Journal of Engineering and Applied Sciences 3 (7): 532-537, 2008
ISSN: 1816-949X

Utility of Meta-Heuristics for Solving Scheduling Problems

[1]E.O. Oyetunji and [2]A.E. Oluleye
[1]Department of Applied Mathematics and Computer Science,
University for Development Studies, Ghana
[2]Department of Industrial and Production Engineering, University of Ibadan, Ibadan, Nigeria

Abstract: Meta-heuristics, are increasingly being applied by many researchers to solve scheduling problems. In this study, we discuss 5 unique characteristics of meta-heuristics that have endeared them to researchers. These are: ability to escape local optima, obtain better solution quality, solve larger instances of problems, suitable for multi-objective scheduling problems and have wide applicability.

Key words: Scheduling, heuristic, meta-heuristic, optimization, local optima

INTRODUCTION

Scheduling deals with the problem of allocating resources (machines) over time to perform a number of tasks (jobs) with the aim of minimizing cost or maximizing profit. The problem is essentially that of decision-making. However, because of the complexity of the cost function, secondary criteria (performance measures), which correlate strongly with costs are usually utilized in evaluating the performance of solution methods.

Scheduling problems are combinatorial in nature (French, 1982; Lung and Roert, 1993). As an example, for the n jobs, m machines (n×m) problem, the number of possible schedules is $(n!)^m$. This may look tractable when n and m have small values. However, as the values of n and m increase, the problem becomes complex. It is computationally difficult to enumerate all the possible schedules and select the best based on some objective (performance) measures in good time.

There are basically, 2 methods of solving scheduling problems (Ehrgott and Grandibleux, 2000). These are: Exact and approximation methods. Exact methods yield optimal solutions (e.g., total enumeration method, Hungarian method, Johnson's method for 2-machine sequencing, implicit enumeration method such as branch and bound or dynamic programming methods).

The approximation method, on the other hand, involves the use of heuristic algorithms. These usually involve the use of intuitive approaches or rule of thumb. The Heuristic methods are techniques for obtaining acceptable solutions to scheduling problems at a reasonable computational cost. While, they do not always guarantee optimal results, the techniques are relatively economical in terms of computational resources utilized.

Since, real-life problems can have larger values of n and m, the difficulty in solving scheduling problems by total enumeration (exact method) becomes obvious. The computational complexities associated with the factorial nature of the possible schedules, however, prohibit a practical solution to large sized problems. It is evident from, the above that scheduling problem (s), especially when n and m are large, can take a whole life span to solve.

As a result of the difficulties and computational complexities involved in obtaining optimal solutions to scheduling problems (even the simplest form of the single machine case), research efforts have been directed towards the development of heuristic algorithms. Generally, heuristics have the problem of being trapped in a local optimal solution. This has led to the emergence of meta-heuristics. Given that meta-heuristics attempt to avoid being trapped in local optima, researchers apply them to solve scheduling problems more and more. Meta-heuristics provide a way of considerably improving the performance of simple heuristic procedures. They have been developed to solve complex optimization problems in many areas, with combinatorial optimization being one of the most fruitful (Laguna, 2002; Puchinger and Raidl, 2005).

In this study, 5 common types of meta-heuristics are described. Also, discussed are 5 unique characteristics of meta-heuristics which has endeared them to researchers.

Corresponding Author: E.O. Oyetunji, Department of Applied Mathematics and Computer Science,
University for Development Studies, Ghana

META-HEURISTICS

Meta-heuristics contains all heuristics methods that show evidence of achieving good quality solutions for the problem of interest within an acceptable time. Meta-heuristics (also written as metaheuristics) are heuristic methods for solving a very general class of computational problems by combining traditional heuristics methods in a hopefully efficient way. A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optima.

Heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule (Laguna, 2002; Sevaux, 2003; Bonissone, 2003). The word meta means beyond or of higher level. Generally, metaheuristics are applied to problems for which there is no satisfactory problem-specific algorithm or heuristic or when it is not practical to implement such a method.

Types of meta-heuristics: Meta-heuristics can be classified into 2 classes; population-based methods and point-to-point methods (Osman and Laporte, 1996). In the latter methods, the search invokes only 1 solution at the end of each iteration from which the search will start in the next iteration. On the other hand, the population-based methods invoke a set of many solutions at the end of each iteration. Most commonly used metaheuristics are Genetic Algorithm (GA), Memetic Algorithm (MA), which are examples of population based meta-heuristics, Tabu Search (TS), Simulated Annealing (SA) and Ant Colony Optimization (ACO), which are examples of point-to-point meta-heuristics.

Genetic Algorithm (GA): A Genetic Algorithm (GA) is a procedure that tries to mimic the genetic evolution of a species. GA simulates the biological processes that allow the consecutive generations in a population to adapt to their environment (Yaohua and Chi-Wai, 2007). The adaptation process is mainly applied through genetic inheritance from parents to children and through survival of the fittest. Therefore, GA is a population-based meta-heuristic. GA starts with an initial population whose elements are called chromosomes. The chromosome consists of a fixed number of variables which are called genes. In order to evaluate and rank chromosomes in a population, a fitness function based on the objective function is defined. Three operators (selection, crossover and mutation operators) are specified to construct the complete structure of the GA procedure. The selection operator has the responsibility of selecting an intermediate population from the current one in order to be used by the other operators (crossover and mutation). In this selection process, chromosomes with higher fitness function values have a greater chance to be chosen than those with lower fitness function values. Pairs of parents in the intermediate population of the current generation are probabilistically chosen to be mated in order to reproduce the offspring. In order to increase the variability structure, the mutation operator is applied to alter 1 or more genes of a probabilistically chosen chromosome. Finally, another type of selection mechanism is applied to copy the survival members from the current generation to the next one.

Standard genetic algorithm

Step 1 initialization: Generate an initial population P_0. Set the crossover and mutation probabilities p_c (0, 1) and p_m (0, 1), respectively. Set the generation counter t = 1.

Step 2 selection: Evaluate the fitness function F at all chromosomes in P_t. Select an intermediate population P_t' from the current population P_t.

Step 3 crossover: Associate a random number from (0, 1) with each chromosome in P_t' and add this chromosome to the parents pool set SP_t if the associated number is less than p_c. Repeat the following Steps 4 and 5 until all parents in SP_t are mated.

Step 4: Choose 2 parents p_1 and p_2 from SP_t. Mate p_1 and p_2 to reproduce children c_1 and c_2.

Step 5: Update the children pool set SC_t through $SC_t = SC_t \cup \{c_1, c_2\}$ and update SP_t through $Sp_t = SP_t - \{p_1, p_2\}$.

Step 6 mutation: Associate a random number from (0, 1) with each gene in each chromosome in P_t', mutate this gene if the associated number is less than p_m and add the mutated chromosome only to the children pool set SC_t.

Step 7 stopping conditions: If stopping conditions are satisfied, then terminate. Otherwise, select the next generation P_{t+1} from $P_t \cup SC_t$. Set SC_t to be empty, set t = t + 1 and go to Step 2.

Memetic Algorithm (MA): Algorithms with closer a analogy to cultural evolution than to biological evolution are Called Memetic Algorithms (MA). MA is an example of a population-based meta-heuristic. Basically, MA combines local search heuristics with crossover

operators. They are inspired by models of adaptation in natural systems that combine evolutionary adaptation of populations of individuals with individual learning within a lifetime (Fleury *et al*., 2005). Under different contexts and situations, MAs are also known as hybrid evolutionary algorithms (HEA).

MA is based on a population of agents and has proved to be of practical success in a variety of problem domains. They constitute one of the most successful approaches for combinatorial optimization in general and for the approximate solution of NP-Hard Optimization problems in particular (Yavuz *et al*., 2006). MAs are concerned with exploiting all available knowledge about the problem under study. They also combine global and local search by using an Evolutionary Algorithm (EA) to perform exploration, while, the local search method performs exploitation.

Standard memetic algorithm (in pseudocode)

```
Begin
 initialize population;
 for each individual do local-search individual;
  repeat
   for individual =1 to # crossovers do
     select two parent individual1, individual2\in
     population randomly;
     individual3: = crossover(individual1,individual2);
     individual3: = local-search(individual3);
     add individual i3 to population;
   end for;
   for individual=1 to # mutations do
     select an individual of population randomly;
     individual {m}: = mutate(individual);
     individual {m}: = local-search(individual {m});
     add individual {m} to population;
   end for;
   population: = select(population);
   if population converged then
     for each individual of best populations
       do individual: = local-search (mutate (individual);
      end for
    end if
  until terminate = true;
 end for
end
```

Simulated Annealing (SA): Simulated Annealing (SA) is a variant of meta-heuristics which successively generates a trial point in a neighborhood of the current solution and determines whether or not the current solution is replaced by the trial point based on a probability depending on the difference between their function values (Ku and Karimi, 1991). Convergence to an optimal solution can theoretically be guaranteed only after an infinite number of iterations controlled by the procedure called cooling schedule. The main control parameter in the cooling schedule is the temperature parameter T. The role of T is to let the probability of accepting a new move be close to 1 in the earlier stage of the search and to let it be almost zero in the final stage of the search.

Standard simulated annealing algorithm

Step 1 initialization: Choose an initial solution x_0 and fix the cooling schedule parameters; initial temperature T_{max}, epoch length M, cooling reduction ratio λ (0, 1) and minimum temperature T_{min}. Set the temperature $T = T_{max}$ and $k = 0$.

Step 2 epoch loop: Repeat the following steps (3-5) M times.

Step 3: Generate a trial point y_k in the neighborhood of the current solution x_k.

Step 4: Evaluate f on the trial point y_k and compute $p = 1$, if $f(y_k) < f(x_k)$; or $p = \exp(-Df/T)$, otherwise, where $Df = f(y_k)-f(x_k)$.

Step 5: Choose a random number u from (0, 1). If $p = u$, set $x_{k+1} = y_k$. Otherwise, set $x_{k+1} = x_k$. Set $k = k + 1$.

Step 6 termination condition: If the cooling schedule is completed ($T = T_{min}$), terminate. Otherwise, decrease the temperature by setting $T = \lambda T$ and go to step 2.

Tabu Search (TS): Tabu Search (TS) is a meta-heuristic method. The main feature of TS is its use of an adaptive memory and responsive exploration. A simple TS combines a local search procedure with anti-cycling memory-based rules to prevent the search from getting trapped in local minima. TS restrict returning to recently visited solutions by constructing a list of recently visited solutions called Tabu List (TL). In each iteration, TS generates many trial solutions in a neighborhood of the current solution. The trial solutions generation process is composed to avoid generating any trial solution that is already recently visited. The best trial solution found among the generated solutions will become the next solution. Therefore, TS can accept uphill movements to avoid getting trapped in local minima. TS can be terminated if the number of iterations without any improvement exceeds a predetermined maximum number.

Standard tabu search algorithm:

Step 1: Choose an initial solution x_0. Set the Tabu List (TL) to be empty and set the counter k = 0.

Step 2: Generate neighborhood moves list $M(x_k) = \{x': x'; N(x_k)\}$, based on tabu restrictions, where $N(x_k)$ is a neighborhood of x_k.

Step 3: Set x_{k+1} equal to the best trial solution in $M(x_k)$ and update TL.

Step 4: If stopping conditions are satisfied, terminate. Otherwise, go to step 2.

Ant Colony Optimization (ACO): Real ants have developed an efficient way of finding the shortest path from a food source to their nest without using visual information. While, searching for food, ants deposit pheromone on the ground and follow, in high probability, pheromone previously deposited by other ants. Assuming a single food source, more than one way to reach the source and initially equal probability for an ant to choose a path, more ants will visit the shortest path on average and therefore, pheromone accumulates faster if they walk with approximately the same speed. If new ants arrive at a point where they have to decide on one or another path they prefer to choose the shorter path with higher probability. This in turn increases the pheromone on the shortest path such that after a, while all ants will choose the shortest path (Auer *et al*., 1999).

The Ant Colony Optimization (ACO) is inspired by the behavior of real ants. In the ACO algorithm, the solution to a combinatorial optimization problem is constructed by agents (ants), which choose the values for the decision variables constituting a feasible solution. Each choice is (in analogy to real ants) a probabilistic choice proportionate to a global variable representing the amount of pheromone. ACO is one of the most successful techniques of the wider field of swarm intelligence. Since, the first ACO algorithms proposed about 15 years ago, there have been many significant contributions on algorithmic variants, challenging application problems and theoretical foundations (Maniezzo *et al*., 2004). These have established ACO as a mature, high-performing meta-heuristic for the solution of difficult optimization problems.

Standard ant colony optimization algorithm (in pseudocode):

```
Choose the number N of ants;
Set Nt: = number of keys;
Set Nc: = number of combination character sets;
Set l: = length of a text string;
Initialize the matrix of pheromones P[Nt][Nc];
Repeat
  Create N text strings S[N][l];
  Create N empty keyboards sets T[N][Nt];
  For I: = 1 to l do
    For k: = 1 to N do
      c: = S[k][i];
      If letter c was not already treated by ant k then
        Choose a position p for c;
        T[k][p]: = c;
        Evaporate P[p][c];
      EndIf;
    EndFor;
  EndFor;
  For all ants k: = 1 to N do
    Evaluate the result for the ant k;
  EndFor;
  Choose the best results;
  Update P[ ][ ];
  Verify Min-Max of P[ ][ ], modify if necessary;
Until satisfying result.
```

Unique characteristics of meta-heuristics: Meta-heuristics are perhaps the most exciting development in approximate optimization techniques of the last 2 decades. They have had widespread successes in attacking a variety of difficult combinatorial optimization problems that arise in many practical areas (Knowles and Corne, 2000; Laguna, 2002). This explains, why there has been widespread adoption of metaheuristics for solving many instances of scheduling problems. But what are the factors responsible for this level of successes and increasing adoption of metaheuristics? Why have they become so useful to researchers working on combinatorial problems? A review shows 5 unique characteristics of metaheuristics.

Meta-heuristics avoids being trapped in local optima: Because of the high computational resources required by exact methods in obtaining solutions to many scheduling problems, researchers prefer the use of approximation methods such as heuristics methods. Although, heuristic methods obtains solution to scheduling problems within a reasonable time, while utilizing less computational resources, sometimes the answers they give are absurdly bad. Generally, heuristics are myopic because they construct schedules based on limited or local information without considering the consequences of implementing those schedules. In an attempt to improve the solution to a problem, a number of heuristics employ some local search strategies to search the neighbors of the initial

solution. In many cases, with a good start and a good exchange rule, the near optimal (local optimal) solution you get is good enough. However, in most cases one may be stuck with a bad one, to escape local optimality may be difficult. This is one of the major drawbacks of heuristic methods. This has led to the emergence of meta-heuristics.

Meta-heuristics mainly invoke exploration and exploitation search procedures in order to diversify the search all over the solution space and intensify the search in some promising areas. Therefore, meta-heuristics are not easily entrapped in local minima. Meta-heuristic methods begin with one or more initial solutions (called population) and employ intelligent search strategies that try to avoid local optima. While, the search for better solutions in the commonly used heuristics often terminates at a local minimum due to their greedy nature, metaheuristics can climb hills (i.e., accept moves that generate solutions of highercost than the present one) and thus be able to dig itself out of a local minimum to search for better minima (Ku and Karimi, 1991).

Meta-heuristics obtain better solution quality: As against the traditional heuristic methods, meta-heuristics has the capacity and ability to produce better and improved solutions. There are 2 factors that are responsible for this. These are initial solution and the move mechanism of the meta-heuristics. Many meta-heuristics generate initial solutions that are population-based (i.e., the starting solution consists of a number of schedules). They also employed population-based strategies to manipulate a collection of solutions rather than a single solution at each stage. Constructing new solutions from either an initial one or previous ones by exploring the neighbors of the old solutions is called the move mechanism. Most meta-heuristic methods also utilize multiple heuristics to generate new population members. This incorporation of multiple heuristics for generating trial solutions, as opposed to relying on a single rule, helps meta-heuristics to obtain solutions of higher quality than their traditional heuristic counterparts.

Meta-heuristics can solve larger instances of problems: Another major characteristic of meta-heuristics is their ability to solve large instances of scheduling problems. Many meta-heuristics use elegant termination conditions such as maximum time available, maximum number of iteration etc. in their exploration strategy. This is achieved by setting a time limit or number of iterations within which the meta-heuristics can carry out extensive search of the solution space. One major drawback of traditional heuristics is that as the problem size grows, the quality of solution obtained becomes poor. But giving a time or iteration limit to meta-heuristics to explore and exploit the search space allows them to still obtain good solutions for larger instances of scheduling problems.

Some researchers who have applied meta-heuristics to scheduling problems corroborated this by submitting that Meta-heuristic methods can provide near-optimal solutions within moderate or acceptable computing time. Therefore, meta-heuristic methods have been found to be more suitable for large-size problems (He and Hui, 2007; Schittekat *et al*., 2007). Their experimental result showed that the proposed meta-heuristic (GA method) found much better solutions than the Random Search (RS) and Mixed-Integer Linear Programming (MILP) model for the large-sized instances. Also, Ku and Karimi (1991) reported that the proposed SA method appeared very attractive as a method for solving large-scale scheduling problems in batch processes.

Meta-heuristics are suitable for multi-objective scheduling problems: The purpose of multi-objective scheduling is different from that of single objective scheduling: In the latter, the goal is to find the best solution, which is the schedule that minimizes (or maximizes) the objective function. In contrast, in multi-objective scheduling there is no single solution that minimizes (or maximizes) all the objective functions at once. Indeed, the objective functions often conflict, as a schedule that decreases one objective will increase another. There are, usually more than one solution (schedule) to multi-objective scheduling problems (Nagar *et al*., 1995).

In recent years, meta-heuristics have been applied more and more to multi-objective problems. Undoubtedly, they are well qualified to tackle problems of a great variety. This asset, coupled with the possession of a population, seems to make them particularly attractive for use in multi-objective problems, where a number of solutions approximating the Pareto front are required (Knowles and Corne, 2000). Some theoretical justification for the use of evolutionary algorithms in multi-objective optimization, in the form of convergence proofs, has been provided by Rudolph (1998a, b). Also, there has been a growing research effort in the use of meta-heuristics within the field of Multiple Criteria Decision Making (MCDM), a branch of operations research (Knowles and Corne, 2000).

Meta-heuristics have wide applicability: Most traditional heuristics are designed for a given problem; hence, they are limited in scope of application. The ability of the meta-heuristics to extensively explore the search space and

exploit solutions that have been found makes them applicable to very wide classes of scheduling problems.

CONCLUSION

Developments in the areas of meta-heuristics in recent times are enormous and on the increase. Many researchers in the area of production scheduling are applying meta-heuristics to many scheduling problems that have been classified as NP-hard. In this study, we have highlighted 5 unique characteristics that have made meta-heuristics the bride of researchers working on combinatorial optimization problems. This study serves as an eye opener for researchers to explore meta-heuristics for scheduling problems. It is also, the desire of the authors to embark on a further research on (development of) meta-heuristics for many scheduling problems that are still classified as NP-Hard.

REFERENCES

Auer, A.B., B. Bullnheimer, R.F. Hartl and C. Strauss, 1999. Applying Ant Colony Optimization to solve the Single Machine Total Tardiness Problem, Report No. 42.

Bonissone, P., 2003. Soft computing and meta-heuristics: Using knowledge and reasoning to control search and vice-versa. In: Proc. SPIE. Applications and Science of Neural Networks. Fuzzy Systems and Evolutionary Computation V., S. Diego, CA., 5200: 133-149.

Ehrgott, M. and X. Grandibleux, 2000. An annotated bibliography of multiobjective combinatorial optimization. Report in Wirtschaftsmathematik, No. 62, Fachbereich Mathematik-Universitat Kaiserslautern.

Fleury, G., P. Lacomme, C. Prins and M. Sevaux, 2005. A memetic algorithm for a bi-objective and stochastic CARP, MIC: The 6th Metaheuristics International Conference, Vienna, Austria, pp: 22-26.

French, S., 1982. Sequencing and Scheduling. Ellis Horwood Limited.

He, Y. and C. Hui, 2007. Genetic algorithm for large-size multi-stage batch plant scheduling. Chem. Eng. Sci., 62 (5): 1504-1523.

Knowles, J. and D. Corne, 2000. M-PAES: A memetic algorithm for multi-objective optimization. In: Proc. Congress on Evolutionary Computation CEC.

Ku, H. and I. Karimi, 1991. An evaluation of simulated annealing for batch process scheduling. Ind. Eng. Chem. Res., 30: 163-169.

Laguna, M., 2002. Global optimization and meta-heuristics, Encyclopedia of Life Support Systems. Theme 6.5, Topic 2.

Lung, C.C. and B.L. Roert, 1993. Complexity of single machine, multi-criteria scheduling problems. Eur. J. Operat. Res., 70 (1): 115-125.

Maniezzo, V., L.M. Gambardella and F. de Luig, 2004. Ant Colony optimization. An Unpublished Report.

Nagar, A., J. Haddock and S. Heragu, 1995. Multiple and bicriteria scheduling: A literature survey. Eur. J. Operational Res., 81 (1): 88-104.

Osman, I.H. and G. Laporte, 1996. Metaheuristics: A bibliography.

Puchinger, J. and G.R. Raidl, 2005. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In: Proc. First Int. Work-Conference on the Interplay Between Natural and Artificial Computation, 3562 of LNCS, pp: 41-53.

Rudolph, G., 1998a. Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets. In: Porto, V., N. Saravanan, D. Waagen and A. Eiben (Eds.). Evolutionary Programming VII. Proceedings of the 7th Annual Conference on Evolutionary Programming, Berlin, Springer-Verlag, pp: 345-353.

Rudolph, G., 1998b. On a Multi-Objective evolutionary algorithm and its convergence to the pareto set. In: Proc. 5th IEEE Conf. Evolutionary Computation, Piscataway, New Jersey. IEEE Press, pp: 511-516.

Schittekat, P., K. S¨orensen, M. Sevaux and J. Springael, 2007. A metaheuristic for solving large instances of the school bus routing problem, MIC. The 7th Metaheuristics. Int. Con. Montreal, Can., pp: 25-29.

Sevaux, M., 2003. Metaheuristics: A quick overview, TEW-Antwerp.

Yavuz, M., E. Akcali and S. Tufekci, 2006. A hybrid meta-heuristic for the batching problem in just-in-time flow shops. J. Math. Modell. Algorithms, 5 (3): 371-393.