Heuristics for application mapping on mesh-based Network-on-Chip architectures

Student Name:

Supervisor Name:

Submitted as part of the degree of MEng Computer Science to the

Board of Examiners in the Department of Computer Science, Durham University

Abstract —

Context/Background - Due to shrinking transistor sizes, more and more processing elements and memory blocks are being integrated on a single die. However, traditional communication infrastructures cannot handle the synchronisation problems of these large systems. Using the network-on-chip (NoC) architecture (Kumar et al. 2002) is a step towards solving this communication problem. Energy and communication-efficient application mapping is a previously studied problem on NoC architectures, but current algorithms either have a significant runtime or do not determine accurate results.

Aims - This work aims to replicate and extend the findings of (Tosun et al. 2015), extend the current literature on scalability of algorithmic approaches, and improve the heuristics of contemporary application mapping algorithms.

Method - A survey of novel and existing algorithms is provided to deliver a level-playing field empirical analysis and ascertain the best such techniques across a range of task graphs. The scalability of these approaches is analysed to a greater extent than current literature by utilising real multimedia benchmarks as well as a set of random task graphs of increasing size.

Results - Despite its optimality, integer linear programming methods are shown to not be viable due to their computational complexity. Deterministic methods such as CastNet are shown to have the fastest runtime, however, the lowest communication-energy costs are attained by population-based algorithms as the application size increases. The novel EvoNet algorithm that we propose in this paper achieves heuristic superiority in almost all cases, and significantly reduces the runtime of the previously-best genetic algorithm.

Conclusions - Population-based heuristic methods are shown to be the strongest candidates for creating optimum or near-optimum solutions to problems where CPU runtime is one of the major evaluation criteria. For reusable architectures where the development cost can be amortised across many applications, we conclude that the optimality of EvoNet makes it the best such candidate mapping algorithm. Our findings significantly further the state-of-art.

Keywords — Network-on-Chip, Heuristic, Mapping, Optimisation, Energy, Communication

I INTRODUCTION

The continual advancement of semiconductor technology and the restrictions due to the transistor power-density law for designing efficient single-core processors together are rapidly driving computer architecture towards the many-core era. However, traditional communication infrastructures such as bus or star cannot handle the synchronisation problems of these large systems. Such infrastructures are plagued by several shortcomings including: poor scalability; low bandwidth; large latency; and high power consumption. Mitigating the development challenges for many-core processors needs an energy and communication-centric cross-layer optimisation method. To address these limitations, the network-on-chip (NoC) introduces a packet-switched fabric for on-chip communication and it becomes the de facto many-core interconnection mechanism.

The network-on-chip has become the central communication paradigm for system-on-chip (SoC) designs after it was introduced over a decade ago (Benini & De Micheli 2002). In the case of large-scale designs, NoC architectures are preferred as they reduce the complexity involved in designing the wire layout and also provide a well-controlled structure capable of increased power, speed and reliability. For high-end SoC designs, the NoC is considered the best integrated solution. An example network-on-chip (shown in Fig.1) is divided into regular tiles where each tile can be a general-purpose processor (referred to as an intellectual property (IP) block). A router is embedded within each tile with the objective of connecting it to its neighbouring tiles. Thus, instead of routing global wires, inter-tile communication can be achieved by routing packets via these embedded routers.

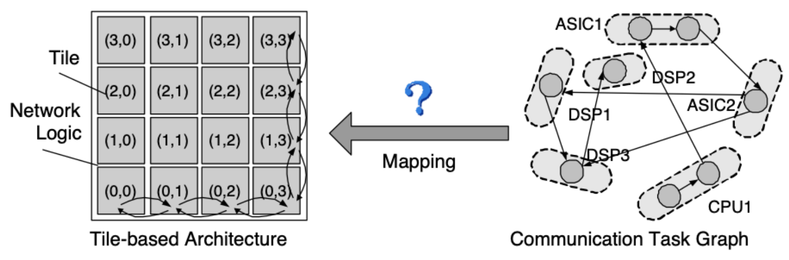

Figure 1: Tile-based architecture and the mapping problem. Taken from (Hu & Marculescu 2003).

Three key concepts come together to make this tile-based architecture very promising: structured network wiring, modularity and standard interfaces. Since the network wires are structured and wired beforehand, their electrical parameters can be very well controlled and optimised, making it possible to significantly reduce power dissipation and propagation delay. Modularity and standard network interfaces facilitate re-usability and interoperability of the modules. Moreover, since the network platform can be designed in advance and later used for many applications, it makes sense to highly optimise this platform as its development cost can be amortised across many applications.

To exploit this regular tile-based architecture, the application design flow is separated into three phases. Firstly, the application needs to be divided into a graph of concurrent tasks. Secondly, using a set of available IPs, the application tasks are assigned and scheduled. Finally, the designer determines the topological placement of these IPs onto different tiles via a mapping function such that the metrics of interest are optimised (such as latency, energy and data communication costs). For applications with only generic processing cores, the problem is reduced to the division of tasks and the mapping of those tasks to tiles. Referring to Fig. (1), the mapping phase determines onto which tile ((3,0), (2,1), (1,3) etc.) each IP (ASIC2, DSP3, CPU1, etc.) should be placed and maps an edge in the communication task graph onto a path within the architecture.

NoC architectures can be constructed using regular topologies or irregular (custom) topologies. Both topology types exhibit advantages and disadvantages: Irregular topologies suit the optimisation of different requirements such as the number of links and the number of routers to be used; however, they are not always reusable. If a regular network-on-chip is designed to fit the requirements of few highly communicative components, it will be largely over-designed with respect to the needs of the remaining components. Consequently, irregular network architectures might be necessary for realising application-specific SoCs. On the other hand, regular topologies can be reused and are easy to design, and as such, most multi-core architectures employ regular mesh-based topologies.

The mapping problem is considered to be one of the most important aspects of NoC design (Ogras et al. 2005) especially in the context of tile-based architectures, as it significantly impacts the energy and performance metrics of the system. Unfortunately, the mapping problem is an instance of the constrained quadratic assignment problem which is known to be NP-hard (Garey & Johnson 2002). The search space of the problem increases factorially with the system size. If the number of cores on the target architecture has enough cores for the given application, $n!$ different solutions can be found for the application with n tasks. This quickly becomes impossible to enumerate for large problems.

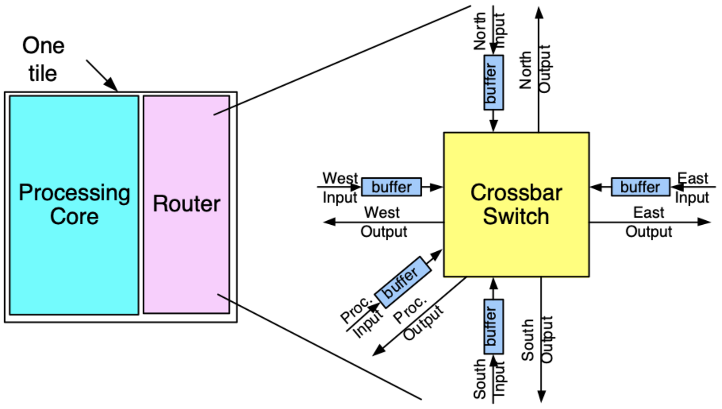

Figure 2: The typical structure of a regular tile. Taken from (Hu & Marculescu 2003).

Energy and communication-efficient application mapping is a previously studied problem for mesh-based NoC architectures; however, there is still a need for intelligent and efficient mapping algorithms since current approaches either have a significant run-time, or generate a non-optimal mapping. This project aims to expand the current literature on the topic of energy-aware application mapping algorithms and heuristics on mesh-based NoC architectures.

A Objectives

The research question posed is: *Can we undertake a level-playing field empirical analysis of mapping algorithms in order to ascertain the best such algorithms across a range of task graphs and network topologies?* To address this research question, the objectives for this project were divided into three categories: minimum, intermediate, and advanced.

- **Minimum:** Establish a 'mapping' framework so as to:
  - Define task graphs T.
  - Define topologies G.
  - Define and evaluate a mapping $F : T \to G$.
- **Minimum:** Implement, compare and evaluate random, simulated annealing and genetic mapping algorithms from (Tosun et al. 2015).
- **Intermediate:** Implement the ILP method and CastNet Heuristic algorithm from (Tosun et al. 2015).

- **Intermediate:** Critically evaluate and compare algorithms against the final achieved communication cost and execution time.
- **Advanced:** Implement, compare and critically evaluate several alternative heuristic/optimisation algorithms such as Artificial Bee Colony and Particle Swarm.
- **Advanced:** Evaluate the performance scalability of different algorithms to a greater extent than the current literature provides by testing each mapping strategy on applications of increasing size and complexity.
- **Advanced:** Enhance an existing heuristic or develop a new algorithm for efficient application mapping that is an improvement on either the execution runtime or the optimality of the resultant mapping.

By successfully fulfilling all of the above objectives, a thorough comparison of contemporary application mapping techniques is delivered, highlighting the effects of various heuristics and optimisation mechanics. We go on to demonstrate that our proposed algorithm **EvoNet** is superior in final communication cost, and significantly reduces the execution runtime for the largest application sizes. For reusable architectures where the development cost can be amortised across many applications, we conclude that the optimality of EvoNet makes it the best such candidate mapping algorithm.

II RELATED WORK

In this section, an overview of different techniques in the field of application mapping is given, highlighting several recent algorithmic and heuristic developments. While several studies have explored better solutions to the mapping problem from different angles, in this paper, our focus is on algorithms that target communication-energy minimisation.

A Application Mapping Algorithms

A binomial IP mapping technique, called BMAP is proposed in (Shen et al. 2007). This algorithm first maps the given application on a 1D mesh. Then, it iteratively applies a binomial merge, which is a simple tree merging problem, until the target mesh topology is achieved. Similarly, another energy-aware mapping algorithm based on a branch-and-bound method is presented in (Hu & Marculescu 2005). A branch-and-bound algorithm consists of a systematic enumeration of candidate solutions by means of state space search: the set of candidate solutions forms a rooted tree with the full solution set at the root. The algorithm explores branches of this tree, which represent subsets of the solution set. Before enumerating the candidate solutions of a branch, the branch is checked against upper and lower estimated bounds on the optimal solution, and is discarded if it cannot produce a better solution than the best one found so far. The algorithm depends on an efficient estimation of the lower and upper bounds of regions/branches of the search space. If no bounds are available, the algorithm degenerates to an exhaustive search. The authors compare their algorithms with a Simulated Annealing based method and show up to 51.7% energy savings. However, despite no comparison with contemporary mapping approaches being made in recent research, our work will not implement either of BMAP or the branch and bound algorithm due to time constraints.

Linear programming (LP) is an optimisation method for problems that can be mathematically formulated by linear equalities and inequalities, and is explored in (Tosun et al. 2009). An LP formulation consists of an objective function and one or more constraints that limit the solution space. ILP is a special case of LP, where the variables are restricted to strictly integer values. An ILP solver searches for the optimum solution in this limited space using different methods such as a simplex algorithm, and if the formulation fully covers the problem behaviour it determines the optimum solution. ILP is NP-complete, and thus seems the best option for linear optimisation problems if the number of variables is limited; however, when the number of variables in the formulations increases, the solution times can become unacceptable, therefore ILP methods are only good candidates for optimisation problems when there are limited numbers of variables. (Tosun et al. 2009) formulates the application mapping problem using 0–1 ILP formulations, in which variables are restricted to the integer values of either 0 or 1. In our work, we will restate the formulations to make this paper self-contained.

There are two types of heuristic mapping methods in the literature for the application mapping problem, the first being constructive and the second, iterative. A constructive mapping algorithm maps the tasks one by one onto the network topology by selecting tasks and cores based on some predefined decision criteria. After the solution is obtained, the position of the tasks on the topology is not changed. Moreover, in constructive mapping algorithms, the principal decisions to be made are the sequence of tasks to be mapped onto the mesh, and on to which cores these tasks will be mapped. For certain algorithms, the task selection criterion is twofold: initial task selection and remaining task selections. The core selection step can also be similarly divided into two parts, consisting of initial core selection for the first task and the core selection for the remaining tasks. A recent constructive heuristic method, Onyx, is presented in (Janidarmian et al. 2009). In this algorithm, the authors propose a method that selects the tasks to be mapped based on the assigned priority. The algorithm then maps the selected task by searching the candidate tiles on a lozenge-shaped path. The more recent research presented in (Tosun 2011*b*) extends this with the CastNet algorithm by using the symmetry of the mesh topologies to select the initial tiles. CastNet is a focus of our paper as results show it to have a fast run time with promising results for small problem sizes (less than 81 tasks). Their method is shown to be effective due to the small number of solutions generated through constructive heuristics. The results of this algorithm form the basis of our research into effective mapping heuristics, making CastNet a primary area of interest.

In the iterative case, once a solution is obtained, the algorithm iteratively tries to improve the solution by swapping the task allocation of cores until some specific condition is met. An iterative heuristic algorithm NMAP, proposed in (Murali & De Micheli 2004), maps the given application onto a 2D mesh under bandwidth constraints targeted to minimise the energy consumption of the final design. In their algorithm, they initially map the tasks based on their communication weights and then calculate the communication cost of this mapping using Dijkstra's shortest-path algorithm. Finally, they refine the mapping by iteratively improving it by swapping the allocation of pairs of tasks.

Genetic algorithms (GA) (Goldberg & Holland 1988) and simulated annealing (SA) are also commonly used meta-heuristics for NP-hard optimisation problems. Inspired by the biological process of evolution, a genetic algorithm mimics the evolutionary functions of selection, mutation, and crossover processes to determine optimum or approximate solutions. In nature, these biological functions are repeated over generations and deliver individuals best fitted to their environment. Individuals with low fitness do not progress to the next population and thus do not pass on any characteristics to the next generation (survival of the fittest). GA has been shown to be very attractive for computationally intense multi-parameter optimisation problems (Goldberg & Holland 1988); thus, we will also employ it for our mapping problem.

Simulated annealing, presented in (Kirkpatrick et al. 1983) is a probabilistic method to find global minimum cost configuration in a search space, and is inspired by the process of annealing in metallurgy. The process starts with an initial configuration and continuously reduces the temperature of the system in search for a better configuration. As the system cools, the acceptance criteria of samples is narrowed to focus on improving movements. If the process obtains a better energy value in a configuration, it directly accepts this configuration, otherwise, it accepts the new solution based on an acceptance probability function. In this way, it makes uphill moves from a local minimum in an attempt to jump to a valley where the global minimum might reside. The slow 'cooling' schedule allows a new low-energy configuration to be discovered and exploited.

SA- and GA-based methods are common within recent research, most notably in: (Lu et al. 2008) where SA was used for cluster-based mapping cores onto 2D mesh NoCs; (Moein-Darbari et al. 2009) where a Chaos-GA was proposed to minimise energy consumption of the design; and in (Ascia et al. 2004), which proposed a GA-based method that maps the application onto the mesh to optimise multi-objective functions. Both an SA and GA based approach will be implemented within this paper to serve as generic meta-heuristic algorithms for comparison. These strategies serve as reliable metrics due to their elasticity within the optimisation field. However, the historic primary downside of each approach is the proneness to terminate before an optimal mapping is found in the case of SA (Lu et al. 2008), and the sometimes unacceptable execution-time-to-optimality trade-off in the case of GA (Goldberg & Holland 1988).

Ant colony optimisation (ACO) (Dorigo & Birattari 2010) is a population-based meta-heuristic for the solution of difficult combinatorial optimisation problems. In ACO, each individual of the population is an artificial agent that incrementally and stochastically builds a solution to the considered problem. Agents build solutions by moving on a graph-based representation of the problem. At each step their moves define which solution components are added to the solution under construction. A probabilistic model is associated with the graph and is used to bias the agents' choices. The probabilistic model is updated on-line by the agents so as to increase the probability that future agents will build good solutions. (Ferrandi et al. 2013) proposes an algorithm based on ACO that simultaneously executes the scheduling, mapping and linear placing of tasks on a reconfigurable NoC, hiding reconfiguration overheads through pre-fetching. The ACO heuristic gradually constructs solutions and then searches around the best ones, cutting out non-promising areas of the design space, and (Ferrandi et al. 2013) demonstrates that their proposal is more general and robust, and finds better solutions (16.5% on average) than an equivalent implementation that does not take into account hardware reconfigurability.

The Particle Swarm Optimisation (PSO) algorithm is presented in (Kennedy & Eberhart 1995), and is inspired by the combined social behaviour of a flock of birds or a school of fish. (Sahu et al. 2013) presents a discrete particle swarm optimisation-based strategy to map applications on both 2D and 3D mesh-connected Networks-on-Chip. Their mapping results, in terms of the overall communication metric, have been compared with well-known techniques reported in the literature and also with exact methods built around integer linear programming. (Sahu et al. 2013) shows that PSO-based results are superior to those from reported techniques, such as Ant Colony, BMAP and NMAP. For smaller benchmarks, the results obtained are same as those corresponding to the ILP formulation, establishing the quality of the solution strategy.

The Artificial Bee Colony (ABC) algorithm, presented in (Karaboga & Basturk 2007) is a multi-variable optimisation algorithm based on the intelligent behaviour of honey bee swarm. The results produced by ABC for the optimisation of various multi-modal functions are compared with GA, PSO and Particle Swarm Inspired Evolutionary Algorithm (PS-EA), and show that ABC outperforms the other algorithms in nearly all cases. Although the ABC algorithm has not yet been applied to the application mapping problem, its demonstrated superiority in other domains is a strong case for its implementation and comparison with the other techniques outlined in this paper.

B Summary

While several studies have explored better solutions to the mapping problem from different angles, in this paper our focus is on algorithms that target energy minimisation. We do not consider the area constraint due to the fact that chips of the current technology are abundant of transistors: instead, we map applications to the minimum sized 2D mesh that will fully encompass all tasks. Literature on the topic of application mapping often compares recently developed algorithms with old benchmarks and optimisation techniques, and so this paper will provide a contemporary comparison between several such techniques in order to give a more level-playing-field empirical analysis of approaches. Artificial Bee Colony, a novel optimisation technique, has not yet been applied to the mapping problem, and so is adapted and implemented to serve as another benchmark. The document constraints of this paper means it would have been impossible to implement all contemporary techniques from the literature, and so several of the more outdated algorithms had to be ignored.

There is a similar but dated comparison work presented in an earlier study (Marcon et al. 2008). Our work differs from this study in three aspects. First, we propose our own methods for each optimisation technique. Second, we use several contemporary heuristic algorithms including our novel algorithm EvoNet developed within this paper to compare with other optimisation methods. Finally, we compare our methods using both real benchmarks and random graphs, and we experiment with significantly larger application sizes (up to 256 tasks, as opposed to 81 tasks in recent literature). This work ultimately aims to replicate and extend the findings of the recent research in (Tosun et al. 2015), extend the current literature on scalability of algorithmic approaches, and improve the heuristics of contemporary application mapping algorithms.

III SOLUTION

The solution designed to implement the objectives is presented in this section. To start, an overview of the general architecture of the solution is given, subsequently, the detailed solution and the underlying algorithms are described in the ensuing sub-sections.

A *Architecture and Implementation Tools*

The solution was implemented in Python, a popular language that provides an extensive toolbox of scientific libraries for all kinds of applied programming problems. The system is categorised into distinct components with some shared functionality. A generalised **Topology** class is provided, allowing an n dimensional regular or irregular mesh to be defined. The class contains several simple helper methods, providing functionality for: swapping the tasks allocated to a pair of cores; generating a random task mapping from a given task graph; calculating the Manhattan distance between two cores, etc. Such methods are commonly called by the majority of the contemporary application mapping algorithms, and so encapsulation in this way makes sense. Individual algorithms are implemented in their own class, and manipulate Topology objects throughout their execution, returning their lowest-cost mapping.

B *The Application Mapping Problem*

The objective is to find a mapping of tasks onto different NoC cores such that the total communication energy consumption is minimised, while guaranteeing the performance of the system. To formulate this problem, we introduce the following concepts:

Definition III.1. A **WCTG** (weighted communication task graph) represents the input application, and is a directed graph $G(V, E)$ where each vertex $v_i \in V$ represents a task in the application and each edge $e_{i,j} \in E$ represents a dependency between two tasks v_i and v_j. The amount of data transferred between v_i and v_j is represented by the weight $w_{i,j}$ for each edge $e_{i,j}$ in bits per second.

Definition III.2. A **TG** (topology graph) represents the target architecture, and is a graph $G(P, L)$ where each vertex $p_i \in P$ denotes a processing core in the topology and each edge denotes a physical link $l_{i,j} \in L$ between p_i and p_j.

Using the above definitions, the application mapping problem can be formulated as follows: Given a WCTG and a TG that satisfy:

$$|V| \leq |P| \tag{1}$$

find a one-to-one mapping function $F : V \to P$ from a WCTG to a TG that minimises the energy consumption of the NoC. We can state the mapping problem mathematically as:

$$\textbf{Minimise} : E_{NoC} \tag{2}$$

such that:

$$\forall v_i \in V, \exists p_k \in P, F(v_i) = p_k \tag{3}$$

$$\forall v_i \neq v_j \in V, F(v_i) \neq F(v_j). \tag{4}$$

Conditions (3) and (4) mean that each task should be mapped to exactly one core, and no core can host more than one task. The definition of E_{NoC} is given in the energy model in the next subsection.

The routing algorithm is an important factor in NoC designs because it establishes the path of the messages between cores. Deadlocks may appear if the routing algorithms are not carefully designed. A deadlock occurs when no message or packet can advance toward its destination because the queues of the message system are full, and arise due to finite resources such as the bandwidth of the links, or the size of the routing buffer at each core. Routing algorithms must be deadlock-free, and must also ensure that the network fulfils all predefined performance/bandwidth constraints. Routing algorithms fall into two categories: deterministic and adaptive. A well-known deterministic example within mesh-based networks is XY routing (Duato et al. 2002), where packets are first routed horizontally, and then vertically from the current router with coordinates $C(x, y)$ to the destination router $D(x, y)$. Note that, for 2D mesh networks with XY routing, equation (7) shows that the average energy consumption of sending one bit of data from processing core p_i to p_j is proportional to the Manhattan distance between the two tiles onto which those cores are mapped. The main advantages of deterministic routing algorithms are their low implementation cost and traffic predictability, and they often require fewer resources than adaptive algorithms. Additionally, deterministic routing algorithms offer a predictable traffic pattern, making it easier for a designer to analyse the network. On the other hand, adaptive routing algorithms have better resilience to deadlocks and can be used in high-traffic networks and in networks with limited bandwidth (Tosun et al. 2015). In this work, we utilise deterministic XY routing.

C Energy Model

Our goal within this paper is not to minimise the energy consumed by the cores within the NoC, as this is independent of the generated network topology, rather, to try to minimise the energy consumed by the network resources. The energy consumed by a NoC is directly proportional to the amount of bit transitions in the network, and so to estimate the total energy consumption of a given NoC architecture, we use an energy model based on the total number of bit transitions. One well accepted energy model is given in (Hu & Marculescu 2005), and takes into account the energy $E_{T_{bit}}$ consumed when one bit of data is transported through the network as:

$$E_{T_{bit}} = E_{S_{bit}} + E_{B_{bit}} + E_{W_{bit}} + E_{L_{bit}} \tag{5}$$

where $E_{S_{bit}}$, $E_{B_{bit}}$, $E_{W_{bit}}$ and $E_{L_{bit}}$ represent the energy consumed by the switch, the buffer, interconnection wires inside the fabric, and the links, respectively. Since the energy consumption of the buffering $E_{B_{bit}}$ and the internal wires $E_{W_{bit}}$ are negligible (Hu & Marculescu 2005), the energy model can be reduced to:

$$E_{T_{bit}} = E_{S_{bit}} + E_{L_{bit}} \tag{6}$$

Consequently, the average energy consumption of sending one bit of data from processing core p_i to p_j can be calculated as:

$$E_{T_{bit}}^{p_i,p_j} = (n_{hops} + 1) * E_{S_{bit}} + n_{hops} * E_{L_{bit}} \tag{7}$$

where $n_{hops} + 1$ is the number of switches the bit passes, and n_{hops} is the number of links it traverses on its way along its path. (7) gives the energy model for the regular tile-based NoC architecture.

Let v_a and v_b be the tasks mapped onto the cores p_i and p_j respectively via a mapping function F. The communication-energy between two tasks with a communication weight $w_{a,b}$ can be calculated as:

$$E_{total}^{v_a,v_b} = w_{a,b} E_{T_{bit}}^{F(v_a),F(v_b)} \tag{8}$$

Finally, we can determine the total energy consumption of the NoC architecture as follows (Tosun et al. 2015), where $e_{a,b} \in E$ denotes the communication dependency between two tasks v_a and v_b:

$$E_{NoC} = \sum_{\forall e_{a,b} \in E} E_{total}^{F(v_a),F(v_b)} \tag{9}$$

As illustrated previously in (7), one significant approach to lessen energy consumption is to limit n_{hops} between related cores in the NoC as much as possible. It goes without saying that this approach would be effective for other performance parameters including latency.

D Algorithm Implementation

D.1 Simulated Annealing

The algorithm is initialised with a random mapping, and we set the temperature to its highest value, after which, the algorithm executes two nested loops. While the external loop searches for global minima, the internal loop tries to refine the local solution. The internal loop randomly swaps the allocation of two cores to determine a new solution. It then evaluates if the new solution is better than the solution at hand, and accepts it to be the current solution if it is. Otherwise, it generates a random variable α, where $0 \leq \alpha \leq 1$ and compares it with the acceptance probability function $e^{(-\Delta C)/temperature}$. If the result of the function is higher than α, the new move is accepted. At high temperatures, the acceptance probability is also high, but decreases with the temperature of the system. After each iteration, the temperature is decremented, and we start a new iteration accepting the solution at hand as our initial solution for the new iteration. In this way, SA is able to avoid becoming stuck in local minima. (Marcon et al. 2008) states that the algorithm obtains good results when the initial temperature is selected to be $\lceil 10 \ln |P| \rceil$, where $|P|$ is the number of tiles in the mesh, and that the number of external loop iterations is set $|P|^2$. A downside of the algorithm is that it only manipulates one solution throughout its execution, therefore if this initial mapping has a poor total communication cost, the algorithm could potentially be very slow to converge if at all.

D.2 Genetic Algorithm

In our GA method, we initially create a chromosome structure that represents a valid mapping. In this representation, a chromosome is constructed by a string of genes, each of which represents a tile of the mesh (shown in Fig. 3). Gene i represents tile i in a chromosome, meaning that the size of a chromosome is limited to the number of tiles in our mesh. Initially, a set of n randomly generated chromosomes form generation zero. The fitness of each chromosome is then calculated as the total communication cost as given in Eq. (9).

The optimisation process starts from the initial population and repetitively applies crossover, mutation, and selection operators to generate new individuals. In the crossover operation given in (Tosun et al. 2015), we first pick two parent chromosomes from the population using a probabilistic policy whereby an individual has a higher chance of selection if it has a higher fitness. This probabilistic selection is a roulette wheel selection mechanism which is described as equation below:

$$P_i = \frac{1/fit_i}{\sum_j 1/fit_j} \tag{10}$$

A cut index is then randomly determined at which to chop and swap the second portion of each chromosome, returning two child individuals. However, the newly generated chromosomes are generally invalid because some of the tasks occur twice in a chromosome while some are missing from the same chromosome. A repair procedure to produce a pair of valid child chromosomes is then applied that first removes the doubly listed tasks from the chromosomes then randomly assigns all the missing tasks to empty genes until all tasks are present in the chromosome once. Next, with probability p_{mutate} we randomly swap two genes, creating new individuals in the population (equivalent to swapping the task allocation of two cores) - this is the mutation step. The child with the best fitness of the two newly generated chromosomes is then added into the population. In order to calculate the fitness of a chromosome, it must be converted back to its original structure. After a crossover operation, n individuals in the population doubles to $2n$ individuals. In the selection process, we select n individuals from the population list based on their fitnesses and carry them to the next iteration. We apply the crossover and mutation operators until our stopping criterion is met. As our stopping criterion, we set a limit on the number of iterations that bear no improvement, and then select and return the solution with the best communication cost value.

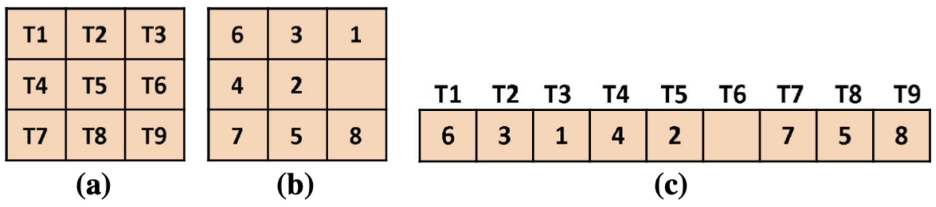

Figure 3: a.) The sequence for a 3x3 mesh, b.) mapping of a generic WCTG, c.) chromosome representation of the mapping. Taken from (Tosun et. al 2015).

The exact mechanics of the crossover operation can significantly affect the quality of successive population generations, therefore care is taken to define a crossover mechanism that maintains a balance between genetic inheritance and diversity. The chromosome structure encodes the structure of the mapping, but conversion to the chromosome structure, crossover, mutation and conversion back to the mapping structure has the potential to remove any underlying similarity between the parents and the newly generated child individual. Having such a characteristic could negatively affect the algorithm, as successive generations would essentially be populations of random individuals with no resemblance to the previous iteration.

D.3 ILP

(Tosun 2011a) formulates the problem as follows, where each task is assigned to a tile, and where some tiles can be empty if there are fewer tasks than tiles:

Table 1: Constants and variables used in the ILP formulation (Tosun et al. 2009)

Symbol	Quantity
n	The number of tasks in the WCTG.
$w_{i,j}$	The communication weight between tasks i and j in the WCTG.
$X\text{dim}$, $Y\text{dim}$	The size of the mesh in the x and y dimensions respectively.
$\alpha_{i,x,y}$	Binary variable. $\alpha_{i,x,y} = 1$ if task i is mapped to the coordinates (x, y), otherwise $\alpha_{i,x,y} = 0$.
$X_{i,j,a}$	Binary variable. $X_{i,j,a} = 1$ if the distance in the x dimension between tasks i and j is equal to a. Otherwise, $X_{i,j,a} = 0$
$Y_{i,j,b}$	Binary variable. $Y_{i,j,b} = 1$ if the distance in the y dimension between tasks i and j is equal to b. Otherwise, $Y_{i,j,b} = 0$
$X\text{cost}$, $Y\text{cost}$	The total communication cost in the x and y dimensions respectively.

$$\sum_{x=0}^{Xdim} \sum_{y=0}^{Ydim} \alpha_{i,x,y} = 1, \forall i \qquad \sum_{i=1}^{n} \alpha_{i,x,y} \leq 1, \forall x, y \tag{11}$$

(12)

To determine the total communication cost, we first find the Manhattan distance between two communicating tasks. We use constraints (13) and (14) to capture the distances a and b in dimensions x and y, respectively.

$$\begin{aligned} X_{i,j,a} \geq \alpha_{i,x_i,y_i} + \alpha_{j,x_j,y_j} - 1, \forall i, j.e_{i,j} \in E \\ 0 \leq x_i, x_j \leq X\text{dim}, 0 \leq y_i, y_j \leq Y\text{dim s.t. } a = \|x_j - x_i\| \end{aligned} \tag{13}$$

$$\begin{aligned} Y_{i,j,b} \geq \alpha_{i,x_i,y_i} + \alpha_{j,x_j,y_j} - 1, \forall i, j.e_{i,j} \in E \\ 0 \leq x_i, x_j \leq X\text{dim}, 0 \leq y_i, y_j \leq Y\text{dim s.t. } b = \|y_j - y_i\| \end{aligned} \tag{14}$$

Using (15) and (16), we then calculate the cost in the x and y dimensions. In these formulas, a and b represent the number of hops in the x and y dimensions, respectively. Multiplying these values with the communication weight $w_{i,j}$ of two communicating tasks i and j gives us the total communication cost of these two allocations in the architecture:

$$X\text{cost} \ = \sum_{e_{i,j} \in E} \sum_{a=1}^{X\text{dim}} w_{i,j} \times a \times X_{i,j,a} \qquad Y\text{cost} \ = \sum_{e_{i,j} \in E} \sum_{b=1}^{Y\text{dim}} w_{i,j} \times b \times Y_{i,j,b} \tag{15}$$

(16)

Thus, the objective function (Tosun et al. 2015) is expressed as:

$$\textbf{Minimise: } \text{Comm} = X\text{cost} + Y\text{cost} \tag{17}$$

Due to the computational intensity of ILP-based methods, and the magnitude of IPs we hope to map to NoC architectures in the modern world, it is likely that the only performance improvements of ILP-based strategies in the future will be down to faster hardware, making them less viable than contemporary heuristic approaches.

D.4 CastNet

CastNet (Tosun 2011*b*) is a low-complexity heuristic algorithm for application mapping onto mesh-based NoC architectures. The algorithm aims to map highly communicating tasks close to each other in order to minimise the number of links the data has to travel through. The CastNet algorithm is a constructive algorithm, however it finds more than one solution based on the symmetry properties of the mesh topology. The number of solutions is equal to the number of cores in a minimal symmetric core region in the mesh.

Two important decisions for the mapping problem are the selection of the first task to map and the selection of the initial core to map this task to. In the CastNet algorithm, we select the tasks based on their priorities. We assign the priorities based on the tasks' total communications with their neighbours and their average communication for each task within the WCTG. We select the task with the highest priority as the first task to be mapped onto the mesh. For the initial core selection, we pick the maximum number of alternative mapping positions as the number of cores in a symmetric core region on the mesh. For a given 2D square mesh, the symmetric core regions on the mesh are found when we equally divide the mesh into eight symmetric regions with vertical, horizontal, and two diagonal cuts. Cores intersected on a region are also included in that region. We choose initial core locations from one region, as this means we will have explored all possible symmetric solutions. We then apply the mapping algorithm for each selected initial core, mapping our highest priority task onto this initial core. After the first task is mapped to the first core, the algorithm maps all remaining tasks one by one onto the mesh until it arrives at a complete and valid mapping. Successive tasks are selected by calculating each unmapped task's total communication with the mapped ones, and choosing the task with the highest communication. Ties are broken by choosing the task with the highest priority. For this selection, we do not need to search for every task in the unmapped tasks; we only look for the tasks that have edges with mapped tasks. The generated mapping with the lowest total communication cost is returned as the best mapping.

D.5 Discrete Particle Swarm Optimisation (DPSO)

Particle swarm optimisation is a population-based stochastic technique developed in (Kennedy & Eberhart 1995), inspired by the social behaviour of bird flocking or fish schooling. In a PSO system, multiple candidate solutions coexist and collaborate simultaneously. Each solution, called a particle, flies in the problem space according to its own experience as well as the experience of neighbouring particles. Each particle's movement is influenced by its local best-known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

In the general DPSO framework, the position of a particle in an n-dimensional space at the kth iteration is $p_k =< p_{k,1}, p_{k,2}, ..., p_{k,n} >$, and represents a mapping of n tasks. For the ith particle, the quantity is denoted as p^i_k. Let $pbest^i$ be the local best solution that particle i has seen so far, and $gbest_k$ be the global best particle of iteration k. The new position of particle i is calculated as:

$$p^i_{k+1} = (s_1 \times I \oplus s_2 \times (p_k \rightarrow pbest^i) \oplus s_3 \times (p_k \rightarrow gbest_k)) \cdot p^i_k \tag{18}$$

In the above expressions, $a \rightarrow b$ represents a sequence of swaps applied on components of a to transform it to b. The operator $\oplus$ is the fusion operator. For two swap sequences a and b, $a \oplus b$ is equal to the sequence in which the sequence of swaps in a are followed by the sequence of swaps in b. The constants s_1, s_2, and s_3 are the inertia, self-confidence, and swarm confidence values. The quantity $s_i \times (a \rightarrow b)$ means that the swaps in the sequence $a \rightarrow b$ will be applied with a probability s_i. I is the sequence of identity swaps, and corresponds to the inertia of the particle to maintain its current configuration. Equation (18) shows how each particle is updated stochastically by a sequence of swaps that move it towards both its previous best and the global best. (Sahu et al. 2013) experimentally determines that the values of $s_1 = 1.0$, $s_2 = 0.04$ and $s_3 = 0.02$ are observed to give good results for most applications.

D.6 Artificial Bee Colony

A colony of honey bees can successfully accomplish tasks through social cooperation. In the ABC algorithm, there are three types of bees: employed bees, onlooker bees, and scout bees. Employed bees search around the food source (solution) in their memory; meanwhile they share the information of these food sources to onlooker bees. Onlooker bees prefer to select rich food sources from those found by employed bees when deciding where to conduct their own searching. Food sources that have higher quality (fitness) will have a greater chance of being selected by onlooker bees than those of lower quality. Employed bees that abandon their food sources become artificial scouts and search for new food sources.

The first half of the swarm consists of employed bees, and the second half constitutes onlooker bees. The number of employed or onlooker bees is equal to the number of solutions in the swarm. Initially, a randomly distributed population of SN solutions (food sources) is generated, where SN denotes the swarm size. Let $X_i = \{x_{i,1}, x_{i,2}, ..., x_{i,n}\}$ represent the i^{th} solution in the swarm, where n is the number of tasks in the application.

Each employed bee X_i generates a new candidate solution V_i by performing a random swap of the allocation of two tasks. The bee greedily memorises the solution X_i or V_i that has a better fitness. After all employed bees complete the search process, they share the information of their food sources with the onlooker bees. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with the roulette wheel selection policy given in (10). Each onlooker bee attempts to improve the solution it was drawn to by performing a random swap to the allocation of two tasks. If a solution cannot be improved over a predefined number of cycles, then the solution is abandoned and a new random solution is determined.

D.7 Proposed EvoNet

As the number of tasks in the application increases, the average cost of a random mapping compared to the cost of an optimal mapping grows significantly for a given topology. In the case of GA which initialises its execution with a population of random solutions, for increasing application sizes, the time taken to approach an optimal mapping increases significantly because the number of swaps from a random to an optimal mapping is a function of the number of all possible solutions, and thus grows significantly with the problem size.

We propose the **EvoNet** two-stage algorithm, which aims to both reduce the execution time and improve the optimality of the resultant mapping. Stage one is the population creation step, and aims to be a significant improvement over the standard GA due to seeding a subset of the initial population with the set of strong mappings generated by the CastNet heuristic algorithm. The remaining individuals in the population are randomly generated to maintain solution variance. CastNet constructs as many close-to-optimal solutions as there are locations in a symmetry core region for a given topology, and this number increases with the size of the application. Thus, for a square mesh topology to allocate 128 tasks there are 21 solutions generated, and for the allocation of 256 tasks there are 36 solutions generated. With this in mind, EvoNet seeds the initial population of the GA with this set of close-to-optimal solutions in order to significantly increase the initial average fitness as opposed to a population of random solutions. From here, the evolutionary phase executes as described previously in the GA.

CastNet's primary success was its rapid time to solution, but its downside was that the resultant mapping it generated diverged from optimality progressively as the application grew, leaving heuristic superiority to GA (Tosun et al. 2015). Through the use of its genetic crossover mechanism EvoNet will be able to yield a superior optimality in its resultant mapping (as GA was shown to in (Tosun et al. 2015)) but with a substantially diminished execution time due to CastNet's ability in the population generation phase to provide a 'headstart'. EvoNet is a simple amalgamation of the iterative and constructive heuristics of CastNet and GA, combining the best aspects of each respectfully in order to deliver the lowest-cost resultant mappings.

IV RESULTS

In this section, the results and analysis of the comparison of the outlined mapping algorithms are presented. To effectively evaluate the quality of the algorithms implemented and developed within this project, original data sets from cited papers were tested on as a control with interleaved analysis from several larger sources. The number of vertices and edges are important parameters for the mapping problem because they heavily affect the mapping result and execution time.

We tested the aforementioned mapping methods on real multimedia benchmarks and on randomly generated task graphs. Six video applications were selected from the literature: the video object plane decoder (VOPD) and the MPEG-4 decoder from (Janidarmian et al. 2009); the multi-window display (MWD) from (Chang & Chen 2008); and the 263 decoder (263 Dec.), 263 encoder (263 Enc.) and MP3 encoder (MP3 Enc.) from (Srinivasan et al. 2006). Furthermore, we experimented with randomly generated weighted communication task graphs with varying numbers of tasks (32 up to 256) from the Mendeley Data Set (Boveiri 2018). Problems of this size are sufficiently large enough to determine the asymptotic scalability of our proposed methods. For all population-based meta-heuristic approaches, a population size equal to the number of tasks n in the application was used in order to ensure fair comparison.

For the random method, we arbitrarily map tasks onto cores of the mesh, producing 1,000 solutions and selecting the one with the best energy savings. There is no intelligent optimisation technique in this method except to choose the better of two candidates. Additionally, each run may output different solutions. This method was implemented to serve as a baseline.

A Multimedia Benchmarks

As highlighted in Table (2) and Figure (4), we see that apart from the Random algorithm, all heuristic approaches fare similarly. Each strategy achieved within 3-4% of the optimal communication cost for a mapping of the multimedia benchmarks, however SA was consistently the worst heuristic. Among the methods presented above, the ILP method guarantees the optimal solution. However, ILP runtime complexity is the highest by far, thus will likely not produce solutions within acceptable time limits for large problems. CastNet achieved inward of 2% of the optimal mapping cost in the fastest execution time for all benchmarks. GA and PSO both obtained very similar communication costs (with both achieving optimality in two benchmarks), however GA has a longer execution time, being second slowest overall. The algorithmic improvements of EvoNet are apparent, realising a significant speedup of at least 40% (Table (3)), and achieving optimality in all but one benchmark.

Table 2: Communication cost comparison of mapping methods

Benchmark	Total communication (Mbit/s)							
	Random	GA	EvoNet	SA	CastNet	ILP	ABC	DPSO
VOPD	5,974	4,141	**4,119**	4,290	4,167	**4,119**	4,243	4,151
MPEG-4	5,066	**3,567**	**3,567**	3,631	3,631	**3,567**	3,631	**3,567**
MWD	1,504	1,152	**1,120**	1,344	**1,120**	**1,120**	1,312	**1,120**
263 Dec.	27,916	19,871	**19,646**	20,836	20,010	**19,646**	20,582	19,666
263 Enc.	299,052	230,432	230,432	231,017	230,432	**230,407**	230,997	230,432
MP3 Enc.	19,491	**15,521**	**15,521**	16,141	15,546	**15,521**	15,781	15,541

Table 3: Execution time comparison of mapping methods

Benchmark	V	E	Execution Time (s)							
			Random	GA	EvoNet	SA	CastNet	ILP	ABC	DPSO
VOPD	16	20	0.47	20.0	5.53	19.4	**0.33**	13,334	133	8.99
MPEG-4	12	13	0.41	16.8	3.91	18.6	**0.32**	479	122	9.05
MWD	12	12	0.48	16.3	2.32	28.6	**0.36**	5,849	112	7.19
263 Dec.	14	15	0.45	22.8	15.6	21.8	**0.36**	11,221	152	13.8
263 Enc.	12	12	0.46	14.2	10.2	25.8	**0.36**	3,843	92	8.41
MP3 Enc.	13	13	0.48	16.4	10.9	21.0	**0.36**	14,449	149	9.49

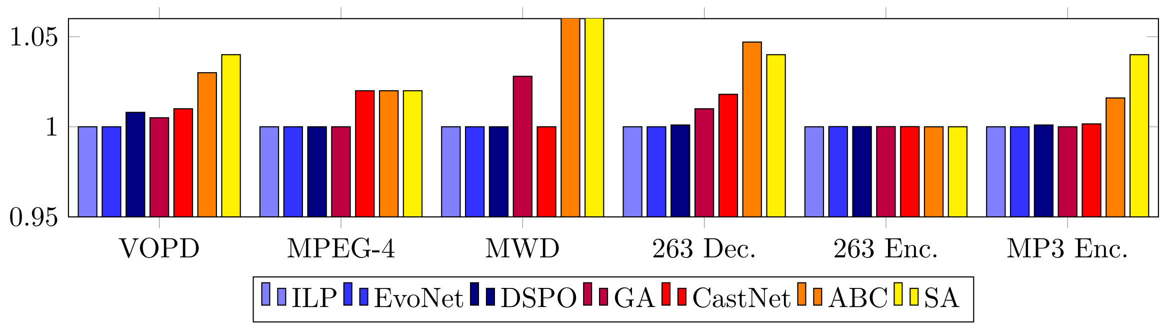

Figure 4: Normalised energy consumption values with respect to results generated by ILP

The SA and GA methods are meta-heuristics that rely on probabilistic decisions, and they are common optimisation techniques even though they do not guarantee optimal results. They start with randomly generated initial mappings and iteratively improve them through local search and the crossover function respectively. GA has a greater time complexity than SA due to the increased number of solutions it manages, but it is also shown to obtain better results than SA in all cases, albeit at the expense of higher execution times. This is the primary downfall of SA, as we see that the number of swaps required to transform a single random solution into the optimal mapping grows substantially with the problem size.

As seen in Figure (4), despite PSO, GA and EvoNet not achieving optimality for all benchmarks, all obtained a total communication cost within a fraction of a percentage point of the best mapping, highlighting their promise as effective optimisation strategies (with EvoNet in particular). The ABC algorithm performs poorly on the benchmarks despite its proven success in other optimisation fields. If the cut-off value within ABCs solution abandonment characteristic is too low, the neighbourhood search space for a particular solution will not be explored significantly enough to gain an improvement. This leads to that potentially viable solution being abandoned and replaced by a worse, randomly generated solution. However, increasing the cut-off value merely slows the execution runtime of the algorithm. This characteristic, compounded with the relatively inefficient search strategy (shared with SA) is likely the reasoning behind ABCs disappointing performance.

B Mendeley Dataset

The Mendeley Dataset (Boveiri 2018) contains 125 random task-graphs for multiprocessor task scheduling varying in size, communication-to-computation ratio (CCR), and parallelism. CCR is defined as a measure of how communication/computation-intensive a task graph is, and parallelism refers to the average number of data-dependencies for each task, contributing to the connectivity of the resulting task-graph. Each mapping approach was evaluated on the task-graph with middling parallelism and CCR characteristics for applications with a number of tasks in the set $\{32, 64, 128, 256\}$. This set was chosen in order to give a valid comparison on some of the hardest available problems to date, significantly harder than previous research explores.

Table 4: Communcation Cost comparison of mapping methods

Vertices	Edges	Total Communication (Mbit/s)						
		Random	GA	EvoNet	SA	CastNet	ABC	DPSO
32	120	103,937	72,372	**71,380**	92,533	75,971	88,852	72,077
64	272	323,848	203,795	**198,744**	300,043	217,016	298,946	212,875
128	590	1,113,736	638,958	**603,601**	1,007,858	691,063	1,005,840	684,009
256	1,221	3,042,281	(2,159,577)	(**1,877,775**)	2,990,497	1,906,422	2,963,115	1,888,406

Table 5: Execution Time comparison of mapping methods

Vertices	Edges	Execution Time (s)						
		Random	GA	EvoNet	SA	CastNet	ABC	DPSO
32	120	0.9	1,201	161	605	0.9	900	506
64	272	2.0	51,614	33,306	10,223	6.8	14,000	1,874
128	590	6.0	593,201	406,438	54,032	273	60,322	8,640
256	1,221	24.6	(604,800)	(604,800)	354,985	5,552	355,433	40,032

Our experiments showed that despite ILP obtaining optimal results for the multimedia benchmarks, its runtime complexity prevented it from generating a solution for graphs with more than 16 tasks within an acceptable time limit.

Concerning accuracy, random mapping is the worst method among the candidates because there is no associated intelligence or improvement mechanism with it; each mapping is independent from the others. Even when generating an order of magnitude more random mappings, the solution quality barely improves by more than a few percent.

The previously highlighted negative effects of a poor local search mechanism in both SA and ABC are clearly exacerbated for the larger task graphs, leading to both a longer execution time and a poorer final communication cost when compared to CastNet.

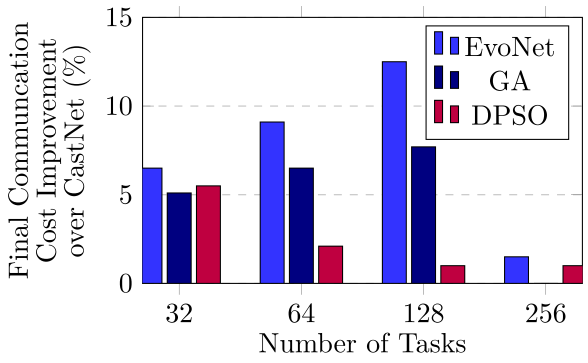

Figure 5: Energy consumption improvements of GA, EvoNet and DPSO against CastNet

The DPSO formulation utilises a less intensive crossover, but because particles only ever move towards each other, the swarm can quickly converge and become trapped in a local optimum. GA and EvoNet are able to improve their solutions with larger jumps for a greater number of iterations due to the more explorative nature of their improvement mechanism combined with their 'survival of the fittest' law. This means that weaker solutions are never kept.

CastNet was designed primarily for its rapid time-to-solution, however for significantly large application sizes ($\geq$ 256 tasks), even the execution time of CastNet becomes substantial, highlighting the need for efficient search strategies. The fundamental problem that all optimisation algorithms face is the potential of getting trapped in local minima (Tosun et al. 2015), thus, effective heuristic optimisation algorithms must utilise a calculated compromise between exploration and exploitation in order to generate efficient mappings (Sahu et al. 2013). Algorithms that maintain a population of solutions at any one time, particularly as the application size increases, are able to mitigate such a problem to a greater degree. However, the results obtained by strategies employing poor exploration schemes such as CastNet and DPSO diverge from the optimum, thus leaving the heuristic leadership to GA and EvoNet, as shown in Table (4). The bracketed results in Tables (4) and (5) denote that the execution time-out value of 7 days was reached for the task graph containing 256 nodes, however it is worth noting that within this time EvoNet was still able to achieve the lowest communication cost out of all the heuristic strategies, and given extra time would likely continue improving its result.

EvoNet's significant improvement to the field is due to the ability of the initial population generation phase to make a sizeable jump towards the area in the search space containing the global optimum. This avoids wasting precious time improving random solutions, as shown in Table (5). The 'headstart' means that the entirety of the evolutionary phase of its execution is spent improving an already strong set of solutions, ultimately leading to the lowest resultant mapping communication cost.

For large optimisation problems, CPU runtime is one of the major algorithm evaluation criteria. In our mapping problem, determining the solution in a short time is vital since time-to-market is a crucial parameter in the integrated circuit field. In such a case, heuristic methods are shown to be the strongest candidates for creating optimum or near-optimum solutions. Our results show that the novel EvoNet algorithm is able to achieve significant improvements of between 2.6% and 12% over CastNet for the largest applications, with a 40% reduction in execution time compared to GA, establishing its heuristic supremacy within the field when it comes to optimality.

V CONCLUSIONS

Our project achieved significant success in fulfilling all set aims. In this paper, we have provided a comprehensive commentary on the state of current research and presented several application mapping algorithms for mesh-based network-on-chip architectures to reduce energy consumption. These algorithms are based on ILP, and heuristic methods, and were compared on several real multimedia benchmarks and on randomly generated task graphs of varying sizes up to 256 tasks: significantly larger than previous research has experimented on.

CastNet and DPSO obtained very promising results for the real applications with 16 tasks or fewer, however when the number of tasks in the application graphs increases, the results obtained by CastNet and PSO diverge from the optimum leaving the heuristic leadership to GA for graphs with more than 64 tasks. This led to the development of the novel two-stage hybrid algorithm EvoNet, presented in this paper, which combines the CastNet algorithm with evolutionary meta-heuristics of GA in order to obtain the greatest improvement in results.

The primary strength of the solution is that through the implementation of a variety of probabilistic, heuristic and linear optimisation methods this work has delivered a broad survey of contemporary mapping techniques. Our solution was able to successfully leverage recent optimisation heuristics alongside more established mapping strategies, and ultimately allowed deeper insights to be drawn into their effect and endurance as the problem size scales significantly. The approach of the project was highly suitable and the results structure allowed for the effective appraisal of all outlined algorithms. Our solution allowed us to ascertain the best mapping algorithms across a range of task graphs and network topologies through a level playing-field empirical analysis.

We conclude that ILP is the best used for small applications, while EvoNet is best used for applications of any size, albeit with an increase in energy consumption. Heuristic methods can be used for mapping problems where execution time is crucial and are especially important for NP-hard problems, such as the one presented in this paper. Our work was shown to produce results which were replicated within other papers and comparable to those of the current state of the art solutions. We have deduced and reinforced the findings of (Tosun 2011*b*), and have provided improvements to their CastNet algorithm in the form of EvoNet, by using its outputs to seed the population of an evolutionary stochastic search strategy. We have demonstrated that our proposed algorithm EvoNet is superior in final communication cost, and significantly reduces the execution runtime for the largest application sizes when compared to GA. For reusable architectures where the development cost can be amortised across many applications, we conclude that the optimality of EvoNet makes it the best such candidate mapping algorithm. Our findings further the state-of-art.

A suitable direction for future work would be the investigation of an improved crossover mechanism within EvoNet, as this would be a step towards reducing the execution run time particularly for the largest applications because the evolution phase is where the algorithm spends most of the execution time. An idea for this would be an adaptive crossover mechanism that maintains a greater genetic similarity between the child and parent chromosomes over time to reduce the likelihood that generated children have a lower fitness than both parents. Another path of investigation would be the efficient parallelisation of both phases of EvoNet, as this would lead to the most significant decrease in the final execution time.

References

Ascia, G., Catania, V. & Palesi, M. (2004), Multi-objective mapping for mesh-based noc architectures, *in* 'Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis', pp. 182–187.

Benini, L. & De Micheli, G. (2002), 'Networks on chips: A new soc paradigm', *Computer* **35**(1), 70–78.

Boveiri, H. R. (2018), '125 random task-graphs for multiprocessor task scheduling'.
URL: *http://dx.doi.org/10.17632/4fycv9td56.2*

Chang, K.-C. & Chen, T.-F. (2008), 'Low-power algorithm for automatic topology generation for application-specific networks on chips', *IET Computers & Digital Techniques* **2**(3), 239–249.

Dorigo, M. & Birattari, M. (2010), *Ant colony optimization*, Springer.

Duato, J. et al. (2002), 'Interconnection networks: An engineering approach, m. kaufmann pub', *Inc., USA* .

Ferrandi, F., Lanzi, P. L., Pilato, C., Sciuto, D. & Tumeo, A. (2013), Ant colony optimization for mapping, scheduling and placing in reconfigurable systems, *in* '2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)', IEEE, pp. 47–54.

Garey, M. R. & Johnson, D. S. (2002), *Computers and intractability*, Vol. 29, wh freeman New York.

Goldberg, D. E. & Holland, J. H. (1988), 'Genetic algorithms and machine learning', *Machine learning* **3**(2), 95–99.

Hu, J. & Marculescu, R. (2005), 'Communication and task scheduling of application-specific networks-on-chip', *IEE Proceedings-Computers and Digital Techniques* **152**(5), 643–651.

Janidarmian, M., Khademzadeh, A. & Tavanpour, M. (2009), 'Onyx: A new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip', *IEICE Electronics Express* **6**(1), 1–7.

Karaboga, D. & Basturk, B. (2007), 'A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm', *Journal of global optimization* **39**(3), 459–471.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* 'Proceedings of ICNN'95-International Conference on Neural Networks', Vol. 4, IEEE, pp. 1942–1948.

Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

Kumar, S., Jantsch, A., Soininen, J.-P., Forsell, M., Millberg, M., Oberg, J., Tiensyrja, K. & Hemani, A. (2002), A network on chip architecture and design methodology, *in* 'Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002', IEEE, pp. 117–124.

Lu, Z., Xia, L. & Jantsch, A. (2008), Cluster-based simulated annealing for mapping cores onto 2d mesh networks on chip, *in* '2008 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems', IEEE, pp. 1–6.

Marcon, C. A. M., Moreno, E. I., Calazans, N. L. V. & Moraes, F. G. (2008), 'Comparison of network-on-chip mapping algorithms targeting low energy consumption', *IET Computers & Digital Techniques* **2**(6), 471–482.

Moein-Darbari, F., Khademzade, A. & Gharooni-Fard, G. (2009), 'Cgmap: a new approach to network-on-chip mapping problem', *IEICE Electronics Express* **6**(1), 27–34.

Murali, S. & De Micheli, G. (2004), Bandwidth-constrained mapping of cores onto noc architectures, *in* 'Proceedings design, automation and test in Europe conference and exhibition', Vol. 2, IEEE, pp. 896–901.

Ogras, U. Y., Hu, J. & Marculescu, R. (2005), Key research problems in noc design: a holistic perspective, *in* 'Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis', ACM, pp. 69–74.

Sahu, P. K., Shah, T., Manna, K. & Chattopadhyay, S. (2013), 'Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **22**(2), 300–312.

Shen, W.-T., Chao, C.-H., Lien, Y.-K. & Wu, A.-Y. (2007), A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network, *in* 'First International Symposium on Networks-on-Chip (NOCS'07)', IEEE, pp. 317–322.

Srinivasan, K., Chatha, K. S. & Konjevod, G. (2006), 'Linear-programming-based techniques for synthesis of network-on-chip architectures', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **14**(4), 407–420.

Tosun, S. (2011*a*), 'Cluster-based application mapping method for network-on-chip', *Advances in Engineering Software* **42**(10), 868–874.

Tosun, S. (2011*b*), 'New heuristic algorithms for energy aware application mapping and routing on mesh-based nocs', *Journal of Systems Architecture* **57**(1), 69–78.

Tosun, S., Ozturk, O. & Ozen, M. (2009), An ilp formulation for application mapping onto network-on-chips, *in* '2009 International Conference on Application of Information and Communication Technologies', IEEE, pp. 1–5.

Tosun, S., Ozturk, O., Ozkan, E. & Ozen, M. (2015), 'Application mapping algorithms for mesh-based network-on-chip architectures', *The Journal of Supercomputing* **71**(3), 995–1017.