Hardware Software Partitioning Problem in Embedded System Design Using Particle Swarm Optimization Algorithm

Alakananda Bhattacharya[1], Amit Konar[1], Swagatam Das[1], Crina Grosan[2] and Ajith Abraham[2]
Department. of Electronics and Telecommunication Engg, Jadavpur University, Kolkata, India[1]
b_alaka2@hotmail.com, konaramit@yahoo.co.in,, swagatamdas19@yahoo.co.in
Faculty of Mathematics and Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania
cgrosan@cs.ubbcluj.ro
Center of Excellence for Quantifiable Quality of Service[2]
Norwegian University of Science and Technology, Norway
ajith.abraham@ieee.org

Abstract

Hardware/software partitioning is a crucial problem in embedded system design. In this paper, we provide an alternative approach to solve this problem using Particle Swarm Optimization (PSO) algorithm. Performance analysis of the proposed scheme with Integer Linear Programming, Genetic Algorithm and Ant Colony Optimization technique has been compared using standard benchmark datasets, and the computer simulations reveal that the proposed approach outperforms all the meta-heuristic based existing techniques with respect to cumulative runtimes for several runs of the same program. The Integer Linear Programming has been found to yield the optimal solutions, and the proposed swarm scheme yields sub-optimal solution, sufficiently close to the reported results obtained for integer programming.

Keywords: Genetic Algorithm, Hardware/Software Partitioning, Integer Linear Programming, Particle Swarm Optimization

1. Introduction

An embedded system is a computing system rather than desktop computers/laptops/palmtops, capable of reacting spontaneously with sensory inputs in real time and designed for dedicated applications. Typical applications that employ embedded systems include fax machines, copiers, printers, scanners, cash registers, alarm systems, card readers, mobile phones, digital cameras, washing machines, DVD players, speech recognizers and many more. Like typical computer systems, embedded systems too include hardware and software components. In other words, it is common to use both application-specific hardware accelerator circuits and general-purpose programmable units with appropriate software for embedded system design. Usually application specific hardware is much faster than software and also more power efficient, but expensive at the some time. Software, on the other hand, is cheaper, but slow and consumes much power when implemented on a general purpose processor. Hence for faster realization or power-critical situations, hardware based systems are preferred, whereas non-critical modules of embedded systems are realized in software. Consequently a trade-off between cost, power and performance needs to be devised to realize an embedded system on a mixed hardware/software platform.

Among the most crucial steps in embedded system design, partitioning, that is, deciding which components/modules of the system should be realized on hardware and which ones in software is a fundamental problem. This is referred to hardware/software partitioning problem in embedded systems literature [1]. Traditionally hardware/software partitioning was accomplished manually. However, with the increased complexity in embedded systems, researchers currently prefer an automatic approach to handle this problem.

Classically there exist two approaches, exact and heuristic, to handle the hardware/software partitioning problem. The exact algorithms include branch and bound [2], dynamic programming [3], [4] and integer linear programming [1], [5], [6]. Most of the partitioning algorithms in the existing literature,

0-7695-3109-1/08 $25.00 © 2008 IEEE
DOI 10.1109/CISIS.2008.5

however, are heuristic. This is due to the fact that partitioning is an NP hard problem, and therefore exact solutions tend to be quite slow for bigger dimensions of the problem [2], [7]. Among the well known heuristic based algorithms, Genetic Algorithm (GA) [8-11], simulated annealing [12-14], tabu search algorithm [12], greedy algorithms [15], [16] and Ant Colony Optimization (ACO) [17] are most common.

Besides the heuristic algorithms referred to above, sometimes family of heuristics such as hierarchical clustering [7], [18], [19], [20]. Kernighan-Lin heuristics [21] are equally useful for application in partitioning problem. Scheduling best algorithms, which could be the third variety of partitioning algorithms, have also been used in the recent literature [6], [9], [15], [22], [23].

In this paper, we propose a meta-heuristic algorithm, which was originated from the sociological behavior of the intelligent creatures such as eagles, bees, ants. These creatures compete for food and thereby follow a dynamics to obtain optimal food from a given pool of resource food-grains. Such algorithms based on the collective behavior of these creatures are referred to as *swarm intelligence algorithm*. Particle Swarm Optimization (PSO) is one such well known swarm intelligence algorithm. In PSO, we define particles to represent agents to search optima of a given non-linear and rough search landscape. Classical optimization techniques that employ derivatives to find optima cannot be used in many engineering problems because of several discontinuities of the search surface. PSO is a possible scheme to determine optima for such engineering optimization problems. Hardware/software partitioning problem can be formulated as an optimization problem, and multi-agent search such as PSO can be invoked to find optimal solution to the partitioning problem. Here each particle position denotes a trial solution to the problem. Trial solutions are iterated using the steps of the PSO algorithm to determine better candidate solution and the process is repeated until no further improvement in solution is detected. The position of the best particles at this stage is considered as the final solution to the problem.

In this paper, we compare PSO with ACO, GA and other heuristic/meta-heuristic algorithms using standard benchmarks available in the literature. The paper has been divided into six major Sections. Section 2 offers a formal definition to the problem. In Section 3 we briefly outline the main steps of the PSO algorithm, and also illustrate the scope of PSO in hardware/software partitioning problem. Computer simulation and comparison with other heuristic/meta-heuristic algorithm are presented in Section 4 and conclusions are listed in Section 5.

2. Formal Definitions

Arato *et al.* [8] formalized the hardware/software partitioning problem by an undirected graph G = (V, E), where V denotes the set of vertices and E denotes the set of edges. The vertices refer to tasks, and edges refer to communication between the selected pair of vertices. They took an attempt to partition the set of vertices V into V_H and V_S, where V_H denotes the tasks to be realized on a hardware and V_S denotes the tasks to be realized on software. Obviously $V_H \cap V_S = \varnothing$ and $V_H \cup V_S = V$. In this paper, we, however formalize the hardware/software partitioning problem by using a task graph following Marwedel [24].

A task graph is a directed graph consisting of nodes/vertices V and edges E. Thus the task graph T = <V, E> where V = {V_1, V_2, ……, Vn} denotes the set of tasks and E = {e_{ij}} for i, j = 1 to n denotes the set of edges from vertex i to vertex j. The edges here refer to dependencies. In other words, the existence of a directed edge e_{ij} indicates that task representing node i has to be executed prior to execution of the task denoted by node j.

3. Hardware/Software Partitioning by Particle Swarm Optimization Algorithm

In this Section we first briefly outline the particle swarm optimization (PSO) algorithm, and then demonstrate the scope of the algorithm in the partitioning problem.

3.1 Classical PSO Algorithm

Motivated by the behavioral and sociological characteristics of bees and flies, Eberhartt and Kenedy [25] proposed the PSO algorithm. It has been observed that bees usually identify their food by a collective effort. The dynamics of a bee to move towards the target position (location of food resources) depends on three factors: i) the current direction of its motion, ii) the global best position identified by all its fellow bees until this time, and iii) the local best position that the bee has experienced so far.

Let $x_{i\,(t)}$ be the current position of the ith particle at time t, $v_{i\,(t)}$ be the velocity of the ith bee at time t, $p_i^{\,l}$ (t) be the local best position experienced by the ith bee until time t, p^{g} (t) be the global best position of all the bees at time t, the dynamics of the ith bee then can be described by the following two equations:

$$v_i(t) = w\, v_i(t-1) + \alpha_t^{l}\,(p_i^{l}(t) - x_i(t)) + \alpha_t^{g}\,(p^{g}(t) - x_i(t)) \quad (1)$$

$$x_i(t) = x_i(t-1) + v_i(t) \quad (2)$$

where w, α_t^{l} and α_t^{g} denote the inertial velocity, local acceleration coefficient (LAC) and global acceleration coefficient (GAC). The second equation apparently seems to have unmatched dimensions on the two sides of the equality. The confusion regarding dimension can be resolved by considering the following interpretation:

$$x_i(t) = x_i(t-1) + (t - t - 1)\,\overline{v_i}\,(t) \quad (3)$$

The coefficient of the velocity term in the last equation being one is implicitly mentioned by the original equation (1).

The PSO algorithm has been used for solving optimization, search and machine learning problems. In this section, we would like to illustrate the scope of PSO in optimization problems. Classical optimization problems usually require computing partial/total derivatives of the given objective function to be optimized. Unfortunately, in many engineering and scientific optimization problems, the surface of the non linear objective function being discontinuous at several points, derivative based optimization techniques can no longer be employed for such problems. PSO is one such derivative free optimization technique, where given the objective function, we can determine the optima by executing the PSO algorithm. For convenience let us consider a simple two dimensional surface $z = x_1^2 + x_2^2$, which has the minima at the origin; (0, 0). To make the PSO amenable for such optimization problem, we define

$$f = x_1^2 + x_2^2$$

as the fitness function, which will measure the fitness of a *particle* in the PSO algorithm.

The ants/bees/swaps in PSO algorithm are modeled by *particles*. Suppose a number of bees are left on a surface $z = x_1^2 + x_2^2$, where they have a food resource at the origin, the minima of the surface. Here, each bee (or the particle) has two dimensions x_1 and x_2 and they can measure their height by evaluating $f = x_1^2 + x_2^2$. Each particle in his trial motions over iterations remembers its local best position so far attained, and reports the local best position to a black board manager who determines the minimum of the local best positions attained by all the particles at the end of a iteration. The particles change their position following the basic PSO dynamics presented in equations (1) and (2). The basic PSO algorithm is outlined below.

PSO Algorithm

Input: Initial position $x_i(0)$ and velocity $v_i(0)$ for each particle i, the fitness function f (•);

Output: The global best position attained by the best particle;

Step 1: For each particle i,

Evaluate $f(x_i)$;

If $f_i(x_i) < p_i^{l}(t)$,

$p_i^{l}(t) \leftarrow n_i(t)$,

end for;

$p^{g}(t) \leftarrow \text{Min}\,(p_i^{l}(t))$ $\forall i$

Step 2: Evaluate the particle's next position by executing the basic PSO equations (1) and (2) in order.

Step 3: Repeat steps 1 and 2 until convergence occurs by exhibiting $|x_i(t) - x_i(t-1)| < \delta$, $\forall i$, where δ is a pre-assigned small positive number.

3.2. PSO in Hardware/Software Partitioning Problem

The PSO can be employed for hardware/software partitioning problem. Here, we consider each particle to be *n* dimensional, where *n* denotes the number of tasks on the given task graph.

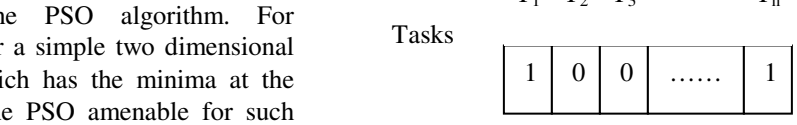

Figure 1: Representation of a particle by a n-dimensional string

Figure 1 provides one way of representing a particle by a n-dimensional binary string, where a "1" and a "0" respectively denote hardware and software representation of the task. Each particle thus attempts to determine the optimal solution for the hardware/software partitioning problem. The fitness function of the ith particle is defined by

$$F_i = \sum_{j=1}^{N} A_j\, T_j, \quad (4)$$

where A_j denotes the area required for the VLSI implementation of the task and T_j denotes the execution time on that platform. When the task j is realized on software, A_j is considered unity and T_j denotes the execution time of the task on a given software platform.

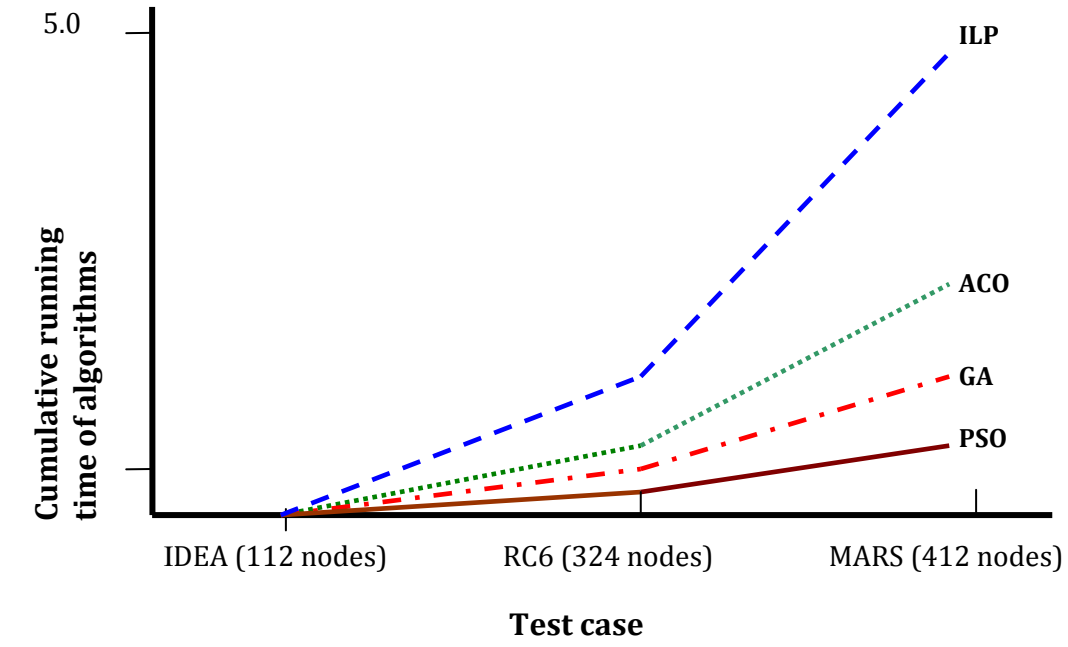

Figure 2. Results of computer simulations for cumulative running time of the algorithms on benchmark problems.

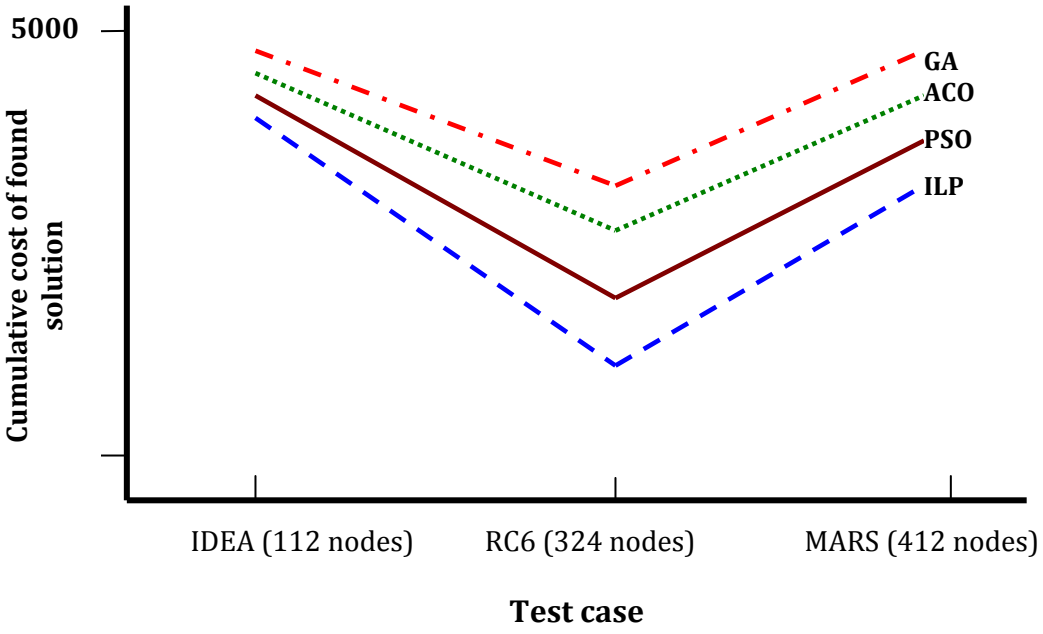

Figure 3. Results of computer simulation for cumulative cost of found solution of different algorithms on different benchmark problems.

The particles thus traverse in a N-dimensional space of tasks, and attempt to determine the optimal solution by minimizing the fitness function. The local best position and the global best position here refers to the best position of a particle with respect to smaller fitness function locally and globally respectively until the current iteration. The solution for the classical hardware/software partitioning problem is obtained by identifying the position of the best particle that has the smallest fitness function. Each dimension of the ith particle in the present case being binary (1 for hardware and 0 for software realization), the solution for the best particle is a binary string describing the hardware/software realization of all the n tasks.

4. Experimental Results

A computer simulation of the PSO, ACO, ILP and GA was performed for the proposed hardware/software partitioning problem using three standard benchmarks, such as IDEA, RC6 and MARS, and the results of the cumulative runtimes and cost of the solutions found by the algorithms are illustrated in Figures 2 and 3 respectively. It is known that ILP always find the optimum [26], so we were interested to examine the performance of the meta-heuristic algorithms: GA, PSO and ACO that can give rise to a near-optimal solution, acceptable to embedded system designers. This is due to the fear that ILP is too costly to realize for its massive computations. An examination of Figure 2 reveals that PSO outperforms all meta-heuristic algorithms from the point of views of cumulative runtimes.

Cumulative cost of found solutions in each benchmark is given in Figure 3. Figure 3 also reveals that the cumulative cost, which gives a measure of the quality of solution, is found to be the best for ILP and next for PSO. So, the role of PSO as a meta-heuristic to the partitioning problem becomes evident from the results of computer simulations.

5. Conclusions

In this paper, we introduced a new simplified model for hardware/software partitioning problem. Although it is NP hard in general, we could find an efficient approach to solve the problem using Swarm Intelligence techniques. We compare the performance of the proposed algorithms with already reported results on ILP [8], GA [8] and ACO [27]. The empirical test revealed that ILP-based solution works most efficiently for graphs with as many as few thousand nodes and yields optimal solutions, whereas the PSO gives near optimal solution on an average. It is further observed that the PSO based algorithm outperforms GA, ACO and ILP with respect to runtime requirements for the given partitioning problem.

6. References

[1] Z. A. Mann and A. Orban, "Optimization problems in system-level synthesis," in *Proc.3rd Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, 2003.

[2] N. N. Binh, M. Imai, A. Shiomi, and N Hkichi, "A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate counts," in *Proc.33rd Design Automation Conference*, 1996.

[3] J. Madsen, J. Grode, P. V. Knudsen, M. E. Petersen, and A. Haxthausen, *LYCOS: The Lyngby co-synthesis system. Des. Automat. Embedd. Syst.* 2, 2, 195-236, 1997.

[4] M. O'Nils, A. Jantsch, A. Hemani, and H. Tenhunen, "Interactive hardware-software partitioning and memory allocation based on data transfer profiling," in *Proc. International conference on Recent Advances in Mechatronics,"* 1995.

[5] R. Niemann, *Hardware/Software Co-Design for Data Flow Dominated Embedded Systems,"* Kluwer Academic Publishers, Norwell, MA, 1998.

[6] R. Niemann and P. Marwedel, An algorithm for hardware/software partitioning using mixed integer linear programming, *Des. Automat. Embedd. Syst*, Special issue: Partitioning Methods for Embedded Systems 2, pp. 165-193, 1997.

[7] F. Vahid and D. Gajski, "Clustering for improved system-level functional partitioning," in *Proc.* 8th International Symposium on System Synthesis, 1995.

[8] P. Arato, S. Zuhasz, Z. A. Mann, A. Orban, and D. Papp, "Hardware/software partitioning in embedded system design," in *Proc. of the IEEE International Symposium on Intelligent Signal Processing*, 2003.

[9] R. P. Dick and N. K. Jha, "MOGAC: A multiobjective genetic algorithm for hardware-software co-synthesis of hierarchical heterogeneous distributed embedded systems," *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 17*, 10, pp. 920-935, 1998.

[10] G. Quan, X. Hu, and G. Greenwood, "Preference-driven hierarchical hardware/software

partitioning," in *Proc. of the IEEE/ACM International Conference on Computer Design*, 1999.

[11] V. Srinivasan, S. Radhakrishnan, and R. Vemuri, "Hardware software partitioning with integrated hardware design space exploration," in *Proc. of DATE*, 1998.

[12] P. Eles, Z. Peng, K. Kuchcinski, andA. Doboli, System level hardware/software partitioning based on simulated annealing and tabu search, *Des. Automat. Embedd. Syst. 2,* 1 (Jan.), pp. 5-32, 1997.

[13] R. Ernst, J. Henkel, and T. Brenner, Hardware/software cosynthesis for microcontrollers, *IEEE Des. Test Comp. 10*, 4, pp. 64-75, 1993.

[14] M. Lopez-Vallejo, J. Grazal, and J. C. Lopez, "Constraint–driven system partitioning," in *Proc. of DATE*, pp. 411-416, 2000.

[15] K. S. Chatha and R. Vemuri, "MAGELLAN: Multiway hardware-software partitioning and scheduling for latency minimization of hierarchical control-dataflow task graphs," in *Proc. of CODES 01*, 2001.

[16] J. Grode, P. V. Knudsen, and J. Madsen, "Hardware resource allocation for hardware/software partitioningin the LYCOS system," in *Proc. of Design Automation and Test in Europe (DATE '98)*, 1998.

[17] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MA, MIT Press, 2004.

[18] T. F. Abdelzaher and K. G. Shin, "Period-based load partitioning and assignment for large real-time applications," in *IEEE Trans. Comput. 49,* 1, pp. 81-87, 2000..

[19] E. Barros, W. Rosenstiel, and X. Xiong, "Hardware/software partitioning with UNITY," in *Proc. 2nd International Workshop on Hardware-Software Codesign*, 1993.

[20] F. Vahid, "Partitioning sequential programs for CAD using a three-step approach," *ACM Trans. Des. Automat. Electron. Syst. 7*, 3 (July), pp. 413-429, 2002.

[21] B. W. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, *Bell Syst. Techn. J. 49*, 2, pp. 291-307, 1970.

[22] A. Kalavade and E. A. Lee, The extended partitioning problem: Hardware/software mapping, scheduling and implementation-bin selection," *Des. Automat. Embedd. Syst. 2*, 2, pp. 125-164, 1997.

[23] M. Lopez-Vallezo and J. C. Lopez, On the hardware-software partitioning problem: System modeling and partitioning techniques, *ACM Trans. Des. Automat. Electron. Syst.* 8, 3 (July), pp. 269-297.

[24] P. Marwedel, *Embedded System Design*, Springer, 2006.

[25] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Symp. Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39-43.

[26] L. Wolsey, *Integer Programming*, John Wiley & sons, 1998.

[27] G. Wang, W. Gong and R. Kastner, "Application partitioning on programmable platforms using the ant colony optimization," *J. of Embedded Computing*, vol. 11, no. 19, 2006.