International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 12, Number 10 (2019), pp. 1615-1620

A Preliminary Study on the Complexity of Some Heuristics for Solving Combinatorial Optimization Problems

[1]Emmanuel Oluwatobi Asani [2]Peace O. Ayebga [3]Joyce A. Ayoola

[4]Aderemi E. Okeyinka [5]Ayodele A. Adebiyi

[1,2,3,4,5]*Department of Computer Science, Landmark University, Nigeria.*
ORCIDs: 0000-0002-6774-8529 (Emmanuel), 0000-0001-6186-314X (Aderemi), 0000-0002-3114-6315 (Ayodele), 0000-0002-0830-7811 (Peace O.), 0000-0003-0713-8128 (Joyce A.), 0000-0001-8404-4294 (Goodness),

Abstract

Combinatorial Optimization Problems (COP) are mostly NP-Hard and therefore, recurse is made to the use of heuristics for solving them. The goal of this study is to find out how efficient the approximate methods are and in what circumstance, they can be applied to the solution process of optimization problems. Constructive, Improvement, and Partitioning and Decomposition, or Compound heuristics are considered in this study. Our methodology includes the implementation of these methods on the classical Travelling Salesman Problem as a typical combinatorial optimization problem, as well as computation of their complexity using both Analytical and Computational speed approaches. This study is a research in progress; we have presented in this paper the work done thus far viz: a synthesis of selected approximate algorithms, the graph-theoretic illustration of a typical Travelling Salesman Problem and its solution using exhaustive, that is brute-force approach, as well as a heuristic method. The result shows that (as expected though), the solution derived using the Nearest Neighbour heuristic is not optimal. Further research includes implementation of the other heuristics, obtaining and comparing their complexity and a further study of the complexity implication of hybridizing some of the methods used.

Keywords: Complexity, Heuristics, Combinatorial, Optimization, Nearest Neighbour.

1.0. INTRODUCTION

The task of solving complex, mostly impracticable computational problems with limited resources remains a research conundrum which continues to generate interests in the field of mathematics and computing. This scientific technique of finding the best solution that helps optimise given cost function is referred to as Combinatorial Optimization. Combinatorial Optimization is concerned with the task of obtaining the best or close to optimal set of solutions of a finite set, subject to predefined conditions or constraints [1]. We depict these sets of possible solutions using formal mathematical notations or structures, such as graphs, matroids, among others.

The Combinatorial Optimization problem is defined [2, 3, 4] as follows:

Suppose that $\mathbb{F}$ is a family of subsets of set E with finite elements $E = \{e_1 \dots e_n\}$ and

$w: E \to \mathbb{R}$ be a weight function defined as real numbers assigned to the elements of E. The aim of the combinatorial optimization problem is to obtain $F^* \in \mathbb{F}$ such that

$$w(F^*) = min_{F\in\mathbb{F}} w(F)$$

Where $w(F) \coloneqq \sum_{e\in\mathbb{F}} w(e)$

In order to convert this into an optimization problem in $\mathbb{R}^E$, we substitute each $F \in \mathbb{F}$ by its incidence vector. Let $X_e^F = 1$ if $e \in F$ and $X_e^F = 0$ otherwise.

Then if we let $S = \{X^F : F \in \mathbb{F}\} \subseteq \{0,1\}^E$ be the set of incidence vectors of the sets in $\mathbb{F}$, the corresponding optimization problem is:

$$\min\{w^T x : x \in S\}.$$

Combinatorial Optimization spans the fields of Bioinformatics, Artificial Intelligence, Mathematics, Operations Research, Computer Science to complete tasks such as memory register allocation, planning and scheduling, project management, Internet data packet routing, protein structure prediction and so forth. Models are built to formulate and solve real life problems. Examples include completing a Hamiltonian Cycle in the shortest time/cheapest cost known as Travelling Salesman Problem (TSP), Satisfiability Problems (SAT), Graph Colouring Problems (GCP), Cutting Stock Problem, Minimum Spanning Tree (MST), Constraint Satisfaction Problem (CSP), Bin Parking Problem (BPP) etc. [4, 5]. COPs are categorised as either P-problems or NP-hard problems. COPs whose solutions can be obtained in polynomial time are referred to as P-problems. They are mostly decision problems and their solutions space can be built in polynomial time *p*. The COPs whose solutions are obtainable in non-deterministic polynomial time are referred to as NP- hard Problems [6].

Some of these problems can be solved using either exact algorithms or approximate methods. However, because most of these problems are NP-hard problems and since the search space of the factorial number of solutions becomes so large that

they are impractical to solve using exhaustive processing, the use of heuristics is often resorted to.

Combinatorial Optimization aims to provide solutions by deploying efficient algorithmic techniques whose runtime is bounded by a polynomial in the input size. Thus, in solving Combinatorial Optimization Problems, our concerns are:

i. How quickly can we obtain one (or all) optimal solution(s)?

ii. And in cases where, due to complexities, we are unable to obtain the optimal solution, what is the most appropriate approximate solution we can find using efficient algorithmic techniques?

Solving Combinatorial Optimization Problems using Exact technique include Explicit enumeration often referred to as brute force which involves traversing and building all the admissible solutions space to obtain the optimal solution. There are instances where we are able to solve Combinatorial Optimization Problems efficiently, especially those with small degree of search space, using exact algorithms. An example is the problem of finding the shortest paths on a graph, under some amenable assumptions usually met in practice. This can be tackled optimally in polynomial time by the "Dijkstra or Bellman-Ford algorithms" [7]. More complex problems, with no "efficient" algorithms may be approached by modelling the problem as a Mixed Linear Programming (MILP) model and solving it by a MILP solver (e.g. Cplex, Gurobi, Xpress, AMPL, OPL etc.). This utilizes the general-purpose exact algorithms which guarantees optimal solutions at least hypothetically. The computational complexity of these techniques are exponential in nature, thus, the time required to provide their solutions grows exponentially with its solution space [7]. In this case we use heuristics. It may also be achieved using Implicit enumeration which means that all the admissible solutions are considered and implicitly evaluated but are not explicitly built, for instance, tree search with "Branch and Bound" or Branch and Cut. Another Exact solution involves modelling the problems with integer programming models [5, 8, 9].

While exact methods have the potential, at least in theory, to obtain optimal solutions, it is not always practicable. This is owing to two issues that are concurrent in practice vis: the complexity of COPs which are mostly NP-Hard problems, and the constraint of time. This has motivated the deployment of heuristics.

2.0. HEURISTICS

Heuristics are approximate techniques or 'rules of thumb' for solving problems albeit without the guarantee of getting optimal solutions. As opposed to exact methods, heuristic methods do not guarantee optimum solution, rather, they yield good enough or near optimal solutions in reasonable time by drastically cutting down the solution space [10]. A good heuristic must provide near optimal solutions, be easy to implement, flexible and ultimately provide solutions in reasonably short time.

Aside from the need to solve hard problems in polynomial time p, other motivations for using heuristic methods in literature [11, 12, 13, 7] include:

- Unavailability of optimal methods for solving the problems
- The heuristic is part of a broader optimal solution procedure
- Incompatibility of existing exact solutions to available hardware
- The heuristic is more flexible the available exact method and can integrate constraints that are difficult to model.

It is difficult to adequately group heuristic methods into classes, because they are many and were designed in many cases to solve unique problems, thus ruling out the possibility of generalising them. However, in this study, we identify three (3) broad categorisation of heuristic methods in literature [11, 12, 13, 14] which are Constructive Heuristics, Improvement / Local Search Heuristics, Compound Heuristics.

2.1. Constructive Methods

The Constructive Heuristic techniques build solutions, step by step by following a set of predefined guidelines. These guidelines have to do with:

- Initialization: decision on the starting point or initial sub-cycle;
- Selection criterion;
- Position to insert the new element.

The constructive heuristic techniques have been used extensively in solving classic combinatorial optimization problems. We describe some well-known constructive heuristic methods briefly in Table 1 below:

Table 1: Description of some well-known constructive heuristic methods [12, 13]

Heuristic	Description
Nearest Neighbour (NN)	Start at node i (arbitrary or fixed) and find node $k+i$ not yet chosen but closest to node i to form a sub-tour. Find the next node which is unconnected but closest to the last node of the tour and join this node to the last node. If all nodes have been selected STOP, else repeat the process.
Nearest Insertion (NI)	Start at node i (arbitrary or fixed) and find node j not yet chosen but closest to node i to form a sub-tour. Find the next node k which is unconnected but closest to the last node of the tour and insert between two nodes of the sub-tour such that $C_{ik} + C_{kj} - C_{ij}$ is minimized. If all

Heuristic	Description
	nodes have been selected STOP, else repeat the process.
Farthest Insertion (FI)	Start at node i (arbitrary or fixed) and find node j not yet chosen but farthest to node i to form a sub-tour. Find the next node k which is unconnected and farthest from the last node of the tour and insert in such a way as to minimise the cost. Repeat until all nodes are inserted
Cheapest insertion (CI)	This is similar to the Nearest Insertion heuristic. Start at node i (arbitrary or fixed), find cities $k, i\ and\ j$ (i and j being the extremes of an edge belonging to the partial tour and k not belonging to that tour) for which $C_{ik} + C_{kj} - C_{ij}$ is minimized. If all nodes have been selected STOP, else repeat the process.

2.2. Improvement/Local Search Methods

In contrast to the Constructive Heuristic methods discussed in section 2.1., improvement or local search technique attempts to optimize feasible solutions by applying 'iterative improvements'. Improvement is iteratively applied to solutions from previous step and it terminates when for a solution, the termination criterion has been met, that is, there is no other solution that improves it. This is based on the ideology that by iteratively improving and making small changes on the quality of a particular solution, we can obtain close to optimal solution. Examples found in literature include Cheapest Insertion, 2-opt Inter-Route, 2-opt Intra-Route, 3-opt algorithm, *Lin-Kernighan* Algorithms among others [15, 16].

2.3. Compound Methods

The constructive and local search methods form the foundations of the Compound heuristic procedures. In this approach, two or more constructive and improvement heuristics are applied separately and the best solution is chosen [17, 18, 19]. Examples include CCAO (Convex Hull, Cheap Insertion, Largest Angle and OR-Opt) [20], GENIUS [21] among others.

3.0. THE TRAVELING SALESMAN PROBLEM (TSP)

The Travelling Salesman Problem (TSP) is a NP-hard problem as it can be solved in non-deterministic polynomial time.

The Travelling Salesman Problem (TSP) is depicted as shown below:

The travelling salesman has to traverse the cities $1\ to\ n$ in a Hamiltonian cycle. That is, he is expected to start from city 1 through to the remaining $n-1$ cities in arbitrary order, and return to the starting point with the object of touching the cities once in minimal time. The distance $d(i,j)$ depicts the distance from city $i\ to\ j$.

We represent TSP formally below [22]:

$$F = \min \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$\sum_{j=1}^{n} x_{ij} = 1; i = 1, \dots, n$$

$$\sum_{i=1}^{n} x_{ij} = 1; j = 1, \dots, n$$

The objective function is marked with *F*. With a limitation,

$$x_{i_1 i_2} + x_{i_2 i_3} + \dots + x_{i_r i_1} \leq r - 1.$$

$x_{ij}\, x_{ij}$ are the binary variables

$$x_{ij} = \begin{cases} 1 & if\ the\ salesman\ travels\ from\ city\ i\ to\ city\ j \\ 0 & if\ the\ salesman\ is\ not\ travelling\ from\ city\ i\ to\ city\ j \end{cases}$$

$d_{ij}\, d_{ij}$ is the distance from city *i* to city *j*.

A typical exact algorithm checks all $O(n!)$ permutations. Below is the exact algorithm based on dynamic programming as designed by [23].

For every $S \subseteq [2, \dots, n\}$ and for every city $i \in S$, we denote by $OPT[s; i]$ the length of the shortest path that starts in city 1, then visits all cities in $S - \{i\}$ in arbitrary order, and finally stops in city i.

Clearly, $OPT[\{i\}; i] = d(1, i)$ and

$$OPT[S; i] = \min\{OPT[S - \{i\}; j] + d(j, i) : j \in S - \{i\}\}.$$

By working through the subsets S in order of increasing cardinality, we can compute the value $OPT[S; i]$ in time proportional to $[S]$.

The optimal travel time length is given as the minimum value of $OPT[\{2, \dots, n\}; j] + d(j, 1)$ over all j with $2 \leq j \leq n$.

This yields an overall time complexity of $O(n^2 2^n)$, hence $O^*(2^n)$

This algorithm although old has one of the best results till date in term of time complexity [23].

4.0. EXAMINING A 6-CITY TSP.

The 6-city network depicted on a graph in Figure 1 is solved by the use of the Nearest Neighbour Heuristic as well as by exhaustive enumeration.

The Nearest Neighbour Heuristic is one of the oldest and best performing approximate methods [24]. It follows a greedy pattern in solving the TSP. Nearest Neighbour starts with a city, depicted on graph as a node, which may be either fixed or arbitrary and adds the nearest node which has not yet been visited to the last node in the tour. This process is iterated until all the nodes have been added. The Nearest Neighbour algorithmic process are described as follows:

[1]. ***Initialization*** – Start with city (node) i, fixed or selected arbitrarily;
[2]. ***Selection*** – find city $k + 1$ not yet chosen but nearest to city i to form a sub-tour;
[3]. ***Insertion*** – Insert $k + 1$ at the end of the partial tour.
[4]. If all cities are inserted then STOP, else go back to 2.

A program was written to evaluate the exhaustive solutions. The results are shown on Table 2.

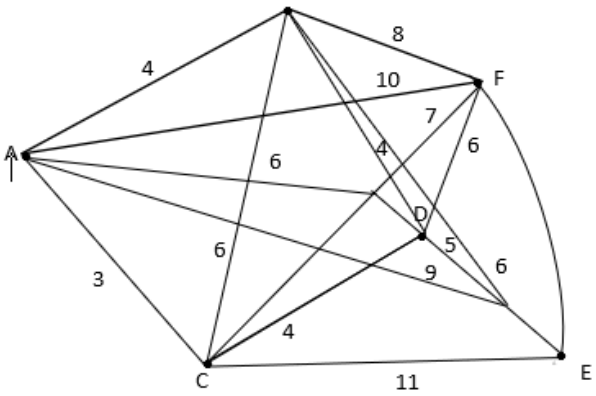

Figure 1: A 6-city Travelling Salesman Problem.

The preceding figure can be represented in the following way as a table:

Table 2: Matrix representation of Graph of Salesman Tour.

Vertex	A	B	C	D	E	F
A	0					
B	4	0				
C	3	6	0			
D	6	4	4	0		
E	9	6	11	5	0	
F	10	8	7	6	5	0

The application of the heuristic gives ACDBEFA with cost 32 as the solution. From the exhaustive enumeration however, 28 is the cost, that is the shortest distance. This is therefore a case of trading-off optimality for computational efficiency.

Many solution techniques are available for the travelling salesman problem. A large number of these solution techniques rely heavily on advanced results in integer linear programming, non-linear programming and dynamic programming.

Heuristics provide solutions that usually are within a few percent of the optimum. Thus, for problems of realistic sizes, heuristics represent a practical solution approach.

ABDEFCA	28	AFDCBEA	41	AFCBDEA	41
AFDCEBA	41	ABEDFCA	31	AFDBECA	40
AFDECBA	42	AFDEBCA	36	ABEFCDA	32
AFEBDCA	33	AFDBCEA	46	AFEDCBA	34
ABEFDCA	28	AFEBCDA	37	AFECDBA	38
AFECBDA	42	ABFEDCA	29	AFBEDCA	36
AFBDECA	41	AFBDCEA	46	ACBEFDA	32
AFBCDEA	42	AFBECDA	45	AFBCEDA	46
ACDFEBA	28	ACDBEFA	32	ACDEFBA	29
ACFEBDA	31	ACFDEBA	31	ACFEDBA	28
ADFEBCA	32	ADCFEBA	32	ADBEFCA	31
ABCDFEA	34	AFEBDCA	32	ABCDEFA	34
ABCFEDA	33	ABCEDFA	42	ABCEFDA	38
ABDCFEA	33	ABCFDEA	37	ABDCEFA	38
ABDFCEA	41	ABDECFA	41	ABDFECA	33

ABECFDA	40	ABEDCFA	36	ABECDFA	41
ABFECDA	38	ABFDECA	37	ABFDCEA	42
ACBDEFA	33	ABFCEDA	41	ABFCDEA	37
ACBFEDA	33	ACBDFEA	33	ACBEDFA	36
ACDEBFA	36	ACBFDEA	37	ACDBFEA	33
ACEDFBA	37	ACDFBEA	36	ACEDBFA	41
ACEFBDA	37	ACEBDFA	40	ACEBFDA	40
ACFBEDA	35	ACEFDBA	33	ACFDBEA	35
ADCBFEA	38	ACFBDEA	36	ADCBEFA	37
ADCFBEA	40	ADCEBFA	45	ADCEFBA	38
ADBECFA	44	ADBCEFA	42	ADBCFEA	37
ADEBCFA	40	ADBFECA	37	ADBFCEA	45
ADECFBA	41	ADEBFCA	35	ADECBFA	46
ADFBECA	40	ADEFCBA	33	ADEFBCA	33
ADFCEBA	40	ADFBCEA	46	ADFECBA	38
AECDFBA	42	ADFCBEA	40	AECDBFA	46
AECFBDA	45	AECBDFA	46	AECBFDA	46
AEDCFBA	37	AECFDBA	41	AEDCBFA	42
AEDFBCA	37	AEDBCFA	41	AEDBFCA	36
AEBDFCA	35	AEDFCBA	37	AEBDCFA	40
AEBFCDA	40	AEBCDFA	41	AEBCFDA	40
AEFDCBA	34	AEBFDCA	36	AEFDBCA	33
AEFCBDA	37	AEFBDCA	33	AEFBCDA	38
AFCDBEA	40	AEFCDBA	33	AFCDEBA	36
AFCBEDA	40	AFCEDBA	41	AFCEBDA	44

5.0. CONCLUSION

We have examined Travelling Salesman Problem vis-à-vis the concept of heuristics and indeed its applicability as demonstrated in the Nearest Neighbour Heuristic. A 6-city TSP is proposed and solved by use of both heuristic and exhaustive enumeration. From this study, it is realized that the effort required to apply heuristic is by far less than that expended on exhaustive enumeration. Hence in the light of heavy computational complexity, a heuristic approach to optimization problems is preferred. Since whenever it seems hopeless to find optimal solutions, it is still possible to get reasonably good solutions.

In furtherance of this work, we intend to implement other heuristics and then to obtain and compare their complexities. We will equally look into the complexity implications of hybridizing some of the methods used.

REFERENCES

[1]. Dowlatshahi M.B., Nezamabadi-Pour H., Mashinchi M. (2014). A discrete gravitational search algorithm for solving combinatorial optimization problems. Information Sciences, Vol. 258, pp. 94-107

[2]. Consoli, Sergio & Darby-Dowman, K. (2006). Combinatorial Optimization And Metaheuristics. Operational Research Report: Available online: https://bura.brunel.ac.uk/handle/2438/9631. Accessed 13 Feb. 2019.

[3]. Adam Kasperskia and Paweł Zielińskib (2014). Combinatorial optimization problems with uncertain costs and the OWA criterion. Theoretical Computer Science. Vol. 565, pp. 102-112

[4]. Neos (2018). Combinatorial Optimization. Available online: https://neos-guide.org/content/combinatorial-optimization. Accessed 13 Feb. 19.

[5]. Henrique Becker and Luciana S.Buriol (2019). An empirical analysis of exact algorithms for the unbounded knapsack problem. European Journal of Operational Research Available online 12 February 2019 In Press, Accepted Manuscript. https://doi.org/10.1016/j.ejor.2019.02.011

[6]. Gerhard J. Woeginger (2003). Exact algorithms for NP-hard problems: a survey. Combinatorial optimization - Eureka, you shrink! Springer-Verlag New York, Inc. New York, NY, USA. ISBN: 3-540-00580-3, pp. 185 – 207.

[7]. Giovanni L. De (2017). Methods and Models for Combinatorial Optimization: Heuristics for Combinatorial Optimization. Available online:

https://www.math.unipd.it/~luigi/courses/metmodoc1718/m02.meta.en.partial01.pdf. Accessed 13 Feb. 2019.

[8]. Anthony Przybylski and Xavier Gandibleux (2017). Multi-objective branch and bound. European Journal of Operational Research. Vol. 260 (3), pp. 856-872

[9]. Claudio Contardoa, Manuel Iorib and Raphael Kramerb (2019). A scalable exact algorithm for the vertex p-center problem. Computers & Operations Research. Vol. 103, pp. 211-220.

[10]. P.M.Todd P.M. (2001). Heuristics for Decision and Choice. International Encyclopedia of the Social & Behavioral Sciences, 2001. Pages 6676-6679.

https://doi.org/10.1016/B0-08-043076-7/00629-X

[11]. Marti R and Reinelt G. (2011). Heuristic Methods. The linear Ordering Problem Exact and Heuristic Methods in Combinatorial Optimization. Springer, -Verlag Berlin Heidelberg. ISBN: 978-3-64-16728-7. Pp 17-40. DOI: 10.1007/978-3-642-16729-4 2

[12]. Oliveira J. F. and Carravilla M. A. (2009). Heuristics and Local search. Available online: https://paginas.fe.up.pt/~mac/ensino/docs/OR/CombinatorialOptimizationHeuristicsLocalSearch.pdf. Accessed 13 Feb. 2019.

[13]. Kyritsis M., Gulliver S. R., Feredoes E. and Ud Din S. (2018). Human behaviour in the Euclidean Travelling Salesperson Problem: Computational modelling of heuristics and figural effects. Cognitive Systems Research Vol. 52, pp 387-399.

[14]. Manfred Gilli (2004). An Introduction to Optimization Heuristics. Available online: http://www.unige.ch/ses/dsec/static/gilli/CyprusLecture2004.pdf. Accessed 13 Feb. 2019

[15]. Mtenzi F. (2006). Improvement heuristics for the Sparse Travelling Salesman Problem. In Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006, pp101-108.

[16]. Tavares L. G., Lopes H. S. and Lima C. R. E., (2009). Construction and improvement heuristics applied to the capacitated vehicle routing problem. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, 2009, pp. 690-695. doi: 10.1109/NABIC.2009.5393467

[17]. Frederickson, G., Hecht, M., and Kim, C., (1978). Approximation Algorithms for Some Routing Problems," SIAM J. Computing T, 178--193.

[18]. Langston, M. A. (1987). A Study of Composite Heuristic Algorithms. Journal of Operational Research Society. 38,539-544.

[19]. Yao, A. C., (1980). New Algorithms for Bin Packing," J. Ass. Comput. Mach. 27,207-227.

[20]. Golden B.L. and Stewart W.R. (1985). Empirical Analysis of Heuristics in *The Travelling Salesman Problem:* A guided tour of Combinatorial Optimization. E. W. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Smoys (eds), Wiley, Chichester, pp. 207-249.

[21]. Gendreau M., Hertz A., and Laporte G. (1992). New Insertion and Postoptimization Procedures for the Travelling Salesman Problem. Operations Research, Vol. 40, pp. 1086-1094.

[22]. Mataija M., Rakamarić M. Šegić and Jozić F (2016). Solving the travelling salesman problem using the Branch and Bound Method. Zbornik Veleučilišta u Rijeci, Vol. 4 (1), pp. 259-270

[23]. Held M. and Karp R.M. (1962). A dynamic programming approach to sequencing problems. Journal of SIAM. Vol. 10, pp. 196-210

[24]. Chauhan C., Gupta R. and Pathak K. (2012). Survey of Methods of Solving TSP along with its Implementation using Dynamic Programming Approach. International Journal of Computer Applications, vol. 52 (4), pp. 34-38.