Parallel Applications Mapping onto Network on Chip Based on Heterogeneous MPSoCs Using Hybrid Algorithms

Dihia Belkacemi, Mehammed Daoui, Samia Bouzefrane, Youcef Bouchebaba

▶ **To cite this version:**

Dihia Belkacemi, Mehammed Daoui, Samia Bouzefrane, Youcef Bouchebaba. Parallel Applications Mapping onto Network on Chip Based on Heterogeneous MPSoCs Using Hybrid Algorithms. International Journal of Distributed Systems and Technologies, 2019, 10 (2), pp.37-63. 10.4018/IJDST.2019040103 . hal-02411294

HAL Id: hal-02411294
https://hal.science/hal-02411294

Submitted on 5 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parallel applications mapping onto network on chip based on heterogeneous MPSoCs using hybrid algorithms

Dihia Belkacemi
LaRI Lab, University of Tizi-Ouzou, Algeria.

Mehammed Daoui
LaRI Lab, University of Tizi-Ouzou, Algeria.

Samia Bouzefrane
CEDRIC Lab, CNAM, France.

Youcef Bouchebaba
ONERA-Toulouse, France.

ABSTRACT

Mapping parallel applications onto a Network on chip (NoC) that is based on heterogeneous MPSoCs is considered as an instance of an NP-hard and a multiobjective problem. Various multiobjective algorithms have been proposed in the litterature to handle this issue. Metaheuristics stand out as highly appropriate approaches to deal with this kind of problem. These metaheuristics are classified into two sets: population based metaheuristics and single solution based ones. To take advantage of the both sets, the trend is to use hybrid solutions that have shown to give better results. In this paper, we propose to hybridize these two metaheuristics' sets to find good Pareto mapping solutions to optimize the execution time and the energy consumption simultaneously. The undertaken experimental results have shown that the proposed hybrid algorithms give high quality non dominated mapping solutions in a reasonable runtime.

Keywords: Hybridation, Multi-objective optimization, Mapping, Network on chip (NoC), Heterogeneous Multiprocessor Systems-on-Chips (MPSoCs)

INTRODUCTION

Todays's applications that run on embedded systems such as multimedia (Sanchez et al., 2015), automative context (Stoychev et al. 2016; Rettkowski et al. 2015), signal processing (Meloni et al. 2015), and healthcare (Iranfar et al., 2018; El Mimouni & Karim 2014) have performance requirements which can no longer be satisfied by uniprocessor systems (Tafesse & Muthukumar, 2013). Heterogeneous Multi-processor Systems-on-Chips (MPSoCs) are widely accepted as a solution for future embedded systems (Leupers et al., 2017). With the increase of the number of processing elements in Systems-on-Chip (SoCs), it becomes very difficult to continue with the traditional non-scalable bus-based communication systems (Mukherjee & Chattopadhyay, 2017). The Network on chip (NoC) (Benini &Micheli, 2002) paradigm has been introduced as a promising interconnection solution to these systems. NoCs offer a

scalable and flexible-communication infrastructure with a highly supported modularity and a powerful performance (Hesham et al., 2016).

One critical issue of NoC based on heterogeneous MPSoCs is how to map an application on this platform, which is known to be NP-hard (Garey & Johnson, 1979). Furthermore, searching for an optimal mapping involves a frequent optimization of multiple objectives (functions) simultaneously which are often conflicting (e.g., overall completion time with energy consumption or load balancing with communication cost). So it is not possible to find a single mapping solution which optimizes all the objectives simultaneously. Instead, there exists a set of mapping solutions known as a Pareto optimal set. Various multiobjective algorithms have been proposed in the litterature to handle this issue. Metaheuristics stand out as a highly appropriate approaches to deal with this kind of problem in order to find good trade-off solutions (Talbi, 2009). These metaheuristics are broadly classified into two classes: Single solution based metaheuristics (S-metaheuristics) and Population based ones (P-metaheuristics). S-metaheuristics start from one solution's candidate and try to find better solutions in its neighborhood (local area search) so that this kind of metaheuristics focuses on exploitation. On the other hand, P-metaheuristics start with a set of solutions called population, instead of one solution. The power of the latter is its capability to explore a huge search space, hence allowing these metaheuristics to focus on exploration. Combining these two sets (i.e., hybridization) gives a well-balanced approach in both exploration and exploitation. Archived Multiobjective Simulated Annealing (AMOSA) (Bandypadhyay et al.,2008) is applied in this work to further improve the P-metaheuristics' Pareto front.
In this paper, we discuss our contribution followed by the related work. Then, we describe the multi-objective optimization principles and we formalize the problem to be solved. After that, we present the proposed hybrid algorithms and discuss the experimental results. Finally, we conclude the paper.

OUR CONTRIBUTION

Since the mapping problem is considered as the one of the most design challenges in NOC based heterogeneous MPSoC which affects greatly the final system performance, design of robust optimizers is required. Several optimization methods such as exact methods and metaheuristics have been proposed in the litterature to tackle this problem. Due to the todays' high systems complexities, the exact methods have quickly reached their limitations because of their unacceptable runtime. Metaheuristics stand out as highly appropriate approaches to tackle this problem. As stated above, these metaheuristics are classified into S-metaheuristics and P-metaheuristics. These latter especially Multiobjective Evolutionary Algorithms (MOEA) are widely used and have proved their success in solving this problem. Our solution is to further improve these MOEAs using local search based metaheuristic called AMOSA and gives new approaches which draw advantages from both metaheuristics' kind and consequently supply designers with a best trade-off mapping solutions.

RELATED WORK

Several multi-objective approaches have been proposed to solve the NoC mapping problem (Ascia et al., 2005; Zhou et al., 2006; Jena & Sharma, 2007; Tornero et al., 2009; Nedjah et al., 2011; He & Guo, 2013; Zhu et al., 2015; Chatterjee et al., 2016; Bruch et al., 2017). Ascia et al. (2005) present an approach for exploring the mapping design space while optimizing performance and power consumption by using SPEA2 (Ziztler et al., 2001) meta heuristic. Zhou et al. (2006) consider the mapping problem as a two conflicting objective optimization problem that attempts to minimize the average number of hops and achieves a thermal balance using NSGA (Srinivas & Deb, 1995) meta heuristic. Jena and Sharma (2007) propose an approach that tries to optimize energy consumption and bandwidth requirements by using NSGAII (Deb et al., 2002). To determine the Pareto-optimal configuration, a multi-objective genetic algorithm (MOGA) is used to optimize the average delay and the routing robustness by Tornero et al. (2009). Nedjah et al. (2011) propose solution which relies on multiobjective evolutionary algorithms

NSGAII (Deb et al., 2002) and MicroGA (Coello & Pulido, 2001) to minimize the hardware area, the execution time and the total power consumption. An ant colony optimization approach (ACO) is used by He and Guo (2013) to optimize the communication power consumption and the delay. Another mapping approach is proposed by Zhu et al. (2015) where a trade-off between temperature and latency is made. A constructive heuristic based method is proposed by Chatterjee et al. (2016) to solve the mapping problem where both network communication cost and system reliability are optimized. Bruch et al., (2017) use NSGAII (Deb et al., 2002) meta-heuristic to find trade-off solutions that improve the deadline compliance, reduce both the number of virtual channels used in the routers and the static energy consumed by the network. Some of the aforementioned works (He &Guo,2013;Zhu et al., 2015; Chatterjee et al., 2016) aggregate several objectives in a unified cost function. The drawback of these approaches comes from its difficulty to adjust the weights. Some other cited works (Zhou et al.,2006; Jena & Sharma, 2007; Nedjah et al., 2011) do not take into account the dynamic effect on network on chip (NoC) such as contention, and assume that the NoC is contention-free when they explore the mapping space. Most of these works use metaheuristics approaches, especially population based ones (P-metaheuristics) to find the approximation of Pareto optimal set (For instance: SPEA2 by Ascia et al. (2005) , MOGA byTornero et al. (2009), NSGAII by Bruch et al. (2017), etc.). From this, one can understand that metaheuristics (P-metaheuristics) are appropriate approaches used to find a set of optimal or sub optimal mapping solutions. Combining these meta-heuristics' set with other search techniques, referred to as hybrid (or memetic) metaheuristics, prove their effectiveness and achieve better performance compared to non hybrid ones (Blum et al.,2011). Several hybrid approaches are proposed in the context of the NoC and MPSoCs mapping (Wu et al., 2012; Wang et al., 2016;Yan et al., 2017; Guo et al., 2018).Wu et al. (2012) propose a new mapping algorithm called GA-MMAS based on Genetic Algorithm GA and MAX-MIN Ant System Algorithm (MMAS) to optimize energy consumption and latency for NoC. An adaptive memetic algorithm (AMA) to solve the mapping problem is proposed byWang et al. (2016), which combines an adaptive GA and an effective local search algorithm. Another hybrid approach is presented byYan et al. (2017), a multi objective hybrid algorithm (MOHA), which integrates a Pareto local search into NSGAII (Deb et al., 2002) metaheuristic is proposed. Guo et al. (2018) propose a novel IP-core mapping algorithm called CGSA (Cataclysm Genetic based Simulated Annealing). Their proposed algorithm integrates genetic with an improved simulated annealing algorithm assorted with cataclysm strategies. Some of the aforementioned hybrid approaches(Wang et al.,2016; Guo et al., 2018) consider only one cost function such as communication cost (Wang et al.,2016) and reliability (Guo et al., 2018). Some other works use a unified cost function(Wu et al.,2012), or target homogeneous MPSoCs (Yan et al.,2017).In this work, the multi-objective hybrid approach which merges P metaheuristics and S-metaheuristicsis proposed to solve our mapping problem and unlike other works which use analytical models to provide a fast evaluation of a given mapping (Wu et al.,2012;Wang et al.,2016), in this work, a simulation model is used to evaluate the two objectives we consider (execution time and energy consumption) where the dynamic effect of the network on chip is considered.

MULTI-OBJECTIVE OPTIMIZATION

As the name suggests, the multi-objective optimization problem involves a simultaneous optimization of multiple objectives that are often conflicting. When solving such problems, it is not possible to find a single solution which optimizes all the objectives simultaneously. Instead, there exists a set of trade-off optimal solutions known as Pareto-optimal set solutions. More formally, a multi-objective optimization problem can be formulated in the following as in given by Zhou et al. (2011):

$$"minimize" \; F(x) = [f_1(x), \dots f_m(x)]^T \mid x \in \Omega \tag{1}$$

where Ω is the decision space and $x \in \Omega$ is the decision vector. $F(x)$ consists of m objective functions $f_i: \Omega \to R, i = 1, 2, \dots, m$, where R^m is the objective space. The objectives in formula (1) are often conicting. The Pareto optimality is defined as follows.

Definition 1.

A vector $u = (u_1, \ldots, u_m)^T$ is said to dominate another vector $v = (v_1, \ldots, v_m)^T$, denoted as $u \prec v$, if $\forall\, i \in \{1, \ldots, m\}, u_i \leq v_i$ and $u \neq v$.

Definition 2.

A feasible solution $x^* \in \Omega$ of problem (1) is called *a Pareto optimal solution*, if $\nexists y \in \Omega$ such that $F(y) \prec F(x^*)$. The set of all the Pareto optimal solutions is called the Pareto set (PS), denoted as: *The image of the PS in the objective space is called the Pareto front (PF).*

$$PF = \{F(x) |\, x \in PS\}$$

The mapping problem can be considered as an instance of the multi-objective optimization problem. In this work, a set of multi-objective hybrid optimization algorithms has been proposed to explore the mapping space and to find the set of Pareto optimal solutions.

MAPPING PROBLEM FORMALIZATION

The mapping problem can be formalized as the assignment of an application model onto a platform model in such a way that the metrics of interest (cost functions) will be optimized under a set of constraints. In this section, we introduce our mapping system models, including the application and the platform models. The cost functions and the constraints used for the evaluation of the mapping solutions are also presented.

1. **Application model**

The application model is represented by a task graph denoted by A(T, E), where T is a non-empty set of vertices (tasks) t_i and E is a non-empty set of edges e_i. Each task t_i is annotated with two vectors V_i and C_i which contain respectively the energy consumption and the execution time of task t_i on each type of processing element. Each edge e_i in E corresponds to a dependency relation between two tasks connected by e_i labeled with volume (e_i) representing the amount of data exchanged between these tasks.

2. **Architecture model**

From the architecture point of view, the high level platform model contains a set of heterogeneous processing elements (PEs). Each PE is connected to a router and has the following characteristics: Id, type, frequency and location. Routers are interconnected with each other through bidirectional links which are assumed to be homogeneous (same data rate D) and transfer data through a NoC topology in the form of packets (set of flits). Figure 1 depicts the NoC's topology and the router-based architecture that we consider in our work. Each router has five direction links: North (N), South (S), East (E), West (W) and Local (L). The local port is connected to the processing element (PE) and the other ports are connected to neighboring routers. Input and output ports are assumed to have each a fixed-size buffer.

Figure 1: (a) Example of Heterogeneous MPSoC with four types of processors interconnected using 2D Torus NoC Topology. (b) Router's architecture

In order to communicate between PEs through the NoC, a method of routing is required. An XY routing algorithm is assumed as a deterministic routing algorithm which first routes packets on X direction to the

correct column and then on the Y direction toward the destination. A wormhole switching techniqueis also assumed in our work due to the small buffering space available in NoC's routers. The Round Robin (RR) technique is used as an arbitration mechanism to solve contentions, and the credit-based flow control technique is used to check the availability of router's buffers, hence avoiding data overflows.

3. Cost functions

We consider, in this work, two cost functions: the execution time (overall completion time) as well as the energy consumption described using simulation models.

a) Simulation model

To compute the both fitness functions (the overall completion time and the energy consumption), a discrete event based simulation model has been developed. The advantages of this model over analytical ones is its capability to analyze and represent the NoC system behavior in sufficient details. For example, the user can specify switching techniques, routing algorithms, flow control and arbitration policies, etc. In addition, our model allows capturing the waiting time due to the dynamic effects of NoC based MPSoCs systems like contentions. As in all discrete event based simulation models, we have maintained an event list to store the system events in a chronological order. Each event occurs at a particular instant of time. In our work, the following events have been considered:

i. *Event1*. Execute_Task (t_i,P_j) : The simulation starts by executing the ready tasks of each processor.
ii. *Event2.*Generate_Packets: Once the task ends its execution, the generation of packets occurs at the network interface (NI). Each packet contains a header, a payload and tail flits.
iii. *Event3*.Transfer Flits (ProcessorToRouter): Packets' flits will be sent flit by flit from the processor to the router.
iv. *Event4*.Apply Routing:As soon as a header flit arrives at the router's input buffer, the next hop is calculated according to the assumed routing protocol.
v. *Event5*.Apply Arbitration:According to the header flits, if several packets request the same output port, Round Robin arbitration policy is applied to select one of them.
vi. *Event6*.Traverse Router:The selected packet sends its flits through the router if there is enough space in its output buffer.
vii. *Event7*.TraverseLink:By applying the credit-based flow control technique, flits are transmitted between two neighboring routers through the link if the credit value of a source router's output buffer is higher than zero.
viii. *Event8*.Transfer Flits (RouterToPocessor): This event occurs if the final destination corresponds to the router's local port.

The output of this simulation model returns the two considered metrics measures (i.e. the overall completion time and the energy consumption) for a given mapping configuration from a single simulation run. In the following, we describe how these two metrics are computed using this simulation model.

b) Overall completion time using simulation model

To explain how the overall completion time of a given application is computed using the simulation model, let us take an example of a given mapping configuration (see Figure 2(c)).The simulation starts by executing the *first event* (Execute_Task (T_1,P_5)) as given by algorithm 1.

Figure 2: Example illustrating overall completion time computation using a simulation model

```
Algorithm1: Execute_Task (T1,P5)

for each Rlist(P5)do
T1=Rlist(P5).peek()
if(P5.state()= free)then
Schedule(Execute_Task (T1,P5), current_time)
ST(T1,P5) = current_time
FT(T1,P5) = ST(T1,P5)+ C15
Schedule(Generate_Packets, FT(T1,P5))
Rlist(P5).remove(T1)
else
delta = FT(Tk,P5)- current_time
Schedule(Execute_Task (T1,P5), current_time+delta)
endif
endfor
```

where $ST(T_1,P_5)$ and $FT(T_1,P_5)$ are respectively the Start Time and Finish Time of task T_1on a processor P_5, $FT(T_k,P_5)$ is the finish time of the last task running on the same processor P_5 where T_1 is mapped, C_{15} is the execution time of the task T_1 on a processor P_5, $R_{list}(P_5)$ is a list of ready tasks assigned to a processor P_5 and *delta* is the waiting time required to release the processor P5 so the task T1 can start its execution on it.

Once the task T_1 ends its execution on P5, its communication can be started after a packetization phase (*event 2*). Flits are sent from the processor P5 where $T_1$is mapped to the local port of router R5 attached to it, T_{P5R5} is the flit traversal time between the processor P5 to the router R5. As soon as the header flit arrives to R5' input buffer, a next destination is searched according to the control techniques of this router including the routing algorithm, arbitration and control policies. Let T_{R5} is the R5 router traversal time which is computed as the sum of routing (T_{ro}), arbitration (T_{arb}) and flow control (T_{ctr}) times. It is worth mentioning that the waiting time is added if concurrent communications (contentions) occur. After applying the routing algorithm *(XY in our case study),* flits traverse the link l_{R5R6} connecting the two routers R5 and R6 (*event 7*). T_{R5R6} is the link traversal time between R5 and R6. As the R6' local port is not attached to the final destination (*i.e., the processor P9*), flits traverse the router R6 (T_{R6}) towards next hop R9 through the link $l_{R6R9.}$ By invoking control techniques again at router R9 (i.e. *event 4, event 5 and event 6*), event 8 occurs and $T_{R9P9}$is the flit traversal time between router R9 and processorP9. According to this example and since our simulation is flit-based level, the communication time of a given flit $T_{flit(P5P9)}$ from P5 to P9 where T1 and T2 are mapped respectively can be computed as following:

$$T_{flit(\mathrm{P5P9})} = T_{P5R5} + T_{R5} + T_{R5R6} + T_{R6} + T_{R6R9} + T_{R9} + T_{R9P9} \quad (2)$$

In general, the simulation model pulls off and executes event by event until all the application tasks have been finished (i.e.the event list becomes empty). Hence, the overall completion time of an application consists of the time elapsed between the execution of the first event and the execution of the last one.

c) Energy consumption

As the overall completion time cost function, the energy consumption of NoC based on a heterogeneous MPSoC's system is estimated using the simulation model. This model computes the overall energy consumption of the system as the sum of energy's system components at each clock cycle including processing (i.e. processors' energy) and communication energies (i.e. NoC components' energy). The advantage of using this model comes from its capability to take into account the additional energy consumed by routers' buffer in presence of contentions.

4. Application and architecture constraints

In this paper, we consider the following constraints.

a) Task's assignment

Each task is assigned to exactly one processor, i.e:

$$\sum_{j=0}^{P-1} x_{ij} = 1, \forall i \in [0; NBT - 1] \quad (3)$$

where *P* is the number of processors (PEs) and *NBT* is the number of tasks. x_{ij} is a decision variable defined as follows: x_{ij}=1 if Ti is assigned to processor Pj, x_{ij}=0 otherwise.

b) Pre-assignment

In some cases such as hardware constraints (like dedicated accelerators), we can predefine a tasks assignment on specific processors for better performance purposes.

THE MULTI-OBJECTIVE MODEL

As mentioned above, the multiobjective model considered in this work consists in minimizing both the overall completion time and the energy consumption"$minimize$" $F(x) = [f_1(overallcompletiontime), f_2(energyconsumption)]$ under the two considered constraints. The two cost functions considered in this work are in Pareto. Hence, reliable trade-off mapping solutions which simultaneously optimize these two contradictory metrics are required. In this paper, we propose a hybrid multi-objective approach which combines the power of two metaheuristics' sets to solve this mapping problem.

HYBRID ALGORITHMS PROPOSAL

In this section, new hybrid metaheuristics are proposed for mapping applications onto NoC based on heterogeneous MPSoCs. Our hybridization approach is divided into two phases as shown in Figure 3. In the first phase, a set of well-known MultiObjective Evolutionary Algorithms (MOEAs) including NSGAII (Deb et al., 2002), SPEA2 (Ziztler et al., 2001), PESA2 (Corne et al., 2001), FastPGA (Eskandari et al., 2007) and IBEA (Zitzler Zitzler & Künzli, 2004) have been applied to get promising mapping solutions. In the second phase, the obtained solutions from the first phase and randomly generated solutions are used to initialize AMOSA's archive to further improve the P-metaheuristics' Pareto front and to ensure diversity by using random generated solutions. First, we describe how these

MOEAs metaheuristics are adapted to solve the mapping problem; and then how AMOSA is used to refine Pareto mapping solutions.

Figure 3: Hybridization approach

1. Solving the mapping problem using P-metaheuristics algorithms

Population based metaheuristics called P-metaheuristics have been proved as an ideal candidates to solve Multi-Objective Problems (MOPs). These metaheuristics work with multiple solutions called population rather than a single solution. Due to the population-based property, they can find several solutions of the Pareto optimal set in a single run, which lead them to be major and efficient approches recognized in solving multiobjective optimization. The most representative set of this kind of algorithms are Multiobjective Evolutionary Algorithms (MOEAs). In this section, we give an overview and describe briefly the five well-known MOEAs selected to be used in our proposed hybrid algorithms:

a) NSGA II: Nondominated Sorting Genetic Algortithm II is the most popular MOEA proposed by Deb et al. (2002), it uses in its selection operator: Pareto nondominated sorting and crowding distance.The Pareto nondominated sorting consists in dividing individuals into ranked non dominated fronts as follows: from a given solutions'set S, we have to find a set NS1 which contains the non dominated solutions according to non dominance relation applied to S.All the solutions that belong to the NS1's set are assigned to rank 1. This NS1 set is removed from S, and the same process is repeated on S-NS1, the next non dominated solutions' set NS2 is found from S-NS1. All the solutions belonging to the NS2's set are assigned to rank 2. This process goes on until S becomes empty. Solutions from a given rank are ranked again according to a crowding distance which is used to estimate the spread of solutions. To find a good Pareto front in terms of convergence and diversity, NSGAII selects the solutions with a lower rank, and if the solutions have the same rank, the crowding distance is used to select more diverse solutions.
b) SPEA2: Strength Pareto Evolutionary Algorithm 2 has been widely used in MOPs, initially proposed by Zitler et al. (2001).This algorithm uses both population and an archive in its operation.The archive is initially empty and size limited. It is used to save non dominated solutions found during the search, and if the number of non dominated solutions exceeds the archive size's limit, a truncation method is applied to preserve the boundary solutions. A fine grained fitness assignment is also used in this algorithm, which incorporates both the concept of Pareto dominance and density information.
c) PESA2: Pareto Envelope-based evolutionary algorithm 2 is proposed by Corne et al. (2001). This algorithm divides the search space in hyper boxes and instead of selecting individuals according to their fitness, one hyper box is selected and a random solution from the selected hyper box is kept. So this kind of selection is called region based selection.
d) FastPGA: Fast Pareto Genetic Algorithm is proposed by Eskandari et al. (2007). It classifies the parents and offspring solutions into two sets: the first one contains the non dominated solutions and the second one contains dominated ones. The fitness value of the solutions from the first set is calculated using the crowding principle presented in NSGAII algorithm (Deb et al., 2002). For the other solutions, the fitness-value procedure looks like the one used in SPEA2 (Ziztler et al., 2001). In addition, FastPGA uses a population regulation operator to dynamically regulate the size of the population.
e) IBEA: Indicator Based Evolutionary Algorithm is proposed by Zitzler and Künzli (2004). This algorithm uses an arbitrary quality indicator, such as Hypervolume in its selection phase.Unlike other algorithms cited above, IBEA algorithm does not require additional diversity-preservation mechanisms.

The five above cited algorithms are already implemented in jMetal framework (jMetal) which we have adapted to solve our mapping problem.For this purpose, we have defined the solution coding and adapted the different mutation and crossover operators, as described in the following.

i. Solution coding

For our mapping, we used an integer coding. Each mapping solution is represented by a tuple $(x_1, x_2, \ldots, x_n)$, where x_i gives the PE on which the task t_i has been mapped. The value of each variable x_i is chosen from the set Pt_i which is the list of all permissible processors for the task t_i. Figure 4 gives a chromosome example where 5 tasks are mapped on a given heterogeneous platform.

Figure 4: chromosome example

Let us assume that $Pt_0=\{1, 3, 8\}$ is the permissible processors of the task t_0. For example the value of the task t_0 is 3 which is chosen from its Pt_0.

ii. Mutation and crossover

After applying crossover and mutation operators, it is possible that some solutions (chromosomes) will be invalid due to the assignment of a given task to unfeasible processor. An example of invalid solution after applying a mutation operator (for instance a polynomial mutation) is depicted in Figure 5 (the task t_0 is assigned to the processor 22 which is not in its permissible set $Pt_0=\{1, 3, 5, 9, 11, 15\}$).

Figure 5: Example of invalid solution after applying Polynomial mutation operator

To tackle this problem, we first transform the initial coding to an intermediate one by using the indices of Pt_i elements. As shown in Figure 6, the parent's chromosome (15, 7, 9, 4, 15) will be transformed to (5, 1, 3, 0, 2). 5 is the index of the element 15, 1 is the index of the element 7, 3 is the index of the element 9, etc. The set Pt_i will be transformed to $P't_i$. The aim of this new representation is to minimize invalid solutions' numbers.

Figure 6:Intermediate coding

Unfortunately, we do not eliminate all of them. An example of invalid solution after applying this new representation is given in Figure 7.

Figure 7: Example of invalid solution with the new representation after applying Polynomial mutation.

Thus task t_0 is assigned to processor with index 7 which is not in the set of processors' indexes $P't_0=\{0, 1, 2, 3, 4, 5\}$. To correct these invalid solutions, we propose Algorithm 2.

Algorithm 2:Correct invalid solutions to valid ones.

```
for all invalid solutions do
for all tasks do
```

```
if (val(t_i) ∉P'(t_i)) then
generate a random valueinside P'(t_i)
replace val(t_i) by a generated random value.
endif
endfor
endfor
```

*Figure 8:*Correct invalid solution byapplying algorithm 2

The result's solution after applying Algorithm 2 is depicted in Figure 8. At the last step, we transform the obtained intermediate coding to the initial one (see Figure 9), the chromosome (1, 1, 3, 0, 2) is then transformed to (3, 7, 9, 4, 15). 3 is the element with index 1 in the set Pt_0, 7 is the element with index 1 in the set Pt_1, 9 is the element with index 3, etc.

*Figure 9:*Return to the initial coding

Although these P-metaheuristics give a good trade-off mapping solutions used by several authors in the litterature, these metaheuristics can be further improved by a local search method, we have selected AMOSA metaheuristics as an S-metaheuristic which use the P-metaheuristics' results as a good start points in its archive as well as random solutions' set to ensure more spread solutions.

2. Refinement of P-metaheuristics results using AMOSA algorithm

Archived Multiobjective Simulated Annealing proposed by Bandyopadhyay et al. it is a multiobjective version of the simulated annealing (SA) algorithm which is a well-known single solution based search algorithm. It incorporates the concept of archive to store the non-dominated solutions during search process as well as to determine its acceptance probabilities. We have used AMOSA in this study as local reseach to further refine the mapping solutions. In this section**,** before describing AMOSA' archive initialization which is the heart of this novel proposal, we first show how we adapt AMOSA to solve the mapping problem.

a) Solving mapping problem using AMOSA

To apply this algorithm to our mapping problem, we have to specify a solution representation and a corresponding neighbourhood move operator.

i. Solution representation

The potential solution (point) in AMOSA algorithm is like the chromosome representation shown in Figure 4. As AMOSA is a single solution based metaheuristic, the search process starts from an initial solution and tries to find a new one (new search area) by applying the neighbourhood move operator that is described below.

ii. Neighbourhood move operator

For AMOSA algorithm, the neighbourhood move operator is like the mutation operator which we have adapted above. So all the mutation operators like Flip Mutation, Polynomial Mutation, Uniform Mutation and Non Uniform Mutation can be specified as neighbourhood move operators in AMOSA algorithm.

b) Initialize AMOSA' archive

Our approach consists in injecting the Pareto mapping solutions provided by P-metaheuristics as well as random ones in AMOSA's archive in order to further improve solutions given by standard MOEAs and give new well balanced approaches in both exploration and exploitation. These new approaches are called according to the P-metaheuristic used preceded by H. For instance HNSGAII, use AMOSA algorithm to further improve the standard NSGAII (see Algorithm 3).

```
Algorithm 3: HNSGAII-main.

Step1: execute NSGAII
S: is a set which contains non-dominated solutions returned by NSGAII
R: is a set which contains non-dominated solutions generated randomly
A: is AMOSA' archive
HL: is the maximum size of the Archive on termination

Add all the solutions of S to the A
for each solution r∈R do
if (l≺ r) then|l∈A
        delete r
else
add r to A
endif
endfor
if (A.size()> HL) then
   Apply clustering

Step2: execute AMOSA
```

Experimental results

In this section, our proposed hybrid algorithms are evaluated using various mapping problem's instances (small, medium and large).These instances differ from each other regarding the task graph's and platform's size used as depicted in Table 1. The implementation of P-metaheuristics used in the proposed hybrid approach is the one of jMetal framework (jMetal) which we have adapted to our mapping problem. We have also extended this framework by adding new algorithms' implementations such as the S-metaheuristic algorithm (AMOSA) and the Multi-objective exact one called Multi-objective Branch and Bound (MBB). It is worth mentioning that the exact method is used to check the efficiency of the proposed hybrid multiobjective algorithms in solving small and medium mapping problem's instances. For our experiments, TGFF(TGFF) is used to generate a set of synthetic task graphs by varying the number of nodes (tasks). On the other hand, the architecture model (Platform) consists of k types of processors interconnected using 2D Torus NoC topology. Table 2 gives the NoC's parameters used in our experiments.

Table 1. Mapping problem's instances

Mapping problem	**Task graph**	**Platform**
Small	6	3x3 torus topology
Medium	10	4x4 torus topology
Large	100	8x8 torus topology

Table 2. NoC's parameters

NoC topology	2D Torus
Switching technique	Wormhole switching
Routingtechnique	XY routing algorithm
Arbitration technique	Round Robin (RR)
Flow control	Credit-based

Table 3. Algorithm's Parameterization

NSGAII/FastPGA	
Population Size	100
Max Iterations	10000
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.9
SPEA2/PESA2/IBEA	
Population Size	100
Archive Size	100
Max Iterations	10000
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.9
AMOSA	
Initial temperature (T_0)	800
Final temperature (T_1)	0.001
Cooling rate α	0.9
Max Iterations	100
HL	100
SL	110
Gamma	1.8

The proposed multiobjective hybrid algorithms are evaluated according to their quality of the Pareto solutions returned and their computational time (runtime). To measure the hybrid algorithms' quality solutions, two properties are usually required: convergence and uniform diversity. A number of quality indicators for measuring these two criteria are included in jMetal framework (jMetal). In this paper, we have considered the following evaluation metrics:

1. Inverted Generational Distance (IGD) (Nebro et al., 2006): this metric measures both convergence and diversity. It uses the optimal Pareto front and measures the distance of each of its elements and the computed approximation.
2. Epsilon (Durillo& Nebro, 2011): measures the smallest distance it would be necessary to translate every solution in a computed front for a problem. So it dominates the optimal Pareto front of this problem. This metric measures only convergence.

The smaller the IGD and Epsilon values are, the closer the approximation set is to the reference Pareto front (better convergence). A small value of IGD means also a good diversity of the obtained solutions.When IGD and Epsilon values are 0, it implies that all the generated solutions of a given algorithm are in the optimal Pareto set of the problem. These applied metrics require an optimal Pareto set to be computed. To this end, we have considered the Pareto front returned by MBB as an optimal Pareto set (reference Pareto front) for small and medium mapping problems. For the large mapping problem, the

optimal Pareto is obtained by collecting the results of several runs of the different algorithms. In all our experiments, we have performed 30 independent runs and the obtained tables (4, 5, 6, 7, 10 and 11) represent statistical informations of the quality indicator applied as well as algorithms' runtime. Table 3 gives algorithms'parameter settings used in the following experiments. Notice that these parameters have been chosen after a primary phase where sensitivity analysis is done to find the parameters which give good results for all instances in terms of both solutions' quality and runtime. All the calculations were performed on a PC Intel(R) Core(TM) i7 CPU, 2.7GHz, with 8 Go of RAM.
Tables (4,5, 6 and 7) show respectively a comparative study between the proposed hybrid algorithms called (HNSGAII, HSPEA2, HPESA2, HFastPGA, HIBEA) and the non hybrid ones (NSGAII, SPEA2, PESA2, FastPGA, IBEA) on small and medium mapping problem's instances. The fronts returned by the different algorithms are compared with the optimal Pareto set provided by the multiobjective branch and bound (MBB). We have limited the number of permissible processors for each task, so that the maximum search space's size of small and medium mapping problems considered in these experiments are respectively 21600 and 139968 instead of 9^6 and 16^{10}.
From Tables (4,5, 6 and 7), one can see that the proposed hybrid approaches present better mean and median values compared to non hybrid ones for both small and medium mapping instances. These enhancements are well viewed by boxplots given in Figures 10 and 11 (in terms of Epsilon). This shows that for the small and medium mapping problem instances, our proposed hybrid algorithms outperform the non hybrid ones according to the two performance metrics applied (IGD and Epsilon). It is important to note that except HIBEA metaheuristic which outperforms IBEA in all runs (HIBEA's boxplots are above IBEA' ones), hybrid algorithms do not outperform non hybrid ones in all runs. For instance, *Tables 8* and *9* compare first ten consecutive runs among 30 runs where hybrid and non hybrid algorithms have been compared according to the Epsilon quality indicator.As seen from *Tables 8* and *9*, the hybrid algorithms outperform non hybrid ones in almost all the runs and never deteriorate non hybrid algorithms' results. It can also be concluded that our proposed hybrid algorithms provide results which are the same or close to those returned by MBB in a very reasonable runtime. For example, more than 75% of HNSGAII's runs give the same Pareto front as MBB while NSGAII gives lower than 50% best runs (see Figure 9). This confirms the optimality of the proposed hybrid algorithms' results for the small and medium mappings problems. Consequently, we can trust our proposed algorithms' results for solving large mapping problem instances.

Table 4. Statistical indices. IGD. Hybrid vs non Hybrid algorithms (small mapping's instance)

Algorithms	IGD					Runtime
	Mean	Standard deviation	Median	min	max	
NSGAII	5.72e-04	5.8e-04	4.89e-04	0.00e+00	1.84e-03	4,759 s
HNSGAII	1.42e-04	3.6e-04	0.00e+00	0.00e+00	1.83e-03	10,683s
SPEA2	1.55e-03	9.8e-04	1.46e-03	7.91e-04	6.79e-03	5,132s
HSPEA2	1.83e-04	3.8e-04	0.00e+00	0.00e+00	1.56e-03	11,127s
PESA2	4.66e-03	2.4e-03	3.13e-03	1.21e-03	8.98e-03	4,140s
HPESA2	1.81e-03	2.0e-03	5.18e-04	0.00e+00	5.19e-03	10,171s
FastPGA	1.74e-03	1.0e-03	1.54e-03	5.72 e-04	6.79e-03	7,938s
HFastPGA	1.07e-03	1.6e-03	4.89e-04	0.00e+00	5.01e-03	13,933s
IBEA	1.15e-02	2.2e-03	1.15e-02	7.37 e-03	1.79e-02	0,169s
HIBEA	3.50e-03	2.3e-03	5.01e-03	0.0e+00	8.01e-03	6,093s
MBB	0.00e+00	0.0e+00	0.0e+00	0.0e+00	0.0e+00	43,663s

Table 5. Statistical indices. Epsilon. Hybrid vs non Hybrid algorithms (small mapping's instance)

Algorithms	Epsilon					Runtime
	Mean	Standard deviation	median	min	max	
NSGAII	3.53e+00	5.1e+00	1.00e+00	0.00 e+00	1.40e+01	4,759 s
HNSGAII	6.33e-01	2.5e+00	0.00 e+00	0.00 e+00	1.40e+01	10,683s
SPEA2	5.83e+00	8.6e-01	6.00 e+00	5.00 e+00	9.00e+00	5,132s
HSPEA2	9.00e-01	2.0e+00	0.00 e+00	0.00 e+00	8.00 e+00	11,127s
PESA2	2.56e+01	3.6e+01	1.40e+01	5.00 e+00	1.68e+02	4,140s
HPESA2	3.87e+00	5.1e+00	2.00e+00	0.00 e+00	2.10e+01	10,171s
FastPGA	6.90e+00	1.5e+00	6.00 e+00	5.00 e+00	9.00e+00	7,938s
HFastPGA	2.03e+00	2.0e+00	1.00 e+00	0.00 e+00	6.00 e+00	13,933s
IBEA	1.86e+02	6.2e+01	18.85e+01	9.1e+01	3,06e+02	0,169s
HIBEA	6.73e+00	1.5e+01	4.00e+00	0.00e+00	8.40e+01	6,093s
MBB	0.00e+00	0.0e+00	0.00 e+00	0.00e+00	0.00 e+00	43,663s

Table 6. Statistical indices. IGD. Hybrid vs non Hybrid algorithms (medium mapping's instance)

Algorithms	IGD					Runtime
	Mean	Standard deviation	median	min	max	
NSGAII	1.28e-03	4.6e-04	1.38e-03	0.0e+00	2.55e-03	0,584m
HNSGAII	3.65e-04	5.5e-04	0.00 e+00	0.00 e+00	1.38e-03	1,322m
SPEA2	1.98e-03	4.9e-04	1.83e-03	1.38e-03	3.21e-03	0,574m
HSPEA2	9.93e-04	5.6e-04	1.22e-03	0.00 e+00	1.64e-03	1,315m
PESA2	4.36e-03	3.8e-03	2.78e-03	8.96e-04	1.70e-02	0,559m
HPESA2	9.53e-04	6.5e-04	1.24e-03	0.00 e+00	1.87e-03	1,328m
FastPGA	2.65e-03	6.5e-04	2.49 e-03	1.58 e-03	4.26e-03	0,597m
HFastPGA	6.12e-04	6.3e-04	4.06e-03	0.00 e+00	1.56e-03	1,344m
IBEA	1.24e-02	3.4e-03	1.21e-02	6.82e-03	2.08e-02	0,018m
HIBEA	8.89e-04	7.5e-04	1.16e-03	0.00 e+00	2.96e-03	0,770m
MBB	0.00e+00	0.0e+00	0.00 e+00	0.00 e+00	0.00 e+00	27,065m

Table 7. Statistical indices. epsilon. Hybrid vs non Hybrid algorithms (medium mapping's instance)

Algorithms	Epsilon					**Runtime**
	Mean	Standard deviation	median	min	max	
NSGAII	1.13e+01	5.9e+00	1.05e+01	0.00 e+00	2.20e+01	0,584m
HNSGAII	3.00e+00	5.2e+00	0.00 e+00	0.00 e+00	1.90e+01	1,322m
SPEA2	2.09e+01	8.3e+00	2.35e+01	7.00 e+00	3.80e+01	0,574m
HSPEA2	6.47e+00	4.5e+00	7.0 e+00	0.00 e+00	1.90e+01	1,315m
PESA2	6.01e+01	8.1e+01	2.60e+01	1,00e+01	3.27 e+02	0,559m
HPESA2	7.07e+00	6.1e+00	7.00 e+00	0.00 e+00	2.20e+01	1,328m
FastPGA	2.63e+01	8.6e+00	2.60e+01	1.00e+01	3.80e+01	0,597m
HFastPGA	4.47e+00	4.9e+00	5.00 e+00	0.00 e+00	1.90e+01	1,344m
IBEA	2.03e+02	9.1e+01	2.03 e+02	6.30e+01	4.02 e+02	0,018m
HIBEA	6.67e+00	1.0e+01	7.00 e+00	0.00 e+00	5.50e+01	0,770m
MBB	0.00e+00	0.0e+00	0.00 e+00	0.00 e+00	0.00 e+00	27,065m

Table 8. Epsilon metric according to Table 5

Algorithms	Run 0	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9
NSGAII	1.0	0.0	0.0	1.0	14.0	0.0	0.0	0.0	1.0	0.0
HNSGAII	1.0	0.0	0.0	0.0	14.0	0.0	0.0	0.0	0.0	0.0
SPEA2	5.0	6.0	6.0	6.0	5.0	6.0	6.0	6.0	5.0	6.0
HSPEA2	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0
PESA2	8.0	21.0	14.0	14.0	14.0	14.0	14.0	168.0	19.0	14.0
HPESA2	3.0	1.0	4.0	1.0	6.0	4.0	0.0	4.0	0.0	1.0
FastPGA	8.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	5.0	9.0
HFastPGA	3.0	0.0	1.0	0.0	5.0	1.0	1.0	0.0	0.0	3.0
IBEA	257.0	211.0	251.0	122.0	154.0	125.0	109.0	182.0	240.0	168.0
HIBEA	1.0	4.0	3.0	3.0	5.0	4.0	4.0	4.0	4.0	4.0
MBB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 9. Epsilon metric according to Table 7

Algorithms	Run 0	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9
NSGAII	3.0	19.0	19.0	7.0	7.0	7.0	19.0	4.0	13.0	7.0
HNSGAII	3.0	19.0	7.0	0.0	0.0	0.0	0.0	0.0	7.0	0.0
SPEA2	21.0	19.0	11.0	24.0	24.0	24.0	20.0	10.0	7.0	11.0
HSPEA2	7.0	7.0	0.0	0.0	11.0	19.0	4.0	7.0	7.0	11.0
PESA2	38.0	24.0	24.0	24.0	26.0	10.0	16.0	327.0	26.0	202.0
HPESA2	10.0	0.0	14.0	13.0	19.0	0.0	0.0	7.0	13.0	7.0
FastPGA	38.0	38.0	20.0	24.0	10.0	38.0	38.0	26.0	24.0	38.0
HFastPGA	0.0	7.0	7.0	0.0	7.0	3.0	0.0	7.0	0.0	0.0
IBEA	180.0	63.0	220.0	361.0	83.0	246.0	132.0	148.0	148.0	248.0
HIBEA	0.0	55.0	0.0	0.0	7.0	3.0	7.0	0.0	0.0	0.0
MBB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Figure 10: Epsilon. Hybrid vs non hybrid algorithms (small mapping's instance)

Figure 11: Epsilon. Hybrid vs non hybrid algorithms (medium mapping's instance)

In Figure 12, a comparative study has been carried out between the two previous algorithms' sets (hybrid and non hybrid algorithms) for large mapping's instances (see *Table 1*). In this experiment, we have mapped a task graph with 100 tasks on a heterogeneous MPSoC platform with six types of processors interconnected using 8x8 Torus topology.

Figure 12: Mapping of randomly generated graph with 100 tasks on a heterogeneous MPSoC platform with six types of processors interconnected using 8x8 Torus topology. Two cost functions are optimized (execution time and energy consumption)

The plotted graphs, as depicted in Figure 12, represent the non dominated mapping's solutions of each algorithm after 30 runs. As seen from *Tables* (*10* and *11*) *and Figures* (*12* and *13*), hybrid algorithms yield the most promising results (give better Pareto front) compared with non hybrid ones in both IGD and Epsilon metrics at the expense of time. This additional time of hybrid algorithms compared with non hybrid ones is due to the time needed by Archived Multi-Objective Simulated Annealing (AMOSA) to improve the P-metaheuristics' Pareto front.

Figure 13: Hybrid vs non hybrid algorithms (large mapping's instance)

Table 10. Statistical indices.IGD.Hybrid vs non Hybrid algorithms (large mapping's instance)

Algorithms	IGD					Runtime
	Mean	Standard deviation	median	min	max	
NSGAII	7.97e-01	4.2e-02	7.87e-01	7.21e-01	8.75e-01	3,12m
HNSGAII	1.80e-01	5.0e-02	1.70e-01	9.10e-02	2.82e-01	6,354m
SPEA2	5.28e-01	6.3e-02	5.43e-01	3.99e-01	6.30e-01	2,933m
HSPEA2	1.46e-01	3.2e-02	1.45ee-01	8.95e-02	2.08e-02	6,121m
PESA2	3.61e-01	9.5e-02	3.49 e-01	1.98 e-01	5.81e-01	2,639m
HPESA2	1.57e-01	5.1e-02	1.48 e-01	3.57e-02	2.99 e-01	5,83m
FastPGA	4.76e-01	9.6e-02	4.67e-01	2.74 e-01	6.84 e-01	2,87m
HFastPGA	1.63e-01	5.6e-02	1.59 e-01	4.35e-02	3.21e-01	6,049m
IBEA	1.14e+00	5.8e-02	1.15e+00	9.04 e-01	1.22e+00	0,1m
HIBEA	1.77e-01	5.4e-02	1.66e-01	9.25 e-02	3.21e-01	3,339m

Table 11. Statistical indices.epsilon.Hybrid vs non Hybrid algorithms (large mapping's instance)

Algorithms	EPSILON					Runtime
	Mean	Standard deviation	median	min	max	
NSGAII	8.62e+03	5.5 e+02	8.72e+03	7418 e+03	9.59e+03	3,12m
HNSGAII	2.79e+03	7.0 e+02	2.78e+03	1.36 e+03	4.14e+03	6,354m
SPEA2	6.07e+03	7.6 e+02	6.11 e+03	4.70 e+03	7.76e+03	2,933m
HSPEA2	2.22 e+03	7.4e+02	2.32 e+03	1.44 e+02	3.30 e+03	6,121m
PESA2	4.67 e +03	9.8 e+02	4.57 e+03	3.14 e+03	6.96 e+03	2,639m
HPESA2	2.48e +03	7.6e+02	2.50 e+03	9.82 e+02	4.31e+03	5,83m
FastPGA	5.52 e+03	9.4e+02	5.32e+03	3.70 e+03	7.57e+03	2,87m
HFastPGA	2.61e +03	7.3e+02	2.65e+03	1.08e+03	4.66 e+03	6,049m
IBEA	1.18 e +04	4.9 e+02	1.190e+03	1.03e+04	1.26e+04	0,1m
HIBEA	2.65e +03	7.9 e+02	2.52e+03	1.32 e+03	4.73e+03	3,339m

Lastly, we provide experiments that show how our proposed hybrid algorithms are sensitive to their parameters. Figures 14 and 15 present two experiments on the effect of some parameters like AMOSA's

Max iterations (Figure 14) and AMOSA's Cooling rate (Figure 15) on the performance of an example of the proposed hybrid algorithms HNSGAII. Note that we did not show all the proposed hybrid algorithms' Pareto fronts in order to have clear figures. In these experiments, we have varied one parameter at one time for HNSGAII algorithm. In the first experiment (Figure14), the AMOSA's Max iterations values setting are as follows: 50, 100 and 500 and in the second one (Figure15), cooling rate α values are the following: 0.5, 0.7 and 0.9 and other parameters have been fixed (see Table 12). A large mapping instance has been used in this experiment.

Table 12. HNSGAII's Parameterization

HNSGAII	
Population Size	100
Max Iterations (NSGAII)	10000
Mutation Probability	1.0/L (L : individual length)
Crossover Probability	0.9
Initial temperature (T_0)	800
Final temperature (T_1)	0.001
Cooling rate α	0.9 (0.5, 0.7)
Max Iterations (AMOSA)	100 (50, 500)
HL	100
SL	110
Gamma	1.8

Figure 14: AMOSA's Max iterations' effect

Figure 15: AMOSA's Cooling rate' effect

From Figures 14 and 15, it is clear that HNSGAII is very sensitive to its input parameters which strongly determine the mapping solutions' quality. For instance, in Figure 14, we observe that the higher AMOSA's max iterations value gives better mapping results at the expense of time. So compromise between both the solution quality and the runtime required must be taken into account when exploring a given mapping problem.

Table 13. Notation list

Term	Description
T_i	Task i
P_j	Processor j
$ST(T_i,P_j)$	Start Time of task i on processor j
$FT(T_i,P_j)$	Finish Time of task i on processor j
$R_{list}(P_j)$	List of ready tasks assigned to a given processor Pj
E_{ij}	The energy needed to execute the task t_i on a processor P_j
C_{ij}	The execution time of the task t_i on a processor P_j
T_{ro}	The time required for routing
T_{arb}	The time required for arbitration
T_{ctr}	The time required for flow control

T_{Ri}	Router's execution time
$T_{flit\ (Ps,Pd)}$	Time required to transfer a given flit from source processor towards destination one.
P	The number of processors (PEs)
NBT	The number of tasks
x_{ij}	T_i is assigned to processor PE_j (Binary variable)

DISCUSSION

Through experimental results conducted on several mapping instances (small, medium and large), the performance of our proposal is proved. So, selecting these new approaches in the mapping phase of a given system design flow may decrease its implementation cost and has a good impact on its final behavior since these approaches give a high quality trade-off mapping solutions compared with standard ones. However, the proposed hybrid algorithms are very sensitive to their parameters since they combine two metaheuristics' sets which are both sensitive to their parameters. It is the reason why an initial experimental phase where a sensitivity analysis for each parameter of each hybrid algorithm must be carried out in order to determine the most appropriate settings to tackle a given problem type.

CONCLUSION AND FUTURE WORK

In this paper, a set of hybrid algorithms which combine P-metaheuristics and S-metaheuristics are proposed to solve the mapping problem in a NoC architecture. To prove the optimality of the proposed algorithms, we first compared them to the exact method (MBB) for small and medium mapping instances. From our experimental results, our proposed hybrid algorithms provide the same results or close to those given by the exact method (MBB) in low runtime while exceeding the non hybrid ones in terms of results' quality.This confirms that the S-metaheuristics (local search) can effectively improve the P-metaheuristics. As a future work, we plan to explore a couple of directions, including, exploring the proposed algorithms by considering other design system's metrics like communication cost, load balancing, etc.; further improvement of the P-metaheuristics by injecting a local search in their initialization phase or their operators like mutation, crossover, etc.; and by considering other NoC's characteristics (other topologies, router's architecture, etc.).

REFERENCES

Leupers, R., Aguilar, M.A., Eusse, J.F., Castrillon, J., &Sheng, W. (2017). MAPS: ASoftware Development Environment for Embedded Multicore Applications. *Springer Science+Business Media Dordrecht*, 917-949.

Mukherjee, P., & Chattopadhyay, S. (2017). Low Power Low Latency Floorplan-aware Path Synthesis in Application-Specific Network-on-Chip Design.*Integration, the VLSI Journal*, 58, 167-188.

Benini, L., Micheli, G. (2002). Networks on chip: A new SoC paradigm. *IEEE Computer*, 35 (1), 70-78.

Hesham, S., Rettkowski, J., Goehringer, D., & Abd El Ghany, M.A. (2016). Survey on Real-Time Networks-on-Chip.*IEEE Transactions on Parallel and Distributed Systems*.

Garey, M.R., Johnson, D.S. (1979). Computers and Intractability, A Guide to the Theory of NP-Completeness.

Talbi, EG. (2009). *Metaheuristics: From Design to Implementation*. United States, John Wiley and Sons Ltd.

Bandypadhyay, S., Saha, S., Maulik, U., &Deb, K.(2008). A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA.*IEEE Transactions onevolutionary Computation,* 12 (3).

Ascia, G., Catania, V.,& Palesi, M. (2005). Mapping Cores on Network-on- Chip.*International Journal of Computational Intelligence Research (IJCIR)*, 109-126.

Zhou, W., Zhang, Y & Mao, Z. (2006). Pareto based Multi-objective Mapping IP Cores onto NoC Architectures.*Circuits and Systems,APCCAS*, 331-334.

Jena, R.K., &Sharma, G.K. (2007). A multi-objective evolutionary algorithm-based optimisation model for network on chip synthesis. *International Journal of Innovative Computing and Applications*, 977-982.

Tornero, R., Sterrantino, V., Palesi, M,& Orduna, J.M. (2009). A multi-objective strategy for concurrent mapping and routing in networks on chip. *International Symposium on Parallel and Distributed Processing*, 1-8.

Nedjah, N., da Sliva , M.V.C, &Mourelle, L.de.M. (2011). Customized computer-aided application mapping on NoC infrastructure using multi-objective optimization. *Journal of System Architecture*, 57(1), 79-94.

He, T., Guo, Y. (2013). Power consumption optimization and delay based on ant colony algorithm in network-on-chip.*Engineering Review*, 33(3),219-225.

Zhu, D., Chen, L., Pinkston, T.M., & Pedram, M. (2015). TAPP: Temperature-Aware Application Mapping for NoC-Based Many-Core processors, *Design Automation and Test in Europe Conference and exhibition*, 1241-1244.

Chatterjee, N., Reddy, She., Reddy, Shi., &Chattopadhyay.S. (2016). A reliability awareapplication mapping onto mesh based Network-on-Chip. *International Conference on Recent Advances in Information Technology(RAIT)*, 537- 542.

Bruch, J.V., Sliva, E.A.da., Zeferino, C.A., &Indrusiak, L.S. (2017). Deadline, Energy and Buffer-Aware Task Mapping Optimization in NoC-Based SoCs Using Genetic Algorithms.*Symposium on Computing Systems Engineering (SBESC)*, 86-93.

Blum, Ch., Puchinger, J., Raidl, G.R., & Roli, A. (2011).Hybrid Metaheuristics in Combinatorial Optimization: A Survey.*Applied Soft Computing*, 11(6), 4135- 4151.

Wu, N., Mu, Y., & Ge, F. (2012). GA-MMAS: an Energy- and Latency-aware Mapping Algorithm for 2D Network-on-Chip. *IAENG International Journal of Computer Science*.

Wang, X., Liu, H., & Yu, Z. (2016). A novel heuristic algorithm for IP block mapping onto mesh-based networks-on-chip. The Journal of Supercomputing, 72(5), 2035-2058.

Yan, R., Zhou, Y., Yan, Y., Yin, M., Yu, M., Ma, F., & Huang, K. (2017). A Hybrid Multi-objective Evolutionary Algorithm for Energy-Aware Allocation and Scheduling Optimization of MPSoCs.*International Conference on Tools with Artificial Intelligence*, 701-708.

Guo, L., &Ge, Y., Hou, P., Cai, Q., & Wu, J. (2018). A Novel IP-Core Mapping Algorithm in Reliable 3D Optical Network-on-Chips. *Optical Switching and Networking*.

Deb, K., Pratap, A., Agarwal, S.,& Meyarivan, T. (2002). A fast and elitist multi- objective genetic algorithm: NSGA-II. *IEEE Transactions on EvolutionaryComputation*, 6(2), 182197.

Zitzler, E.,Laumanns, M., &Thiele, L. (2001). SPEA2: improving the performance of the strength Pareto evolutionary algorithm.*Technical Report 103, Computer Engineering and Communication Networks Lab (TLK)*, Swiss FederalInstitute of Technology.

Corne, DW., Jerram, NR., Knowles, JD., & Oates, MJ. (2001).PESA-II: region-basedselection in evolutionary multiobjective optimization, *In Proceedings of theGenetic and Evolutionary Computation Conference*, San Francisco, California, USA.

Eskandari, H., Geiger, CD., & Lamont, GB. (2007). FastPGA: a dynamic populationsizing approach for solving expensive multi-objective optimization problems, *4th International Conference on Evolutionary Multi-Criterion Optimization*,Matsushima, Japan.

Zitzler, E., & Kunzli, S. (2004). Indicator-based selection in multiobjective search, In:Yao X et al., editors. Parallel problem solving from nature (PPSN VIII).Berlin, Germany: Springer Verlag, 832- 842.

Srinivas, N., & Deb, K. (1995). Multi-objective optimization function optimizationusing non-dominated sorting genetic algorithms, 2(3), *Evolutionary Computation*,221-248.

jMetal. The jMetal framework. from http://jmetal.sourceforge.net/

TGFF. Task Graph For Free. fromhttp://ziyang.eecs.umich.edu/projects/tgff/index.html

Nebro, AJ., Luna, F., Alba, E., Beham, A., &Dorronsoro, B. (2006). AbYSS: Adapting Scatter Search for Multiobjective Optimization, Tech Rep. ITI-2006-2, De partamento de Lenguajes y Ciencias de la Computacin, University of Mlaga.

Durillo, JJ., & Nebro, AJ. (2011). jMetal: A Java framework for multiobjective optimization, *Advances in Engineering Software (Thomson Reuters)*,42(10), 760-771.

Coello, CA.C., & Pulido, G. (2001). A micro-genetic algorithm for multiobjective optimization. , Lecture Notes in Computer Science, 126-140.

Zhou, A., Qu, BY., Lui, H., Zhao, SZ., Suganthan, PN., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. Swarm and Evolutionary Computation, 1(1), 32-49.

Tafesse, B., &Muthukumar, V. (2013). Framework for simulation of heterogeneous MpSoC for design space exploration.,VLSI Design,1–16.

Sanchez G., Agostini L., Sousa L. & Marcon C. (2018). Parallelism exploration for 3D high-efficiency video coding depth modeling mode one. In *Journal of Real-Time Image Processing*, 1–11.

Stoychev I. et al. (2016). Sensor data fusion with MPSoCSim in the context of electric vehicle charging stations. *In the Proc. of IEEE Nordic Circuits and Systems Conference* (NORCAS), 1–6.

Rettkowski J., Wehner P., Schülper M.,& Göhringer D. (2015). A Flexible Software Framework for Dynamic Task Allocation on MPSoCs Evaluated in an Automotive Context. In: Sano K., Soudris D., Hübner M., Diniz P. (eds) *Applied Reconfigurable Computing*. Lecture Notes in Computer Science, vol. 9040. Springer, Cham.

Meloni P., Tuveri G., Pani D., Raffo L., Palumbo F. (2015). Exploring custom heterogeneous MPSoCs for real-time neural signal decoding.*In the Proc. ofConference on Design and Architectures for Signal and Image Processing* (DASIP), 1–8.

Iranfar A., Pahlevan A., Zapater M., Žagar M., Kovač M., Atienza D. (2018). Online efficient bio-medical video transcoding on MPSoCs through content-aware workload allocation. *In the Proc. ofDesign, Automation & Test in Europe Conference & Exhibition* (DATE), 949 – 954.

El Mimouni E.-H.& KarimM. (2014). A MicroBlaze-based Multiprocessor System on Chip for real-time cardiac monitoring. *In the Proc. ofthe International Conference on Multimedia Computing and Systems* (ICMCS), 331 – 336.