 5.3.4, $t_{\text{int}}$ is experimentally selected and rand() is a random function with the range $[0 : 1]$.

5.3.6 Fixed Interval with First Packet Randomly Scheduled (rde)

This method places the first packet at every node by random selection. The upper-bound of the first schedule is selected for every particle by Eq. (12) given in Sect. 5.3.5. The scheduled time of the first packet is

$$t_{(x,y,0)} = dw \cdot \text{rand}().$$

The succeeding packets are scheduled only with a simple interval time $t_{\text{int}}$:

$$t_{(x,y,s)} = t_{(x,y,0)} + t_{\text{int}} \cdot s.$$  

As discussed in the previous section, random initial placement offers uniform distribution at some level. This means that there exist a limited number of packets at the initial stage of communication and that packet transfer capability is not sufficiently utilized since many of routers have considerable rooms for further transfer. To minimize the duration time, we should inject many packets at the initial stage for exploiting the maximal capability of the network. Thus, this method firstly evaluates the duration time based on Eqs. (14) and (15) as phase-1, and then, it shifts the schedule times in phase-2. Amount of the shift time $t_{\text{sft}}$ is varied within the range $0 \leq t_{\text{sft}} \leq t_{\text{int}}$ and the $s$-th packet is scheduled at

$$t'_{(x,y,s)} = \begin{cases} 0 & t_{(x,y,s)} < t_{\text{sft}} \\ t_{(x,y,s)} - t_{\text{sft}} & \text{otherwise}. \end{cases}$$

5.4 Evaluation Results

Figure 4 summarizes the duration times where $N = 16$ and $S = 8$. Except the transpose (trns) pattern, where any of the methods could not improve performance, other traffic patterns show considerable performance improvements especially in the rep and exp models.

Table 4 shows speed-up ratios of optimization models used in this paper. In this table, the second left column
shows the duration times in the bst model as comparison bases, and other columns from the third left show speed-up ratios from the bst model. The rep and exp models achieve significant performance improvement in bcmp, brev, brot and shfl traffic patterns. The maximal improvement ratio is 1.72 times (relative to the bursty (bst) communication). We will discuss about the evaluation results from various viewpoints in the next section.

We further executed extensive evaluations for all combinations of \((N = 8, 16) \times (S = 1, 4, 8)\). Tables 5, 6, 7, 8 and 9 show the remaining results. In each table, similar to Table 4, the second left column shows the duration times in the bst model as comparison bases. Other columns from the third left show speed-up ratios from the bst model. Each table shows outstanding performance improvement by the PSO-based method (either rep or exp model).

6. Discussions

6.1 Effects of Injection Intervals

The simple fixed interval method (evn) shows better performance than that of the Entropy throttling method, except brot. The Entropy throttling employs guard time that inhibits the next packet injection for a certain period of time and the guard time contributes largely in performance enhancement. The guard time can be considered as an interval between consecutive packet injection.

From the viewpoint of injection interval, ern, rde, rep, and exp models share a common feature of injection interval. These models achieve relatively good performance, whereas the rdo and pso models suffer insufficient improvement in the duration time. These differences suggest that appropriate injection intervals support communication performance.

6.2 Effects of Search Space

The random-only model (rdo) does not exceed the performance of the evn model. However, the search space of rdo clearly includes the position given by evn. Thus, it is natural to understand that the essential search space of our problem is too large to find appropriate solutions by a simple random method such as rdo. On the other hand, by comparing
Table 7  Speed-up ratios from bursty communication ($N = 8, S = 8$).

| traffic pattern | bst duration | speed-up ratio from bst | speed-up ratio from bst |
|----------------|--------------|-------------------------|-------------------------|
|                |              | evn | rdo | ern | rde | pso | rep | exp |
| bcmp           | 146          | 1.000 | 1.000 | 1.000 | 1.007 | 1.007 | 1.007 | 1.007 |
| brev           | 326          | 1.094 | 1.109 | 1.144 | 1.185 | 1.177 | 1.235 | 1.240 |
| brot           | 244          | 1.162 | 1.025 | 1.070 | 1.814 | 1.070 | 1.208 | 1.208 |
| shfl           | 244          | 1.104 | 1.025 | 1.070 | 1.814 | 1.052 | 1.208 | 1.214 |
| torn           | 262          | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| trns           | 265          | 1.000 | 1.000 | 1.004 | 1.004 | 1.004 | 1.004 | 1.004 |

Table 8  Speed-up ratios from bursty communication ($N = 8, S = 4$).

| traffic pattern | bst duration | speed-up ratio from bst | speed-up ratio from bst |
|----------------|--------------|-------------------------|-------------------------|
|                |              | evn | rdo | ern | rde | pso | rep | exp |
| bcmp           | 82           | 1.000 | 1.000 | 1.000 | 1.012 | 1.012 | 1.012 | 1.051 |
| brev           | 166          | 1.064 | 1.099 | 1.169 | 1.161 | 1.186 | 1.229 | 1.229 |
| brot           | 133          | 1.188 | 1.108 | 1.147 | 1.209 | 1.137 | 1.255 | 1.255 |
| shfl           | 127          | 1.085 | 1.067 | 1.085 | 1.155 | 1.076 | 1.210 |
| torn           | 134          | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| trns           | 137          | 1.000 | 1.000 | 1.007 | 1.007 | 1.007 | 1.007 |

Table 9  Speed-up ratios from bursty communication ($N = 8, S = 1$).

| traffic pattern | bst duration | speed-up ratio from bst | speed-up ratio from bst |
|----------------|--------------|-------------------------|-------------------------|
|                |              | rdo | pso | rep |
| bcmp           | 30           | 1.000 | 1.000 | 1.111 |
| brev           | 46           | 1.150 | 1.179 | 1.179 |
| brot           | 40           | 1.143 | 1.212 | 1.250 |
| shfl           | 40           | 1.081 | 1.176 | 1.212 |
| torn           | 38           | 1.000 | 1.000 | 1.000 |
| trns           | 41           | 1.000 | 1.025 | 1.025 |

Fig. 5  Trajectories of Gbest scores in naive and repetitive models, $N = 16, S = 8$, bcmp traffic.

PSO methods with rdo, our PSO methods reasonably improve performance. This shows that the PSO methods employ considerable search capability.

6.3 Search Behaviors of Naive PSO Models

Figure 5 shows the trajectories of Gbest in the pso and rep models. In this figure, only updated scores are drawn and pso1, pso2, and pso3 show pso1, pso2, and pso3 models, respectively. Furthermore, re1, re2, and re3 show three variants of the repetitive models (rep). Two optimization runs are illustrated for each of the PSO models. This shows that optimization behaviors vary in each run.

Figure 5 shows Gbests of the three pso models, however, it does not show clear difference in overall characteristic in optimization behavior. We counted the number of particles that share the same Pbest score. Figure 6 shows the
distribution of the number of the same Pbest scores along the PSO operation steps. In this figure, dark red color shows frequent occurrence of Pbest scores. In the two cases of Figs. 6(a) and (b), distribution region of Pbest is gradually decreasing as the PSO step proceeds, while Fig. 6(c) does not show remarkable changes.

The \textit{pso1} method shows wider Pbest distribution than that of \textit{pso2} and the distribution is unbalanced in the \textit{pso2} method. The difference clearly comes from update and re-initialization rules. \textit{pso3} extremely distributes Pbest values as Fig. 6(c) shows. This wide distribution corresponds to unbalanced scheduling at initialization of each particle (Sect. 5.2.1).

We then compare the naive PSO models with \textit{ern} and \textit{rde} models. The naive models do not always achieve the best performance. The \textit{rde} model shows better performance than those of \textit{ern}, and it outperforms the naive models in many traffic patterns as Fig. 4 shows. However, the practical effects of \textit{rde} are not stable. In some cases, \textit{rde} outperforms the naive models at a considerable level, whereas, in some other cases, performances are comparable. Although we can guess that the reason comes from large search space and strong attraction in local-minima phenomena, we currently require detailed evaluations and further discussions for effective search methods.

6.4 Search Behavior of Naive and Repetitive Models

Figure 5 also illustrates the clear advantage of the repetitive models in comparison to the naive ones. Large difference in Gbest scores comes from the difference of \textit{pso} and \textit{rep} models, where the latter model introduces fixed interval to reduce search space and enhances local-search capability.

For intuitive understanding, we introduced a graphical representation of the network behavior. Figure 7 shows the common legend for the communication behavior charts. The horizontal axis shows the position of computing nodes and routers and the vertical axis shows time. The chart mainly illustrates the packet density in each router by the shade of color, and different color corresponds different virtual channel: red, green and blue colors show virtual channel (VC) #0, 1, and 2, respectively. For example, a blight red-colored pint shows that the corresponding router has many packets in its own VC-0 buffers. As an opposite example, a black-colored point shows that the router has no packets in its packet buffers. Congested situation in multiple VCs are represented by combination colors such as yellow (red(0) and green(1)), cyan (green(1) and blue(2), and magenta (red(0) and blue(2))

The behavior chart also has small vertical slits between consecutive routers. White segments in the slits show actual packet injection timing (not the predetermined scheduled time). Thus, we can intuitively understand the network situation of packet injection and congestion.

Figure 8 shows the communication behavior of the naive model (\textit{pso2}). Figure 9 is a behavior chart of the \textit{rep} model. As shown in Fig. 8, we cannot distinct 1-packet sessions in the naive model result, while Fig. 9 clearly shows repetitive patterns.

Our initial anxiety in search space reduction was optimization capability, since the reduced methods never answer strict solutions. However, Fig. 4 denies the anxiety. Figures 8 and 9 support the repetitive models. In communication behavior charts, bright colors indicate heavy congestion where packet density is high, and black portions indicate no packets stay in the router. By comparing Figs. 8 and 9, the former has many bright portions where as the latter is relatively controlled.

![Fig. 7](image)

**Fig. 7** Common legend for communication behavior charts.

![Fig. 8](image)

**Fig. 8** Communication behavior of \textit{pso} result, $N = 16$, $S = 8$, bcmp traffic.
6.5 Effect of Intervals in Repetitive Model

Figure 10 shows the behavior charts of \texttt{exp} results. The base 1-packet session (shown in Fig. 10 (a)) is repeatedly placed according to the \texttt{exp} model. The only difference between Figs. 10 (b) and (c) is interval, whose actual interval values are 36 and 37, respectively. This shows that even a small difference in timing cause large difference in communication performance.

Figure 11 shows an alternative representation of the \texttt{exp} model. In this figure, the horizontal axis shows PSO operation step and the vertical axis shows the number of in-flight packets in the interconnection network. The blue curve shows the behavior of the base 1-packet session and other three curves show expanded schedule results for intervals 36, 37, and 38 [cycles]. As the figure shows, the \texttt{exp} model shows repetitive behaviors in the number of in-flight packets. All of the expanded schedule show the same behavior with the 1-packet session in their first session. The
Fig. 11  Behaviors of the expansion model results, $N = 16$, $S = 8$, bcmp traffic.

Fig. 12  Gradient descent effects, $N = 16$, $S = 8$, bcmp traffic.

graph shows that succeeding sessions interfere their preceding sessions.

6.6 Effect of Local Search Behavior

Figure 12 shows effects of local-search introduced in the rep model (Sect. 5.2.2). Each of Figs. 12 (a) and (b) includes 50 lines. Each line corresponds to a local move by a certain velocity vector $v$. The line shows the duration time results according to the successive movement along its own velocity $v$, where $v = (v_1, v_2, \ldots, v_M)$ and $v_i \in \{-1, 0, 1\}$. At $i$-th step, a particle is placed at $x = x_0 + i \cdot v$, where the initial position of the particle is $x_0$.

Figures 12 (a) and (b) show $||v|| = 1$ and $||v|| = 32$ cases, respectively. These results reveal that a large velocity, which has large norm, has high potentials in search operation. The results further indicate that the evaluation function $F(x)$ can be partially differentiated. Note that the differential coefficient does not guarantee monotonically increasing/decreasing features in the duration time function, however, we can expect strong search capability than the naive PSO models.

6.7 Local-Minima

As we can expect the capabilities of the local search, we should clarify the limitation of the local search and show strong effectiveness of the PSO operation. We additionally ran local-search operations of 200 particles for 2,000 steps. Initially each particle is placed at a randomly-selected position in the search space. In every step, each particle generates its velocity vector at random and moves along the velocity vector for 20 times. The particle starts the next step from the best-score position in the 20 trials. Figure 13 shows the trajectories of two particles that are illustrated by red and blue colors. Solid lines in the figure show the global best score at each step.

This figure suggests that the search behavior is readily captured by strong local-minima that performs undesirable duration time, as compared to the PSO results in Fig. 5. Thus, we cannot expect satisfactory solutions only by the local-search. Difference between Figs. 5 and 13 shows strong capability of the PSO operations.

6.8 Major Factors in Repetitive and Expansion Models

As Table 4 shows, the exp model achieves the best performance in bcmp and shfl traffic patterns. As stated in Sect. 5.2.3, the exp model uses all of the intermittent Gbest schedules throughout the PSO operation steps. In many cases of the PSO methods, multiple Gbest schedules are acquired. Each schedule shows the best score at that PSO operation step, although, its simple repetition by the expansion model varies in the duration time. Furthermore, even if two Gbest schedules perform the same duration time in the 1-packet session, their repetitive expansions of eight packets may differ in duration times.

The exp model can be regarded as it piles the basic 1-packet session to the time-axis as a building-block. If the building-block has a thin rectangular shape, we can expect
that the piled blocks are also thin, which correspond to remarkable communication performance. On the other hand, if the building-block is in a concave shape and shapes of its upper and lower curves match to each other, the thickness of the resulting piled blocks will also be sufficiently thin. Thus, if a 1-packet session has a concave shape in the behavior chart, the session never shows the best duration time by itself. However, the expansion model has possibility in achieving the shortest duration time by expanding the 1-packet session.

We further investigate to clarify the major factor that affects the total duration time of the piled blocks. Figure 14 shows the results. Each dot in the graph corresponds to an individual Gbest schedule and the final best-score point is marked blue. All of Figs. 14 (a) to (c) have the same vertical axis that shows duration time. X-axes in Figs. 14 (a), (b) and (c) are base duration time of 1-packet session, thickness of the 1-packet session, and the number of packet interferences in 1-packet session, respectively. We use thickness of the 1-packet session as the largest duration time from the first active time to the last one in every buffer at every router, where active means that a buffer keeps one or more flits of packets. Furthermore, we count interferences when a packet is blocked by another active packet that is transferred to a neighboring router (or the current computing node).

Figures 14 (a) and (b) show weak correlation of the base duration time in the a-packet session and the total duration time. Against our expectation that thin behavior of 1-packet session achieves good performance, the level of correlation is weak. This corresponds to the result depicted in Fig. 10, where only a small difference in the interval parameter results a large difference in the total duration time.

Figure 14 (c) shows that the number of interferences does not always affect the total duration time, whereas we expect that interferences are eliminated by the optimal packet scheduling. The figure suggests that some level of dense packet transfer is necessary for enhancing communication performance.

6.9 Application of the PSO Method

As shown in the evaluation results given in the previous sections, the PSO method offers considerable performance improvement in many traffic patterns. However, we should discuss drawbacks of the method for its effective use.

We recognize that the largest and only drawback is long computation time. As far as it is based on a metaheuristics algorithm, in general, long computation time is inevitable. Furthermore, our PSO method requires a simulation run for every evaluation score of $F(x)$ (in Sect. 3.1), which results in long computation time. For example, one session of the repetitive model (rep) requires about seven days to complete 2,000 PSO steps for 200 particles. Furthermore, from the point of uncertainty, multiple PSO sessions are required to determine reliable and satisfactory results. This paper performs at least four runs for each model. Thus, prior to the parallel execution, we should run necessary PSO sessions to determine packet schedule so that all of the necessary collective communications are scheduled in advance.

7. Conclusions

This paper discusses static packet scheduling methods

---

Footnotes:
1Figure 14 (b) uses ‘nearly busy duration time.’
2The simulator is parallelized by means of MPI (Message Passing Interface) and runs on Intel Core i7-3770S (3.1 GHz clock, 8 GB main memory, Ubuntu 14.04 (server)). Each simulation run requires only less than one second, however, a vast number of runs are executed.
to minimize duration times of collective communication, where each computing node has its own schedule of packet injection that is determined by the scheduling method in advance. The major point of our discussion is to introduce the particle swarm optimization (PSO) method to find the optimal scheduling results with the reasonable complexity in a large search space. This paper firstly introduced the PSO principle to apply to the optimization problem of packet scheduling. As naive applications of PSO, we present three PSO methods through the discussions for avoiding local-minima. The naive methods essentially have vast search space that restrains search results. Thus, we discussed reduction schemes of the search space and proposed three variants, the repetitive model, expansion model, and coding model.

Evaluation results reveal that our PSO methods achieve considerable performance improvement in collective communication, by comparing with several non-PSO methods. Whereas the naive PSO methods do not achieve satisfactory results in some traffic patterns, the repetitive model and expansion model, which are our proposed models, outperform non-PSO methods. These models achieve at most 1.71 time improvement compared to the bursty communication condition.

The PSO algorithm does not guarantee strictly optimal solutions in general. Although this paper achieves some outstanding solutions, these solutions will not be truly optimal and we can expect further optimization. This paper discussed the PSO-based search methods from various points of views with some additional experiments. These discussions conducted important new knowledge on search behaviors. We will continue further discussions for optimal packet communication as our future work.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Numbers 15K00068 and 16K00068.

References

[1] W.J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[2] J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An Engineering Approach, Morgan Kaufmann, 2003.
[3] M. Thottethodi, A.R. Lebeck, and S.S. Mukherjee, “Exploiting global knowledge to achieve self-tuned congestion control for k-ary n-cube networks,” IEEE Trans. Parallel Distrib. Syst., vol.15, no.3, pp.257–272, 2004. DOI: 10.1109/TPDS.2004.1264810.
[4] T. Yokota, K. Ootsu, and T. Ohkawa, “Entropy throttling: Towards global congestion control in interconnection networks,” Proc. 3rd International Symposium on Computing and Networking (CANDAR’15), pp.40–49, Dec. 2015. DOI: 10.1109/CANDAR.2015.58.
[5] T. Yokota, K. Ootsu, and T. Ohkawa, “Enhancing entropy throttling: New classes of injection control in interconnection networks,” IEICE Trans. Inf. & Syst., vol.E99-D, no.12, pp.2911–2922, Dec. 2016. DOI: 10.1587/transinf.2016PAP0007.
[6] E. Baydal, P. López, and J. Duato, “A family of mechanisms for congestion control in wormhole networks,” IEEE Trans. Parallel Distrib. Syst., vol.16, no.9, pp.772–784, 2005. DOI: 10.1109/TPDS.2005.102.
[7] P. López, J.M. Martínez, and J. Duato, “DRIL: Dynamically reduced message injection limitation mechanism for wormhole networks,” Proc. 1998 International Conference on Parallel Processing, pp.535–562, 1998. DOI: 10.1109/ICPP.1998.708527.
[8] M.S. Obaidat, Z.H. Al-Awwami, and M. Al-Mulhem, “A new injection limitation mechanism for wormhole networks,” Computer Communications, vol.25, no.11-12, pp.997–1008, 2002. DOI: 10.1016/S0140-3664(02)00014-2.
[9] T. Yokota, K. Ootsu, F. Furukawa, and T. Baba, “Entropy throttling for maximizing packet mobility in interconnection networks,” IPSJ Trans. Advanced Computing Systems, vol.47, no.SIG13 (ACS 15), pp.1–11, 2006 (in Japanese).
[10] Y. Morie, N. Sueyasu, T. Matsumoto, T. Naori, H. Ishihata, K. Inoue, and K. Murakami, “Optimization of MPI rank allocation considering communication timing for reducing contention,” IPSJ Trans. Advanced Computing Systems, vol.48, no.SIG13 (ACS 19), pp.192–202, Aug. 2007 (in Japanese).
[11] T. Soga, K. Kurihara, T. Naori, M. Kurokawa, and K. Murakami, “Dynamic optimization of load balance in MPI broadcast,” IPSJ Trans. Advanced Computing Systems, vol.1, no.3, pp.67–82, Dec. 2008 (in Japanese).
[12] Y. Gong, B. He, and J. Zhong, “Network performance aware MPI collective communication operations in the cloud,” IEEE Trans. Parallel Distrib. Syst., vol.26, no.11, pp.3679–3689, Nov. 2015. DOI: 10.1109/TPDS.2013.96.
[13] A. Touzene, “On all-to-all broadcast in dense Gaussian network on-chip,” IEEE Trans. Parallel Distrib. Syst., vol.26, no.4, pp.1085–1095, April 2015. DOI: 10.1109/TPDS.2014.2314689.
[14] H. Shibamura, H. Miwa, R. Susukita, T. Hirao, Y. Ajima, I. Miyoshi, T. Shimizu, H. Ishihata, and K. Inoue, “Optimization and simulation evaluation of an all-to-all communication using packet pacing,” IPSJ Trans. Advanced Computing Systems, vol.4, no.3, pp.56–65, 2011 (in Japanese).
[15] H. Shibamura, H. Miwa, I. Miyoshi, and K. Inoue, “Influence of load/network imbalances on collective communication using packet pacing,” IPSJ SIG Technical Report, vol.2012-HPC-135, no.2, pp.1–8, 2012 (in Japanese).
[16] W.J. Dally and H. Aoki, “Deadlock-free adaptive routing in multicomputer networks using virtual channels,” IEEE Trans. Parallel Distrib. Syst., vol.4, no.4, pp.466–475, April 1993. DOI: 10.1109/71.219761.
[17] W. Guo, J. Li, G. Chen, Y. Niu, and C. Chen, “A PSO-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks,” IEEE Trans. Parallel Distrib. Syst., vol.26, no.12, pp.3236–3249, Dec. 2015. DOI: 10.1109/TPDS.2014.2386343.
[18] Y. Xu, K. Li, L. He, L. Zhang, and K. Li, “A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems,” IEEE Trans. Parallel Distrib. Syst., vol.26, no.12, pp.3208–3222, Dec. 2015. DOI: 10.1109/TPDS.2014.2385698.
[19] T. Saito, “Particle swarm optimizers and nonlinear systems,” IEICE Fundamentals Review, vol.5, no.2, pp.155–161, 2011 (in Japanese).
[20] T. Yokota, K. Ootsu, and T. Ohkawa, “Introducing PSO for optimal packet scheduling of collective communication,” Proc. 4th International Symposium on Computing and Networking (CANDAR’16), pp.429–435, Dec. 2016. DOI: 10.1109/CANDAR.2016.0080.
Takashi Yokota  received his B.E., M.E., and Ph.D. degrees from Keio University in 1983, 1985 and 1997, respectively. He joined Mitsubishi Electric Corp. in 1985, and was engaged in several research projects in special-purpose, massively parallel and industrial computers. He was engaged in research and development of a massively parallel computer RWC-1 at Real World Computing Partnership as a senior researcher from 1993 to 1997. From 2001 to 2009, he was an associate professor at Utsunomiya University. Since 2009, he has been a professor at Utsunomiya University. His research interests include computer architecture, parallel processing, network architecture and design automation. He is a member of IPSJ, IEICE, ACM and the IEEE Computer Society.

Kanemitsu Ootsu  received his B.S. and M.S. degrees from University of Tokyo in 1993 and 1995 respectively, and later he obtained his Ph.D. in Information Science and Technology from University of Tokyo in Japan. From 1997 to 2009, he is a research associate and then an assistant professor at Utsunomiya University. Since 2009, he is an associate professor at Utsunomiya University. His research interests include high-performance computer architecture, multi-core multithread processor architecture, mobile computing devices, binary translation, and dynamic optimization. He is a member of IPSJ, IEICE, and ISCIE.

Takeshi Ohkawa  received his B.E. and M.E. and Ph.D. degrees in Electronics Engineering from Tohoku University in 1998, 2000 and 2003, respectively. He was a postdoctoral fellow at Tohoku University, where he engaged in the reconfigurable LSI and software/hardware co-design in 2003. In 2004, he joined National Institute for Advanced Industrial Science and Technology (AIST), Japan as a researcher, where he started working on distributed object technology and its application to FPGA. From 2009, he started work in start-up company TOPS systems, where he engaged in R&D of heterogeneous multi-core processor and its software platform. From 2011, he is an assistant professor in Utsunomiya University. His research interests include reconfigurable technology, software/hardware co-design, parallel and distributed architecture and component-based design technology. He is a member of ACM, IEICE, and IPSJ.