Journal of Universal Computer Science, vol. 18, no. 7 (2012), 901-916
submitted: 8/8/11, accepted: 27/3/12, appeared: 1/4/12 © J.UCS

Mapping and Scheduling in Heterogeneous NoC through Population-Based Incremental Learning

Freddy Bolanos
(Universidad Nacional de Colombia, Medellin - Colombia
fbolanosm@unal.edu.co)

Jose Edison Aedo and Fredy Rivera
(Universidad de Antioquia, Medellin - Colombia
{joseaedo, farivera}@udea.edu.co)

Nader Bagherzadeh
(University of California, Irvine - USA
nader@uci.edu)

Abstract: Network-on-Chip (NoC) is a growing and promising communication paradigm for Multiprocessor-System-On-Chip (MPSoC) design, because of its scalability and performance features. In designing such systems, mapping and scheduling are becoming critical stages, because of the increase of both size of the network and application's complexity. Some reported solutions solve each issue independently. However, a conjoint approach for solving mapping and scheduling allows to take into account both computation and communication objectives simultaneously. This paper shows a mapping and scheduling solution, which is based on a Population-Based Incremental Learning (PBIL) algorithm. The simulation results suggest that our PBIL approach is able to find optimal mapping and scheduling, in a multi-objective fashion. A 2-D heterogeneous mesh was used as target architecture for implementation, although the PBIL representation is suited to deal with more complex architectures, such as 3-D meshes.

Key Words: Multiprocessor-System-on-Chip (MPSoC) and Network-on-Chip (NoC), Population-Based Incremental Learning (PBIL), Computer-Aided Design (CAD)
Category: C.1.2, I.2.6, J.6

1 Introduction

Modern embedded systems are based on Multiprocessor-System-on-Chip (MPSoC), in order to deal with the increasing performance requirements and affordable power consumption [Ceng et al. 2008, Wolf 2004]. MPSoC integrates several processing elements or PEs in a single chip. These PEs can be different from each others, namely, heterogeneous. For connecting this set of PEs, usually a communication network is used, which is referred to as Network-on-Chip (NoC).

When designing an embedded system, mapping and scheduling are critical stages. These stages affect, sometimes simultaneously, several figures of merit that are often in conflict [Hu and Marculescu 2004, Zhang et al. 2006]. Some approaches treat with mapping and scheduling independently, for the sake of dealing with its complexity [Ghosh et al. 2009, Raina and Muthukumar 2009, Holzenspies et al. 2008]. Some other reported solutions are able to solve mapping and scheduling at once, but only for homogeneous architectures [Chen et al. 2008, Shin and Kim 2004, Xian et al. 2007, Xu et al. 2007]. In the same way, there are some reported solutions which do not take into account the communication overhead [Goraczko et al. 2008, Yang et al. 2009].

The work described in [Singh et al. 2011] performs mapping and scheduling for heterogeneous architectures. A hybrid strategy is used, in order to deal with both complexity and performance requirements. Hybrid means that part of the optimization job is done in design time, and the remaining job is done dynamically, at execution time. The computation of tradeoff points and resource throughput analysis is performed in design time, because these computations are the most computing-intensive. A run-time optimizer chooses the best tradeoff point for a given application which is going to run in the system. This approach provides dynamical behavior and adaptability for different sets of applications. The main problem with this approach is related with its inability for finding new tradeoff points, if new applications need to be mapped onto the NoC.

The work reported in [Huang et al. 2011] describes design time static task mapping for heterogeneous MPSoC systems. Computation and communication aspects are taken into account in the optimization process as well as energy-aware objectives. Part of the design is treated as an optimization problem, which is solved by using a simulated annealing algorithm. In such a kind of algorithms, a single solution is adjusted progresively in order to perform the design space search, which may lead to local-optimum issues [Bertsimas and Nohadani 2010]. By adjusting several solutions at once, population-based approaches are more robust in the optimization of complex search spaces, since it is easier to scape from local optimums [Bai and Zhao 2006, Pindoriya et al. 2010]. This parallel feature also facilitates dealing with multi-objective problems.

[Ramin et al. 2011] describes a solution based on genetic algorithms for mapping and scheduling, for heterogeneous architectures. Genetic algorithms are the most known instance of population-based techniques. The algorithm optimizes the power consumption of the final implementation, while deals with timing constraints. The target architecture is a 2-D mesh, and a wormhole routing schema has been considered for simulating the system. Instead of representing each population individual independently, as is usual in genetic algorithms, Population-Based Incremental Learning (PBIL) algorithms use a compact representation for the whole population. This feature may lead to speed up the convergence process [Pang et al. 2006].

This paper presents the results of simulating a PBIL algorithm, aimed to perform mapping and scheduling of tasks onto an NoC, which is organized as a 2-D heterogeneous mesh. An adaptive and multi-objective version of the PBIL approach was implemented. A simple X-Y routing algorithm was chosen for simulation of the potential solutions, although more elaborated routing schemes may be included. Four objectives were considered in the optimization process: Total completion time for application's tasks, total power consumption, peak bandwidth, and number of hops in the communication network.

The remaining sections of the paper are organized as follows. Section 2 defines briefly the optimization problem, concerning mapping and scheduling of executable tasks over a 2-D mesh. Section 3 describes the basic PBIL algorithm and the modifications proposed in order to make it suitable for the problem at hand. Section 4 shows the simulation results. Concluding remarks and future work appear on Section 5.

2 Task Mapping and Scheduling

Figure 1 shows an Acyclic Directed Annotated Graph (ADAG) which represents an application by means of a set of executable tasks, and their dependencies. Tasks are represented by vertices in the graph, while dependencies or links among tasks are represented by the edges. Annotations provide information about the potential implementation of each given task or dependence in the available resources, and serves for guiding the optimization process, supplying a way for comparing several solutions. Such information may include, but is not restricted to power consumption, execution time, bandwidth, and any other result of implementing tasks or links on available resources of the target architecture.

The proposed PBIL algorithm was tested by means of several synthetic problems, generated by the TGFF software tool [Valerio 2008]. These problems resemble several mapping and scheduling scenarios, i.e. applications of different sizes and complexities, several mesh sizes and availability of different kind of implementation resources. TGFF generates ADAGs in a pseudo-random way, and it is widely used for synthetic benchmarking. The ADAG depicted in Figure 1, and all the input instances used for testing the PBIL algorithm described in this paper, were generated using TGFF.

Although a conjoint approach is proposed in this paper, mapping and scheduling are going to be described independently, for the sake of clarity.

2.1 Scheduling of tasks

Since the target architecture is heterogeneous, there is a set of potential resources available for each task's implementation. The first issue to solve is choosing a specific resource for implementing each task on the system. Scheduling deals with this issue, as well as with the timing order of each task on specific PEs. It

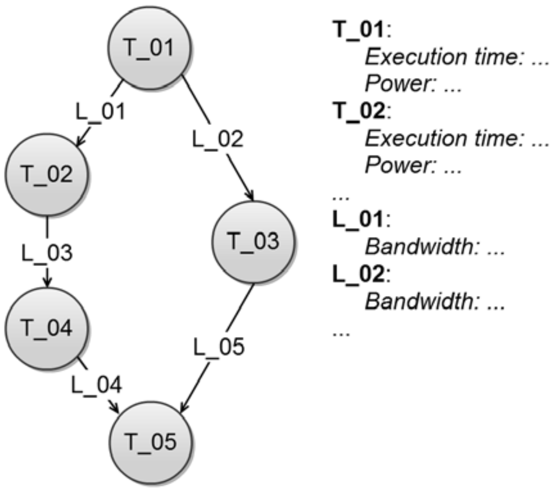

Figure 1: An ADAG generated by TGFF.

is assumed that the tasks are executed sequentially on each PE, and that the execution of a given task is not interrupted until such task finishes.

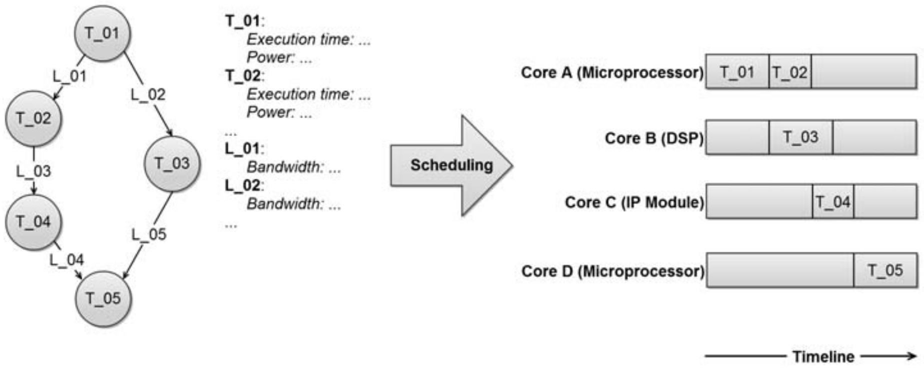

Figure 2: Scheduling example in a heterogeneous context.

Figure 2 depicts the scheduling process in a heterogeneous context, where the tasks of a given input ADAG are allocated to a set of available resources for implementation. As in Figure 1, the input application is comprised of five tasks (namely from T_01 to T_05) and there is a total of four resources available for task's implementation: two of them are generic microprocessors, and the remaining ones are a Digital Signal Processor (DSP), and an Intellectual Property (IP) block.

A complete scheduling solution must provide a binding among each task on the input ADAG and an available resource in the target architecture. In our scheduling approach, the only constraint imposed to the solutions is the number of total resources necessary for system's implementation. No prior assumption is made about the amount of resources of each kind that must be present on the solution. For instance, although in Figure 2 there can be up to four PE's, nothing precludes that all the PE's to be generic microprocessors or DSPs. Each scheduling solution affects directly the timing features for each task on the system, as well as the power consumption associated with computations.

2.2 Core Mapping

Figure 3 shows the process that follows to the scheduling step. It is referred as core mapping or simply mapping, and consists of placing each of the PEs on a specific place (tile) of a 2D mesh of finite size. The left side of Figure 3 is called a core graph, and corresponds to the previous stage's output, i.e. scheduling. The core graph depicts all PEs involved with system's implementation (each vertex in Figure 3 represents a single PE) and the dependencies among such PEs (depicted as edges in that figure). The mapping process is aimed to choose a specific tile on the mesh for each PE in the core graph.

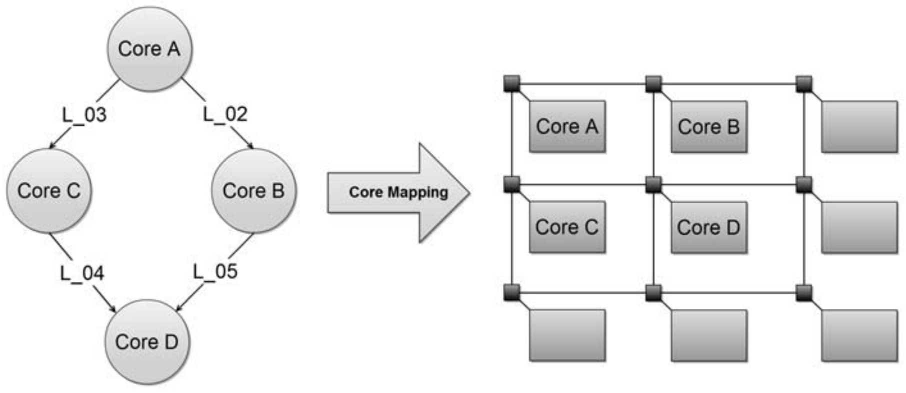

Figure 3: Core mapping on a tile mesh.

As can be seen, mapping affects power consumption and delay for the communications, as well as the bandwidth requirements for the physical links on the mesh. Core mapping has a direct impact in completion time of tasks, because of the delays induced by the network links. In the same way, mapping may affect the power consumption drastically, because as reported in [He et al. 2011], in a mesh network, the communications power may represent up to 25% of the total power consumption. For the optimization of several figures of merit, mapping and scheduling are strongly related processes. Next section describes a PBIL algorithm for performing mapping and scheduling in a conjoint fashion, taking into account several figures of merit at the same time.

3 The PBIL algorithm

PBIL algorithms are stochastic search methods that obtain directional information from the previous best solutions. Such algorithms have been used in design automation for embedded systems with promising results [Fan et al. 2007, Bolanos et al. 2010]. PBIL techniques are a special case of a larger group of optimization approaches based on population. The heart of PBIL approach is an array of probabilities which converges progressively to an optimal solution. In the case of binary problems, the PBIL array takes the form of a vector, which stores a probability value for each attribute of the problem to be optimized. In the case of non-binary problems, where there are more than two potential choices for each attribute, it is necessary to work with a probability matrix, in order to take into account all the solutions space. In both cases, the idea is to update iteratively the probability values in the array. As the PBIL algorithm converges, some values in the array become higher (i.e. some attributes become more probable), which means that the array is close to approximate an optimal solution.

Let us suppose an optimization problem that can be completely solved by answering a set of N questions or attributes. For each attribute, there can be up to M decision choices. A PBIL probability matrix for this problem is depicted in Figure 4. In such a figure, $P(i, j)$ represents the probability of attribute j of being optimized using choice i.

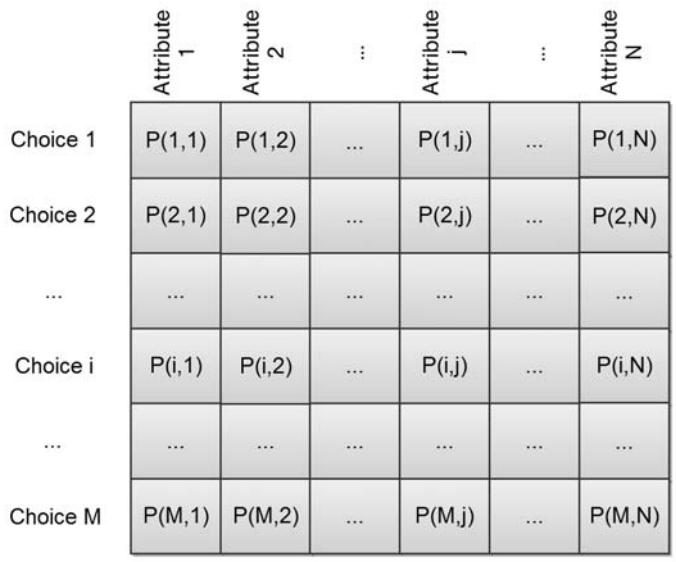

Figure 4: A PBIL probability matrix.

Algorithm 1: Basic PBIL algorithm.

```
Input: An $M \times N$ probability matrix, called $P$
Output: An optimized solution for the problem at hand
begin
    $P(i,j) = \frac{1}{M}$; $\forall\ 1 \leq i \leq M$ and $1 \leq j \leq N$;
    repeat
        $Pop = Create\_Population\,(P)$;
        $Fitness = Evaluate\_Population\,(Pop)$;
        $Best = Choose\_Best\,(Pop, Fitness)$;
        $E = Entropy\,(P)$;
        $LR = Learning\_Rule\,(E)$;
        $P = Update\_Array\,(P, Best, LR)$;
    until $(E > Tolerance)$;
    return $Best$;
end
```

A basic version of the adaptive PBIL approach is shown in Algorithm 1. Algorithm 1 starts with a probability array, namely P, with dimensions M by N as shown in Figure 4. The probabilities on the array are initialized to $1/M$, which is the value that ensures maximum population diversity, in such a way that at first, all potential solutions to the problem are being considered in a equitative way.

At each algorithm's iteration, a new population (*Pop*) is generated, based on the probabilities of the array, by means of the *Create_Population* routine. Those attributes with highest probability values will be more frequently present on the populations individuals. All the individuals of the population just created, are meant to be assessed, using the *Evaluate_Population* function. The fitness values allow to choose the best solution for the optimization problem at hand. The *Choose_Best* routine is used to this end.

The learning rate parameter or *LR*, is a way to control the convergence speed of the PBIL algorithm. Higher values of *LR* will lead to fast convergences, although the quality of the solutions might not be satisfactory. If *LR* is reduced, quality will improve at the expenses of longer convergence time.

In our adaptive approach, the LR parameter must be adjusted dynamically in order to allow both exploration and exploitation of the PBIL search space. The entropy (E) of the probability array is calculated and used as an estimation of the population's diversity. In Algorithm 1, routine *Learning_Rule* represents the way in which the *LR* parameter is tuned as a function of the P array's entropy. Once the *LR* parameter is calculated, the P array must be updated in order to adjust the probabilities, according to the best solutions found in the population. Function *Update_Array* is used with this aim.

The value of the E parameter in Algorithm 1 is calculated as the systemic entropy of the PBIL array, just as is done in information theory. Equation (1) depicts the calculations performed inside the *Entropy* routine, aimed to entropy calculation.

$$E = -\frac{1}{N} \times \sum_{i=1}^{M} \sum_{j=1}^{N} P_{(i,j)} \times Log_M \left(P_{(i,j)} \right) \qquad (1)$$

According to Equation (1) entropy values are going to be in the range from 0 to 1. $E = 1$ means that there is maximum population's diversity (this only happens when all values on the PBIL matrix are equal to $1/M$). When $E = 0$, it means that the PBIL matrix points to a unique and completely defined solution.

Since the Entropy's magnitude decreases as the algorithm converges to a given solution (i.e. the values of probability tend to be concentrated on unique entries of each column of the P array), an usual termination condition for the PBIL algorithm is to compare the value of E with a given tolerance, close to zero. Testing if the entropy value is equal to zero is a restrictive and time-consuming condition for the PBIL algorithm's termination.

The way in which the *LR* parameter is changed as a function of entropy is often referred as the learning rule. Figure 5 shows a linear learning rule, which

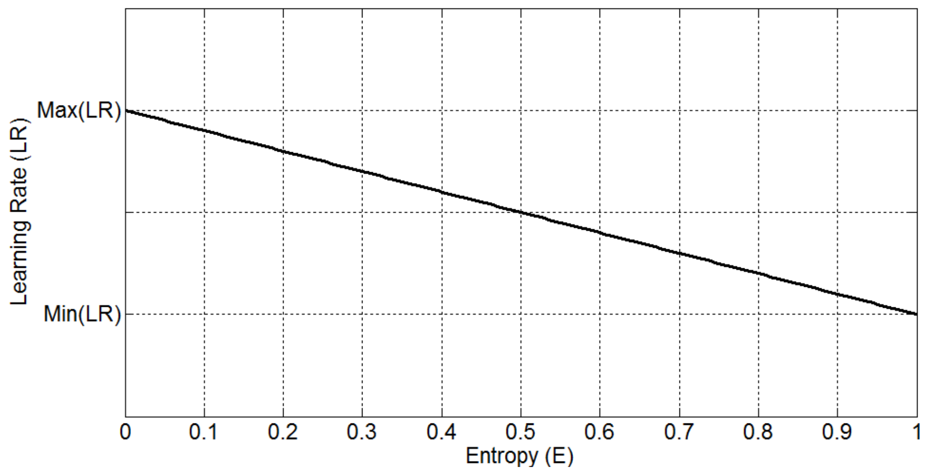

Figure 5: Linear Learning rule.

was used inside the *Learning_Rule* routine. The idea is to keep the LR parameter low at the beginning of the algorithm, when there is a higher population's diversity and the values of E are close to one. When entropy's value decreases, i.e. when the population approaches to a given optimal, LR parameter is increased to speed up the convergence process.

For each attribute of the optimization problem or, equivalently, for each column in the PBIL array, the function *Update_Array* must increase the probability associated with the attribute which resulted to be the best solution. Since each single column in the PBIL matrix represents a conjoint probability event, the probabilities sum along such a column must be equal to one. Therefore, when a given probability in the array is increased, the remaining ones in that column must be decreased accordingly. Equation (2) shows the probability's updating formulae, which are based on the Hebbian learning rule [White 1991]. In Equation (2), it is supposed that for a given attribute j, the best solution obtained is the choice k.

$$P_{(i,j)NEW} = \begin{cases} P_{(i,j)OLD} + \left(1 - P_{(i,j)OLD}\right) \times LR & \text{if } i = k \\ \left(1 - P_{(k,j)NEW}\right) \times \dfrac{P_{(i,j)OLD}}{1 - P_{(i,j)OLD}} & \text{if } i \neq k \end{cases} \tag{2}$$

The PBIL matrix depicted in Figure 4 is not suitable for representing the issues related to mapping and scheduling, as described in the prior section. Figure 6 shows a proposed PBIL representation for the mapping and scheduling combined optimization problem. In this case, two probability matrixes are required, although these arrays do not represent the mapping and scheduling processess directly. The first matrix, with dimensions M by N, defines which type of resource or PE, is going to be placed on each of the available tile spaces. That means that there could be up to M different kinds of PEs and that there are N places in the network for PE allocation. The second matrix, with dimensions N by T, defines where the tasks of the initial specification will be executed. The number T defines the total number of tasks on the system.

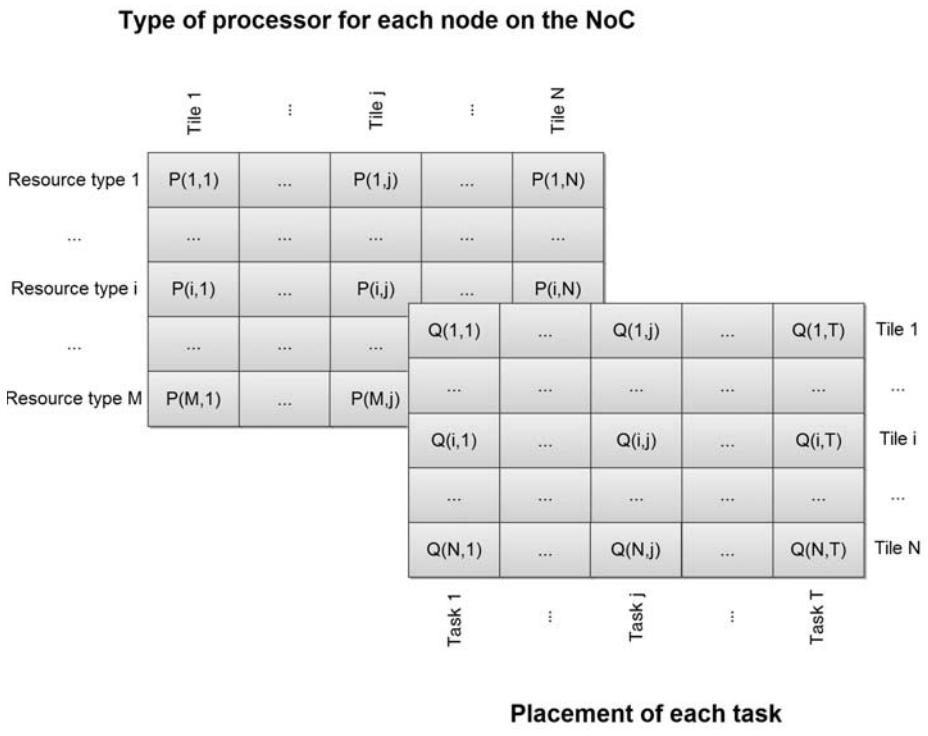

Figure 6: Proposed PBIL matrix representation.

The two sets of probabilities in Figure 6, namely, P and Q, may be updated accordingly with the optimization process described in Algorithm 1, with very slight modifications. The initialization of these probabilities must be performed according to each array's dimensions, as explained before. A complete solution for the mapping and scheduling problem may be derived by using the probability values of each column on both arrays. The total entropy can be calculated as the mean of the entropies for both matrixes.

Regarding the multi-objective implementation of the algorithm, an aggregation approach was used to join all the algorithm's objectives on a unique representative value. By means of a weighting vector in the aggregation calculation, the designer may give more relative importance to some optimization objectives, above the remaining ones. In order to obtain several solutions with different tradeoffs among the optimization objectives, Algorithm 1 is executed several times in a sequential way. A kernel approach based on distance [Silverman 1986] was used for avoiding that different executions of the algorithm converge to the same optimal results.

4 Simulation Results

Figure 7 shows the evolution of the four objectives considered as a function of the number of iterations of the optimization process, as described in Algorithm 1. The mapping and scheduling shown in Figure 7, were performed with an input task graph with thirty tasks, over a target 2-D mesh with a size of 5 by 5 tiles. The learning rate (LR parameter) was settled to change between 0.15 and 0.4. In Figure 7, the completion time, which is given in seconds, is calculated as the sum of the individual completion time for each task on the system. The power consumption is given in watts. The peak bandwidth values are normalized with respect to a reference value of 100 MHz.

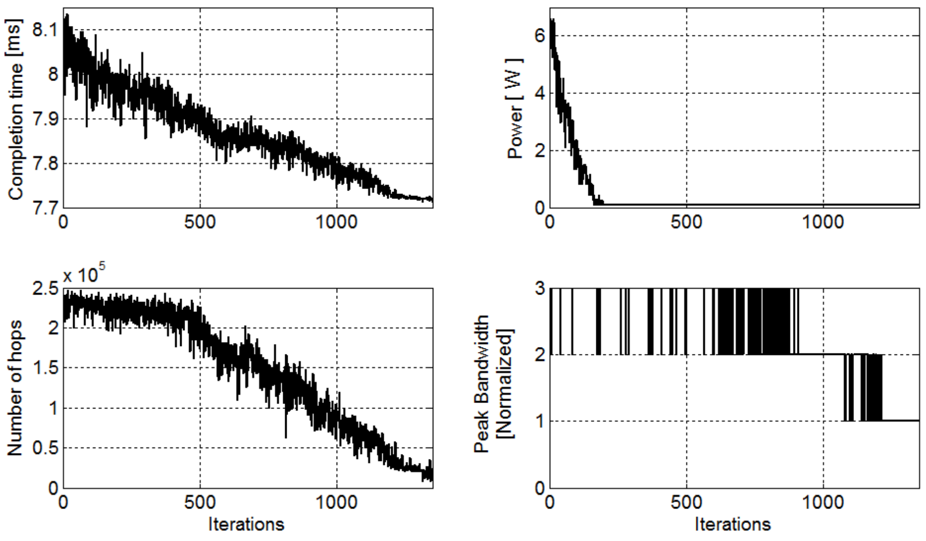

Figure 7: Evolution of the four objectives considered.

Figure 7 shows how the four objectives converge progresively toward their minimum values, as the algorithm iterates. The evolution of the bandwidth objective exhibits step changes, as integer numbers, because such objective is calculated with respect to a reference (base) value. The knees exhibited by some of the graphics (such as the one shown in the Completion Time objective, around the iteration 700), are a consequence of the algorithm climbing out of a local minima and falling back to lower values.

Some previous works suggest that the performance of the PBIL convergence is a predictable function of the size of the optimization problem [Bolanos et al. 2010].

Figure 8 shows the convergence time for optimization problems of several sizes. In Figure 8, the X axis represents the size of the problem, in terms of the number of tasks of the input specification, with the remaining conditions as described for the Figure 7.

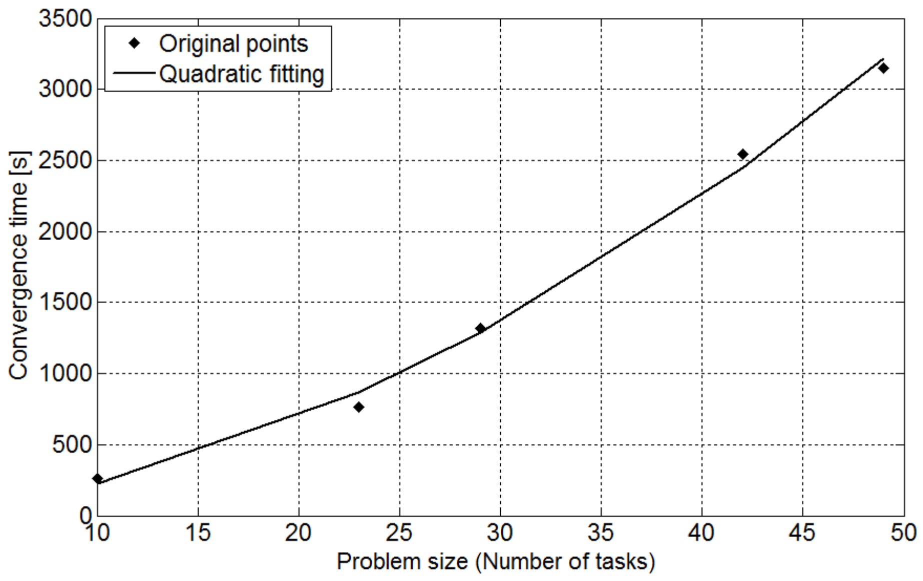

Figure 8: Relationship between convergence time and problem's size.

Figure 8 shows that convergence times have a behavior very close the quadratic fitting function. This might be explained by inspecting Figure 6. Such figure shows that the size of the problem (number of tasks in the application) is one of the dimensions of the PBIL array, which is the representation that such algorithm uses for the population of solutions. If one of the dimensions is increased, it is logical to expect that the number of operations that the algorithm must perform on the whole array, shall be increased accordingly.

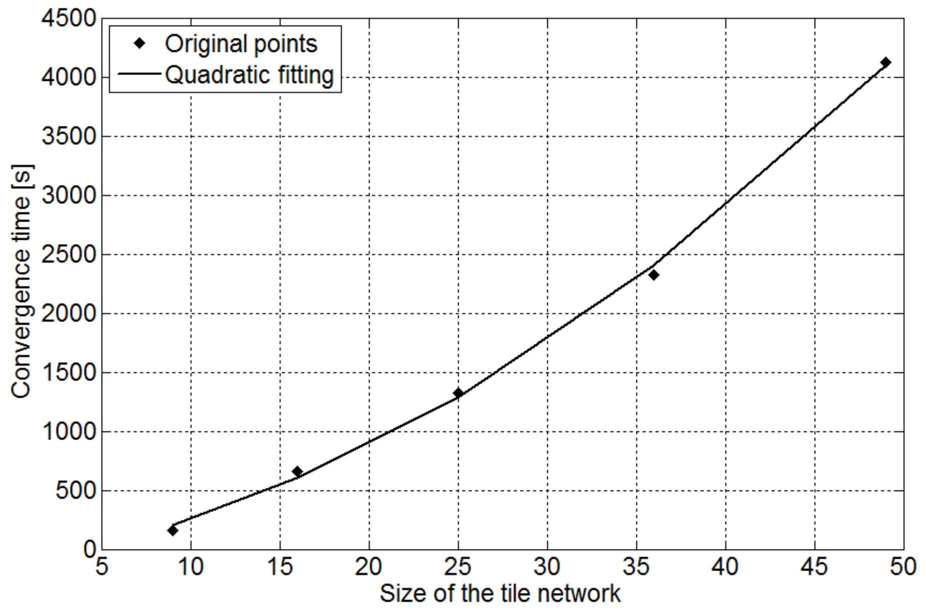

Figure 9: Relationship between convergence time and network's size.

In order to further inspect this idea, Figure 9 shows the dependence between the convergence time and the size of the tile network. Several instances of the algorithm were executed with the amount of tasks fixed and equal to thirty. The tile network size were changed from 9 (3 by 3 mesh) to 49 (7 by 7 mesh), letting the rest of parameters unaltered. Again, there seems to be a quadratic relationship between these two values, since the tile network size is also one of the dimensions of the PBIL matrix representation. Similar tests were conducted regarding the dependence of convergence time with respect to the number of resource types. The quadratic behaviour appeared again.

5 Conclusions and Future Work

A mapping and scheduling algorithm based on PBIL has been developed and simulated, with promising results. The main features of this algorithm are:

– The proposed PBIL representation, as depicted in Figure 6, does not rely in a specific network topology, despite a mesh network was used for simulations in this paper. Any other topology, such as a 3-D network, might be used without drastic changes in the population's representation, because such representation deals only with the size of the network (number of tile spaces).

- Similarly, the proposed technique is able to perform mapping and scheduling by including more elaborated routing schemes. The only limitation is the ability to simulate the routing conditions in a reasonable time, because several potential solutions must be assessed before the algorithm reaches an optimal solution, as depicted in Figure 7.
- The proposed approach does not require excessive formalization. The PBIL algorithm exhibits less complexity and more flexibility, when compared with some other reported approaches, such as ILP [Derin et al 2011]. Figures 8 and 9 depict a quadratic behavior of the convergence time with respect to the problem size, in contrast to the exponential behavior of the ILP algorithms, in which some speedup strategies must be introduced, in order to make it feasible for practical applications [He et al. 2011].
- The results suggest that, the proposed PBIL technique may be used online (i.e. at execution time). This would allow to adjust the mapping and scheduling schemas to changing conditions, such as failures in the resources of the network.

Acknowledgements

The authors would like to thank to ARTICA, to COLCIENCIAS, to ICT Ministry of Colombia, to National University of Colombia and to University of Antioquia, for their support in the development of this work.

References

[Bai and Zhao 2006] H. Bai and B. Zhao, "A survey on application of swarm intelligence computation to electric power system," in Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, vol. 2, pp. 7587 7591, 2006.

[Bertsimas and Nohadani 2010] D. Bertsimas and O. Nohadani. "Robust optimization with simulated annealing". J. of Global Optimization 48, 2 (October 2010), 323-334. 2010.

[Bolanos et al. 2010] F. Bolanos, J. Aedo, and F. Rivera, "System-level partitioning for embedded systems design using population-based incremental learning," in CDES (H. R. Arabnia and A. M. G. Solo, eds.), pp. 7480, CSREA Press, 2010.

[Ceng et al. 2008] J. Ceng, J. Castrillon, W. Sheng, H. Scharwachter, R. Leupers, G. Ascheid, H. Meyr, T. Isshiki, and H. Kunieda, "MAPS: an integrated framework for MPSoC application parallelization" in DAC, 2008.

[Chen et al. 2008] G. Chen, F. Li, S. W. Son, and M. T. Kandemir, "Application mapping for chip multiprocessors," in DAC, 2008.

[Derin et al 2011] O. Derin, D. Kabakci, L. Fiorin, "Online task remapping strategies for fault-tolerant Network-on-Chip multiprocessors," Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on, pp. 129-136, 1-4 May 2011.

[Fan et al. 2007] L. J. Fan, B. Li, Z. Q. Zhuang, and Z. Q. Fu, "An approach for dynamic hardware/software partitioning based on DPBIL," in Natural Computation, 2007. ICNC 2007. Third International Conference on, vol. 5, pp. 581-585, 2007.

[Ghosh et al. 2009] P. Ghosh, A. Sen, and A. Hall, "Energy efficient application mapping to noc processing elements operating at multiple voltage levels," in NOCS, 2009.

[Goraczko et al. 2008] M. Goraczko, J. Liu, D. Lymberopoulos, S. Matic, B. Priyantha, and F. Zhao, "Energy-optimal software partitioning in heterogeneous multiprocessor embedded systems," in DAC, 2008.

[He et al. 2011] O. He, S. Dong,W. Jang, J. Bian, D. Z. Pan, "UNISM: Unified Scheduling and Mapping for General Networks on Chip," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , no. 99, pp.1-14, 2011.

[Holzenspies et al. 2008] P. K. F. Holzenspies, J. Hurink, J. Kuper, and G. J. M. Smit, "Run-time spatial mapping of streaming applications to a heterogeneous multiprocessor system-on-chip (MPSOC)," in DATE, 2008.

[Hu and Marculescu 2004] J. Hu, R. Marculescu, "Energy-Aware Communication and Task Scheduling for Network-on-Chip Architectures under Real-Time Constraints, in Proc. Design," in Proc. Automation and Test in Europe Conf., Paris, France, Feb. 2004.

[Huang et al. 2011] J. Huang, C. Buckl, A. Raabe, A. Knoll, "Energy-Aware Task Allocation for Network-on-Chip Based Heterogeneous Multiprocessor Systems," Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on , pp.447-454, 9-11 Feb. 2011.

[Pang et al. 2006] H. Pang, K. Hu. Z. Hong, "Adaptive PBIL algorithm and its application to solve scheduling problems," Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE , pp.784-789, 4-6 Oct. 2006

[Pindoriya et al. 2010] N. Pindoriya, S. Singh, and K. Lee, A comprehensive survey on multi-objective evolutionary optimization in power system applications, in Power and Energy Society General Meeting, 2010 IEEE, pp. 18, 2010.

[Raina and Muthukumar 2009] A. Raina, V. Muthukumar, Traffic Aware Scheduling Algorithm for Network on Chip, in Proc. of the 6th International Conference on Information Technology: New Generation, 27-29 Apr 2009, Las Vegas, USA. pp. 877-882.

[Ramin et al. 2011] R. Ramin, H. Shaahin, V. B. Vosoughi, "An energy-aware methodology for mapping and scheduling of concurrent applications in MPSoC architectures," Electrical Engineering (ICEE), 2011 19th Iranian Conference on , pp.1-6, 17-19 May 2011.

[Shin and Kim 2004] D. Shin and J. Kim, "Power-aware communication optimization for networks-on-chips with voltage scalable links," in CODES+ISSS, 2004.

[Silverman 1986] B. W. Silverman, "Density estimation: for statistics and data analysis". 1986.

[Singh et al. 2011] A. K. Singh, A. Kumar, and T. Srikanthan. "A hybrid strategy for mapping multiple throughput-constrained applications on MPSoCs". In Proceedings of the 14th international conference on Compilers, architectures and synthesis for embedded systems (CASES '11). ACM, New York, NY, USA, 175-184. 2011.

[Valerio 2008] K. Valerio, "Task graphs for free (tgff v3.0)," Available at `ziyang.eecs.umich.edu/dickrp/tgff/manual.pdf`, April 2008, last time revisited: 09/02/2011.

[White 1991] R. White, "Competitive hebbian learning," in Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on, vol. ii, p. 949 vol.2, July 1991.

[Wolf 2004] W. Wolf,"The future of multiprocessor systems-on-chips." in DAC, 2004.

[Xian et al. 2007] C. Xian, Y.-H. Lu, and Z. Li, "Energy-aware scheduling for real-time multiprocessor systems with uncertain task execution time," in DAC, 2007.

[Xu et al. 2007] R. Xu, R. G. Melhem, and D. Mosse, "Energy-aware scheduling for streaming applications on chip multiprocessors," in RTSS, 2007.

[Yang et al. 2009] C.-Y. Yang, J.-J. Chen, T.-W. Kuo, and L. Thiele, "An approximation scheme for energy-efficient scheduling of real-time tasks in heterogeneous multiprocessor systems," in DATE, 2009.

[Zhang et al. 2006] Q. Zhang, A. B. J. Kokkeler, G. J. M. Smit, "A Reconfigurable Radio Architecture for Cognitive Radio in Emergency Networks," in Porc. European Conference on Wireless Technology, 10-15 September 2006, Manchester, UK. pp. 35-38.