


# Task Scheduling for Heterogeneous Systems Using a Hybrid Deep Neural Network and Genetic Algorithm Approach

Yutao Han <sup>a</sup>

*Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada*

**Keywords:** Heterogeneous System, Task Scheduling, Deep Neural Network, Genetic Algorithm.

**Abstract:** Task scheduling in heterogeneous computing systems is a highly complex and challenging problem due to the diverse architectures and varying computational capabilities of different hardware resources. Efficiently allocating tasks to these resources to optimize performance is a significant challenge in such environments. This study addresses this challenge by combining the deep neural network with the genetic algorithm to create an efficient task scheduling approach. The research focuses on constructing a deep neural network that progressively learns from the task scheduling schemes generated by the genetic algorithm, aiming to accelerate the scheduling process. The method involves using the genetic algorithm to generate initial scheduling solutions and training a Deep Neural Networks (DNN) to learn from these solutions. The results show that it is difficult for the network to fully reproduce the performance of genetic algorithm-based scheduling, but the network significantly reduces the time required to generate effective scheduling plans. This hybrid model highlights the potential of leveraging machine learning techniques to enhance the efficiency of task scheduling in heterogeneous computing systems.


## 1 INTRODUCTION

With the rise of big data, machine learning, and real-time analytics, the need for higher performance and specialized processing capabilities has become more urgent. Traditional CPU-based architectures struggle with the diverse and intensive workloads these fields demand (Lee et al., 2010). Heterogeneous computing systems have emerged to solve these challenges by integrating different types of processors, such as GPUs, FPGAs, and specialized accelerators, into a single system. This approach utilizes the unique strengths of each processor to significantly improve performance and efficiency. Since each processor has its own execution logic and advantages, an appropriate task scheduling approach among the different processors becomes critical in heterogeneous systems to achieve maximum utilization.

A proper scheduling plan should assign tasks to different processors/machines effectively which can minimize the total processing time of all jobs. Due to the dependencies of tasks in the real world, scheduling has been proven to be an NP-complete

problem (Ullman, 1975). According to the huge solution space, complex constraints and multi-objective optimization characteristics, the genetic algorithm (GA) is suitable for scheduling problems.

Researchers have developed updates for different procedures of the basic genetic algorithm. The most intuitive approach to improve is having more reasonable population initialization and adaptive parameters for mutation/crossing to increase the convergence speed (Fang et al., 2020). HGAAP involves a heuristic algorithm like HEFT to generate good initial solutions as the population based on the earliest finish time (Ding et al., 2017). Another approach is replicating individuals with high fitness to the next generation to increase the probability of keeping outstanding genes (Cheng & Xu, 2020). Besides GA, the list scheduling algorithm is also famous for heterogeneous computing scheduling. PSLS further expands the basic list algorithm with a downward length table (DLT), which is used to measure the time difference between a task being affected by its subsequent tasks when it is executed on any processor (Zhao et al., 2019). The selection of processors will depend on the DLT, allowing this

<sup>a</sup> <https://orcid.org/0009-0006-1970-5731>

algorithm to consider the global effect when assigning tasks. An algorithm also enhances task prioritization by using an improved weight that considers execution time differences across processors and incorporates communication costs. Then selecting processors by a randomized decision mechanism of balancing local and global optimizations (AlEbrahim & Ahmad, 2017).

Although the above methods are effective and efficient, generalization is a big defect for them. The optimal solutions yielded by applying algorithms on a specific structure of computing tasks are not helpful for other different structures. All of them need to be re-evaluated from the beginning. To improve the generalization, the artificial neural network (ANN) based scheduler has been introduced. Based on the statistics of CPU instructions and threads as parameters, a lightweight single-layer ANN can be constructed to predict which processors different tasks should be assigned to (Gupta et al., 2020). While ANNs are promising in making quick decisions based on learned patterns, their training from scratch can be computationally expensive and challenging.

This research proposes a hybrid scheduling approach that combines the deep neural network (DNN) with GA to leverage the strengths of both methods. The DNN is trained from scratch initially. In the early stages, GA plays a critical role in optimizing task scheduling, and the solutions are then used to train the DNN. The DNN gradually learns to predict efficient scheduling strategies based on GA's feedback. Over time as the DNN becomes more proficient, the scheduler will more depend on the DNN to improve the system's overall efficiency and capability of generalization. The training process of DNN is similar to supervised learning where is target is produced by GA. This approach also does not require a pre-train network.

The main objective of this research is to develop a robust and efficient task scheduling framework that can adapt to the dynamic nature of heterogeneous computing environments. By combining the rapid decision-making capabilities of DNNs with the optimization power of GA, this approach aims to achieve a balance between computational efficiency and scheduling accuracy. This study will evaluate the proposed method against traditional scheduling algorithms to demonstrate its effectiveness in handling complex, dependent tasks within a heterogeneous computing system.

## 2 METHODOLOGY

This methodology addresses task scheduling in heterogeneous systems using a combination of DNN and GA. The task scheduling problem is modelled as a Directed Acyclic Graph (DAG), with tasks assigned to processors based on computation and communication costs. The DNN predicts task allocations through learning from the GA's solution, while the GA optimizes scheduling plans to minimize execution time.

### 2.1 Scheduling Problem Modelling

The task scheduling problem is often modelled as a Directed Acyclic Graph (DAG), which is used broadly in research (Ding et al., 2017; Cheng & Xu, 2020; Heydari & Shahrhoseini, 2011; Zhao et al., 2019). Each node in DAG represents a task and the arrowed edges between nodes indicate the dependencies of tasks. There are communication costs for dependent tasks, shown as edge weight in the graph. The acyclic nature of the graph ensures that no state is revisited, thus reflecting the progressive nature of scheduling tasks. When running a task scheduler on a heterogeneous system consisting of multiple processors, different tasks will have different performances on different. This also needs to be considered.

Due to the above, a heterogeneous system could be presented by  $P$ , a set of  $p$  processors where each one is denoted as  $p_i$ . A task scheduling problem could be defined as  $G = (T, E)$ .  $T$  is the set of tasks needed to execute and each task is denoted as  $t_i$ . There will be a list of length  $p$  encoded in each node, representing the computation cost of the task  $t_i$  on each processor  $p_j$ .  $E$  is the set of edges including weights. Each edge from node  $i$  to node  $j$  is denoted as  $e_{ij}$  and its value is the communication cost from task  $t_i$  to  $t_j$  if these two tasks are not executing on the same processor. The paper assumes the number of tasks is constant, but the structure of the DAG could be any. Figure 1 gives an example instance of a scheduling problem of six tasks on a heterogeneous system of three processors.

Based on the task scheduling problem, a scheduling plan could be defined as assigning each task a processor to execute. The execution order follows the topological order to satisfy the dependencies between tasks.

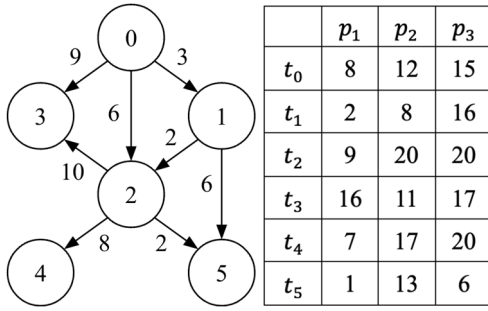


Figure 1: A DAG example with communication costs and the computation costs for each task on each processor. (Picture credit : Original)

## 2.2 Deep Neural Network

In this research, the DNN architecture is inspired by the encoder-decoder structure with graph attention network (GAT) and multi-head attention (Lee et al., 2021). Figure 2 shows the overview of the structure. The feature extractor layer of the proposed DNN will receive the computation cost of all tasks on all processors and the communication cost among tasks. For tasks that are not the child task of the current one, simply set the communication cost to 0. There will be affine operations on two types of costs, concatenating the result gives the initial task representation vectors. The encoder then leverages multiple GAT layers to encode the task relationships into each task vector. The final vector representation will be the sum of the normal GAT result and inversed graph GAT result to have a better task feature embedding from the given DAG structure.

The allocation predictor of the DNN will receive the features of each task from the extractor and use

the attention mechanism to predict the allocation. For each task, it combines embeddings from previous and subsequent tasks, along with the last task's embedding, into a context vector. This context vector is used to query the embeddings via multi-head attention. Then the attention output of each task is passed through the corresponding fully connected layer to produce logits. With Softmax, the decoder will finally produce processor allocation probability for each task as the final output of the network. A scheduling plan could be derived by taking the argmax of probabilities as the allocated processor for each task.

Since the output is probabilities, the scheduling problem could be treated as a classification problem. Based on task features, the DNN will predict which processor (class) should a task be allocated to. The loss function is the sum of Cross Entropy Loss of each task prediction. Moreover, to ensure the property of processors (classes) remain the same, for each input DAG, the ID numbers of processors are always sorted by ascending computation costs, which means that,  $p_0$  is one with minimum computation cost,  $p_1$  is the one with second minimum computation cost, and so on.

## 2.3 Genetic Algorithm

GA is an optimization algorithm that searches for the optimal solution in the possible solution space by simulating biological evolution mechanisms. In this research, the GA is implemented as following. The algorithm first generates an initial population of random solutions, where each individual represents a possible scheduling plan.

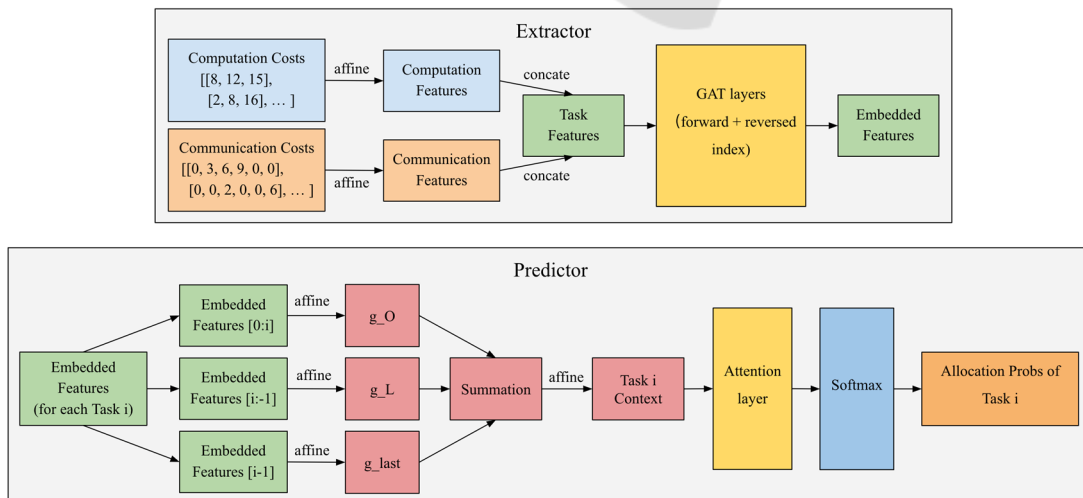


Figure 2: Structure of the DNN to predict processor allocation probability of each task. (Picture credit : Original)

After the population initialization, tournament is used for selection. Each tournament will randomly choose three individuals and only keep the one with best fitness. Since the goal is finding the best

scheduling plan, this research uses the Makespan as the fitness of each plan, which could be calculated with the following formulas:

$$MakeSpan = \text{Max}_p(\text{ExecuteTime}(p)) \quad (1)$$

$$\text{ExecuteTime}(p) = \text{Max}_{t \in p}(\text{EndTime}(t)) \quad (2)$$

$$\text{StartTime}(t) = \text{Max}_{u \in \text{parent}(t)}(\text{EndTime}(u) + \text{Comm}(u, t)) \quad (3)$$

$$\text{EndTime}(t) = \text{StartTime}(t) + \text{Cost}(t) \quad (4)$$

$$\text{Comm}(u, t) = \begin{cases} \text{Comm}(u, t), & u, t \text{ are executed on a same } p \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where  $p$  represents the processor,  $t \in p$  are tasks that allocated to processor  $p$ ,  $\text{Cost}(t)$  is the computation cost of task  $t$  on the processor it belongs to, and  $\text{Comm}(u, t)$  is the communication cost between task  $u$  and its child task  $t$ . Notes that the communication costs only occur when these two tasks are executed on different processors. These formulas show that, Makespan of a scheduling plan is the greatest execution time of the tasks among all processors.

In the next step, the algorithm will sequentially choose two individuals from the selection above to do crossing and mutation to generate offspring. For crossing, two individuals will swap cells after a randomly selected index. Since the cells are always ordered in tasks' topological order, therefore the new offspring will not contain cells with duplicate tasks or cells with missing tasks. For mutation, the algorithm will randomly allocate a new processor to the task in each cell of every individual in the population.

The algorithm also takes advantages from the method of saving elite individuals (Ding et al., 2017). The elite group consists of the top 10% of individuals with the best fitness in the population. After offspring generation, the new population will be created by concatenating the elite group and all offspring, and the best individual in the new population will be recorded. After running for a specified number of generations, the algorithm will return the individual with the best fitness during this period.

## 2.4 Combination of DNN and GA

At the earlier stage, the scheduling model provided in this paper mainly depends on GA with long generation time to product qualified schedule plan. As the DNN's loss and the Makespan difference between DNN and GA's scheduling plan converge, the model

can start to use DNN for inference. Depending on the performance of DNN's inferred result, the model can gradually decrease the number of generations or even eliminate the application of GA to accelerate the entire scheduling process.

## 3 RESULT

### 3.1 Performance of DNN

This research runs through some experiments to check if the designed network is able to converge and learn from the GA's scheduling plan. There are three groups are tested: (20, 3), (50, 5), and (100, 10), where the first number of each group is the number of tasks, and the second one is the number of processors. 1500 DAGs are generated randomly for each test group for training, with range of computation cost being 1 to 20 and range of communication cost being 1 to 10. There will be only one DAG input for each epoch to simulate the training process while GA doing task scheduling in the real world. The population size is set to 200 for all testing groups, and the generations are set as 200, 350, 500 iterations for tasks numbers of 20, 50, and 100 respectively.

Figure 3, 4, 5, 6, 7, and 8 demonstrate the loss during training process and the Makespan difference between DNN and GA's scheduling plan for each test group. It could be observed that the performance of the DNN remains the same after around 1000 epochs' training. Since GA is a heuristic method that explores solutions through stochastic search, the generated targets lack a consistent structure or coherent scheduling rules, which causes the instability of the DNN's performance. This is the biggest limitation of the model proposed in this research. Despite the



fluctuations observed in both the training loss and makespan difference, the overall downward trend demonstrates that the network can capture some useful patterns from the input data and learn effective scheduling strategies to a certain extent.

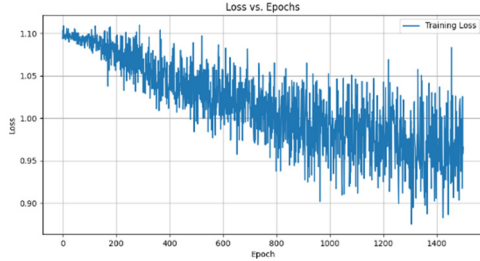


Figure 3: Training loss of test group (20, 3).  
(Picture credit : Original)

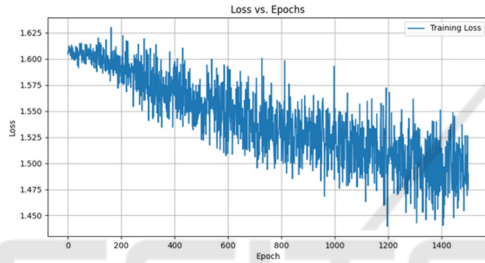


Figure 4: Training loss of test group (50, 5).  
(Picture credit : Original)

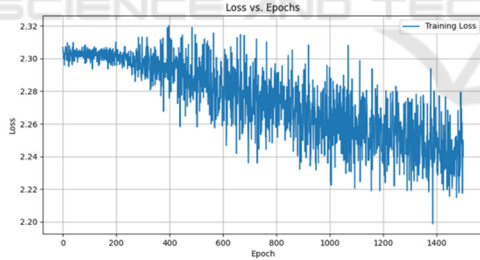


Figure 5: Training loss of test group (100, 10).  
(Picture credit : Original)

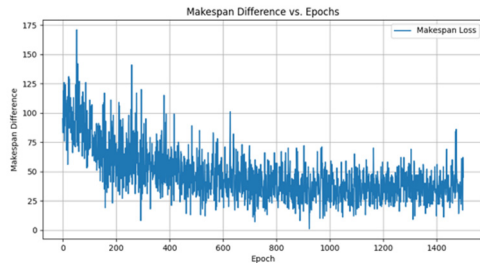


Figure 6: Makespan difference between DNN and GA's scheduling plan during training of test group (20, 3).  
(Picture credit : Original)

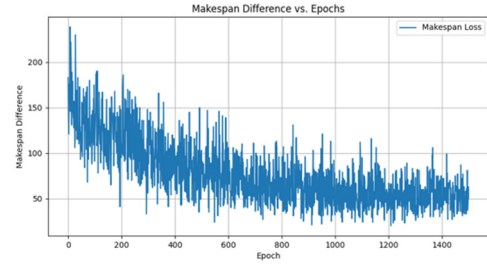


Figure 7: Makespan difference between DNN and GA's scheduling plan during training of test group (50, 5).  
(Picture credit : Original)

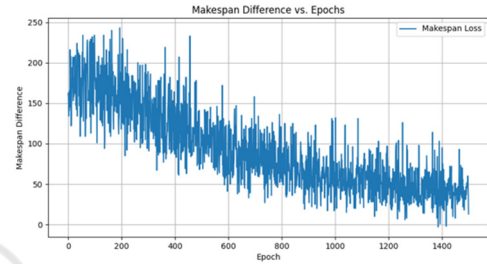


Figure 8: Makespan difference between DNN and GA's scheduling plan during training of test group (100, 10).  
(Picture credit : Original)

1000 DAGs are also randomly generated for each task group for evaluation. Figure 9, 10, and 11 shows the evaluation result. Combined with Figure 6, 7, and 8, it could be derived that test groups (20, 3) and (100, 10) have smaller Makespan difference and better performance. Group (20, 3) has the minimum number of tasks, which reduces the complexity of the scheduling problem. Suboptimal scheduling plans do not result in significant Makespan differences. For the group (100, 10), while the problem complexity increases due to the larger number of tasks, the availability of more processors does increase in trainable parameters which allow the network to capture more patterns.

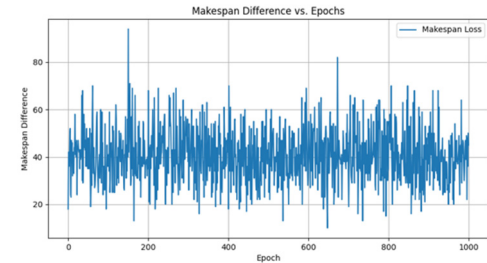


Figure 9: Makespan difference between DNN and GA's scheduling plan during evaluation of test group (20, 3).  
(Picture credit : Original)

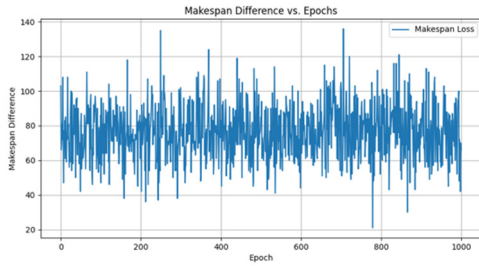


Figure 10: Makespan difference between DNN and GA's scheduling plan during evaluation of test group (50, 5). (Picture credit : Original)

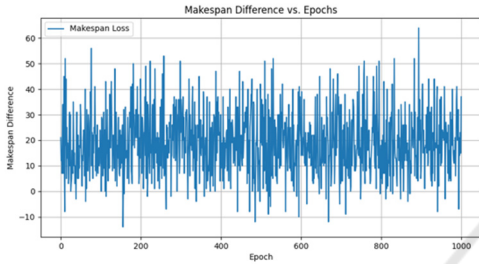


Figure 11: Makespan difference between DNN and GA's scheduling plan during evaluation of test group (100, 10). (Picture credit : Original)

### 3.2 Acceleration of Using DNN

Overall, the scheduling plans generated by the DNN still show a great Makespan difference compared to the GA-generated optimal solutions. Therefore, the model still needs to rely on GA to do further optimization. However, since the DNN's output already has relatively good fitness, the entire scheduling process could be accelerated by using the DNN's output to create the initial population and reducing the number of generations for GA.

This research also runs experiments to compare the performance of hybrid and pure GA models. For the hybrid model, after obtaining outputs from DNN, half of the initial population used in GA is generated by mutating DNN's output, and the second half is randomly generated. Then running GA based on this initial population with half the number of generations. Table 1 shows the time consumption between these two models. Figure 12, 13, and 14 presents the Makespan difference between these two models.

Even though GA uses only half the number of generations in the hybrid model, the resulting scheduling plans have a minimal Makespan difference compared to those generated by the pure GA. The reduction in generations leads to a significant speedup. The hybrid model is nearly twice as fast as the pure GA, with the DNN's inference time

being almost negligible. This highlights the efficiency of using the DNN as a pre-processing step to guide GA, ultimately reducing the computational burden while maintaining competitive performance in scheduling quality.

Table 1: Time consumption between the hybrid and pure GA model. (Table credit: Original)

	Hybrid	Pure GA	Epochs
Test group (20, 3)	662.366 seconds	1292.870 seconds	1000
Test group (50, 5)	5125.380 seconds	10274.914 seconds	1000
Test group (100, 10)	11312.046 seconds	22949.441 seconds	500

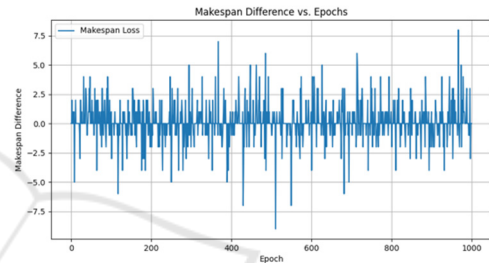


Figure 12: Makespan difference between the hybrid and pure GA's scheduling plan of test group (20, 3). (Picture credit : Original)

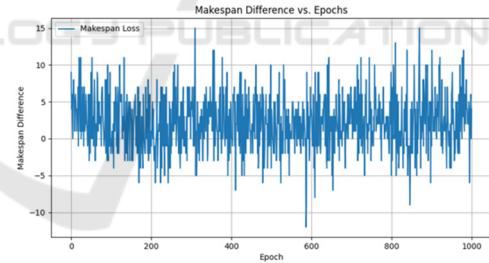


Figure 13: Makespan difference between the hybrid and pure GA's scheduling plan of test group (50, 5). (Picture credit : Original)

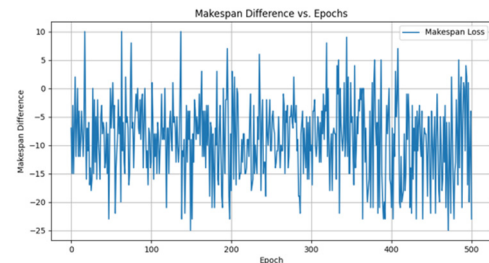


Figure 14: Makespan difference between the hybrid and pure GA's scheduling plan of test group (100, 10). (Picture credit : Original)

## 4 CONCLUSION

This study explores the integration of DNN with GA for scheduling tasks represented by DAG on multiple processors. The DNN is trained to predict the probability of task allocation on processors based on task computation and communication costs. GA-generated optimal solutions are the targets. The experiment results indicate that while the DNN shows significant fluctuations during training, the overall trend demonstrates its ability to learn effective scheduling patterns. Specifically, for test groups with fewer tasks or more processors, the DNN's predictions have a noticeable reduction in the makespan difference compared to the GA targets.

The inherent randomness of GA poses a challenge to the DNN's learning process. As a result, the DNN is not yet sufficient to produce optimal scheduling solutions only. The model utilizes DNN's output to initialize GA's population. This approach reduced the number of GA generations needed to produce a scheduling plan while maintaining minimal makespan differences.

The impact of this research lies in the potential to significantly accelerate scheduling optimization for complex DAG tasks, reducing the reliance on purely heuristic methods. Future work could focus on improving the DNN's generalization abilities by employing reinforcement learning approaches. Additionally, further exploration into hybrid methods could lead to more scalable and efficient solutions for real-time and large-scale scheduling problems.

## REFERENCES

- AlEbrahim, S., & Ahmad, I. (2017). Task scheduling for heterogeneous computing systems. *The Journal of Supercomputing*, 73(6), 2313–2338. <https://doi.org/10.1007/s11227-016-1917-2>
- Cheng, X., & Xu, R. (2020). Research on Task Scheduling of Heterogeneous Multi-core Processor based on Replication Genetic algorithm. *Proceedings of the 4th International Conference on Intelligent Information Processing*, 454–460. <https://doi.org/10.1145/3378065.3378151>
- Deepa, R., Srinivasan, T., Doreen, D., & Miriam, H. (2006). An Efficient Task Scheduling Technique in Heterogeneous Systems Using Self-Adaptive Selection-Based Genetic Algorithm. *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, 343–348. <https://doi.org/10.1109/PARELEC.2006.14>
- Ding, S., Wu, J., Xie, G., & Zeng, G. (2017). A Hybrid Heuristic-Genetic Algorithm with Adaptive Parameters for Static Task Scheduling in Heterogeneous Computing System. 2017 IEEE Trustcom/BigDataSE/ICCESS, 761–766. <https://doi.org/10.1109/Trustcom/BigDataSE/ICCESS.2017.310>
- Fang, J., Zhang, J., Lu, S., & Zhao, H. (2020). Exploration on Task Scheduling Strategy for CPU-GPU Heterogeneous Computing System. 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 306–311. <https://doi.org/10.1109/ISVLSI49217.2020.00063>
- Gupta, M., Bhargava, L., & Indu, S. (2020). Artificial Neural Network based Task Scheduling for Heterogeneous Systems. 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), 74–79. <https://doi.org/10.1109/ICETCE48199.2020.9091745>
- Heydari, F., & Shahhoseini, H. S. (2011). Adaptive algorithm for task scheduling in the distributed heterogeneous systems using harmony search. 7th International Conference on Networked Computing, 11–16. <https://ieeexplore.ieee.org/document/6058937>
- Lee, H., Cho, S., Jang, Y., Lee, J., & Woo, H. (2021). A Global DAG Task Scheduler Using Deep Reinforcement Learning and Graph Convolution Network. *IEEE Access*, 9, 158548–158561. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2021.3130407>
- Lee, V. W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A. D., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P., Singhal, R., & Dubey, P. (2010). Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU. *Proceedings of the 37th Annual International Symposium on Computer Architecture*, 451–460. <https://doi.org/10.1145/1815961.1816021>
- Ullman, J. D. (1975). NP-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3), 384–393. [https://doi.org/10.1016/S0022-0000\(75\)80008-0](https://doi.org/10.1016/S0022-0000(75)80008-0)
- Zhao, Y., Cao, S., & Yan, L. (2019). List Scheduling Algorithm Based on Pre-Scheduling for Heterogeneous Computing. 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCLOUD/SocialCom/SustainCom), 588–595. <https://doi.org/10.1109/ISPA-BDCLOUD-SustainCom-SocialCom48970.2019.00089>