Static Scheduling of a Time-Triggered Network-On-Chip based on Memetic Algorithm

Heyuan Shi
School of Software
Tsinghua University
Beijing, China
Email: hey.shi@foxmail.com

***Abstract*—Time-Triggered Network-on-Chip (TTNoC) is one of the practical applications of real-time systems, focused on the reliable and safety-critical communication, for modern multiprocessor system. The TTNoC allows designer to achieve communication with more nodes and messages, but the scheduling of each message is a challenging problem because designer must consider both path contention and the temporal constrains. We introduce a genetic algorithm, and then integrate local search into the genetic algorithm, so called memetic algorithm, with stochastic ranking strategy to solve this problem. The experimental results show that the better performance of our memetic algorithm in compare with genetic algorithm.**

I. Introduction

Many safety-critical real-time embedded systems desire the predictable communication latency. The temporal constrains of system must be guaranteed []. Time-Triggered system is one of the real-time architectures which isolate the inherent fault and meet the temporal predictability of communication. Therefore, Time-Triggered Network-on-Chip (TTNoC) is efficient and ideal for safety-related embedded systems because it satisfies the strict timing and dependability requirements [].

At present, time-triggered communication comes in various fields. e.g. SAFEbus deployed in the avionic industry [], TTP and FlexRay adopted in automotive [] and TTEthernet in the domain of railway industry []. In particular, many time-triggered Networks-on-Chip architectures have been proposed (e.g., GENESYS MPSoC, Aethereal) [].

A traditional time-triggered architecture is bus-based system where all messages are scheduled in time slots and for every time slot, there is only one message which can be transmitted. To design such systems, the designer pays attention to the time domain to guarantee that all messages are non-overlapping in time domain. Each message owns its unique offset time point. In this case, a feasible schedule is an order for messages such that no two messages can be sent at the same time.

The Time-Triggered Network-on-Chip has been proposed because the current bus-based time triggered systems cannot meet the requirements of development of Multiprocessor System-on-Chip (MPSoC) []. While NoCs provide more link resources to forward messages, it is more complex to synthesize a feasible scheduling for TTNoC since the conflict-free paths through the network-on-a-chip must be considered. The TTNoC allows several message to be transmitted at the same time slot, if their routing paths are non-overlapping. It entails

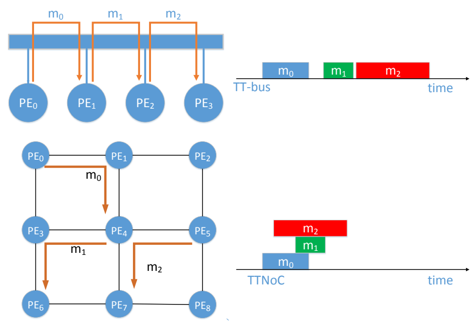


Fig. 1. Difference between bus- and network-based time-triggered architecture

messages share the time slot on the TTNoC. In this case, the designer must consider the feasible scheduling of messages in both time-domain and space-domain. In other words, any two messages cannot occupy the same network link at the identical time point. If messages coexist in time slot, the forwarding path of these messages should be conflict-free. If messages are using identical link to be forwarded, the existence time of these messages cannot be overlapped.

Time-Triggered system resolves contention on the network and avoids dynamic arbitration of intercommunicate resources distribution by the synchronizing time of communication entities and static schedule upon the global synchronized time []. With the control of a global time imposed by the system, the local time of every node on the chip is synchronized to each other. The offset time point of each message (the injection of each messages), from any nodes on the network, is defined by a static schedule table, and the switches on the network also transmit the messages in terms of the schedule table. Therefore, all the messages are received, transmitted or sent at a deterministic time point with the correctness and predictability of real-time communication. However, synthesizing such an off-line schedule table is rather complicated which is viewed as a bin-packing problem, known as NP-complete [].

This paper presents a static scheduling approach to solve the TTNoC scheduling problem by the Memetic Algorithm. Memetic Algorithm is widely used as a synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search.

The rest of this paper is organized as follows. We first review the related work in the TTNoC scheduling Section II. The problem definition and the constrains about the problem (Section III). Then the memetic algorithm with stochastic ranking has been proposed to find the feasible solution (Section IV). Finally, the memetic algorithm was implemented to compare with other SMT-based static scheduling approaches, and the related discussion was also included (Section V).

II. System Model

A. Architecture Model

A TTNoC is a system with a set of m real-time tasks/applications $\Gamma = \{\tau_1, \ldots, \tau_m\}$ mapped on a multi-core interconnection platform, composed of a set of identical processing elements interconnected by physical links and network switches. The topologies of a TTNoC can be either regular or irregular.

The system model is formulated by a direct graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$. The set of vertexes $\mathcal{V} = \{v_1, \ldots, v_m\}$ represents the set of m identical communication nodes (the processing elements and switches) while the edges $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{V}$ comprises the communication physical links connecting the nodes.

The physical links are full-duplex, allowing thus communication in both directions. Therefore we have that $\forall v_a, v_b \in \mathcal{V}, (v_a, v_b) \in \mathcal{L} \Rightarrow (v_b, v_a) \in \mathcal{L}$, where (v_a, v_b) is an ordered tuple denotes a directed logical link connecting from a source node $v_a \in \mathcal{V}$ to a sink node $v_b \in \mathcal{V}$. A network link (v_a, v_b), between the source node v_a and the sink node v_b, is defined by the tuple $\langle (v_a, v_b).w, (v_a, v_b).sd, (v_a, v_b).pd, (v_a, v_b).mt, (v_a, v_b).h \rangle$, where $(v_a, v_b).w$ is the width of physical links, $(v_a, v_b).sd$ is the switching delay in the switches on network link, $(v_a, v_b).pd$ is the link delay per hop which means propagation cost between adjacent nodes, $(v_a, v_b).mt$ is the macrotick of network denotes the time-line granularity of the physical link and $(v_a, v_b).h$ is the number of hops from v_a to v_b on the link (v_a, v_b).

For a communication node $v_i \in \mathcal{V}$, it offer d input links and d output links, d is the switch degree. Therefore a switch is able to connected with at most d port in full-duplex connection. The adjoint switches $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ which connect with each other through physical link rather than the forwarding by other switches, is defined by the sequence $< v_i, v_j >$. Therefore the network link (v_a, v_b) can be split in several connection among the adjoint switches $v_i \in \mathcal{V}$, and we have that $(v_a, v_b) = \{< v_a, v_1 >, \ldots, < v_i, v_j >, \ldots, < v_n, v_b >\}$.

B. Message Model

The application on the TTNoC consists of tasks, which transmitted by messages. Only periodic messages are discussed in this paper because it is the typical case in real-time embedded systems. But in practice, the fixed time slot can be preserved in terms of the route and deadline periodically for sporadic messages. According to the timing specification of TTNoC, the period must be a positive power of two in terms of macrotick, i.e. the messages are harmonic [].

We denote the set of m time-triggered messages on the TTNoC by $\Gamma = \{\tau_1, \ldots, \tau_m\}$. Each message τ_i is defined by the tuple $\langle \tau_i.s, \tau_i.t, \tau_i.d \rangle$, where $\tau_i.s$ denotes the size of message in byte, $\tau_i.t$ denotes the period of message and $\tau_i.d$ is the relative deadline of message.

C. Communication Model

When a communication happened on the TTNoC, i.e. the communication source node $v_a \in \mathcal{V}$ send a message τ_i to the communication sink node $v_b \in \mathcal{V}$ on the TTNoC, the message τ_i is transmitted on the network link (v_a, v_b) from the source node v_a to the sink node v_b. And the communication nodes on the link (v_a, v_b) forward the message.

Based on the architecture and message model, we can model the set of message communication $\mathcal{M} = \{m_1, \ldots, m_i\}$, where $m_i \in \mathcal{M}$ denotes the communication scheduling of a message $\tau_i \in \Gamma$. For each message $\tau_i \in \Gamma$ on the link (v_a, v_b) where $v_a, v_b \in \mathcal{V}$, we model the communication $m_i \in \mathcal{M}$ as a tuple $\langle m_i^{(v_a,v_b)}.\phi, m_i^{(v_a,v_b)}.\mathcal{T}, m_i^{(v_a,v_b)}.\mathcal{D}, m_i^{(v_a,v_b)}.\mathcal{L} \rangle$, where $m_i^{(v_a,v_b)}.\phi$ is the offset of communication, $m_i^{(v_a,v_b)}.\mathcal{T}$ is the period of communication, $m_i^{(v_a,v_b)}.\mathcal{D}$ is the relative deadline of communication, $m_i^{(v_a,v_b)}.\mathcal{L}$ is the duration of communication and (v_a, v_b) in the tuple is the network link for communication.

The granularity of communication is macrotick, for a communication m_i, we have that

$$m_i^{(v_a,v_b)}.\mathcal{T} = \lceil \frac{\tau_i.period}{(v_a, v_b).mt} \rceil \tag{1}$$

$$m_i^{(v_a,v_b)}.\mathcal{D} = \lceil \frac{\tau_i.deadline}{(v_a, v_b).mt} \rceil \tag{2}$$

$$m_i^{(v_a,v_b)}.\mathcal{L} = \lceil \frac{(v_a, v_b).h \times (v_a, v_b).d \times \lceil \frac{\tau_i.s + (v_a,v_b).w}{(v_a,v_b).w} \rceil}{(v_a, v_b).mt} \rceil \tag{3}$$

where $(v_a, v_b).d = (v_a, v_b).sd + (v_a, v_b).pd$ denotes the delay of propagating and switching delay per hop.

For $m_i \in \mathcal{M}$, The $m_i^{(v_a,v_b)}.\mathcal{T}$, $m_i^{(v_a,v_b)}.\mathcal{D}$ and $m_i^{(v_a,v_b)}.\mathcal{L}$ is derived by formula (1), (2) and (3). Formula (1) and (2) denote the period and deadline of the communication model in macrotick. (3) is the delay denotes the time cost in macrotick, from the first byte sent by source node to the last byte received by sink node. And according to store-and-forward(SAF) switching [].

To synthesize a set of communication $\mathcal{M}$, We define $\mathcal{M}.\Phi$ $\mathcal{M}.\mathcal{T}$, $\mathcal{M}.\mathcal{D}$ and $\mathcal{M}.\mathcal{L}$ as the set of offset, period, relative deadline and delay of each communication $m_i \in \mathcal{M}$.

Synthesizing the set of communication $\mathcal{M}$ is to determine the offset $m_i^{(v_a,v_b)}.\phi$ for each communication $m_i \in \mathcal{M}$. We define the set of offset for each communication, i.e. $\mathcal{M}.\Phi = \{m_i.\phi \mid m_i \in \mathcal{M}\}$. Therefore the key of scheduling of TTNoC is to determine $\mathcal{M}.\Phi$ based on the given architecture model $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ and message model Γ with scheduling constrains.

III. SCHEDULING CONSTRAINS

Scheduling each communication $m_i \in \mathcal{M}$ should resolve the constrains on the TTNoC. The synthesized scheduling must satisfy the separation among the set of communication in time domain or space domain, which mainly include offset constrains and link constrains.

A. *Offset Constrains*

For a scheduled communication m_i , the offset $m_i^{(v_a,v_b)}.\phi$ must be positive values and should guarantee communication will be completed before the deadline of the communication. Therefore we have that

$$(m_i^{(v_a,v_b)}.\phi \geq 0) \cap (m_i^{(v_a,v_b)}.\phi + m_i^{(v_a,v_b)}.\mathcal{L} \leq m_i^{(v_a,v_b)}.\mathcal{D}) \tag{4}$$

B. *Link Constrains*

For each communication, the message occupies the link in terms of routing strategy without contention, until all the message is received by sink node. The scheduling $\mathcal{M}$ should guarantee that no two messages that are transmitted on the same time at any time.

Therefore for any two of the communication $m_i, m_j \in \mathcal{M}, m_i \neq m_j$, we define $overlap(m_i, m_j)$, denotes whether there is common routing link between m_i and m_j or not. $overlap(m_i, m_j)$ is equals to 1 if two communications share at least one hop routing link. Therefore we have that

$$overlap((m_i^{(v_a,v_b)}, (m_j^{(v_c,v_d)}) = \begin{cases} 0 & (v_a, v_b) \cap (c_c, v_d) = \emptyset \\ 1 & (v_a, v_b) \cap (c_c, v_d) \neq \emptyset \end{cases} \tag{5}$$

The duration of communication m_i in kth period is defined by $m_i^{(v_a,v_b)}(k)$, which equals

$$[k \times m_i^{(v_a,v_b)}.\mathcal{T}, k \times m_i^{(v_a,v_b)}.\mathcal{T} + m_i^{(v_a,v_b)}.\phi + m_i^{(v_a,v_b)}.\mathcal{L}] \tag{6}$$

Therefore for any $m_i^{(v_a,v_b)} \in \mathcal{M}, m_j^{(v_c,v_d)} \in \mathcal{M}$, we have that

$$(v_a, v_b) \cap (c_c, v_d) \neq \emptyset \implies m_i^{(v_a,v_b)}(p) \cap m_j^{(v_c,v_d)}(q) = \emptyset \tag{7}$$

where $p, q \geq 0$, $p, q \in \mathcal{Z}$.

IV. ASSUMPTION

Our assumptions are given as follows.

The topology of TTNoC $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ is given.

The mapping between a message and a communication node $\{< \tau_i, v_j >| \tau_i \in \Gamma, v_j \in \mathcal{V}\}$ is given. Each message $\tau_i \in \Gamma$ is allocated to a specific process element(communication node) $v_i \in \mathcal{V}$, thus the source node v_a and sink node v_b in $(v_a, v_b) \in \mathcal{L}$ are determined.

The routing strategy of each message is given. Therefore the link (v_a, v_b) and the set of switches for forwarding the message is deterministic.To transmit a message $\tau_i \in \Gamma$, the forwarding path is split in terms of hops by specific routing, i.e. $(v_a, v_b) = \{(v_a, v_m), (v_m, v_n), \ldots, (v_p, v_q), (v_q, v_b)\}$.

The switching technique is store-and-forward.

The latency of forwarding in switch as well as the propagation latency on links is constant.

In our experiment, for simplicity, the relative deadline is equal to the period for each message. The 2-ary mesh network is used and we randomly generate the mapping on the network. The X-Y routing is adopted as the routing strategy. And the relative deadline for each message equal its period. However the algorithm we introduced can be extended to resolve the scheduling problem with the general topology and arbitrary deterministic routing.

V. PROBLEM FORMULATION

The problem we are addressing in this paper can be formulated as follow. Given

(1) the architecture of network $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ consists of the set of communicating nodes $\mathcal{V} = \{v_1, \ldots, v_m\}$ and interconnection links $\mathcal{L} = \{(v_a, v_b) \mid v_a, v_b \in \mathcal{V}\}$.

(2) the mapping of message to communication node $\{< \tau_i, v_j >| \tau_i \in \Gamma, v_j \in \mathcal{V}\}$.

(3) the routing strategy.

(4) the set of tt-messages $\Gamma = \{\tau_1, \ldots, \tau_n\}$ with $\langle \tau_i.s, \tau_i.t, \tau_i.d \rangle$ for $\tau_i \in \Gamma$.

we aim to find a set of communication scheduling $\mathcal{M} = \{m_1, \ldots, m_i\}$ for each $\tau_i \in \Gamma$ to satisfy the set of given scheduling constrains.

Deriving the set of communication scheduling $\mathcal{M}$ means determine $\langle m_i^{(v_a,v_b)}.\phi, m_i^{(v_a,v_b)}.\mathcal{T}, m_i^{(v_a,v_b)}.\mathcal{D}, m_i^{(v_a,v_b)}.\mathcal{L} \rangle$ for each $m_i \in \mathcal{M}$. The $m_i^{(v_a,v_b)}.\mathcal{T}$,$m_i^{(v_a,v_b)}.\mathcal{D}$ and $m_i^{(v_a,v_b)}.\mathcal{L}$ can be derived easily through formula (1),(2) and (3), respectively. Nevertheless, our goal is to find the set of offset $\mathcal{M}.\Phi$ such that satisfies the offset and link constrains. If it is impossible to synthesize a set of scheduled communication, then we try to minimize the amount of infeasible messages which violate the scheduling constrains.

VI. MEMETIC ALGORITHM

Memetic algorithm(MA) is a population-based hybrid genetic algorithm(GA) coupled with an individual learning procedure capable of performing local refinements [].

Our memetic algorithm consists of two parts. We deploy the genetic algorithm as global searching for the set of communication to be scheduled. And choosing the subset of infeasible communications as the element to local search.

The MA requires genetic representation, fitness function as well as evolution strategy, i.e. crossover, mutation and local search. We firstly give an example of communication scheduling to explain the genetic representation. Next the pseudocode is shown. And then the detail will be discussed.

A. *An Example of Scheduling on TTNoC*

To show our memetic algorithm, we firstly give an example of a communication set $\mathcal{M}$ on a 3×3 mesh NoC, shown in figure II. There are a set of communication $\mathcal{M} = \{m_0, m_1, m_2, m_3, m_4\}$ to be scheduled. Each communication $m_i \in \mathcal{M}$ own its period, duration as well as link in terms of the given routing strategy, shown in TABLE I. Since the relative

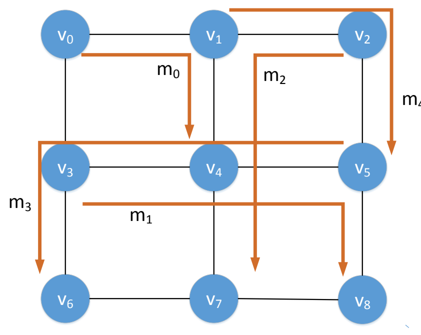


Fig. 2. An example of communication set on the TTNoC

TABLE I
An Example of a communication set $\mathcal{M}$

communication	link	period	duration	possible offset
m_0	$(0,4)$	2	1	$\{0,1\}$
m_1	$(3,8)$	4	1	$\{0,1,2,3\}$
m_2	$(2,7)$	4	1	$\{0,1,2,3\}$
m_3	$(5,6)$	8	2	$\{0,1,2,3,4,5,6\}$
m_4	$(1,5)$	8	1	$\{0,1,2,3,4,5,6,7\}$

deadline is equal to period for each message, the possible offset $m_i^{(v_a,v_b)}.\phi$ of a communication is derived in terms of offset constrains. Therefore we have that $0 \leq m_i.\phi \leq m_i.\mathcal{D} - m_i.\mathcal{L}$. The set of possible offset $m_i.\Phi = \{0, \ldots, m_i.\mathcal{T} - m_i.\mathcal{L}\}$ of each communication is also shown in TABLE I.

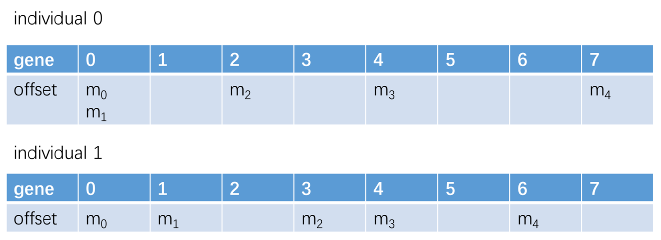
B. Genetic Representation

The genetic algorithm deployed as global search consists of gene, chromosome, individual and population. The population consists of a set of individual and each contains a chromosome which represents a solution to the given question. A chromosome composes of several gene, which as the unit of chromosome [wiki]. We will define them on the field of scheduling on TTNoC.

In our memetic algorithm, we adopt hyperperiod $\mathcal{H}(\mathcal{M}) = LCM\{\mathcal{M}.\mathcal{T}\}$, which is the least common multiple among the set of period of communication set, as the chromosome definition. And the macrotick which is the unit of hyperperiod, is the gene. The length of chromosome equals $\mathcal{H}(\mathcal{M})$, e.g. the hyperperiod is 8 for the given example in TABLE.

The location of a gene on the chromosome denotes the offset $m_i^{(v_a,v_b)}.\phi$ of a communication scheduling $m_i^{(v_a,v_b)} \in \mathcal{M}$. The offset of each communication is distributed on a gene. Therefore a chromosome represents a possible allocation of the offset set of the communication set $\mathcal{M}.\Phi$ It should be noted that a single gene may consists multiple offset of communication, since several communication is able to offset at same time based on the TTNoC architecture.

A set of individual with respective chromosome composes the population. The size of population equals the number of individual. An example of population with two individual, based on TABLE, is shown in Fig.


individual 0

gene	0	1	2	3	4	5	6	7
offset	m_0 m_1		m_2		m_3			m_4

individual 1

gene	0	1	2	3	4	5	6	7
offset	m_0	m_1		m_2	m_3		m_4	

Fig. 3. An example of population with 2 individual

C. Pseudocode

According to the genetic representation, we formal give the algorithm process. The pseudocode is given as follow.

The memetic algorithm begins with an initial population. The size of population as well as the maximum iterations is predefined. For a communication $m_i \in \mathcal{M}$, the offset of each communication $m_i.\phi$ in the initial individual is located randomly among the possible offset $m_i.\Phi$. Therefore the initial population satisfies the offset constrains. The algorithm is iterative executed until a feasible scheduling is found or reaching the given maximum iteration.

When the memetic algorithm get started, firstly the initial population is marked by the fitness function. The detail of fitness function is discussed in *subsection D*. Every time marking the population, a score is derived to evaluate each individual. And the less the number of score, the better the individual. Therefore the individual with score equals zero represent a feasible scheduling of the communication set.

Next, the operation of genetic algorithm, i.e. crossover and mutation, is deployed as global search. And after that, the local search is adopted to the population, if there are no feasible scheduling(individual with score of 0). The detail of search strategy deployed in global as well as local search, is discussed in *subsection E* and *subsection F*, respectively.

After the global and local search, each individual in the population is evaluated by fitness function again. And the new population for the next iteration is generated by selection according to the score of each individual. The individual with less score own more possibility to reserve for the next population. The selection function ensure the size of population is equal to the predefined size in each iteration.

D. Fitness function

To adopt the fitness function to evaluate individuals, we firstly transform chromosome representing the offset of each communication, into the representation of the duration of each communication in the hyperperiod. Next we derive the link constrains among the set of communication according to the routing strategy, as the basis of our fitness function. Then the score of each individual is able to derived by fitness function.

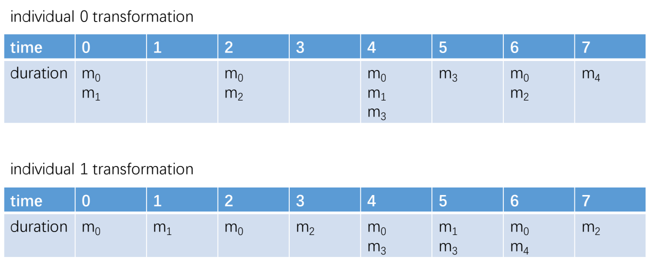
For each communication $m_i \in \mathcal{M}$, The duration of communication m_i in kth period $m_i^{(v_a,v_b)}(k)$, is defined by formula (6). And we have that

$$m_i.\mathcal{H} = \{m_i^{(v_a,v_b)}(k) \mid k \in [0, \mathcal{M}.\mathcal{H}/m_i.\mathcal{T} - 1]\} \quad (8)$$

```
Algorithm 1 Memetic Algorithm()
Input:
    the size of population $scale$
    maximum iterations $maxIteration$
Output:
    Best scheduling $\mathcal{S}$
 1: for $i = 0$ to $maxIteration - 1$ do
 2:   if $i = 0$ then
 3:     // mark the initial population
 4:     generate the initial population $initialPop$
 5:     mark($initialPop$)
 6:     $\mathcal{S}$ = individual with least score in $initialPop$
 7:     $Pop = initialPop$
 8:   else
 9:     // global search
10:     crossover($Pop$)
11:     mutation($Pop$)
12:     mark($Pop$)
13:   end if
14:   // choose the individual best so far after global search
15:   for all individual $indi$ in $Pop$ do
16:     if $indi.score < \mathcal{S}.score$ then
17:       $\mathcal{S} = indi$
18:     end if
19:     if $\mathcal{S}.score == 0$ then
20:       break
21:     end if
22:   end for
23:   localSearch($Pop$)
24:   mark($Pop$)
25:   // choose the individual best so far after local search
26:   for all individual $indi$ in $Pop$ do
27:     if $indi.score < \mathcal{S}.score$ then
28:       $\mathcal{S} = indi$
29:     end if
30:     if $\mathcal{S}.score == 0$ then
31:       break
32:     end if
33:   end for
34:   select($Pop$, $scale$)
35: end for
36: return $\mathcal{S}$
```

where $m_i.\mathcal{H}$ denotes the duration in hyperperiod of a communication in hyperperiod. We define $\mathcal{M}.\mathcal{H} = \{m_i.\mathcal{H} \mid m_i \in \mathcal{M}\}$, denotes the set of duration for each communication in hyperperiod. Therefore, the complete duration in hyperperiod for each communication $\mathcal{M}.\mathcal{H}$ is determined based on the set of communications $\mathcal{M}$. e.g. Fig is the complete duration in a hyperperiod based on the offset in Fig and the period as well as duration in TABLE.

The link constrains should be derived after the transformation of individual. According to the predefined routing strategy, i.e. XY routing, the network link (v_a, v_b) of each communication $m_i \in \mathcal{M}$ is derived. Therefore we have the forwarding path per hop $(v_a, v_b) = \{< v_a, v_1 >, \ldots, < v_i, v_j >, \ldots, < v_n, v_b >\}$, where $m_i^{(a,b)} \in \mathcal{M}$. By checking the routing of each communication by formula (5) in link constrains, the set of overlapped communication for each $m_i \in \mathcal{M}$ is derived. We denote the overlapped set of communication $m_i \in \mathcal{M}$ by $O(m_i)$. And we have that


individual 0 transformation

time	0	1	2	3	4	5	6	7
duration	m_0 m_1		m_0 m_2		m_0 m_1 m_3	m_3	m_0 m_2	m_4

individual 1 transformation

time	0	1	2	3	4	5	6	7
duration	m_0	m_1	m_0	m_2	m_0 m_3	m_1 m_3	m_0 m_4	m_2

Fig. 4. The duration in the hyperperiod based on Fig and TABLE

TABLE II
AN EXAMPLE OF A ROUTING AND OVERLAP SET FOR $\mathcal{M}$

communication	routing link	overlap set
m_0	$< 0, 1 >< 1, 4 >$	m_2, m_4
m_1	$< 3, 4 >< 4, 5 >< 5, 8 >$	m_4
m_2	$< 2, 1 >< 1, 4 >< 4, 7 >$	m_0, m_4
m_3	$< 5, 4 >< 4, 3 >< 3, 6 >$	
m_4	$< 1, 4 >< 4, 5 >$	m_0, m_1, m_2

$$O(m_i) = \{m_j \mid overlap(m_i, m_j) = 1, m_i, m_j \in \mathcal{M}\} \quad (9)$$

i.e. the $overlap(m_0, m_1) = 0$, $overlap(m_0, m_2) = 1$, $overlap(m_0, m_3) = 0$ and $overlap(m_0, m_4) = 1$, thus overlap set $O(m_0)$ is $\{m_2, m_4\}$. Table II shows the routing and overlapped communication for communications based on TABLE I.

To evaluate individuals, for $m_i, m_j \in \mathcal{M}, m_i \neq m_j$, if two communication in the same overlap set, the conflict times between them are derived. We define the mapping $(m_i, m_j) \rightarrow n_{ij}$, where (m_i, m_j) is the conflict communication and n_{ij} is the number of gene which m_i coexist with m_j on a chromosome is n_{ij}, which denotes the conflict times between m_i and m_j.

For each communication $m_i \in \mathcal{M}$, we define $C(m_i)$ to derive the communication which conflicted with m_i and the conflict times. Therefore we have that

$$C(m_i) = \{(m_i, m_j) \rightarrow n_{ij} \mid m_j \in O(m_i), m_i, m_j \in \mathcal{M}\} \quad (10)$$

We define the total conflict times to m_i as $C(m_i).value$. And we have that

$$C(m_i).value = \sum n_{ij} \quad (11)$$

If there is no conflict communication with m_i on the transformation of individual , the value of $C(m_i)$ equals 0. We define $C(\mathcal{M}) = \{C(m_i) \mid m_i \in \mathcal{M}\}$ to denote the conflict mapping set for all the communication set $\mathcal{M}$.

TABLE III
AN EXAMPLE OF MARKING TWO INDIVIDUAL BY FITNESS FUNCTION

$C(\mathcal{M})$ of individual 0	conflict mapping	score
$C(m_0)$	$(m_0, m_2) \rightarrow 2$	4
$C(m_2)$	$(m_2, m_0) \rightarrow 2$	
$C(\mathcal{M})$ of individual 1	conflict mapping	score
$C(m_0)$	$(m_0, m_4) \rightarrow 1$	2
$C(m_4)$	$(m_4, m_0) \rightarrow 1$	

According to $C(m_i)$ for each $m_i \in \mathcal{M}$, the conflict times among any two of communication on the chromosome is determined. The fitness function defined to give each individual a score when executing $mark(Pop)$ in line 5, 12, 24 of the memetic algorithm. The output of fitness function is the sum of n_{ij} in $C(m_i)$ for each communication $m_i \in \mathcal{M}$. Therefore for a individual $indi$ with p communication, we have that

$$fitness(indi) = \sum_{i=0}^{p} C(m_i).value \tag{12}$$

where $m_i \in \mathcal{M}$. The score is assigned to the individual after fitness function. Since the score represents the conflict times, the less the score the better the individual. e.g. The $C(\mathcal{M})$ and the score of two individual is shown in Fig, based on the transformation in Fig and the overlap set in TABLE.

E. Global search strategy

The general operation in genetic algorithm, i.e. crossover and mutation, is used in the step of global search.

For operation of crossover, we select two of individual among the population the parent to join the crossover in terms of the score. The higher scores of the individual, the higher possibility to be selected as parent. the crossover generates a new individual and the chromosome of the new individual depends on the parent. The individual of parent with lower score own the higher possibility to determine the offset location on the chromosome. For the parent individual $indi0$ and $indi1$, we have the function

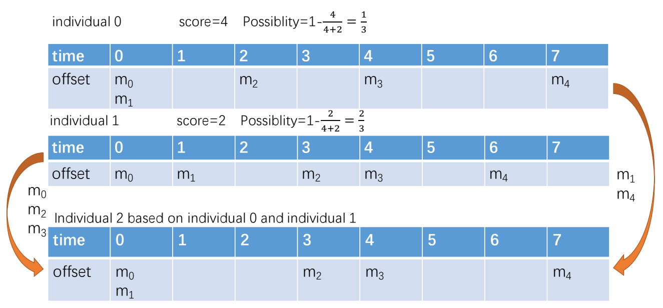
$$possibility(indi0) = 1 - \frac{indi0}{indi0 + indi1} \tag{13}$$

to derive the possibility for each individual. e.g. Fig shows an process of crossover. The individual 2 is generated by crossover based on the two individual in Fig as parent.

The operation of mutation random select an offset of communication. Then selecting a new location of its offset randomly. The probability of mutation for each individual in our implement is 50%.

F. Local search strategy

In the step of local search, for each individual, the communication m_i with the maximum value of $C(m_i).value$ is selected as local search element. If there is two or more individual with the maximum conflict times, the element to be searched is select stochastically among them. After determining the element for local searching, the offset of selected m_i is



individual 0 score=4 Possiblity=1-$\frac{4}{4+2}$ = $\frac{1}{3}$
time 0 1 2 3 4 5 6 7
offset m0 m1 m2 m3 m4
individual 1 score=2 Possiblity=1-$\frac{2}{4+2}$ = $\frac{2}{3}$
offset m0 m1 m2 m3 m4
m0 m2 m3
m1 m4
Individual 2 based on individual 0 and individual 1
offset m0 m1 m2 m3 m4

Fig. 5. crossover

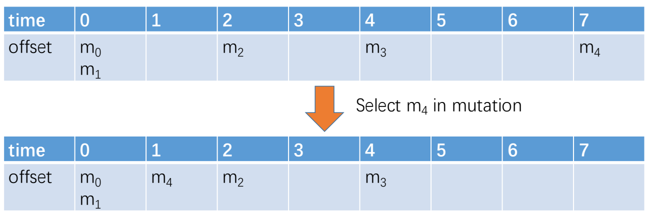


Fig. 6. mutation

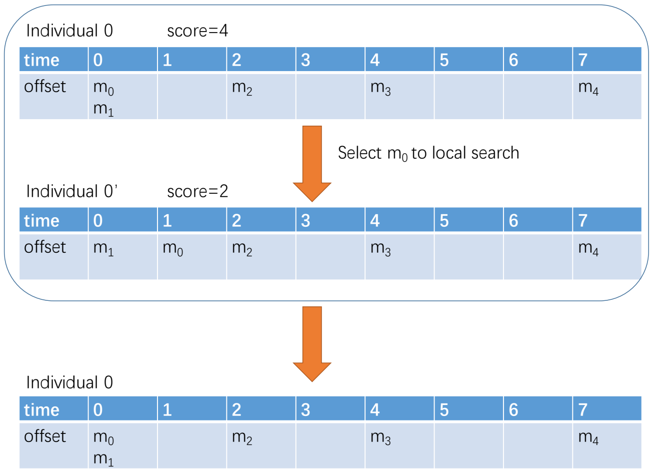


Fig. 7. Local search fig

reallocated in its possible offset interval, to minimum the score of the individual.

Fig shows an example of local search for individual 1 in Fig. According to the $C(\mathcal{M})$ of individual 0 in TABLE III, we random select m_0 as the element. Since the possible offset of m_0 is 0 and 1 shown in TABLE I, we locate the m_0 on gene 0 and gene 1 respectively. After marking two of individual with different offset of m_0, we select the individual which score is 2 as the individual 0 because the score of individual 0' is minimum among the individual which just reallocate the offset of m_0.

In our implement, we naively traverse all the possible offset. However for improving efficiency, many other algorithm in local search can be applied, i.e. tabu search, simulated annealing. And different local search strategy may increase the performance of the memetic algorithm.

VII. Evaluation

A. Experiments Configuration

The network architecture we used is 2D mesh network which scale is 3×3, 5×5 and 7×7, with 9, 25 and 49 switching nodes respectively. the program of our algorithm is implement in JAVA and is running on a Windows machine with 4GHz CPU and 12GB memory. We compare the memetic algorithm with general genetic algorithm without local search.

We generate the set communication with different scale which from 5 to 50. Each communication owns its period and delay which is random generated. The size of population is 100. Therefore there is 100 initial individual. And the maximum iterate times is 100. The memetic algorithm will finish if there is a feasible scheduling, otherwise the algorithm will execute until the iterate 100 times. The case which executes exceeds an hour will be regarded as overtime.

B. Failed Communication Percentage

To compare the performance of the genetic algorithm and memetic algorithm, we define the failed communication as the minimum number of messages, which block the scheduling of other messages. in other word, if the set of failed communication is removed from the given communication set on the TTNoC, the remaining messages will be scheduled.

Fig shows the percentage of failed messages in the set of communication. For each type of case. As can be seen, though the general algorithm spends quite a few time, the rate of failed scheduled messages is much more than the memetic algorithm, especially when the scale of communication set is relatively large. i.e. for the case of 50 messages in 3×3 TTNoC, the failed communication percentage is 17.3% for genetic algorithm while 8% for memetic algorithm. Therefore, the local search in the memetic algorithm is the key step in our algorithm and significantly improve the effect of genetic algorithm.

And it is clear that the failed communication rate decreases when the scale of TTNoC architecture is larger. The reason of this result is that the link resource increases with the growth of architecture and decrease the possibility of contention among the communication on the TTNoC. It also presents that the failed communication rate increases when the number of communication increasing. It is because that the limited link resources have to transmit more messages when there is more communication on the TTNoC.

C. Feasible Case

We consider the number of feasible case in each experiment type. For each test case, we define the case is successful if all the communication are scheduled in feasible location without link contention, otherwise we consider the case is infeasible case. A case is overtime is the case which synthesis time over 90 minutes. Fig presents the result of number of feasible cases

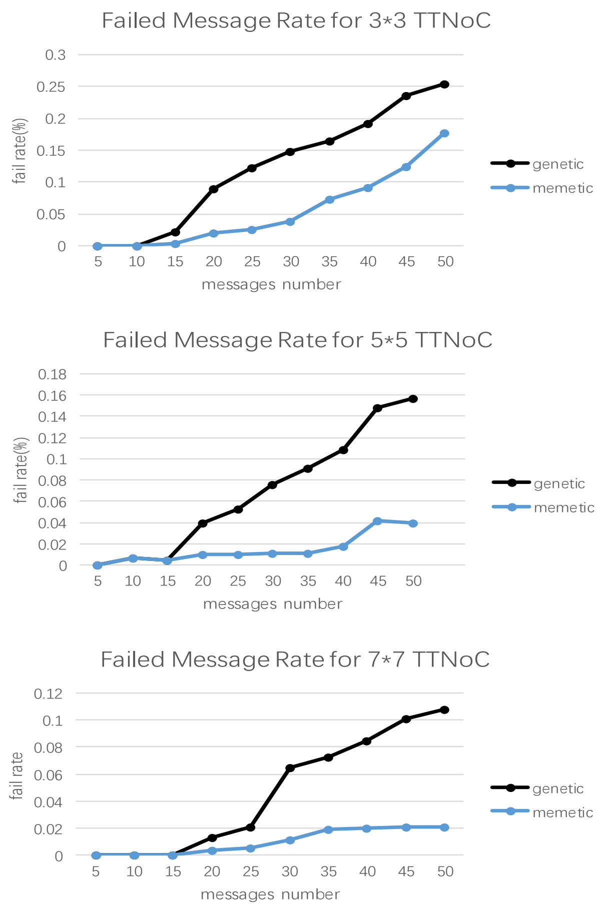


Fig. 8. fail rate

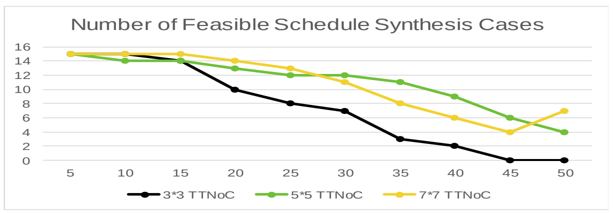


Fig. 9. feasible case

after schedule synthesis. For simple test cases with less than 15 communication, almost cases is feasible. But for complexed test cases more than 30 communication, it is hard for the algorithms to find a feasible scheduling.

VIII. Future Work

Synthesizing a feasible scheduling depends on the number of communication and the scale of TTNoC. It is hard for memetic algorithm to synthesize a feasible scheduling when the set of communication is large and/or the TTNoC architecture is small. However, except of two reason above, the mapping between a message and a communication node and routing strategy of communications also affect generating a feasible solution. Therefore the designer can manually change the map allocation or the routing strategy to the failed messages, and even design a larger network architecture if needed, and after these change, the feasible scheduling can be generated by the memetic algorithm iteratively.

Therefore the mapping between messages and nodes and routing strategy can be integrated to the TTNoC scheduling in our future work. And the different TTNoC architecture, i.e. torus, hypercube, and different local search strategy may be also considered in our experiment.

IX. CONCLUSION

This paper introduces a memetic algorithm to resolve the message scheduling problem on the TTNoC for the real-time communication. We test our memetic algorithm with different scales of TTNoC and various messages. The experiment results shows that our memetic algorithm is efficient to synthesize a scheduling with low fail communication percentage.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.