The Task Scheduling Problem: A NeuroGenetic Approach

Anurag Agarwal, University of South Florida, Sarasota, USA
Selcuk Colak, Cukurova University, Adana, Turkey
Jason Deane, Virginia Tech, USA
Terry Rakes, Virginia Tech, USA

ABSTRACT

This paper addresses the task scheduling problem which involves minimizing the makespan in scheduling n tasks on m machines (resources) where the tasks follow a precedence relation and preemption is not allowed. The machines (resources) are all identical and a task needs only one machine for processing. Like most scheduling problems, this one is NP-hard in nature, making it difficult to find exact solutions for larger problems in reasonable computational time. Heuristic and metaheuristic approaches are therefore needed to solve this type of problem.

This paper proposes a metaheuristic approach - called NeuroGenetic - which is a combination of an augmented neural network and a genetic algorithm. The augmented neural network approach is itself a hybrid of a heuristic approach and a neural network approach. The NeuroGenetic approach is tested against some popular test problems from the literature, and the results indicate that the NeuroGenetic approach performs significantly better than either the augmented neural network or the genetic algorithms alone.

Keywords: Scheduling; Genetic Algorithms; Augmented Neural Networks; NeuroGenetic Approach

INTRODUCTION

The task scheduling problem involves scheduling *n* tasks of a project where they follow a given precedence relationship. Each task requires one unit of a resource for a certain duration (processing time), which is task dependent. The resource could be a machine, labor, CPU, etc. The project faces a resource constraint because only a fixed number of units of the resource are available. It is assumed that preemption is not allowed. Once a task begins, it must reach completion. The objective is to minimize the makespan or the project duration. The task scheduling problem occurs in a variety of situations, ranging from project management to distributed computing environments; in fact, it is at the heart of many scheduling problems and has received considerable attention in the literature. For example, Hu (1961), Coffman and Graham (1972), Graham et al. (1979), Kasahara and Narita (1984), and Agarwal et al. (2003, 2006) have addressed this problem.

The task scheduling problem can be mathematically stated as follows:

Minimize S_{n+1}

Subject to:

$$S_j - S_i \geq d_i \quad i, j \in N \text{ and } (j,i) \in A \tag{1}$$
$$|P(t)| \leq R, \quad t = 1, \ldots, T \tag{2}$$
$$S_i \geq 0, \quad i \in N \tag{3}$$

where *n* represents the number of tasks in the project, *N* represents the set of all tasks, d_i represents the duration or the processing time of task *i*, S_i represents the start time of task *i*, *R* represents the number of units of the resource

available, and A represents the set of arcs or precedence relationships for the project. $P(t)$ represents the set of tasks that are active at time unit t and T represents an upper bound on the project makespan. $|P(t)|$ represents the cardinality of the set $P(t)$ which is the number of active tasks at time t. In the above mathematical formulation, the objective function states that the start time of the $(n+1)^{st}$ task should be minimized. Because the start time of the $(n+1)^{st}$ task is the same as the finish time of the n^{th} or last task in the sequence, this is equivalent to minimizing the makespan of the problem. Constraint set (1) enforces the precedence relationship and processing time requirements. It says that if a task i precedes a task j, then the difference in the start times of tasks j and i cannot be less than the processing time (or duration) of task i. Constraint set (2) enforces the resource constraints. It says that the total number of tasks that can be active at a given time cannot exceed the total units of resources available. Lastly, constraint set (3) is the non-negativity constraint.

Like most scheduling problems, the task scheduling problem is NP-Hard in nature (Kasahara and Narita, 1984) and therefore exact methods for finding the optimal solution are impractical for solving larger instances of the problem. Heuristics and metaheuristics are needed to solve such problems to obtain a near-optimal solution in reasonable computational time. Several heuristics have been proposed for this problem. See Cooper (1976) and Panwalker and Iskander (1977) for a review of heuristics commonly used for similar scheduling problems. For this particular task scheduling problem, the heuristics used most, commonly due to their effectiveness, are: 1) *Highest Level with Estimated Time First* (HLETF), 2) *Critical Path with Most Immediate Successors First* (CP/MISF), 3) *Longest Processing Time First* (LPTF), and 4) *Earliest Finish Time First* (EFTF). The HLETF and CP/MISF heuristics were first proposed in Kasahara and Narita (1984). All these heuristics were used in Agarwal et al. (2003). In addition to these greedy heuristics, Agarwal et al. (2006) proposed some non-greedy heuristics in which some tasks ready to start were made to wait even if resources were available, saving the resources for other more critical tasks that might become ready to start soon.

A number of metaheuristic approaches have also been used for solving this type of problem. Adams et al. (1988) were among the first to propose an iterative (or multi-pass) procedure for solving certain types of scheduling problems. For example, they proposed a shifting bottleneck procedure for the job-shop scheduling problem. Hopfield and Tank (1985) proposed a neural network approach for solving combinatorial optimization problems. Foo and Takefuji (1988) used Hopfield and Tank's neural network approach to solve small job-shop scheduling problems. Agarwal et al. (2003) proposed the augmented neural network (AugNN) approach for solving the task scheduling problem. The AugNN approach is a hybrid of both heuristic and iterative neural network approaches. The AugNN approach is essentially a non-deterministic local search approach in which the initial solution is obtained using a well-known heuristic and then the neighboring solution space is searched iteratively using the principles of neural networks. Agarwal et al. (2006) further improved upon the results of the task scheduling problem by incorporating a non-greedy heuristic approach within the AugNN framework. Genetic algorithms have also been used for similar scheduling problems (Alcaraz and Maroto, 2001, Valls et al., 2008). The genetic algorithm approach is more of a global search approach.

In this paper, a relatively new metaheuristic approach - called NeuroGenetic - is applied to the task scheduling problem. This approach, proposed by Agarwal et al. (2010), is a hybrid of both the AugNN and the genetic algorithm (GA) approaches. It has been applied to the resource constrained project scheduling problem (Agarwal et al., 2011) but has not been applied to the task scheduling problem. The NeuroGenetic approach is applied to the same set of problems used in Agarwal et al. (2003) and Agarwal et al. (2006) and shows the effectiveness of using this approach compared to using the AugNN or a GA approach alone. This relatively new approach is found to be more effective than either the AugNN or the GA approach alone.

The NeuroGenetic Approach

As stated earlier, the NeuroGenetic approach is a hybrid of both the AugNN and GA approaches. In this approach, the AugNN iterations are interleaved with GA iterations. The idea behind interleaving the iterations is to take advantage of the strengths of each technique – the AugNN approach is regarded as a good local search technique while the GA approach is regarded as a good global search technique. Interleaving their iterations provides a better solution than either of these techniques used alone. If AugNN is used by itself, for example, the initial solution is provided by a heuristic and then AugNN iterations search for solutions in the neighborhood of this

initial solution. Therefore, search in the AugNN approach is limited to a single neighborhood. When using genetic algorithms, several good solutions in different neighborhoods are generated as a set of initial solutions and through crossover and mutation, several other solutions are created in various neighborhoods. By interleaving AugNN iterations, the local neighborhoods of several good solutions are basically searched, thus expanding the search space and therefore affording a better chance of finding improved solutions.

Interleaving the AugNN approach with the GA approach is not straightforward because in the AugNN approach, a set of weights of processing elements, in conjunction with a heuristic, determines a solution. The set of weights is modified after each iteration, giving rise to a new solution in subsequent iterations. If AugNN is used by itself, it starts with an initial set of weights and proceeds sequentially from iteration to iteration, generating new solutions with a new set of weights, where the set of weights in iteration t+1 depends on the set of weights in iteration t. The challenge in interleaving the AugNN approach is to take a solution that did not result from a given set of weights and a heuristic and find a subsequent solution in its local neighborhood. This requires some reverse engineering. That is, given a solution and a heuristic, a set of weights has to first be determined that would give the solution (see Agarwal et al., 2010, for details). This set of weights, generated through a reverse engineering process, is then modified so that a new solution in the neighborhood may be generated. Agarwal et al. (2003) developed the AugNN approach that works in conjunction with any greedy heuristics. In this approach, there is one set of weights. Using the reverse engineering approach, it is possible to develop a set of weights that would generate a given solution using a given greedy heuristic. Agarwal et al. (2006) developed a modified AugNN approach that can work in conjunction with a greedy and a non-greedy heuristic. In this approach, there are two sets of weights; one set works with a greedy heuristic and the other works with a non-greedy heuristic. Using the reverse engineering procedure needed in the NeuroGenetic approach to generate these two sets of weights is not possible, which means that the NeuroGenetic approach is limited to utilizing only the greedy heuristic.

Because reverse engineering to generate two sets of weights is not possible, in this paper the NeuroGenetic approach is applied only in conjunction with greedy heuristics and not with non-greedy heuristics. However, the results will be compared with those obtained by both the greedy heuristics and non-greedy heuristics.

The Problems

The task scheduling problem can be explained with the help of an instance of a seven-task problem shown in Figure 1. In this problem, N is 7, R is 2 (because there are two units of machines). The processing times d1, d2, …, d7 are written next to each task. For this problem instance, the precedence relationship is given by the set of arcs A whose elements include (2,1), (3,1), (4,1), (5,2), (6,2), (5,3), (6,3), (7,5), (7,6), (7,4).

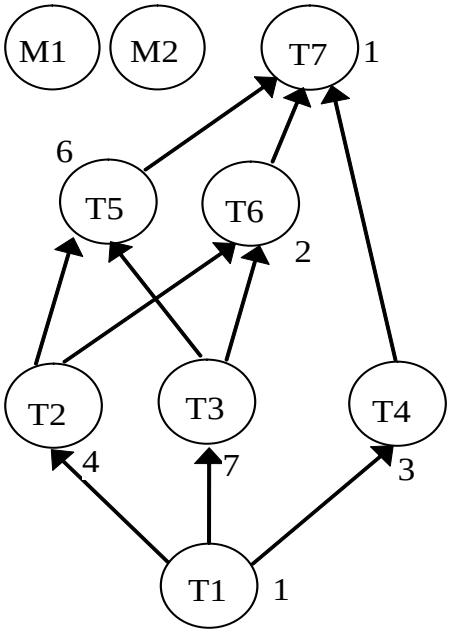

Figure 1: An Instance of the Task Scheduling Problem

In Agarwal et al. (2003), a set of 570 problems was generated to test the effectiveness of the AugNN approach compared to a single-pass heuristic approach. The number of tasks used in these problems ranged from 10 through 100 and the units of resources ranged from 2 through 5. The processing times were generated randomly using a uniform distribution. It was shown that the reduction in gap from the lower bound due to AugNN compared to single-pass heuristic ranged from 39% to 59% for various heuristics. AugNN performed best in conjunction with the HLETF heuristic. The reduction in gap for the HLETF heuristic was 42.5%. It may be noted that if the heuristic is already very good, there is less room for improvement by AugNN. For some heuristics, even though the reduction in the gap was as high as 59%, the net makespans were not the best because the initial solutions were not very good to begin with; in other words, the local neighborhood was not the best for a local search. The least gap from the lower bound due to AugNN was 3.3% using the HLETF heuristic. This dataset will be used for testing purposes and it will be called Uniform-03.

In Agarwal et al. (2006), three new sets of 344 problems each were generated. Again, the number of tasks ranged from 10 through 100 and the number of units of resources ranged from 2 through 5. In the first set, the processing times were generated using a uniform distribution. In the second set, the processing times were generated using a normal distribution. In the third set, they were generated using an exponential distribution. The percent gaps due to AugNN for the three datasets were 1.98%, 2.07% and 1.79%, respectively. The best results were in conjunction with the HLETF greedy heuristic and a non-greedy heuristic. In this study, the authors use these three datasets as well – called Uniform-06, Normal-06 and Exponential-06 - to test the NeuroGenetic approach. In all, four datasets will be used to test the effectiveness of the NeuroGenetic approach.

Computational Experience

The NeuroGenetic approach was coded in Visual Basic.Net (2010) and applied to the four datasets discussed earlier. The best single-pass heuristic for the task scheduling problem in the literature is the Highest Level with Estimated Time First (HLETF). In this study, only this best heuristic is used in conjunction with the NeuroGenetic Approach. Table 1 shows the results of the AugNN alone using the greedy heuristic, the AugNN alone with the greedy and the non-greedy heuristic, and the GA approach alone, in addition to the NeuroGenetic approach. The table shows the results for the four datasets (Uniform-03, Uniform-06, Normal-06 and Exponential-06) individually and also a total of these four datasets for the various approaches. The total or aggregate lower bound (assuming infinite resources) for all the problems for each dataset, the aggregate makespan of all the problems in each dataset for the various approaches, the gaps from the lower bound, and the percent gaps from the

lower bound are also reported. The HLETF heuristic produced solutions with an overall gap of 3.51% from the lower bound. The AugNN approach, used in conjunction with the HLETF heuristic, reduced the gap from 3.51% to 2.21%. Of course, this reduction came at a cost of some extra CPU time, as shown below. The AugNN approach, used in conjunction with the HLETF heuristic and a non-greedy heuristic, reduced the gap further down to 1.96%. Genetic algorithms, by themselves, using the HLETF heuristic to produce the first solution in the initial population, produced solutions with a gap of 1.69%. The NeuroGenetic approach produced the best solutions of any approach with an overall gap of 1.37%. With metaheuristics, one can run them arbitrarily for a very long time. Typically, some type of stopping criteria needs to be used. The stopping criteria used is to find 1,000 unique solutions or run 5,000 iterations, whichever comes first. For smaller problems, it is difficult to find 1,000 unique solutions, therefore the stopping criteria was 5,000 iterations. Larger problems are able to find 1,000 unique solutions easily. When using either the AugNN or GA approach alone or the NeuroGenetic approach, 1,000 unique solutions were produced to keep the comparison reasonable.

Table 1: Makespans, Gaps and Percent Gaps for Various Approaches for the Four Datasets

Problem Set	I (Uniform-03)	II (Uniform-06)	III (Normal-06)	IV (Exponential-06)	Total
Number of Problems	570	344	344	344	1,602
Lower Bound	87,190	142,490	56,804	149,672	436,156
Heuristic Alone (HLETF)	92,311	146,794	58,585	153,781	451,471
Augnn Alone with HLETF Heuristc	90,136	145,312	57,981	152,355	445,784
Augnn Alone with HLETF and A Non-Greedy Heuristic	89,654	145,011	57,944	152,076	444,685
GA Alone (1,000 Unique Solns)	89,267	144,856	57,735	151,652	443,510
Neurogenetic (1,000 Unique Solutions)	88,826	144,478	57,534	151,273	442,111
Gap with Heuristic Alone (HLETF Heuristic)	5,121	4,304	1,781	4,109	15,345
Gap with Augnn Alone (Greedy Heuristic Only)	2,946	2,822	1,177	2,683	9,628
Gap with Augnn Only (Greedy and Non-Greedy Heuristics)	2,464	2,521	1,140	2,404	8,529
Gap with GA Alone (1,000 Unique Solutions)	2,077	2,366	931	1,980	7,354
Gap with Neurogenetic Approach (1,000 Unique Solutions)	1,636	1,988	730	1,601	5,955
Percent Gap with Heuristic Alone (HLETF Heuristic)	5.87	3.02	3.14	2.75	3.51
Percent Gap with Augnn Alone (Greedy Heuristic Only)	3.38	1.98	2.07	1.79	2.21
Percent Gap with Augnn Alone (Greedy and Non-Greedy)	2.83	1.77	2.01	1.61	1.96
Percent Gap with GA Alone (1,000 Unique Solutions)	2.38	1.66	1.64	1.32	1.69
Percent Gap with Neurogenetic Approach (1,000 Unique Solutions)	1.88	1.40	1.29	1.07	1.37

Table 2 shows the CPU times for each dataset for each approach. In terms of CPU time consumption, the AugNN was the most time consuming. Genetic Algorithms consumed much less time than AugNN and NeuroGenetic approaches. The NeuroGenetic approach consumed slightly more time than Genetic Algorithms but significantly less than AugNN. This is because roughly 80% of iterations were performed using the GA approach

and only 20% using AugNN approach, so the CPU time was weighted higher by GA iterations, which are less time consuming than AugNN approaches. In absolute terms, the CPU times per problem were not very significant – 2.23 seconds for NeuroGenetic, 1.8 seconds for GA, and 4.15 seconds for AugNN.

Table 2: CPU Times For All Four Datasets For The Various Approaches

Problem Set	I (Uniform-03)	II (Uniform-06)	III (Normal-06)	IV (Exponential-06)	Total	Per Problem
Number of Problems	570	344	344	344	1,602	
Heuristic Alone (HLETF)	20	14	12	14	60	0.037
Augnn Alone with HLETF Heuristc	1995	1669	1557	1681	6,902	4.308
Augnn Alone with HLETF and A Non-Greedy Heuristic	1909	1614	1532	1606	6,661	4.158
GA Alone (1,000 Unique Solns)	936	652	620	678	2,886	1.801
Neurogenetic (1,000 Unique Solutions)	1125	824	798	841	3,588	2.239

A pair-wise comparison of the makespans over all 1,602 problems, and for each of the four datasets individually, showed that the NeuroGenetic approach was statistically highly significantly better than the Genetic Algorithm alone, with a p-value of < 0.001. It was also better than AugNN alone with a p-value of < 0.0001. Pair-wise comparison to compare AugNN vs. heuristic approach was also performed. The AugNN with greedy approach was better than the heuristic approach with a p-value of < 0.0001. AugNN using greedy and non-greedy heuristics was better than AugNN with greedy alone with a p-value of < 0.01. The GA was better than AugNN with greedy and non-greedy with a p-value of < 0.01. These results clearly demonstrate the effectiveness of the NeuroGenetic approach over other approaches for this type of scheduling problem.

Table 3: P-Values of Pairwise Comparison of Various Approaches

Problem Set	I (Uniform-03)	II (Uniform-06)	III (Normal-06)	IV (Exponential-06)	Total
Number of Problems	570	344	344	344	1,602
AugNN with HLETF heuristic Vs. Heuristic alone (HLETF)	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001
AugNN with HLETF and Non-greedy Heuristic Vs. AugNN with HLETF	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
GA Alone Vs.AugNN with HLETF and A Non-greedy Heuristic	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
NeuroGenetic Vs. GA	< 0.001	< 0.001	< 0.001	< 0.001	< 0.001
NeuroGenetic vs. AugNN	< 0.0001	< 0.0001	< 0.0001	< 0.0001	< 0.0001

SUMMARY AND CONCLUSIONS

In this paper, the NeuroGenetic approach is applied to the well-known task scheduling problem. The NeuroGenetic approach is a hybrid of both the Augmented Neural Network (AugNN) and Genetic Algorithm (GA) approaches. The AugNN approach is itself a hybrid of a heuristic approach and a neural network approach. Both AugNN and GA approaches are iterative approaches. In the NeuroGenetic approach, the AugNN iterations are interleaved with GA iterations. Since the AugNN approach performs a local search and the GA approach performs a global search, the NeuroGenetic approach provides the best of both worlds. With just slightly more CPU time than the GA approach and much less CPU time than the AugNN approach, it provides improved solution quality over both the AugNN and the GA approach alone. These approaches are tested on four different datasets which combined represent over 1,600 problems from the literature, ranging in size from 10 tasks to 100 tasks. The algorithms were coded for these approaches in Visual Basic.Net 2010. For each of the datasets and for all datasets

combined, the results showed that the NeuroGenetic approach performed significantly better than either the AugNN or the GA approach alone.

AUTHOR INFORMATION

Anurag Agarwal is a Professor in the Department of Information Systems and Decision Sciences at University of South Florida, Sarasota, Florida, USA. He earned his Ph.D. from The Ohio State University, Columbus, Ohio, USA (1993) and an MBA from the University of Wisconsin, La Crosse, Wisconsin, USA (1988). He teaches a variety of courses in Information Systems and Statistics and Operations Management, both at the undergraduate and graduate levels. His primary research interests are in heuristics and metaheuristics for various optimization problems.

Selcuk Colak is an Associate Professor in the Department of Business at the Cukurova University, Adana, Turkey. His current research interests are heuristics and metaheuristics, genetic algorithms, neural networks, project and machine scheduling, and distribution planning. He teaches Operations Management, Project Management and Logistics Management courses at both undergraduate and graduate levels. Professor Colak received his B.S. degree in Electrical and Electronics Engineering from the Cukurova University, Adana, Turkey, in 1997 and received his M.S. in Electrical and Computer Engineering and his Ph.D. in Information Systems and Operations Management from the University of Florida, USA, in 2000 and 2006, respectively.

Jason Deane is an Associate Professor in the Department of Business Information Technology in the Pamplin College of Business at Virginia Polytechnic Institute & State University, USA. He received his Ph.D. in Information Systems and Operations Management from the University of Florida, USA, and an M.B.A. and B.S. in Business Administration from Virginia Tech, USA. His current research interests are in the areas of supply chain management, artificial intelligence, computer-aided decision support systems, information system security, large-scale optimization and information retrieval. His teaching interests are in Operations Management and Information Systems.

Terry R. Rakes is William and Alix Houchens Professor of Information Technology at Virginia Tech. He received his Ph.D. in Management Science, M.B.A., and B.S.I.E. from Virginia Tech. His research interests are in analytics and big data analysis, text and data mining, geographic information systems, disaster planning and logistics, information security, and the application of decision support and artificial intelligence methodologies. He has published in *Management Science*, *Decision Sciences*, *Decision Support Systems*, *Annals of Operations Research*, *OMEGA*, *European Journal of OR*, *Operations Research Letters*, *Information and Management*, *Journal of Information Science*, and others.

REFERENCES

1. Alcaraz J. & Maroto C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102, 83–109.
2. Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedures for job shop scheduling. *Management Science,* 34 (3), 391-401.
3. Agarwal, A., Jacob, V.S., & Pirkul, H. (2003). Augmented neural networks for task scheduling. *European Journal of Operational Research*, 151, 481-502.
4. Agarwal, A., Jacob, V., & Pirkul, H. (2006). An Improved Augmented Neural-Networks Approach for Scheduling Problems. *INFORMS Journal on Computing*, 18(1), 119-128.
5. Agarwal, A., Colak, S., & Deane, J. (2010). NeuroGenetic Approach for Combinatorial Optimization: An Exploratory Analysis. *Annals of Operations Research*, 74(1), 185-199.
6. Agarwal, A., Colak, S., & Erenguc, S.S. (2011). A Neurogenetic Approach for the Resource-Constrained Project Scheduling Problem. *Computers and Operations Research*, 38, 44-50.
7. Coffman, E.G. & Graham, R.L. (2000). Optimal scheduling for two-processor systems. *Acta Informatica*, 1, 200-213.
8. Cooper D. F. (1976). Heuristics for scheduling resource–constrained projects: An experimental

investigation. *Management Science*, 22, 1186–1194.

9. Foo, Y.P.S. & Takefuji, Y. (1988). Stochastic neural networks for solving job-shop scheduling: Part 1, problem representation. *Proceedings of Joint International Conference on Neural Networks*, 2, 275-282.
10. Graham, R.L., Lawler, E.L., Lenstra, J.K., & Rinnooy Kan, A.H.G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5, 287-326.
11. Hopfield, J.J. & Tank, D.W. (1985). Neural computation of decisions in optimization problems, *Biological Cybernetics*, 52, 141-152.
12. Hu, T.C. (1961). Parallel sequencing and assembly line problem. *Operations Research*, 9, 841-848.
13. Kasahara, H. & Narita, S. (1984). Practical multiprocessor scheduling algorithms for efficient parallel processing. *IEEE Transactions on Computers*, *C-33*, 11, 1023-1029.
14. Panwalker, S.S. & Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, 25, 45-61.
15. Valls V., Ballestin F., & Quintanilla M. S. (2008). A hybrid genetic algorithm for the resource constrained project scheduling problem. *European Journal of Operational Research*, 185, 495-508.