Comparative Evaluation and Analysis of Hardware/Software Partitioning Algorithms for Embedded System

Babangida Jauro Mohammed[*1], Hussaini Abatcha Geidam[2]

[*]E-mail: jaurobabangida21@gmail.com

[1]*Electrical and Electronics Engineering, Mai Idriss Alooma Polytechnic Geidam, Nigeria*

[2]*Electrical and Electronics Engineering, Federal Polytechnic Damaturu, Nigeria*

Abstract

Hardware/software partitioning has been considered as one of the most crucial steps in the design of embedded systems is i.e. deciding which components of the system should be implemented in hardware and which ones in software. Majority of the hardware/software partitioning problem formulations are N P-hard, this is the reason why most researchers are focusing on developing efficient heuristic methods. This paper compare the most popular heuristic methods after which the most simplest and efficient methods were considered for the design of a combinatorial structure. Two versions of the partitioning problem were considered, one N P-hard, and one with polynomial time solution. This is to understand the real cause of complexity in hardware/software partitioning. The heuristic makes use of problem-specific knowledge, and can thus find high-quality solutions rapidly, and also the polynomial-time algorithm serves as the basis for a highly efficient novel heuristic for the N P-hard version of the problem and it was observed after comparison that multi-level algorithm when implemented gives more efficiency and the different versions when combined supplement each other by eliminating the problems encountered when each of them act alone.

Key words: *Hardware/software partitioning, heuristic, N P-hard, polynomial time solution*

1.0 Introduction

The Hardware/software partitioning algorithms are tools which help researchers in decision making as to which function is to be implemented in hardware and software, to achieve design goals with regards to performance, power, size and cost. This part of the embedded system acts mostly as coprocessors [1].

The operating system (OS) resource managers were mostly used by software partitioning to segments the operating system, which limits the number of CPUs by creating areas where CPU resources are allocated to applications within the same operating system. This is a flexible way of managing data processing resources since the CPU capacity can be changed fairly easily, as additional resource is needed [2]. The Hardware partitioning on the other hand segments a server, by taking a single large server and separating it into distinct small systems. Each separate system acts as a physically independent, self-contained server, with own CPUs, operating system, separate boot area, memory, input/output subsystem and network resources [3].

Many algorithms have been implemented by researchers worldwide to obtain a solution to hardware and software partitioning problems. Some implemented single algorithm while some combined few of the methods to obtain a hybrid algorithm.

After review of so many of these algorithms, this work found that one of the best combination to make and achieve a better hybrid is to combine Particle swam optimization (PSO) algorithm, and Genetic algorithm (GA) because PSO and GA shared some similarities. Both of them begin with a randomized population and each population has their own fitness value for evaluation. They update the population and search for the optimum with random technique. Their differences been that while PSO has no evolution operators such as crossover and mutation GA do have [14]. Also in PSO, particles update themselves with the internal velocity and has memory to store the parameters and is simpler and faster than GA [12].

PSO algorithm is population-based selection, where a set of convenient solutions will be obtained through a set of potential solutions. Each potential solution in search space will adjust its movement according to its own moving experience as well as the moving experience of other solutions. The solution will move towards a promising area to get the global optimum. In short, the purpose of this algorithm is to find the global optimum of the fitness function defined in a given area.

GA algorithm is natural-based selection. This method modifies a population of individual solutions repeatedly. At each step, GA picked the individual solution randomly from the current population to be parents and used them to produce the child for the next generation. As the steps keep on repeating, the population will move towards an optimal solution. There are three main rules at each step to obtain the next generation from the parent population, namely selection rules, crossover rules and mutation rules [11].

The combination of these two algorithms to design a hybrid algorithm gives a more or less optimal solution of a partitioning problem, because it utilizes the advantages of the two algorithms and also overcome their disadvantages. A better results in hardware and software partitioning problem is thus obtained. The GA is easy to express in solving a combinatorial optimization problem and PSO has fast convergence speed [12]. If both of these algorithms were combined, it is very obvious that the execution time and partitioning result of hybrid algorithm will be improved.

Two types of algorithms were used to implement the hardware software partitioning. They are; the exact and the heuristic or evolutionary methods. The exact method partitioning algorithms tends to be quite slow for bigger dimensions of the problem. Therefore, this research work uses heuristic method to increase performance while also reducing the cost of the system.

2.0 Problem Statement

Partitioning remain a key challenge that affect embedded system efficiency and optimization. Partitioning were done manually by the designers based on their experience in the olden days [4]. As the embedded system design increased in its complexity over the years, these efforts to do partitioning manually become unrealistic due to the number of components with different characteristic involved in the design [2].

Exact algorithms, such as branch-and-bound and dynamic programming were among the initial automated partitioning design proposed by researchers [6, 7]. But they are slow, heuristic algorithms such as genetic algorithms, particle swarm optimization and simulated annealing were then developed. These heuristic algorithms also have their own limitations [8, 9]. The PSO algorithm tends to back into its local optimum as the size of the given area is high and the size of convergence rate is low during the iteration process. The Genetic algorithm on the other hand has no guarantee of finding global maxima and requires a decent size of population and a large number of generations to obtain good results. In order to improve the performance, many hybrid algorithms were proposed by researchers. This work proposed a multi-level hybrid and compare their performances.

3.0 Objectives

a. To evaluate the performance of multi-level hybrid algorithm in term of number of iteration to obtain stable cost.

b. To compare the algorithm performance of the hybrid algorithms based on their levels to see how the hybrid level affect performance.

4.0 Methodology

A two and three-level hybrid models were designed using PSO and GA algorithms. They were used to optimize the performance of an embedded system by deciding the implementation of specific application or function in software or hardware. The number of iterations to achieve best cost and the time taking to reach the best cost were examined. And based on these parameters a comparison between GA, PSO and hybrid of GA and PSO were made and also a three-level hybrid model of GA-GA-PSO is constructed and compared.

In this work GA and PSO were implemented separately, and then two and three level algorithms were constructed. In the two level hybrid GA was implemented followed by PSO. And in the three level were implemented using two level successive GA algorithms followed by a PSO model. Figure 1(a) and (b) shows the algorithms architecture.

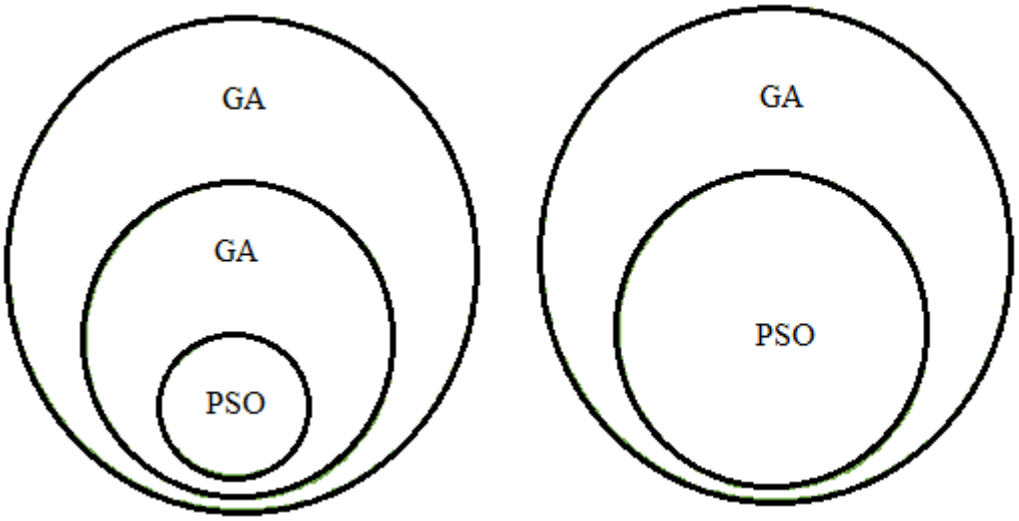

Fig.1. (a) Three level hybrid model architecture (b) two level hybrid model architecture

GA and PSO models were constructed individually in MATLAB environment before combining them into the hybrid model.

GA algorithm imitates the process of natural selection. The first step in GA algorithm is to initialize all the populations of solution and evaluate their fitness after which they will undergo a crossover and mutation operations. The fitness for each of these solutions will then be re-evaluated after crossover and mutation. By sorting all the solutions according to fitness, the extra number of solutions with lowest fitness will be eliminated [15]. The flow chart of the implemented algorithm is shown in Fig. 2.

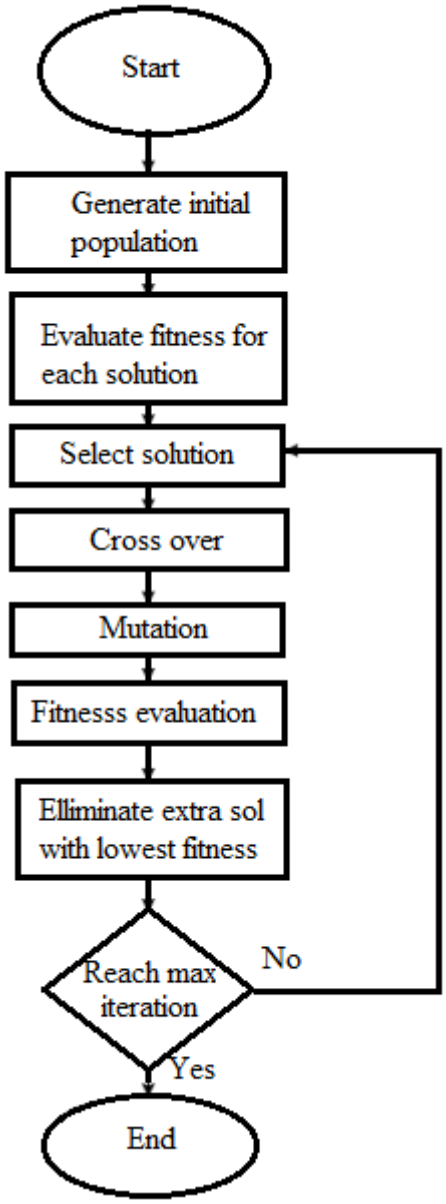

Figure 2: GA algorithm flow chart

PSO algorithm is a population based stochastic optimization technique inspired by social behaviour of bird flocking or fish schooling. The best way to illustrate it is by considering a group of birds searching for food in an area. They don't know where the food is but they know how far the food is. The best strategy to reach the food is to follow the bird nearest to the food. PSO is an inspiration from this scenario [3].

The PSO model, was obtained by initializing the populations of solutions and fitness were evaluated for each solution. The velocity for each solution was initially set to zero. The velocity was then updated using equations 1 and 2.

$$\mathrm{v[i]} = (W * v[i]) + C_1 r_1 (pBest[i] - x[i]) + C_2 r_2 (gBest - x[i]) \quad (1)$$

$$x[i] = x[i] + \mathrm{v[i]} \quad (2)$$

Where

v[i] = velocity of particle

W = Damping inertia factor that takes values downward from 1 to 0 according to the iteration number. (W = W*w_{damp})

C_1 = self-confidence (cognitive) factor

r_1 = random numbers between 0 and 1

C_2 = swarm confidence (social) factor

r_2 = random number between 0 and 1

$[i]$ = current position of particle

$p[i]$ = position vector of best solution that this particle achieved so far

$gBest$ = best position vector obtained so far by any particle in the population

The fitness of each of the particles were evaluated after changing their position and velocity. *pBest* and *gBest* were updated accordingly. These steps were repeated until maximum iterations were reached. Fig. 3 shows the PSO flow chart.

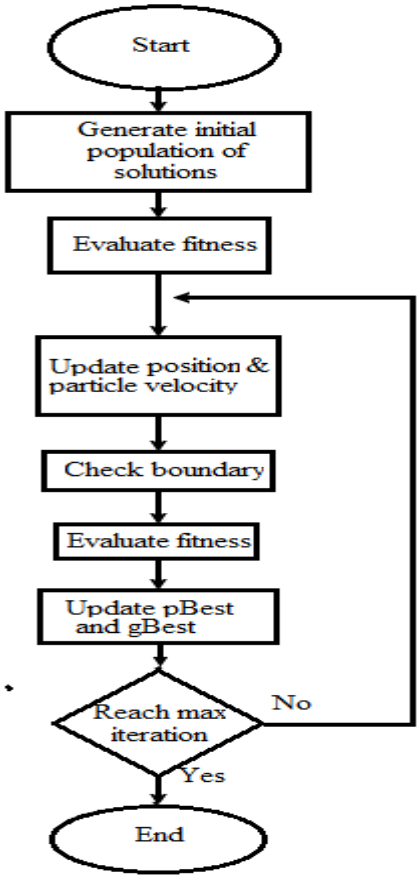

Figure 3: PSO flow chart

4.2

Multi-Level Hybrid Modelling

To implement the multi-level hybrid models, GA was implemented followed by PSO in the two level hybrid, while in the three level hybrid implementation, two successive GA algorithms model were constructed followed by a PSO model. In the two level hybrid, first GA flow chart was utilized up to extra data elimination then the set of data was passed over to PSO algorithm.

In the three level hybrid, the set of data was send over to the next GA algorithm for another round of crossover and mutation after data elimination of the first GA. the data was again passed through second data elimination of the second GA algorithm, after which it was passed over to PSO algorithm. The flow charts of the two level and three level models were shown in figure 4(a) and (b) respectively.

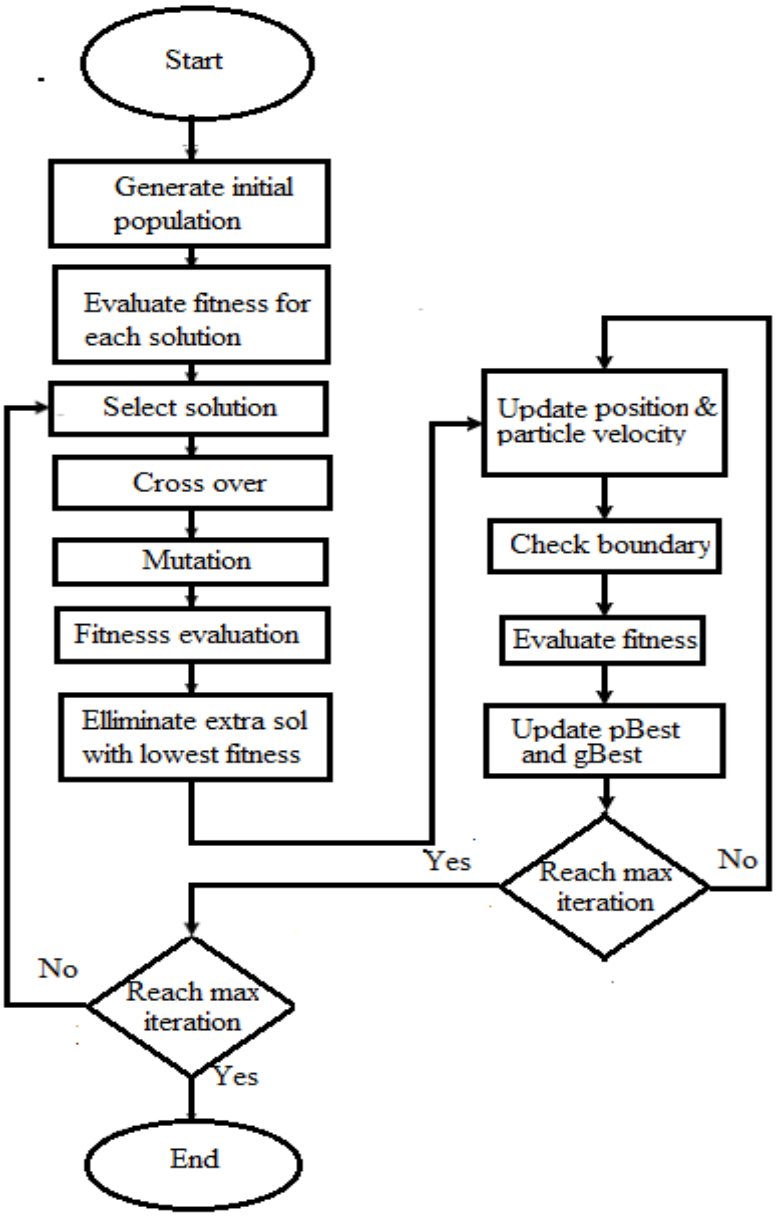

Figure 4(a): GA-PSO algorithm flow chart

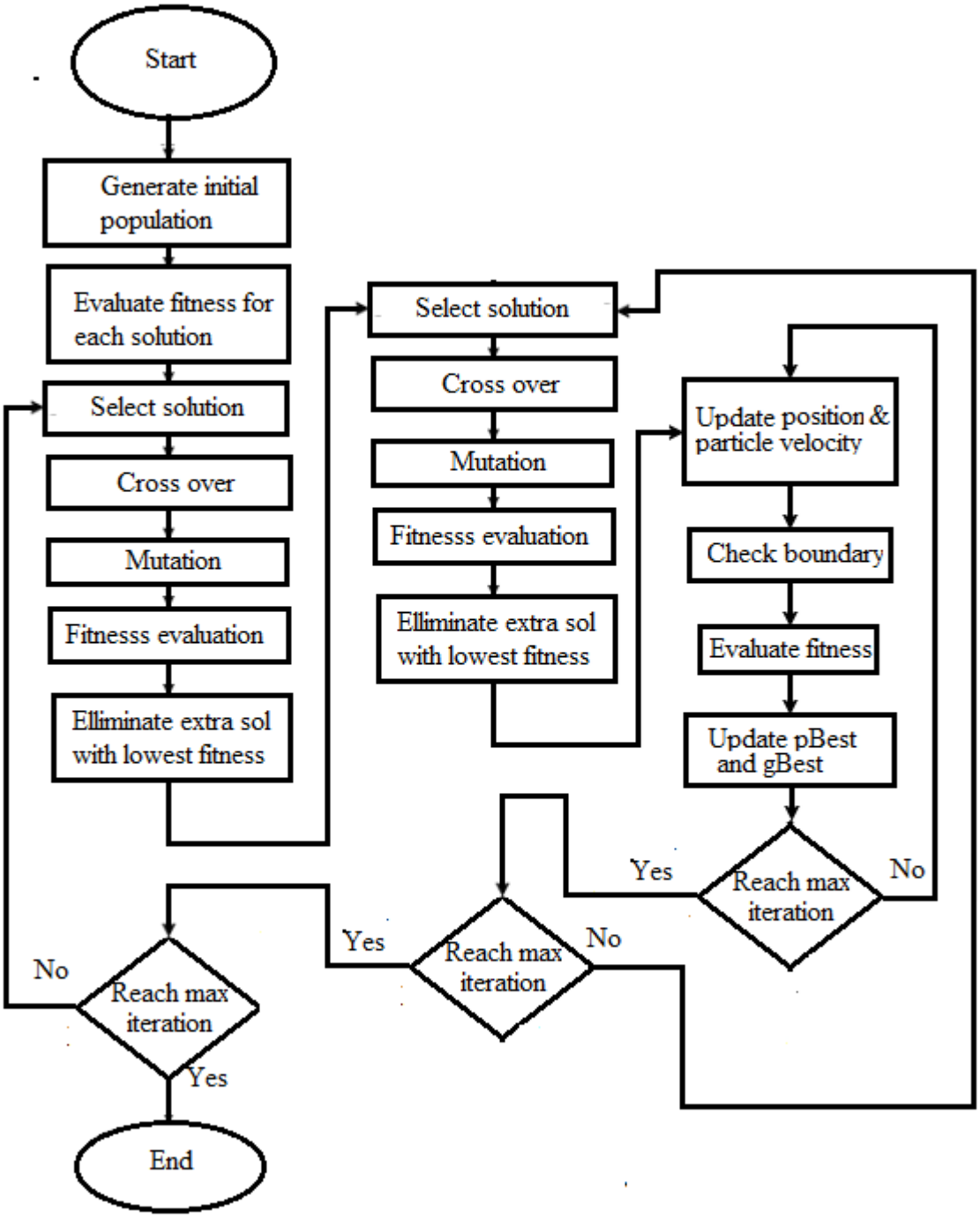

Figure 4(b): GA-GA-PSO algorithm flow chart

4.3. Setting of model and choice of Parameters

Binary solutions were used in this work with hardware node assuming a value of 0 and software node a value 1. Damping coefficient was made to decrease in each iteration by a factor of W_{damp} which is set as 0.98. Both hardware cost and software cost are uniformly and randomly generated in the range from 1 to 99. The cost function is given in Equation 3.

$$\text{Cost} = 100 * \left[\frac{HWcost}{all\ HWcost} + \frac{SWcost}{all\ SWcost} + \frac{PWcost}{all\ PWcost}\right] \quad (3)$$

Where
HWcost is the hardware implementation cost of particle
SWcost is the software implementation cost of particle
PW cost is the power implementation cost of particle
allHWcost is the total of hardware implementation cost of all particle allSWcost is the total of software implementation cost of all particle
allPWcost is the total of power implementation cost of all particle in both software and hardware

Since the node value must be 0 or 1 for a binary problem, the particles were rounded by using hard decision rounding (HDR). a node is mapped to hardware if the node value is lower than 0.5 and mapped to software if node value is greater than 0.5.

GA, crossover probability (Pc) is set to 0.9 and mutation probability (Pm) is set to 0.1. C_1 and C_2 for velocity equation were set to 2 and W was set to 1. Damping value is set to 0.97 for the PSO. Number of particle = 512 Population size = 60 Maximum iterations = 500

Fitness Proportionate Selection method was adopted for this research because of its simplicity and fastness for large number of particles. A random number R between 0 and 1 is chosen. Last individual whose accumulated normalized value is smaller than R was selected.

Heuristic Crossover method was also used for this work. This operator creates one child offspring from two parents. The child gene was obtained using Equation 4.

$$O_1 = P_1 + R\ (P_2 - P_1)_1 = P_1 + R\ (P_2 - P_1) \quad (4)$$

Where
O1 is the child gene
P_1 and P_2 is parent genes
R is a random number between 0 and 1

And finally uniform mutation was applied for mutation operator. This was used to replace the original value of the chosen gene with a uniform random value generated between lower and upper boundary for the gene.

5.0 Results and Discussion

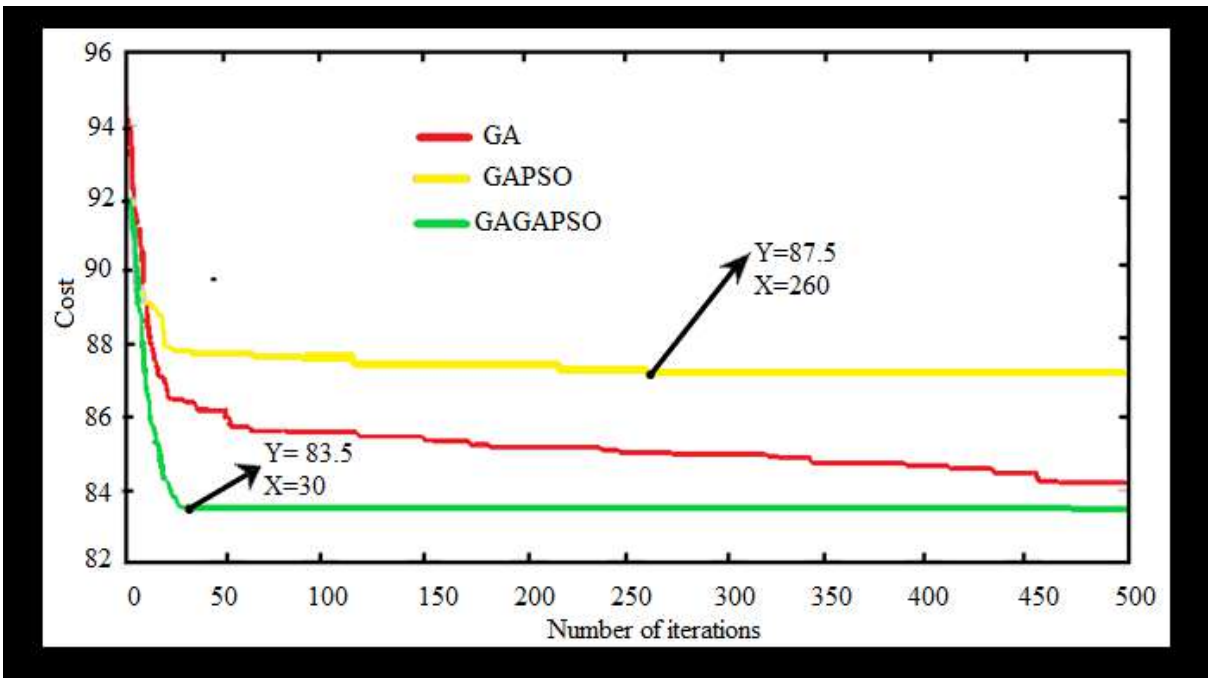

Figure 5: Cost versus Iteration graph of first simulation

The figure above shows the cost versus iteration graph of first simulation. Three algorithms, GA, GA-PSO and GA-GA-PSO were plotted on the graph to show the iterations needed to achieve the best cost. The best cost was said to be reach when the cost is not changing for 450 consecutive iterations. The GA cost keeps changing even after 450 iterations therefore the best cost cannot be determined. The algorithm was not stable to reach the best cost in fewer than 450 iterations. Therefore, the data will not take into consideration. the two level GA-PSO, on the other hand became stable after 260 iterations and the three level GA-GA-PSO was the lowest among the three algorithms and was stable after just 30 iterations, as such it has the best cost. Also, the GA-GA-PSO is able to provide solution with exact value of 0s and 1s while GA algorithm and GA-PSO algorithm show decimal values. The time needed was then calculated from the number of iterations to know the algorithm efficiency using the following formula.

$$t = T_{total} \times \frac{iteration}{500}$$

However, since random particles were used in the simulation, the total time needed is slightly different for each simulation. Hence, 10 trials of the simulations were carried out and the average was taken. The result of 10 simulations were recorded in table 1 and 2. The averaged value was used for comparison and discussion.

Table 1: Result from GA-PSO algorithm

Trial	Number of Iteration	Total time needed (s)
1	260	5.3113
2	319	5.4610
3	172	3.8159
4	398	7.7873
5	300	5.0115
6	340	7.1647
7	313	6.5724
8	299	6.1977
9	386	8.1843
10	266	5.5137
Average		6.1020

From table 1. the average time needed to reach the best cost is 6.1020s. The GA-PSO algorithm is more efficient and stable than GA algorithm alone and require less time to reach the best cost when compared to GA.

Table 2: Result from GA-GA-PSO algorithm

Trial	Number of Iteration	Total time needed (s)
1	30	1.1254
2	21	0.7100
3	15	0.6149
4	25	0.9702
5	15	0.5203
6	23	0.9236
7	21	0.8727
8	16	0.6687
9	12	0.4960
10	23	0.9102
Average		0.7812

From table 2. The time needed to reach the best cost for three level hybrid GA-GA-PSO model is 0.7812s. And is the lowest among the three algorithms.

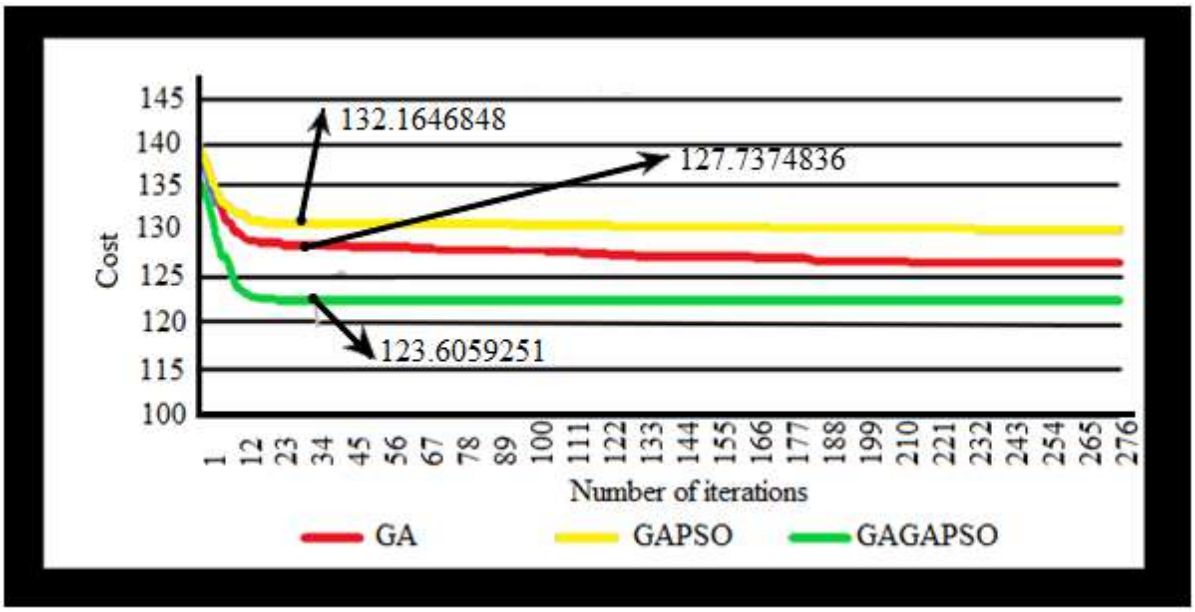

Figure 3. Cost versus iteration for 450 nodes

Figure 3 also shows the cost versus iterations graph of the three algorithms was plotted on the graph and it was seen that the GA-GAP-SO has a smooth graph with few iterations to achieve the minimum cost. The number of iterations to reach the best cost is approximately 12-31 iterations.

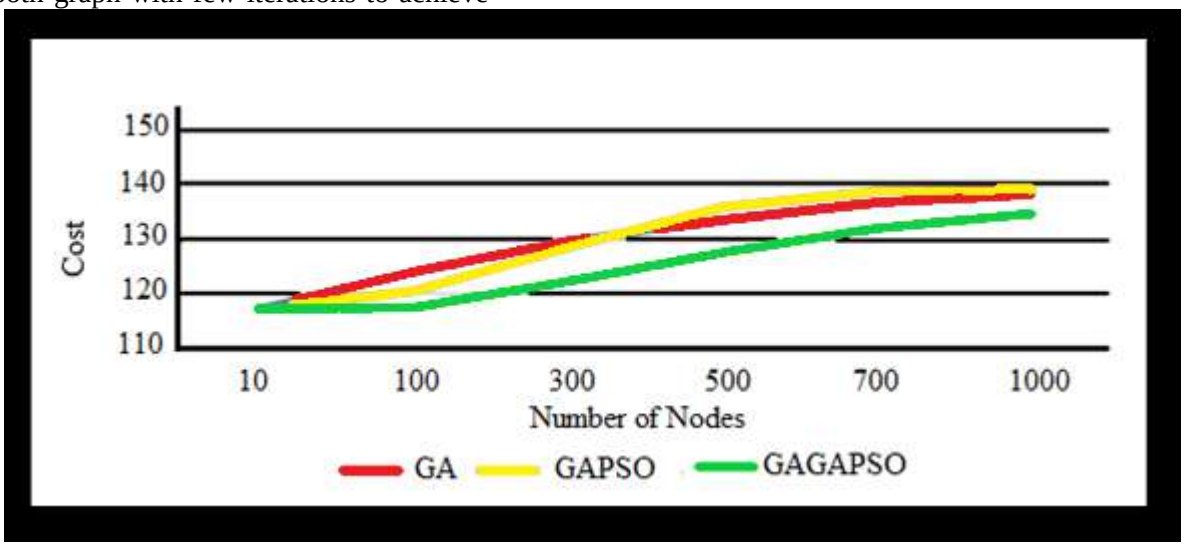

Figure 4. Cost versus number of nodes

Figure 4 shows the cost versus the number of nodes for the three algorithms. From this graph, the GAPSO performs better than GA when the number of nodes is less than 450. If the number of nodes is more than 450, then GA performs better than GAPSO. From the graph also GAGAPSO performs better than GA and GAPSO for all nodes.

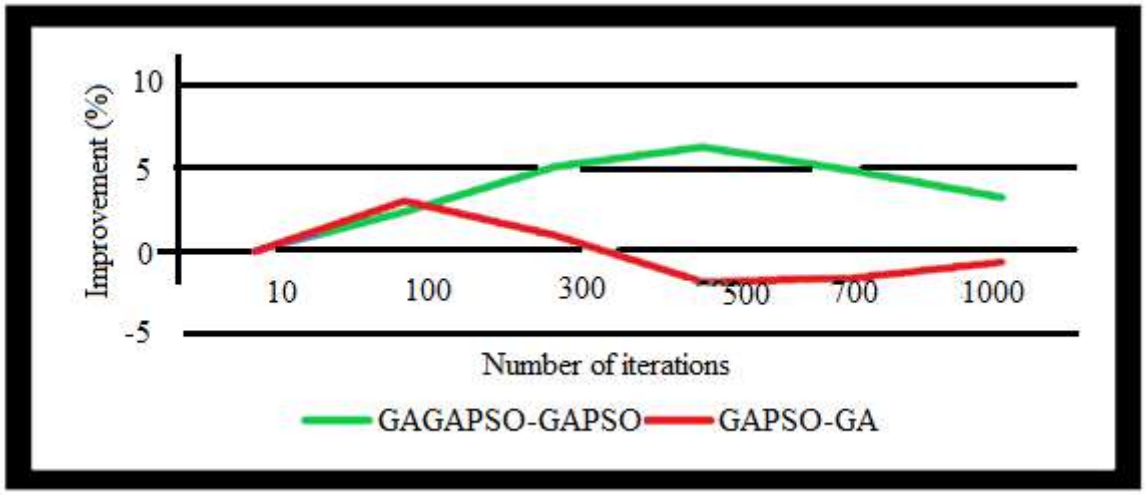

Figure 5. Improvement versus number of nodes

Figure 5 is the plot of percentages of improvement in terms of the minimum cost for GA-GA-PSO over GAPSO and GA-PSO over GA. It was observed from the graph that the maximum improvement of GA-GA-PSO over GA-PSO, is at 450 nodes, with an improvement of 6.3%. After 450 nodes, the improvement of GA-GA-PSO slightly decrease. And for GA-PSO over GA, the maximum improvement was achieved at the 100th node with improvement at approximately 3%. When the number of nodes continuously increase, the performance of GA-PSO also decrease.

5.1 Comparison between algorithms

The GA algorithm was excluded from this comparison because it was not able to give its best cost within 450 iterations. Therefore, it can be concluded that GA algorithm is unstable and require longest time to reach best cost.

Table 3: Result comparison between 3 algorithms

Algorithm	Average time needed (s)
GA	-
GA-PSO	6.1020
GA-PSO-GA	0.7812

The average time needed for GA-PSO algorithm to reach best cost was obtained to be 6.1020s while that for GA-GA-PSO algorithm to reach best cost is 0.7812s. The average time needed for GA-GA-PSO is much lower compared to GA-PSO algorithm and have lower best cost value and computational time. Its solution also consist of exact value of 1s and 0s. Moreover the GA-GA-PSO algorithm also provides better solution for optimization, compared to GA-PSO algorithm and GA algorithm.

6.0 Conclusion

A three-level hybrid GA-GA-PSO algorithm that combines the advantages of successive algorithms into a single model has been designed for software hardware partitioning using MATLAB. This algorithm used shorter iteration to obtain stable cost compared to GA and hybrid GA-PSO algorithm. It also obtains the lowest cost compared to single and two level hybrid algorithms. It also proved to have better average execution time to reach best cost.. The slope was smoother and iterations to achieve best cost was also shorter.

.

7.0 References:

Tiong Reng Xian, Zaini Abdul Halim, Ching Chia Leong, Tan Jiunn Gim
"Hardware-software partitioning using three-level hybrid algorithm for system-on-chip platform"
Bulletin of Electrical Engineering and Informatics Vol. 10, No. 1, February 2021, pp. 466~473 ISSN: 2302-9285, DOI: 10.11591/eei.v10i1.2201

Mohamed b Abdelhalim, A.E. Salama, Serag E. D. Habib. Hardware Software Partitioning using
Particle Swarm Optimization Technique. Conference paper from System-on-Chip for Real-Time Applications, the 6th International Workshop. Source: IEEE Xplore. DOI: 10.1109/IWSOC.2006.348234

Imene Mhadhbi, Slim Ben Othman, and Slim Ben Saoud (2016). An Efficient Technique for Hardware/Software Partitioning Process in Codesign. Scientific Programming Volume 2016 (2016),
Article ID 6382765, 11 pages. dx.doi.org/10.1155/2016/6382765

Marrec, P. L., Valderrama, C., Hessel, F., Jerraya, A., Attia, M., & Cayrol, O. (n.d.). Hardware, software and mechanical cosimulation for automotive applications. Proceedings. Ninth International Workshop on Rapid System Prototyping (Cat. No.98TB100237). doi:10.1109/iwrsp.1998.676692

B., M., & Habib, S. E. (2009). Particle Swarm Optimization for HW/SW Partitioning. Particle Swarm Optimization. doi:10.5772/6740

Binh, N. N., Imai, M., Shiomi, A., & Hikichi, N. (1996). A hardware/software partitioning algorithm for designing pipelined ASIPs with least gate counts. 33rd Design Automation Conference Proceedings, 1996. doi:10.1109/dac.1996.545632

J. Madsen, J Gorde, P. V. Knudsen, M. E. Petersen, A. Haxthausen (1997). "LYCOS: The Lyngby cosynthesis system", Design Automation of embedded Systems, vol. 2, no. 2, pp. 195-236, April 1997.

Z. A. Mann (2004). "Partitioning algorithms for Hardware/Software Co-design", 2004.

J. Henkel, R. Ernst (2001) "An approach to automated hardware/software partitioning using a flexible granularity that is driven by high-level estimation techniques", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, no. 2, pp. 273-289, 2001.

J. Kennedy and R. Eberhart. "Particle swarm optimization," Neural Networks, 1995. Proceedings., IEEE
International Conference on, Perth, WA, 1995, pp. 1942-1948 vol.4. doi: 10.1109/ICNN.1995.488968

Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on (pp. 39-43). IEEE.

Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. In evolutionary computation, 2001. Proceedings of the 2001 Congress on (Vol. 1, pp. 81-86). IEEE.

G.Li, J.Feng, C.Wang, J.Wang. Hardware/Software Partitioning Algorithm Based on the Combination of Genetic Algorithm and Tabu Search, 2015. Engineering Review, Vol 34, Issue 2, pg 151-160.

R.Hassan, B.Cohanum, O.Weck A Comparison of Particle Swarm Optimization and The Genetic Algorithm, 2005. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Material Conference. doi:10.2514/6.2005-1897

Erik D. Goodman Introduction to Genetic Algorithms, 2011. GECCO '11 Proceedings of the 13th annual conference companion on Genetic and evolutionary computation on (pg.839-860).

T.Blickle, L.Thiele A comparison of Selection Schemes used in Genetic Algorithms, 1995. TLK-Report Second Edition.

A.J. Unbarkar, P.D. Sheth Crossover Operators in Genetic Algorithms: A Review, 2015. ICTACT Journal on Soft Computing, Vol 06, Issue 1, pg 1083-1092.

Introduction to Genetic Algorithms, Part XI Crossover and Mutation, 1998. Retrieved from: http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php