Hybrid Metaheuristic Algorithm for Real Time Task Assignment Problem in Heterogeneous Multiprocessors

Poongothai Marimuthu, Rajeswari Arumugam, and Jabar Ali
Department of Electronics and Communication Engineering, Coimbatore Institute of Technology, India

Abstract: *The assignments of real time tasks to heterogeneous multiprocessors in real time applications are very difficult in scenarios that require high performance. The main problem in the heterogeneous multiprocessor system is task assignment to the processors because the execution time for each task varies from one processor to another. Hence, the problem of finding a solution for task assignment to heterogeneous processor without exceeding the processors capacity in general is an NP hard problem. In order to meet the constraints in real time systems, a Hybrid Max-Min Ant colony optimization algorithm (H-MMAS) is proposed in this paper. Max-Min Ant System (MMAS) is extended with a local search heuristic to improve task assignment solution. The Local Search has resulted in maximizing the number of tasks assigned as well as minimizing the energy consumption. The performance of the proposed algorithm H-MMAS is compared with the Modified Binary Particle Swarm Optimization algorithm (BPSO), Ant Colony Optimization (ACO), MMAS algorithms in terms of the average number of task assigned, normalized energy consumption, quality of solution and average Central Processing Unit (CPU) time. From the experimental results, the proposed algorithm has outperformed MMAS, Modified BPSO and ACO for consistency matrix. In case of inconsistency matrix H-MMAS performed better than Modified BPSO, similar to ACO and MMAS, but there is an improvement in the normalized energy consumption.*

Keywords: *Multiprocessors, task assignment, heterogeneous processors, ant colony optimization, real time systems.*

Received September 21, 2014; accepted December 21, 2015

1. Introduction

Embedded Systems are frequently implemented up on hardware a platform that includes different types of processors such as a general-purpose processor, a special-purpose processor, and a coprocessor. In general, a heterogeneous multiprocessor platform is based on different Instruction Set Architectures (ISAs) with configurable and extensible features [10]. This multiprocessor platform meets the computational demands for various applications. Real-time embedded systems are more complex as it includes many heterogeneous components. It is very difficult to implement the real time applications up on the heterogeneous multiprocessor system. Therefore, implementation of real-time application on the heterogeneous platform needs additional effort than the homogeneous platform [4]. The processors are identical in a homogeneous platform, task assignment can be done by solving Bin Packing Problem (BPP) [8]. The complexity increases in such a way that every real time application is in need of different execution times on heterogeneous processors [4]. Hence, Task assignment in a heterogeneous multiprocessor is a combinatorial problem which is an NP hard problem [6, 9]. It can be solved by applying approximate or meta-heuristic algorithms to obtain sub-optimal results within a reasonable time [9]. In this paper, a Hybrid Max-Min Ant Colony Optimization algorithm (H-MMAS) is proposed. It is based on Max-Min Ant System (MMAS) along with the local search algorithms which act as a daemon action to enhance the proposed algorithm further for finding a solution for real-time task assignment to the heterogeneous processors without exceeding the processors computing capacity and fulfilling the deadline constraints. This paper considers the two objectives for the task assignment algorithm. The foremost one is resource objective and the objective is to achieve maximum task assignment in the heterogeneous multiprocessors. The second one is energy objective and the aim is to minimize the cumulative energy consumption for all assigned tasks in a solution.

2. System Model and Problem Statement

In this paper, the heterogeneous multiprocessor environment with m preemptive processors $\{P_1, P2\ldots\ldots P_m\}$ based on CMOS technology is considered [4, 10]. The processors in the heterogeneous environment are operated at different speeds and one instruction per cycle is limited to execute in each processor at variable speed. The energy consumption ($E_{i,j}$) of the task T_i on processor P_j per period is calculated using Equation (1).

$$Energy_{i,j}(E_{i,j}) = Power_{i,j} \approx \left(C_{ef} \cdot \frac{s_{i,j}^{3}}{k^{2}} \right) e_{i,j} = \left(\frac{C_{ef}}{K^{2}} \right) c_i s_{ij}^{2} \quad (1)$$

Where, C_{ef} is the effective switching capacitance related to tasks, k is the constant. $e_{i,j}$ is the execution time for task T_i on processor P_j , $s_{i,j}$ is the speed of P_j for T_i and c_i is the number of clock cycles to execute a task T_i. Equation (1) shows that the energy consumption is directly proportional to the $ci. s_{ij}^{2}$. This equation is significant, because the processors operate at different speeds [2].

A set of N periodic tasks $T= \{T_1, T_2, \ldots \ldots \ldots T_N\}$ is considered. The tasks are assumed to be mutually independent and there is no inter task communication [4, 12]. T_i is defined as $T_i = \{ e_{i,j}, p_{i,j}\}$ where $e_{i,j}$ is the estimated worst-case execution time for task T_i on processor P_j and $p_{i,j}$ is the period of T_i on processor P_j. The task assignment problem considered here is the off-line version, under the condition that the utilization of each processor is less than or equal to 1. In this paper, the partitioned scheme for task assignment and Earliest Deadline First algorithm (EDF) for scheduling the tasks on each processor is considered. The proposed task assignment algorithm has two objectives.

- The first objective aims at maximizing the number of tasks assigned (resource objective) in the heterogeneous multiprocessor under the condition that the cumulative utilization of any processor does not exceed the utilization bound of the EDF algorithm [8], which is considered to be NP-hard problem [2, 4].
- The second objective is to minimize the cumulative energy consumption for all assigned tasks in a solution (energy objective). The resource objective is given precedence over the energy consumption objective.

3. Related Work

Chen *et al.* [4] proposed a new algorithm based on Ant Colony Optimization (ACO) meta-heuristic with 1-Opt and 2-Exchange local search techniques for assigning real-time tasks to heterogeneous processors with the resource and energy objectives. The results are compared with Genetic Algorithm and Linear Programming based approaches and it is shown that the ACO algorithm outperforms the major existing algorithms. Prescilla and Selvakumar [12] proposed Modified Binary Particle Swarm Optimization algorithm (Modified BPSO) and Novel Binary Particle Swarm Optimization to solve the real-time task assignment in a heterogeneous multiprocessor. The results of Modified BPSO, Novel BPSO are compared with ACO and proved that Modified BPSO performs better than Novel BPSO and ACO for consistent utilization matrix and ACO performs better than Modified BPSO and Novel BPSO for inconsistent utilization matrix. From the results, it is observed that these algorithms need more number of iterations to converge and ACO trained input. Braun *et al.* [3] proposed eleven static heuristic algorithms for mapping a class of independent tasks into the heterogeneous distributed computing system to solve the heterogeneous multiprocessor task partitioning problem with the objective of minimizing the makespan. Srikanth *et al.* [13] proposed a task scheduling algorithm using Ant Colony Optimization for scheduling a task set on heterogeneous processors by considering load balancing across the processors. The authors have modeled the heterogeneity of the processors by assuming different utilization times for the same task in different processors. The results are compared and it is observed that the ACO algorithm performs better than the First Come First Serve (FCFS) in terms of waiting time. Although the processor utilization is more for some processors using FCFS algorithm, it is shown that the load is better balanced among the processors using ACO. Jin *et al.* [7] proposed a new feasible algorithm based on Ant Colony Optimization meta-heuristic to solve the multiprocessor control system problem for task assignment and scheduling by taking into account the scheduling performance index and the control performance index as fitness functions of optimization. Wu *et al.* [15] proposed the independent task assignment algorithm for space weapons based on Multi-agent System and Ant Colony Optimization and proved that the proposed model is feasible and effective.

The difference between the MMAS and the proposed Hybrid MMAS (H-MMAS) is that, the former one includes heuristic information to construct solution, whereas the later one excludes heuristic information which is compensated by two local search procedures. To the best of our knowledge, this is the first paper that applies H-MMAS to optimize assignment solution in terms of resource utilization of tasks and energy consumption.

4. MAX–MIN Ant System [MMAS]

Dorigo and Stützle [5] and Stützle and Hoos [14] proposed the Max-Min Ant System. The key feature of MMAS is that the pheromone trails are updated with only one ant, and this ant could be the iteration-best ant or global-best ant which finds the best solution. Moreover, the maximum and minimum values of the pheromones are limited to certain values to escape getting stuck at local solutions. Additionally, pheromone trails initialize to the upper bound τ_{max} to have uniform exploration in the whole search space [1, 5]. It differs from Ant System (AS) mainly in the following three aspects:

1. Only one single ant adds pheromone after each iteration.

2. The range of possible pheromone trails on each solution component is limited to an interval $[\tau_{min}, \tau_{max}]$.
3. The initial pheromone trails are set to τ_{max}.

MMAS uses either the global or iteration best solution for the pheromone trail update. In order to avoid stagnation, the amount of pheromone is restricted to a range $[\tau_{min}, \tau_{max}]$ [1, 5]. The ants are situated at random places initially. At each construction step, ant *k* applies a probabilistic action choice rule to choose a next node to visit next, until a complete solution has been built.

After all ants have constructed a tour, pheromones are updated by applying evaporation as in ant system as given in the Equation (2).

$$\tau(i,j) = (1-\rho)\tau(i,j) \forall (i,j) \in N(s) \quad (2)$$

Where $0 < \rho < 1$ is the pheromone evaporation rate and $\tau(i, j)$ is the pheromone trail. This is followed by the deposit of the new pheromone given by

$$\tau(i,j) = \tau(i,j) + \Delta\tau(i,j)^{best} \quad (3)$$

Where $\Delta\tau(i, j)^{best} = 1/f(s)^{best}$; $f(s)^{best}$ denotes the solution cost of the iteration-best ($s^{best} = s_{ib}$).

The pheromone limits are calculated by:

$$\tau_{\max} = f(s^{best}) / \rho$$
$$\tau_{\min} = \tau_{\max} / (\omega \times In(\theta + 1)) \quad (4)$$

Where Θ is the sequential number of the current iteration starting with 1, ω is a constant and $\omega \geq 1$. s^{best} denotes the iteration best solution.

Initially, all pheromone values are set to τ_{max} and after each iteration pheromone limits are updated. Pheromone trails are evaporated by Equation (2). The pheromones associated with the best solution are increased by Equation (3). Then the validity of limits is checked by the algorithm shown in Algorithm 1.

Algorithm 1: Pheromone update operator

```
Procedure Update_Pheromone( )
{
if τ(i,j) <τmin
{ set τ(i,j) = τmin. }
if τ(i,j) >τmax
{ set τ(i,j) = τmax. }
}//End Procedure
```

If the current best solution has not improved for a certain number of iterations reinitialize the pheromone table by $\tau(i,j) = \tau_{max}$ setting for all *i, j*. Because of pheromone limits, convergence condition is easily formulated: when only one pheromone trail reaches τ_{max} and all other trails become τ_{min}, convergence occurs.

5. Proposed Hybrid Max–Min Ant System for Task Assignment Problem

5.1. Construction Graph and Constraints

For a given set of heterogeneous multiprocessor and task set, each task assigned to one processor by the artificial ant stochastically until each of the tasks is assigned to specific processor without exceeding its computing capacity. The construction graph is shown in the Table1 [4].

Table 1 . Utilization Matrix with '*n*' Tasks '*m*' Processors.

Task _id	**P_1**	**P_2**	**P_3**	**P_4**	**…….**	**P_m**
T_1	$U_{1,1}$	$U_{1,2}$	$U_{1,3}$	$U_{1,4}$	…….	$U_{1,m}$
T_2	$U_{2,1}$	$U_{2,2}$	$U_{2,3}$	$U_{2,4}$	…….	$U_{2,m}$
T_3	$U_{3,1}$	$U_{3,2}$	$U_{3,3}$	$U_{3,4}$	…….	$U_{3,m}$
T_4	$U_{4,1}$	$U_{4,2}$	$U_{4,3}$	$U_{4,4}$	…….	$U_{4,m}$
…….	…….	…….	…….	…….	…….	…….
T_n	$U_{n,1}$	$U_{n,2}$	$U_{n,3}$	$U_{n,4}$	…….	$U_{n,m}$

In Table 1, T_i $(1 \leq i \leq n)$ represents the task *Ti*, P_j $(1 \leq j \leq m)$ represents the j^{th} processor, and $u_{i,j}$ represents the utilization of the i^{th} task on the j^{th} processor. The utilization matrix $u_{i,j}$ is an *nxm* matrix in which *m* is the number of heterogeneous processors and *n* is the number of tasks. The row of utilization matrix represents the estimated amount of computation of a given task on each heterogeneous processor. Similarly, the column of utilization matrix represents the estimated amount of computation for a given processor for each task in the task set. The utilization matrix U_{n*m} holds the real numbers in (0, 1) and infinity. If $U_{ij} = \infty$ means, the particular task is not suitable to execute on a specified processor P_j.[4, 11, 12]. An artificial ant finds to travel across the construction graph based on the constraints given by Equations (5) and (6).

- *Constraint* (1): A task needs to be allocated to only one processor

$$\sum_{j=1}^{m} \Omega_{i,j}^{s} = 1 \quad i=1,2,\ldots\ldots..n \; ; \; j=1,2\ldots\ldots m \quad (5)$$

- *Constraint* (2): The total utilization of each processor does not exceed unity

$$\sum_{i=1}^{n} u_{i,j} . \Omega_{i,j}^{s} \leq 1 \quad (6)$$

where,

$$\Omega_{i,j}^{s} = \begin{cases} 1 & \text{if square } (T_i, P_j) \text{ is used in solution } s \\ 0 & \text{otherwise} \end{cases}$$

5.2. Solution Construction

An artificial ant increases the pheromone value $\tau(i,j)$ at the edge between T_i and P_j which represents the possibility of assigning the task T_i to the processor P_j.

The pheromone values of the ant are initialized as same for solution construction. Each ant builds a tour from a starting pair of task and processor [5]. The probability of selecting the next pair of task and processor is given by Equation (7).

$$p(s,i,j) = \begin{cases} \frac{\tau(i,j)(t)}{\sum_{(i',j')\in N(s)}^{n} \tau(i',j')(t)} & if(i,j) \in N(s \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where, $N(s)$ denotes the set of eligible pairs of task and processor is obtained; $\tau(i,j)$ denotes the pheromone trial of (T_i, P_j).

5.2.1. Exclusion of Heuristic Information

From the literature survey, it can be understood that the existing ACO algorithms for the heterogeneous environment have included heuristic information $(\eta(i,j))$ for deriving the best solution [4, 12, 13]. The heuristic information calculation degrades the performance of the ant system when the values of utilization matrix become very small, thereby heuristic calculation approaches close to the worst case scenario making it difficult for choosing the particular cell for including it in the ant's solution [4]. In order to avoid this, the heuristic information has been excluded in the proposed algorithm. The heuristic information excluded formula is given in the Equation (7).

5.2.2. Inclusion of Two Local Searches

The proposed H-MMAS algorithm includes two local search procedures such as1-Optimal (1-OPT) and 1-Difference (1-DIFF) for compensating the exclusion of heuristic information to improve the task assignment solution after the construction procedure is completed. The local search algorithm starts with an initial task assignment solution, and then search for better solutions using the following neighborhood structures using following the local search procedures

5.2.2.1. Local Search 1: Reducing Average Utilization (1-OPT)

A task is removed from the assigned processor and then assigned to a different processor, only if the overall utilization is reduced (1-OPT). If the total utilization exceeds unity; it is replaced in the initial position.

Algorithm 2: Local Search 1(1-opt)

```
1.  Procedure 1-OPT ( )
2.  {
3.  for each ant k
4.  {
5.    Uavg = (Σ^m_{j=1}Uj)/m
6.      for each task i
7.       {
8.        Remove a task from one processor and assign
          it to the neighborhood processor;
9.        Unew = (Σ^m_{j=1}Uj)/m
10.       If Unew<Uavg; new task assignment is updated;
11.      Elseif Unew>Uavg; old task assignment is
                retained;
12.       }// end for
13.  }//end for
14.  }//end procedure
```

5.2.2.2. Local Search 2: Reducing Difference in Utilization (1-Diff)

A task is removed from the assigned processor and then assigned to a different processor, only if the sum of the differences between individual utilization and overall utilization is reduced (1-Difference). If the total utilization exceeds unity; it is replaced in the initial position.

Algorithm 3: Local Search 2(1-Diff)

```
1.  Procedure 1-DIFF ( )
2.  {
3.  for each ant k
4.  {
5.    Dorg= Σ^m_{j=1}(abs(Unew-Uj))
6.      for each task i
7.      {
8.        Remove a task from one processor and
          assign it to the neighborhood processor;
9.      Compute Dnew = Σ^m_{j=1}(abs(Unew-Uj))
10.     Unew = (Σ^m_{j=1}Uj)/m
11.     if Dnew< Dorg; new task assignment is
              updated;
12.     Elseif Dnew> Dorg ; old task assignment is
                retained;
13.       }//end for
14.}//end for
15.}//end Procedure
```

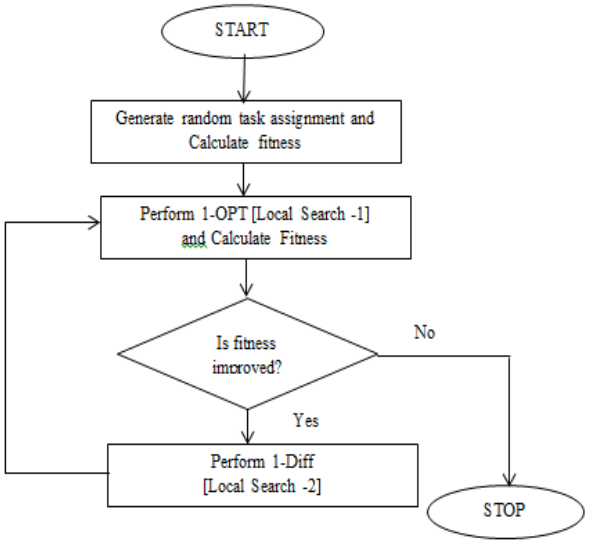


Figure 1. Flow chart of local search algorithm.

Figure 1 shows the flowchart of local search algorithm for each ant. Initially, a solution is constructed randomly and then the local search is

applied onto this solution to improve the quality of the solution. The initial solution is given as input to the 1-OPT and local search is performed [4]. The fitness of the resulting solution is calculated and compared with the previous fitness. To further improve the quality of the solution, another local search (1-DIFF) is proposed. The local search is repeated until no further improving solution is found.

When a solution is constructed, the artificial ants continue to update the pheromone trials by H-MMAS according to the Equation (8) [4]. The pheromone is updated as follows:

$$\tau(i,j) = \begin{cases} (1-\rho)\,\tau(i,j) + f(s^{best}) & \text{if}(i,j) \in s^{best} \\ (1-\rho)\,\tau(i,j) & \text{otherwise} \end{cases} \quad (8)$$

$f(s)$ is a quality function which measures the quality of the solution and is given

$$f(s) = TA(s) + \left(1 - \frac{EC(s)}{\max EC}\right) \quad (9)$$

where,

$$EC(s) = \sum_{j=1}^{m} \left(\sum_{i=1}^{n} E_{i,j} \times \Omega^{s}{}_{i,j} \right) \quad (10)$$

$$MaxEC(s) = \sum_{i=1}^{n} c_i \times \max(s^{2}{}_{i,j}) \quad (11)$$

TA(s) is equal to the number of assigned tasks in the solution *s*. $E_{i,j}$ is the energy utilized by task T_i on processor P_j; c_i is the execution cycle of task T_i ; $s_{i,j}$ is the speed at which task T_i is executed on processor P_j. *EC(s)* represents the energy consumed by the schedule (*s*). *MaxEC* represents the maximum energy consumed by the schedule. *TA(s)* reflects the distance between the solution *s* and a feasible solution. *EC(s)* is normalized by the maximum possible energy consumption *MaxEC*.

The energy consumption of each task is proportional to the square of processor speed ($E_{i,j} = c_i \times s^2_{i,j}$), whereas its computing capacity consumption is inversely proportional to the processor speed ($u_{i,j} = c_i/(s_{i,j}\ p_i)$), which means that the energy objective conflicts with the resource objective, and no single solution can optimize both of them simultaneously. To achieve a better compromise between both objectives, the energy local search algorithm and resource local search algorithm are applied at different stages of the proposed algorithm. Both the resource and energy local search procedures follow the same neighborhood structures such as 1-Opt and 1-Diff as shown in Algorithm 4. In energy local search, the quality of the solution *s* is inversely proportional to the energy consumed by all tasks in it. The proposed H-MMAS algorithm performs energy local search algorithm for every feasible solutions and resource local search algorithm for every infeasible solutions until no improvement can be made to any of them [4]. The pseudo code of proposed H-MMAS algorithm is shown in Algorithm 4.

5.3. Pseudo code of Proposed H-MMAS Algorithm

The algorithm is terminated when a pre-specified number of iterations are completed or the maximum number of iterations is met, and then the present solutions are the task assignment solutions.

Algorithm 4: Proposed H-MMAS

Input: Problem instances
Output: Set of tasks mapped to processor j

1. *Procedure: H-MMAS*
2. *{*
3. *Set parameters, Initialize pheromone trail to τ_{max} , i = 1,2… number of ants*
4. *While (termination condition not met)*
5. *{*
6. *do*
7. *{*
8. *for each ant i;*
9. *{*
10. *Construct Solution S_i under the condition U<=1;*
11. *if (S is a partial solution)*
12. *{*
13. *move ant i one step further;*
14. *}*
15. *else if (S is a feasible solution)*
16. *{*
17. *Apply energy local search algorithm to improve the solution S;*
18. *Calculate quality for each solution*
19. *}*
20. *else if (S is an infeasible solution)*
21. *{*
22. *Apply resource search algorithm;*
23. *Find unassigned task and assign it randomly with EDF bound;*
24. *Calculate quality for each solution*
25. *}*
26. *}// end else if*
27. *Calculate objective function f(s)*
28. $S_{ib} = \{S_j : f(S_j) = \max_{i=1}^{ants} (f(S_j))\}$;
29. *Choose the Ant with the best fitness value of all ants as the g_{best}*
30. *if $f(s_{ib}) > f(s_{gb})$ then $f(s_{gb}) = f(s_{ib})$;*
31. *Update pheromone trails of only the g_{best} solution*
32. *Procedure Update_Pheromone ( )*
33. *}// end for*
34. *} // end do*
35. *}//end- while*
36. *}//end – procedure*

6. Results and Discussion

To test the proposed H-MMAS algorithm, experiments are performed on an Intel core i3 Central Processing Unit (CPU) processor running at 2.27 Gigahertz speed with 1.87 GB RAM. The operating system is MS

Windows 7, 64 bit running the MATLAB R2011b environment.

6.1. Data Set Description

The utilization matrix U_{n*m} holds the real numbers in (0, 1). In the experiments, the utilization matrix is generated for considering real-time heterogeneous environment situations based on task heterogeneity, processor heterogeneity, and consistency for evaluating the performance of the proposed and existing algorithms. The utilization matrix is generated as in [4, 10]. The steps are given below:

1. A *nx1* clock cycle matrix *C* is generated, the number of cycles to execute task T_i is a random number between [100, 1000].
2. A *nx1* task frequency matrix T_B is generated, the task frequency of T_i is a random number between $[1, \Phi_T]$, here Φ_T is task heterogeneity. It is either High Task heterogeneity (HT; $[\Phi_T=100]$) or Low Task heterogeneity (LT; $[\Phi_T=5]$).
3. A *1xm* speed vector is generated for each $T_B(i)$, the speed to execute task T_i on P_j that is $S_i(j)$ to a random number between $[\Phi_T, \Phi_T.\Phi_p]$, here Φ_p is processor heterogeneity; it is either High Processor heterogeneity (HP) or Low Processor heterogeneity (LP). For High processor heterogeneity, Φp=20 and for Low Processor heterogeneity, Φ_p =5.
4. A *nxm* utilization matrix $U_{i,j}$ is generated by $T_B(i)/S_i(j)$ and $U_{i,j} \epsilon [1/(\Phi_T.\Phi_p),1]$
5. The utilization matrix is said to be consistent(*C*) if each speed vector values are sorted by descending with processor P_0 which is always the fastest and processor $P_{(m-1)}$ as the slowest. This implies that a particular processor always runs at same speed for the entire task (i.e., Processor speed doesn't depend on task characteristics). But the inconsistent matrix (*IC*) holds unsorted speed vector values that are random state athey were generated. This implies that a particular processor runs at different speed for different the task (i.e., Processor speed depends on task characteristics).

6.2. Performance Comparison of the Proposed H-MMAS Algorithm with Existing Algorithms Based on Task Heterogeneity, Processor Heterogeneity and Consistency.

In this section, the performance of proposed H-MMAS algorithm is compared with MMAS, Modified BPSO and ACO algorithms [12]. 135 problem instances are generated with each utilization matrix having 15 problem instances for the side by side comparison of the four algorithms.

The four algorithms are run 100 times for each problem instance and the mean and standard deviation of the task assigned are reported. The parameter setting for the H-MMAS algorithm is shown in Table 2.

Table 2. The parameter setting for the proposed H-MMAS algorithm.

Parameters	Value
Number of Ants	10
Number of iterations	250
Initial population	100
ρ (pheromone evaporation rate)	0.02

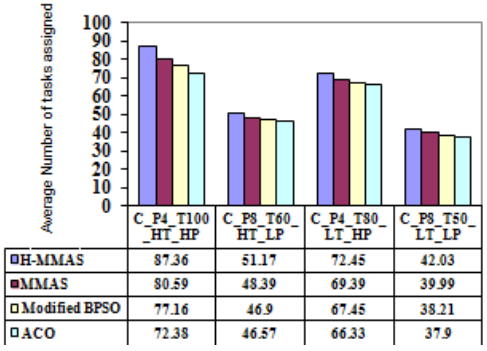
Table 3 shows the comparison of the results obtained by the H-MMAS, MMAS, and Modified BPSO and ACO algorithms. It is observed that the proposed H-MMAS algorithm outperforms the MMAS, Modified BPSO and ACO algorithms in terms of total number of the task assigned.

From Figure 2 and Figure 3, it can be inferred that the number of the tasks assigned by H-MMAS is more compared to MMAS, Modified BPSO and ACO algorithms for consistency and inconsistency utilization matrix. The proposed H-MMAS assigned more number of the tasks than all other algorithms.

Table 3. Comparison of the results obtained by the different algorithms considered for consistency matrix and inconsistency matrix.

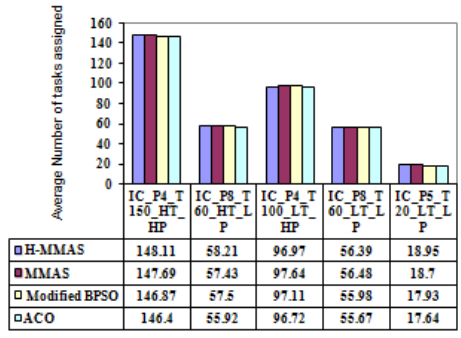
		Proposed H-MMAS			MMAS			Modified BPSO			ACO		
Problem set	**Size**	**Best**	**Mean (μ)**	**Std deviation (σ)**	**Best**	**Mean (μ)**	**Std deviation (σ)**	**Best**	**Mean (μ)**	**Std deviation (σ)**	**Best**	**Mean (μ)**	**Std deviation (σ)**
C_HT_HP	U4*100	88	87.36	1.66	82	80.59	1.72	80	77.16	1.90	76	72.38	1.92
C_HT_LP	U8*60	53	51.17	1.49	51	48.39	1.68	49	46.90	1.72	49	46.57	1.70
C_LT_HP	U4*80	74	72.45	1.08	73	69.39	1.61	71	67.45	1.82	68	66.33	1.84
C_LT_LP	U8*50	43	42.03	1.38	41	39.99	1.42	41	38.21	1.48	40	37.90	1.44
IC_HT_HP	U5*150	150	148.11	1.37	150	147.69	1.64	149	146.87	2.11	150	146.4	2.33
IC_HT_LP	U8*60	60	58.21	1.38	60	57.43	1.62	59	57.5	1.71	59	55.92	1.89
IC_LT_HP	U4*100	99	96.97	1.41	100	97.64	1.68	98	97.11	1.94	100	96.72	2.42
IC_LT_LP	U8*60	58	56.39	1.14	58	56.48	1.26	58	55.98	1.34	58	55.67	1.72
IC_LT_LP	U5*20	20	18.95	0.82	20	18.7	1.12	20	17.93	1.53	20	17.64	1.62

This is because, H-MMAS exploits more strongly the best solutions found during the search and to direct the ants' search towards very high quality solutions and avoids the premature convergence of the ants' search by limiting the pheromone trail strengths between τ_{max} and τ_{min}. The existing Modified BPSO algorithm [12] requires a trained input and more number of iterations for convergence. The proposed H-MMAS algorithm does not need any trained input and proficient of training on itself. So it has a fast convergence rate since it requires less number of iterations.



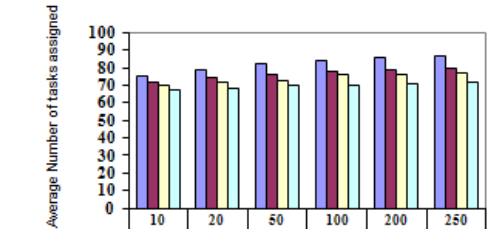
	C_P4_T100_HT_HP	C_P8_T60_HT_LP	C_P4_T80_LT_HP	C_P8_T50_LT_LP
H-MMAS	87.36	51.17	72.45	42.03
MMAS	80.59	48.39	69.39	39.99
Modified BPSO	77.16	46.9	67.45	38.21
ACO	72.38	46.57	66.33	37.9

Figure 2. Comparison of average number of the task assigned for four algorithms of consistency matrix.



	IC_P4_T150_HT_HP	IC_P8_T60_HT_LP	IC_P4_T100_LT_HP	IC_P8_T60_LT_LP	IC_P8_T20_LT_LP
H-MMAS	148.11	58.21	96.97	56.39	18.95
MMAS	147.69	57.43	97.64	56.48	18.7
Modified BPSO	146.87	57.5	97.11	55.98	17.93
ACO	146.4	55.92	96.72	55.67	17.64

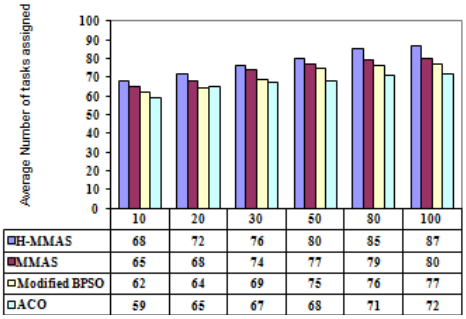
Figure 3. Comparison of average number of the task assigned for four algorithms of inconsistency matrix.



	10	20	50	100	200	250
H-MMAS	75	79	82	84	86	87
MMAS	72	74	76	78	79	80
Modified BPSO	70	72	73	76	76	77
ACO	67	68	70	70	71	72

Figure 4. Comparison of average number of the task assigned for varying number iterations of C_100T_4P_HT_HP utilization matrix.

Figure 4 gives a comparative analysis of the total number of tasks assigned for varying number of iterations for four algorithms and the initial population size is set to 100. Figure 5 gives a comparative analysis of the total number of tasks assigned for varying number of populations for four algorithms and the maximum number iteration is set to 250. From Figure 5 it can be inferred that the H-MMAS outperforms all other existing algorithms because of the balance in the exploitation and exploration of the search space. Thus the task assigned of 87 is achieved for H-MMAS which is comparatively better than the other existing algorithms.



	10	20	30	50	80	100
H-MMAS	68	72	76	80	85	87
MMAS	65	68	74	77	79	80
Modified BPSO	62	64	69	75	76	77
ACO	59	65	67	68	71	72

Figure 5. Comparison of average number of the task assigned for varying population sizes of C_100T_4P_HT_HP utilization matrix.

Table 4. Average normalized energy consumption for four algorithms considering the consistency and inconsistency matrix.

Problem set	Size	Proposed H-MMAS	MMAS	Modified BPSO	ACO
C_HT_HP	U4*100	0.3488	0.4145	0.4324	0.4818
C_HT_LP	U8*60	0.454	0.47	0.5289	0.5437
C_LT_HP	U4*80	0.3284	0.3684	0.394	0.4297
C_LT_LP	U8*50	0.4538	0.471	0.5125	0.5164
IC_HT_HP	U5*150	0.0218	0.0255	0.03144	0.0598
IC_HT_LP	U8*60	0.198	0.2413	0.4067	0.1845
IC_LT_HP	U4*100	0.0375	0.0452	0.0529	0.0427
IC_LT_LP	U8*60	0.1223	0.136	0.1484	0.1286
IC_LT_LP	U5*20	0.2842	0.4189	0.4390	0.4652

The test results from the experiment for average normalized energy consumption for four algorithms considering the consistency and inconsistency matrix are shown in Table 4. From the Table 4, it can be observed that the proposed algorithm has reduced average normalized energy consumption than MMAS, Modified BPSO and ACO for both consistent and inconsistent matrix. The local search yields a better result since it tries to achieve minimized utilization by removing the task from one processor and assigning into the other processor and thereby helping to add more number of tasks. The result obtained is further enhanced by the presence of 1-DIFF local search, and results in minimized average normalized energy consumption.

The execution time of the proposed H-MMAS algorithm takes slightly more time to solve each problem instance compared to ACO and MMAS algorithms. The reason is that the proposed algorithm includes the two local search techniques which enriched MMAS performance by improving the quality of task assignment solutions, but required more execution time to solve each problem instance. H-MMAS takes more time than MMAS algorithm because the heuristic information in MMAS is replaced

by the two exhaustive local search which takes more time.

Table 5. Comparison of Average CPU time (secs) by four algoithm considering the consistency and inconsistency utilization matrix.

Problem set	Size	Proposed H-MMAS	MMAS	Modified BPSO	ACO
C_HT_HP	U4*100	2.2368	1.945	2.886	1.346
C_HT_LP	U8*60	1.99113	1.872	7.781	1.965
C_LT_HP	U4*80	1.8205	1.39	18.146	0.9
C_LT_LP	U8*50	1.639	1.537	10.596	1.379
IC_HT_HP	U5*150	3.535	3.244	21	3.19
IC_HT_LP	U8*60	2.137	1.925	0.429	1.705
IC_LT_HP	U4*100	1.894	1.659	1.962	1.105
IC_LT_LP	U8*60	2.063	1.73	14.609	1.482
IC_LT_LP	U5*20	0.472	0.442	0.187	0.44

H-MMAS requires less execution time compared to Modified BPSO because the H-MMAS requires less number of iterations than Modified BPSO to get the best solution. The results of the comparison are shown in Table 5.

Table 6 shows the Best Quality of the solution for H-MMAS, MMAS, Modified BPSO and ACO algorithms and their ranks. As it can be seen that the quality of the solution of H-MMAS is higher than MMAS, Modified BPSO and ACO algorithms. H-MMAS outperforms Modified BPSO and ACO algorithms due to the searching behavior of the ants enriched with local search algorithm. In case of inconsistency matrix a maximum number of task are assigned similar to Modified BPSO, MMAS and ACO algorithms. The inclusion of local search has its influence in energy part of the solution. From Table 6, it can be observed that the proposed H-MMAS has proved to be the best algorithm for the task assignment optimization problem in the heterogeneous multiprocessor system.

Table 6. Best quality of solution for H-MMAS, MMAS, modified BPSO and ACO and their ranks.

Problem set	Size	Proposed H-MMAS	MMAS	Modified BPSO	ACO	Rank of the solution			
						H-MMAS	MMAS	Modified BPSO	ACO
C_HT_HP	U4*100	88.3948	82.4233	80.4541	75.5418	1	2	3	4
C_HT_LP	U8*60	53.4937	51.5124	49.5371	48.561	1	2	3	4
C_LT_HP	U4*80	74.3561	73.3872	72.4197	68.4553	1	2	3	4
C_LT_LP	U8*50	43.471	41.482	41.53	40.53	1	2	3	4
IC_HT_HP	U5*150	150.0248	150.0298	149.0343	150.0749	1	2	3	4
IC_HT_LP	U8*60	60.2413	60.2815	59.4796	60.1955	1	2	4	3
IC_LT_HP	U4*100	99.0478	100.0491	98.0576	98.0493	1	2	4	3
IC_LT_LP	U8*60	58.1291	58.1308	58.1572	57.1311	1	2	4	3
IC_LT_LP	U5*20	20.397	20.4375	20.4567	20.4841	1	2	3	4

7. Conclusions

In this paper, a Hybrid Metaheuristic algorithm (H-MMAS) is proposed for solving real-time task assignment problem in the heterogeneous multiprocessors. The task assignment solution in heterogeneous multiprocessors is improved using Max-Min Ant System extended with local search algorithms and its performance is compared with MMAS, Modified BPSO and ACO algorithms in terms of number of the task assigned, normalized energy consumption and average CPU time. In consistency matrix, the proposed H-MMAS algorithm has outperformed MMAS, Modified BPSO and ACO algorithms in terms of number of the tasks assigned, normalized energy consumption and average CPU time. The performance in consistency matrix is due to the fact that H-MMAS inherits the property of ACO and MMAS which makes it to run quickly and arrive at an optimal solution. The added feature of two local search ability in H-MMAS makes it to assign more tasks than Modified BPSO, ACO and MMAS algorithms. In case of inconsistency matrix, H-MMAS performs better than Modified BPSO in terms of task assigned, normalized energy consumption, and CPU time. But H-MMAS performs similar to ACO and MMAS in terms of number of the tasks assigned, since variation in speed matrix for each task influences the local search to behave like a normal MMAS, a variant of ACO. Also there is a reduction in normalized energy consumption than MMAS and ACO because of two local search techniques. The average CPU time of the proposed algorithm for inconsistency matrix is slightly more than ACO and MMAS algorithm because H-MMAS algorithm includes local search techniques and less than Modified BPSO algorithm because the proposed algorithm requires less number of iterations to get the best quality of solution.

In our future work, we will work on reducing total execution time and we will investigate the possibility of our algorithm to assign tasks from a task set in which the tasks have precedence constraints and inter-task communication.

References

[1] Babaeizadeh S., Banitalebi A., Ahmad R., and Aziz M., “Solving Optimal Control Problem

Using Max-Min Ant System," *IOSR Journal of Mathematics*, vol. 1, no. 3, pp. 47-51, 2012.

[2] Baruah S., "Partitioning Real-Time Tasks Among Heterogeneous Multiprocessors," *in Proceedings of the IEEE International Conference on Parallel Processing*, Montreal, pp. 467-474, 2004.

[3] Braun T., Siegel H., Beck N., Bölöni L., Maheswaran M., Reuther A., Robertsong J., Theys M., Yao B., Hensgen D., and Freund R., "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing System," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810-837, 2001.

[4] Chen H., Cheng A., and Kuo Y., "Assigning Real-Time Tasks to Heterogeneous Processors by Applying Ant Colony Optimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 1, pp.132-142, 2011.

[5] Dorigo M. and Stützle T., *Ant Colony Optimization*, MIT Press, 2004.

[6] Garey M. and Johnson D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co, 1979.

[7] Jin H., Wang H., Wang H., and Dai G., "An ACO-Based Approach for Task Assignment and Scheduling of Multiprocessor Control Systems," *in Proceedings of International Conference on Theory and Applications of Models of Computation*, Beijing, pp. 138-147, 2006.

[8] Krishna C. and Shink K., *Real-Time System*, McGraw-Hill, 1997.

[9] Narayan V. and Subbarayan G.," An Optimal Feature Subset Selection Using GA for Leaf Classification," *The International Arab Journal of Information Technology*, vol. 11, no. 5, pp. 447-451, 2014.

[10] Poongothai M., "ARM Embedded Web Server Based on DAC System," *in Proceedings of the International Conference on Process Automation, Control and Computing*, Coimbatore, pp. 1-5, 2011.

[11] Poongothai M., Rajeswari A., and Kanishkan V., "A Heuristic Based Real Time Task Assignment Algorithm for the Heterogeneous Multiprocessors," *IEICE Electronic Express*, vol. 11, no. 3, pp. 1-9, 2014.

[12] Prescilla K. and Selvakumar A., "Modified Binary Particle Swarm Optimization Algorithm Application to Real-Time Task Assignment in Heterogeneous Multiprocessor," *Microprocessors and Microsystems*, vol. 37, no. 6-7, pp. 583-589, 2013.

[13] Srikanth G., Maheswari V., Shanthi P., and Siromoney A., "Tasks Scheduling Using Ant Colony Optimization," *Journal of Computer Science*, vol. 8 , no. 8, pp. 1314-1320, 2012.

[14] Stutzle T. and Hoos H., "MAX-MIN Ant System and Local Search for the Traveling Salesman Problem," *in Proceedings of the IEEE International Conference on Evolutionary Computation*, Indianapolis, pp. 309-314,1997.

[15] Wu J., Liu X., Shu J., Li Y., and Liu K., "Independent Task Assignment of Space Warfare Based on MAS and ACO," *Journal of Information and Computational Science*, vol. 10, no. 12, pp. 3861-3867, 2013.

Poongothai Marimuthu is currently an Assistant Professor (Senior Grade) in the Department of Electronics and Communication Engineering, Coimbatore Institute of Technology, Coimbatore 641014 India. Her research areas includes Scheduling in Real-time systems, energy efficient computing systems, low power design and power management of energy harvesting real-time embedded system.

Rajeswari Arumugam is currently a Professor and Head of Department of Electronics and Communication Engineering, Coimbatore Institute of Technology, Coimbatore 641014 India. Her areas of interest include wireless communication, signal processing.

Jabar Ali is currently doing his M.E. (Communication Engineering) in Department of Electronics and Communication Engineering, Coimbatore Institute of Technology, Coimbatore 641014, India. He completed his B.E. in Electronics and Communication Engineering in Mepco Schlenk Engineering College, Sivakasi, India. His areas of interest include Scheduling in real-time embedded systems and computer networks.