PARAMETER-LESS AND METAPHOR-LESS METAHEURISTIC ALGORITHM SUGGESTION FOR SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS

İslam ALTIN[1*], Aydın SİPAHİOĞLU[2]

[1] Eskişehir Osmangazi Üniversitesi, Mühendislik Mimarlık Fakültesi, Endüstri Mühendisliği Bölümü, Eskişehir, ORCID No: https://orcid.org/0000-0002-8133-7806

[2] Eskişehir Osmangazi Üniversitesi, Mühendislik Mimarlık Fakültesi, Endüstri Mühendisliği Bölümü, Eskişehir, ORCID No: https://orcid.org/0000-0001-8743-2911

Keywords

Metaheuristic Algorithms
Rao Algorithm
Discrete Optimization
Traveling Salesman Problem

Abstract

Many optimization problems are complex, challenging and take a significant amount of computational effort to solve. These problems have gained the attention of researchers and they have developed lots of metaheuristic algorithms to use for solving these problems. Most of the developed metaheuristic algorithms are based on some metaphors. For this reason, these algorithms have algorithm-specific parameters to reflect the nature of the inspired metaphor. This violates the algorithm's simplicity and brings extra workload to execute the algorithm. However, the optimization problems can also be solved with simple, useful, metaphor-less and algorithm-specific parameter-less metaheuristic algorithms. So, it is the essential motivation behind this study. We present a novel metaheuristic algorithm called Discrete Rao Algorithm (DRA) by updating some components of the generic Rao algorithm to solve the combinatorial optimization problems. To evaluate the performance of the DRA, we perform experiments on Traveling Salesman Problem (TSP) which is the well-known combinatorial optimization problem. The experiments are performed on different sized benchmark problems in the literature. The computational results show that the developed algorithm has obtained high quality solutions in a reasonable computation time and it is competitive with other algorithms in the literature for solving the TSP.

KOMBİNATORYAL ENİYİLEME PROBLEMLERİNİN ÇÖZÜMÜ İÇİN PARAMETRESİZ VE METAFORSUZ METASEZGİSEL ALGORİTMA ÖNERİSİ

Anahtar Kelimeler

Metasezgisel Algoritmalar
Rao Algoritması
Kesikli Eniyileme
Gezgin Satıcı Problemi

Öz

Pek çok eniyileme problemi karmaşıktır ve çözülebilmesi için önemli miktarda hesaplama çabası gerektirmektedir. Söz konusu eniyileme problemleri araştırmacıların ilgisini çekmiş ve araştırmacılar bu problemlerin çözümünde kullanmak üzere birçok metasezgisel algoritma önermişlerdir. Geliştirilen metasezgisel algoritmaların çoğu metaforlara dayanmaktadır. Bu sebeple algoritmalar ilham alınan metaforların doğasını yansıtmak üzere parametre değerlerine sahiptirler. Bu durum algoritmanın sade olan yapısını bozmakta ve algoritmayı çalıştırmak için fazladan iş yükü getirmektedir. Ancak eniyileme problemleri sade, kullanışlı, metaforsuz ve parametresiz algoritmalarla da çözdürülebilir. Bu çalışmanın temel motivasyonu tam olarak söz konusu sade ve metaforsuz algoritma tasarımıdır. Bu çalışmada kombinatoryal eniyileme problemlerini çözmek için yeni bir metasezgisel yöntem olan Kesikli Rao Algoritması geliştirilmiştir. Kesikli Rao Algoritması (KRA) bilinen Rao algoritmasının bazı bileşenlerinde güncellemeler yapılarak elde edilmiştir. KRA'nın performansı iyi bilinen bir kombinatoryal eniyileme problemi olan Gezgin Satıcı Problemi (GSP) için değerlendirilmiştir. Literatürde yer alan farklı boyutlardaki test problemleri kullanılmıştır. Sonuç olarak, geliştirilen algoritma ile makul çözüm sürelerinde yüksek kaliteli çözümler elde edilmiştir ve geliştirilen algoritmanın GSP için literatürdeki diğer algoritmalarla yarışabilir nitelikte olduğu görülmüştür.

Araştırma Makalesi — Research Article
Başvuru Tarihi : 10.10.2022 — Submission Date : 10.10.2022
Kabul Tarihi : 19.04.2023 — Accepted Date : 19.04.2023

[*] Sorumlu yazar: ialtin@ogu.edu.tr
https://doi.org/10.31796/ogummf.1186895

1. Introduction

Optimization is the process of determining the solution(s) that provide the best value for a given objective function(s). That's why, the problems, the optimal solution is tried to be obtained, are called optimization problems. These problems are encountered in several areas such as industry, engineering, science, finance etc. Optimization problems are basically divided into two classes as continuous and discrete with regard to the structure of decision variables (Papadimitriou and Steiglitz, 1998). In discrete optimization problems, decision variables take integer values, while decision variables are allowed to take continuous values in continuous optimization problems. In the combinatorial optimization problems which are in discrete optimization decision variables are required to belong to discrete set. This discrete set consists of objects belonging to the problem. So, combinatorial optimization problems can be defined as obtaining an optimal solution within a finite solution space (set of possible solutions) (Neos guide website, 2022). Some of the best known combinatorial optimization problems in the literature are knapsack problem, traveling salesman problem, vehicle routing problem and scheduling problem. According to the computational complexity theory, all combinatorial optimization problems are grouped under two classes, P and NP-hard. P class problems are easy and a polynomial time algorithm is available to obtain the optimal solution for this problems. For instance, shortest path, assignment and minimum spanning tree problems are belonging to P class. On the other hand, no polynomial time algorithm has not been discovered yet for any NP-hard class problems. For all of the problems in this class there exist exponential time algorithms. In other words, it will be very time consuming to solve larger instances of NP-hard problems with guaranteed optimality.

Many researchers have concentrated their research on combinatorial optimization problems. Thus, many methods and algorithms have presented to the literature in order to solve these problems. The methods used in the solution of these kind of problems are mainly divided into two classes: exact solution and approximate methods (Talbi, 2009). Exact solution methods guarantee obtaining the optimal solution. Dynamic programming, branch and bound algorithm, cutting plane method are the most frequently used exact solution methods.

Approximate methods can generate high quality solutions in a reasonable computation time and there is no guarantee of obtaining the optimal solution (Talbi, 2009). Heuristic algorithms, which are a member of the approximate methods, are categorized as problem-specific heuristic algorithms and metaheuristic algorithms. Problem specific heuristics are tailored and designed to solve a specific problem. For instance, savings algorithm was proposed to solve the vehicle routing problem, while shifting bottleneck algorithm was developed to solve the job shop scheduling problem.

Metaheuristic algorithms are high-level strategies that guide problem-specific and local search methods to achieve high quality solutions in search space (Hussain, Mohd Salleh, Cheng, and Shi, 2018). These algorithms have various mechanisms to get out of local optima in the search space and are generally stochastic. The most important feature of metaheuristics is that they provide dynamic balance between two conflicting criteria: diversification and intensification. Diversification is exploration of the search space and it is aimed to reach the non-explored regions. Intensification is exploitation of the regions that provide high quality solutions more thoroughly.

Challenging problems have gained the attention of many researchers, and therefore, researchers have developed plenty of metaheuristic algorithms to obtain good solutions of these problems in reasonable time. Accordingly, there are many different metaheuristic algorithms in the literature. These proposed algorithms were detailed reviewed, explained and classified by Ezugwu et al. (2021), Singh and Kumar (2021), Agrawal, Abutarboush, Ganesh, and Mohamed (2021). The authors classified metaheuristic algorithms mainly as single solution based and population based. While the optimization process is performed using one solution in single solution based algorithms, a population of solutions is used in population based algorithms. In fact, almost all single solution based and population based metaheuristic algorithms are metaphor based. That is to say, researchers were inspired by some metaphors while developing these algorithms. These algorithms are sampled below.

Evolutionary based algorithms are inspired by natural evolution. Some of these are genetic algorithm (Holland, 1975), memetic algorithm (Moscato, 1989), genetic programming (Koza, 1992).

Swarm intelligence based algorithms are inspired by behavior of animals in mating, foraging, etc. Some of these are ant colony optimization (Dorigo, 1992), particle swarm optimization (Kennedy and Eberhart, 1995), artificial bee colony (Karaboga and Basturk, 2007), monkey search algorithm (Mucherino and Seref, 2007), cuckoo search (Yang and Deb, 2009), firefly algorithm (Yang, 2009).

Physics based algorithms are inspired by the rules of physics. Some of these are simulated annealing algorithm (Černý, 1985; Kirkpatrick, Gelatt Jr, and Vecchi, 1983), gravitational search algorithm (Rashedi, Nezamabadi-Pour, and Saryazdi, 2009), ray

optimization algorithm (Kaveh and Khayatazad, 2012), black hole algorithm (Hatamlou, 2013), water evaporation optimization algorithm (Kaveh and Bakhshpoori, 2016).

Human behavior based algorithms are inspired by social behavior of human. Some of these are group search optimizer (He, Wu, and Saunders, 2006), league championship algorithm (Kashan, 2009), teaching learning based optimization (Rao, Savsani, and Vakharia, 2011).

As far as we know, all these metaheuristic algorithms do not have obvious advantages over each other, although researchers claim that their algorithms are better. Because these algorithms are inspired by some metaphors, and they have algorithm-specific parameters. These parameters affect the performance of metaheuristic algorithm. Thus, it is necessary to determine the optimal values of the algorithm-specific parameters to get quality results. In this case, parameter tuning phase both increases the workload and may causes bias when comparing the algorithms.

Taking into account all of these, (Rao, 2020) shows a new metaheuristic algorithm, called as Rao Algorithm, with algorithm-specific parameter-less and metaphor-less. This algorithm is a population-based algorithm and has only two parameters to use in solving any problem. These parameters are population size and stopping criteria of the algorithm which are standard parameters for all population based metaheuristics. Rao algorithm was applied to constrained and unconstrained continuous optimization problems and it is stated that it performs well. Therefore, this algorithm has attracted the attention of researchers in a short time. Researchers have applied this algorithm to different fields such as design optimization of mechanical system components (Rao and Pawar, 2020a), optimal power flow problem (Gupta et al., 2021) and optimal design of dome structures (Dede, Atmaca, Grzywinski, and Rao, 2022). Moreover, different kinds of Rao algorithm have been developed such as evolutionary Rao algorithm (Suyanto, Wibowo, Al Faraby, Saadah, and Rismala, 2021), self-adaptive population Rao algorithm (Rao and Keesari, 2021), modified Rao algorithm (Pham and Tran, 2022), quasi-oppositional-based Rao algorithm (Rao and Pawar, 2020b), behavior selection Rao algorithm (Wei, Ouyang, Wu, Li, and Zou, 2022).

The purpose of this study is to develop a new version of the Rao algorithm to solve the discrete optimization problems. This novel Rao algorithm can be called as Discrete Rao Algorithm (DRA) and Traveling Salesman Problem (TSP) was used to show the performance of the developed algorithm. With this study, a metaphor-less and algorithm-specific parameter-less algorithm has been introduced to the literature to be used in any discrete optimization problems.

The following parts of the paper is structured as follows: Rao algorithm, DRA and TSP are explained in Section 2, computational results are presented in Section 3 and Section 4 consists of conclusion and future works.

2. Materials and Method

This study complies with scientific research and publication ethics and principles.

2.1 Rao Algorithm

Rao algorithm was introduced to the literature by Rao (2020). This algorithm is different from many population-based metaheuristic algorithms in the literature with regards to metaphor-less and algorithm-specific parameter-less. The main idea of this algorithm is to update the current solutions using the best and worst solution information in the population. In this way, neighbor solutions will be generated. This algorithm was originally developed for the solution of continuous optimization problems and the success of the algorithm was demonstrated on constrained and unconstrained continuous optimization problems. Its flowchart is demonstrated in Figure 1.

In the flowchart of the algorithm presented in Figure 1, the first point to note is that there are only two parameters to be determined. These are population size and stopping criterion of the algorithm. After these parameter values are determined, the initial solution for all candidates of the population is generated. Next, the neighbor solutions are generated for the candidates in the population. If the obtained neighbor solution is better than the current one based on the objective function value, it is accepted and the current solution is updated, otherwise it continues with the current solution. When the stopping criterion of the algorithm is met, the algorithm is stopped and the best solution in the population is reported.

Three different equations are used to generate neighbor solutions. These are presented below.

$\mathbf{x_{ijk}}$: the value of the i^{th} variable for the j^{th} candidate during the k^{th}iteration.

$\mathbf{x_{ilk}}$: the value of the i^{th} variable for the randomly selected candidate l during the k^{th} iteration.

$\mathbf{x_{ibestk}}$, $\mathbf{x_{iworstk}}$: the value of the i^{th} variable for the best or the worst candidate during the k^{th} iteration.

$\mathbf{r_{1jk}}$, $\mathbf{r_{2jk}}$:random numbers in the range [0,1] for the i^{th} variable during the k^{th} iteration.

$$x'_{ijk} = x_{ijk} + r_{1ik}(x_{ibestk} - x_{iworstk}) \quad (1)$$

$$x'_{ijk} = x_{ijk} + r_{1ik}(x_{ibestk} - x_{iworstk}) + r_{2ik}(|x_{ijk} \; or \; x_{ilk}| - |x_{ilk} \; or \; x_{ijk}|) \quad (2)$$

$$x'_{ijk} = x_{ijk} + r_{1ik}(x_{ibestk} - |x_{iworstk}|) + r_{2ik}(|x_{ijk} \; or \; x_{ilk}| - (x_{ilk} \; or \; x_{ijk})) \quad (3)$$

Equation (1) generates a neighbor solution by sharing the information of the population's the best and the worst solutions to the current solution. On the other hand, with Equations (2) and (3), a neighbor solution is generated by sharing information with the current solution from the best, the worst, and randomly selected solutions in the population. One of them is selected and used in the structure of the algorithm. The algorithm is named with regard to the equation to be used as the neighbor search operator. It is called Rao-1 algorithm if Equation 1 is used, Rao-2 algorithm if Equation 2 is used, and Rao-3 algorithm if Equation 3 is used in the structure of the algorithm. As explained, these algorithms are quite simple and can be easily adapted to solve any continuous optimization problems.

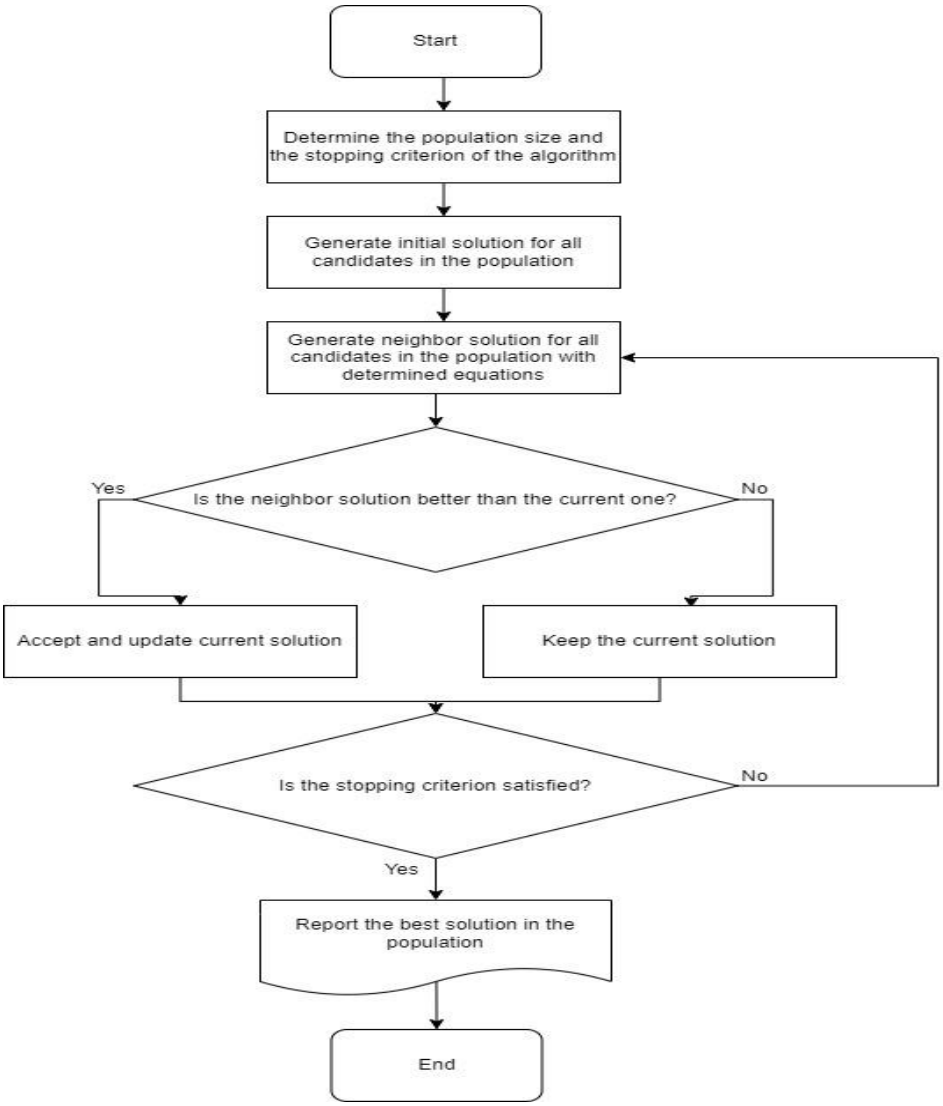

Figure 1. Flowchart of the Rao algorithm

2.2. Traveling Salesman Problem (TSP)

In the 1800s, William Rowan Hamilton created the groundwork for the TSP, which has been the subject of several research to this day. TSP is obtaining the shortest closed tour that allows a salesman to visit each city in a given set just once. This problem is the most well-known combinatorial discrete optimization problem. TSP has various application areas such as drilling of printed circuit boards, overhauling gas turbine engines, x-ray crystallography, computer wiring, the order-picking problem in warehouses and vehicle routing etc. (Matai, Singh, and Mittal, 2010). In

addition, it is the most notable Np-hard problem (Laporte, 1992). That is to say, algorithms that can solve the related problems in polynomial time are not known yet. As the size of the problem increases, the computation time also increases exponentially. Researchers can get detailed information regarding the types of TSP and solution approaches for these problems from Gutin and Punnen (2006) and Applegate, Bixby, Chvátal, and Cook (2011).

Mathematical model of TSP is given below.

Indices:

$i, j = \{0,1, \dots, N\}\ nodes\ (cities)$

Parameter:

c_{ij}: $distance\ value\ between\ i - j$

Decision Variables:

$$x_{ij} = \begin{cases} 1, & If\ salesman\ goes\ from\ i\ to\ j \\ 0, & otherwise \end{cases}$$

$u_i = sub - tour\ elimination\ variables$

Model:

$$\min Z = \sum_i \sum_j c_{ij} x_{ij} \quad (4)$$

$$\sum_i x_{ij} = 1 \qquad \forall j \quad (5)$$

$$\sum_j x_{ij} = 1 \qquad \forall i \quad (6)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad \forall i, j \quad i \neq j, i \geq 2, j \geq 2 \quad (7)$$

$$x_{ij} \in \{0,1\},\ u_i: free \quad (8)$$

Equation (4) is the objective function minimizing the total travelled distance. Equations (5) and (6) guarantee that each node (city) is just visited once. Equation (7) prevents the sub-tours that proposed by Miller, Tucker, and Zemlin (1960). Equation (8) shows the sign constraints.

As the size of the problem increases, the time required to solve the problem increases exponentially. So, this problem belongs to the NP-hard class. For this reason, researchers have proposed heuristic and metaheuristic algorithm to solve it in a reasonable computation time. Accordingly, we proposed a novel DRA to solve this problem efficiently manner.

2.3. Discrete Rao Algorithm (DRA)

The generic Rao algorithm is developed for solving continuous optimization problems. So, some changes are needed in the structure of the algorithm in order to use it in solving combinatorial optimization problems. Since the generic Rao algorithm is algorithm-specific parameter-less, it will be sufficient to modify the neighbor search process. The equations used as the generating neighbor solution in the generic structure of the algorithm are suitable for the continuous optimization problems. Thus, we propose new neighbor search operators to the algorithm. These operators work well for permutation solution representation used for combinatorial optimization problems.

Four different neighbor search operators have been proposed for DRA. These are detailed below.

Two-Point Crossover: This operator will be applied to a randomly selected candidate for each iteration. The information carried by the best or the worst solution in the population will be transferred to the selected candidate. Crossover points are randomly determined for the selected candidate. The part between the determined crossover points is preserved. The other parts of the candidate are updated based on the order of the best or the worst candidate. Please note that up to half of the randomly selected candidate's information should be updated by the best or the worst solution.

Insert Operator: A randomly selected position value in the current solution is inserted to the another randomly generated position. This operator is applied only for the best solution in the population. Thus, the algorithm will continue to focus more on quality solution areas.

Swap Operator: The values of two randomly selected position are exchanged. This operator supports to the intensification mechanism of the algorithm like insert operator. So, it is also applied only for the best solution in the population.

Reverse Operator: For other candidates in the population (except for the randomly selected candidates), the neighbor solution is generated by the reverse operator. With this operator, two different position points are randomly generated in the current solution and the values between these points are reversed. Therefore, it can make major changes to the current solution. Thus, it prevents the algorithm from getting stuck into the local best points.

By using the proposed operators, neighbor solutions are generated for all candidates in the population in each iteration. In other words, they are operated in accordance with the basis of the generic Rao algorithm. The flowchart of the DRA is shown in Figure 2.

Figure 2 indicates schematically the process of the DRA. When the DRA flowchart is analyzed, it is clear that it reflects the spirit of generic Rao algorithm. Because it is algorithm-specific parameter-less and the information from the best solution in the population is employed during the generating neighbor solutions.

Moreover, the proposed algorithm has insert, swap, reverse and two-point crossover operators that will provide intensification and diversification. Thus, the dynamic balance between diversification and intensification mechanisms of the algorithm will be provided. One of the three operators (swap, insert, reverse) is randomly selected to obtain a neighbor solution from the best solution in the population. For a randomly selected solution, the neighbor solution is generated by applying two-point crossover method using either the best or the worst solution. For other candidates of the population, the neighbor solution is obtained using the reverse operator. By this way, all current solutions in the population are updated and new neighbor solutions are generated in each iteration of the proposed algorithm. At the end, we can say that the algorithm is ready for use in solving combinatorial optimization problems with these components.

Figure 2. Flowchart of the DRA

3. Computational Results and Discussion

The developed DRA algorithm is applied to solve the TSP. The components of the algorithm are designed as follows.

Solution representation: Permutation solution representation is implemented in accordance with the TSP. An example of solution representation for TSP with 7 cities is shown in Figure 3.

3	7	5	1	4	6	2

Figure 3. Solution Representation of TSP

Evaluating the solution representation shown in Figure 3, the vehicle route is obtained as Depot – City 3 – City 7 – City 5 – City 1 – City 4 – City 6 – City 2 – Depot. So, the length of this solution representation depends on the number of cities.

Neighbor search operators: Neighbor search operators in the structure of the proposed algorithm are appropriate for permutation solution representation. Figure 4 summarizes the working mechanisms of the neighbor search operators used in the structure of the proposed algorithm.

Swap Operator							
Current Solution	3	7	5	1	4	6	2
Suppose two randomly selected positions are 4 and 6.							
Neighbor Solution	3	7	5	6	4	1	2
Insert Operator							
Current Solution	3	7	5	1	4	6	2
Suppose two randomly selected positions are 1 and 4.							
Neighbor Solution	7	5	1	3	4	6	2
Reverse Operator							
Current Solution	3	7	5	1	4	6	2
Suppose two randomly selected positions are 2 and 5.							
Neighbor Solution	3	4	1	5	7	6	2
Two Point Crossover							
The Best Solution	6	3	2	4	5	7	1
Current Solution	3	7	5	1	4	6	2
Suppose two randomly selected positions are 2 and 4.							
Neighbor Solution	6	7	5	1	3	2	4

Figure 4. The Working Mechanism of the Neighbor Search Operators

As can be seen from Figure 4, neighbor search operators will be used as described in the previous title.

Stopping criterion: The number of iterations is used as the stopping criterion of the proposed algorithm.

The performance of the algorithm has been tested on different sized test problems taken from the literature (TSPLIB website). The information about these problems and the determined values of population size and number of iterations, which are the two parameters of the algorithm used in the solution of these problems, are shown in Table 1.

Table 1. Parameter Values of the DRA

Test Problem	**Number of Nodes**	**Population Size**	**Number of Iteration**
GR17	17	10	1000
GR24	24	10	1000
SWISS42	42	10	10000
GR48	48	10	10000
EIL51	51	10	25000
ST70	70	10	40000
EIL76	76	10	45000
KROA100	100	10	50000
EIL101	101	10	50000
KROA150	150	10	75000
XQG237	237	50	35000
PBL395	395	50	50000
XQL662	662	50	75000

In Table 1, the names of the test problems, the number of nodes (cities) in the test problems, the number of candidates to be included in the population and the number of iterations are presented from the left column to the right respectively. For instance, KROA100 problem contains 100 nodes, and the parameter values of the algorithm specified as (population size, number of iterations) = (10, 50000). As can be seen in Table 1, as the size of the problem increases, the number of iterations required to solve the problem has also increased. This is because as the size of the problem increases, it becomes more difficult to solve. Moreover, for large scale problems such as XQG237, PBL395 and XQL662 the population size is set to 50 in order to prevent the algorithm from getting stuck into the local best points. In addition, the mechanism of generating neighbor solution for these problems is the same as in Figure 2 for the best and randomly selected candidates, while it is different from the structure in Figure 2 for other candidates in the population. The neighbor solution for these candidates

is obtained as follows; half of the candidates to be selected randomly are replaced with the best solution and then reverse operator is applied to these candidates. Thereby, it is intended that the algorithm can be able to better explore for quality solution regions.

The suggested method was coded in Python 3.7 and executed on an Intel Core i5 personal computer at 1.8 GHz with 8 GB of RAM. It has been run 10 times for each test problem and the gap value have been presented in Table 2. The gap value was calculated as presented in Equation (9).

$f(s) =$ The obtained objective function value with DRA.

$f(s^*) =$ The optimal objective function value.

$$\text{Gap} = \left(f(s) - f(s^*)\right)/f(s^*) \quad (9)$$

Table 2. Computational Results of the DRA

Test Problem	The Optimal Results	The Obtained Results	Gap	Computation Times (second)
GR17	2085	2085	-	0.272
GR24	1272	1272	-	0.363
SWISS42	1273	1273	-	5.965
GR48	5046	5046	-	6.434
EIL51	426	426	-	17.587
ST70	675	679	0.006	36.256
EIL76	538	546	0.016	46.935
KROA100	21282	21343	0.003	67.163
EIL101	629	638	0.015	68.454
KROA150	26524	27349	0.031	159.061
XQG237	1019	1074	0.054	950.607
PBL395	1281	1392	0.087	1509.308
XQL662	2513	2794	0.112	4602.709

The computational results of the test problems obtained by the DRA are given in Table 2. In this table, column 2 indicates the gap value between the DRA algorithm's solution and the optimal solution, column 3 shows the computation time of the algorithm in seconds. Researchers can find the optimal solutions of these problems from the TSPLIB website.

The optimal solution of the problems up to the 51 nodes was obtained by the DRA at a reasonable computation time (less than 17 seconds). For the problems with the number of nodes between 70 and 662, solutions can be obtained with acceptable deviations from the optimal solution. Besides, the quality solutions can be obtained in a short computation time even for the large scale problems. For the XQL662 test problem, which is the most challenging one, quality solution could be obtained in approximately 4602 seconds (1 hour 20 minutes) with a deviation of about %11 from the optimal solution. For the KROA150 test problem, high quality solution could be obtained in approximately 159 seconds with a deviation of about 3% from the optimal solution. Namely, the obtained results are promising and the performance of the DRA is high for the TSP. It can be interpreted that the proposed DRA is effective to solve the TSP. In order to demonstrate the convergence capability of the proposed algorithm, the convergence graph of the DRA for KROA150 is indicated in Figure 5.

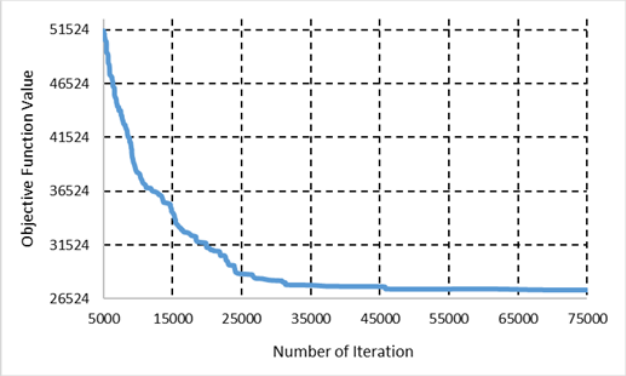

Figure 5. Convergence graph of the DRA

The objective function values on the y axis and the number of iterations on the x axis are represented in the graph. Since the optimal objective function value (total traveled distance) of the KROA150 problem is 26524, the convergence pattern of the DRA is presented to this value. In order to properly display the algorithm's convergence graph, the section after the 5000th iteration of the algorithm is described, and the obtained objective function values in this section are exhibited. The convergence speed of the algorithm is quite high, especially in the first 33% of the total number of iterations. In other words, the algorithm quickly reaches quality solutions. Then, it is clearly seen that the convergence speed of the algorithm decreases gradually as the number of iterations increases. The reason for this is that the algorithm reaches quality solution regions and tries to ensure intensification in these regions. So, the developed algorithm gets the high quality solutions.

4. Conclusion

In this study, a critical approach has been brought to the metaheuristic algorithms used in solving difficult problems in the literature. Such that most of the metaheuristic algorithms in the literature are inspired by some metaphors and have algorithm-specific parameters. Therefore, the use of these algorithms imposes an extra burden on researchers since it is necessary to determine the values of algorithm-specific parameters. Furthermore, as far as we know, the superiority of these algorithms over one another has not been proven yet. This has been the inspiration and starting point for the research. So, Rao algorithm,

metaphor-less and algorithm-specific parameter-less metaheuristic method, has been discussed. Originally, this algorithm was developed to solve the continuous optimization problems. In this paper, we present a novel metaheuristic algorithm called as DRA by updating some components of the generic Rao algorithm to solve the combinatorial optimization problems. The performance of the developed algorithm has been shown on TSP test instances. The obtained results are promising and the developed algorithm has obtained high quality solutions in a reasonable computation time.

The developed algorithm has provided that high quality solutions are obtained. However, since the information of the best and worst solutions is used in the generating neighbor solution stage, the population may contain similar solutions. In this case, similar solutions in the current population may need to be updated. In addition, the proposed algorithm only accepts solutions that improve the current solution. But, it may be possible to benefit from the knowledge of solutions that do not improve the existing solution by using the probabilistic acceptance criterion. So, both the diversity in the population is ensured and it becomes possible to prevent the algorithm from getting stuck into the local best points.

In summary, this study shows that researchers do not have to use the metaphor based metaheuristic algorithms to solve the combinatorial optimization problems. Because, it has been proven that the combinatorial optimization problems can also be solved easily with simple, useful, metaphor-less algorithm. This is a potential area for further research.

It may be useful to develop different versions of the DRA and execute it to solve different combinatorial optimization problems in future works.

Author's Contributions

The authors confirm contribution to the paper as follows: study conception and design: Aydın SİPAHİOĞLU; computational experiments: İslam ALTIN; analysis and interpretation of results: İslam ALTIN and Aydın SİPAHİOĞLU. All authors reviewed the results and approved the final version of the manuscript.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

Agrawal, P., Abutarboush, H. F., Ganesh, T., & Mohamed, A. W. (2021). Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access*, 9, 26766-26791. doi: http://doi.org/10.1109/access.2021.3056407

Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2011). The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton, USA: Princeton University Press.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1), 41-51. doi: https://doi.org/10.1007/bf00940812

Dede, T., Atmaca, B., Grzywinski, M., & Rao, R. V. (2022). Optimal design of dome structures with recently developed algorithm: Rao series. *Structures*, 42, 65-79. doi: https://doi.org/10.1016/j.istruc.2022.06.010

Dorigo, M. (1992). *Optimization, learning and natural algorithms* (Ph. D. Thesis), Politecnico di Milano, Milano.

Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: A comprehensive overview and classification along with bibliometric analysis. *Artificial Intelligence Review*, 54(6), 4237-4316. doi: https://doi.org/10.1007/s10462-020-09952-0

Gupta, S., Kumar, N., Srivastava, L., Malik, H., Anvari-Moghaddam, A., & García Márquez, F. P. (2021). A robust optimization approach for optimal power flow solutions using rao algorithms. *Energies*, 14(17), 5449. doi: https://doi.org/10.3390/en14175449

Gutin, G., & Punnen, A. P. (Eds.). (2006). *The traveling salesman problem and its variations* (Vol. 12). New York, USA: Springer Science & Business Media.

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222, 175-184. doi: https://doi.org/10.1016/j.ins.2012.08.023

He, S., Wu, Q. H., & Saunders, J. R. (2006). A novel group search optimizer inspired by animal behavioural ecology. *Proceedings of the International Conference on Evolutionary Computation*, 1272-1278, Vancouver, Canada.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, USA: University of Michigan Press.

Hussain, K., Mohd Salleh, M. N., Cheng, S., & Shi, Y. (2019). Metaheuristic research: A comprehensive survey. *Artificial Intelligence Review*, 52(4), 2191-2233. doi: https://doi.org/10.1007/s10462-017-9605-z

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3), 459-471. doi: https://doi.org/10.1007/s10898-007-9149-x

Kashan, A. H. (2009). League championship algorithm: A new algorithm for numerical function optimization. *Proceedings of the International Conference of Soft Computing and Pattern Recognition*, 43-48, Malacca, Malaysia.

Kaveh, A., & Bakhshpoori, T. (2016). Water evaporation optimization: A novel physically inspired optimization algorithm. *Computers & Structures*, 167, 69-85. doi: https://doi.org/10.1016/j.compstruc.2016.01.008

Kaveh, A., & Khayatazad, M. (2012). A new meta-heuristic method: Ray optimization. *Computers & Structures*, 112-113, 283-294. doi: https://doi.org/10.1016/j.compstruc.2012.09.003

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the ICNN'95-International Conference on Neural Networks*, 4, 1942–1948, Perth, WA, Australia.

Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671-680. doi: https://doi.org/10.1126/science.220.4598.671

Koza, J. R. (1992). Genetic programming: On the programming of computers by means of natural selection. Cambridge: MIT Press.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2), 231-247. doi: https://doi.org/10.1016/0377-2217(92)90138-y

Matai, R., Singh, S. P., & Mittal, M. L. (2010). *Traveling salesman problem: An overview of applications, formulations, and solution approaches*. D. Davendra (Ed.), In Traveling salesman problem, theory and applications (pp. 1-25). Rijeka, Croatia: IntechOpen.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4), 326-329. doi: https://doi.org/10.1145/321043.321046

Moscato, P. (1989). *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms* (C3P Report 826). Pasadena, USA: California Institute of Technology.

Mucherino, A., & Seref, O. (2007). Monkey search: A novel metaheuristic search for global optimization. *Proceedings of the American Institute of Physics (AIP) Conference on Data Mining, Systems Analysis, and Optimization in Biomedicine*, 953(1), 162-173, Gainesville, USA.

Neos Guide website. (2022). Retrieved from https://neos-guide.org/guide/types/#discrete

Papadimitriou, C. H., & Steiglitz, K. (1998). Combinatorial *optimization: Algorithms and complexity*. North Chelmsford, USA: Courier Corporation.

Pham, H. A., & Tran, T. D. (2022). Optimal truss sizing by modified Rao algorithm combined with feasible boundary search method. *Expert Systems with Applications*, 191, 116337. doi: https://doi.org/10.1016/j.eswa.2021.116337

Rao, R. V. (2020). Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *International Journal of Industrial Engineering Computations*, 11(1), 107-130. doi: https://doi.org/10.5267/j.ijiec.2019.6.002

Rao, R. V., & Keesari, H. S. (2021). A self-adaptive population Rao algorithm for optimization of selected bio-energy systems. *Journal of Computational Design and Engineering*, 8(1), 69-96. doi: https://doi.org/10.1093/jcde/qwaa063

Rao, R. V., & Pawar, R. B. (2020a). Constrained design optimization of selected mechanical system components using Rao algorithms. *Applied Soft Computing*, 89, 106141. doi: https://doi.org/10.1016/j.asoc.2020.106141

Rao, R. V., & Pawar, R. B. (2020b). Quasi-oppositional-based rao algorithms for multi-objective design optimization of selected heat sinks. *Journal of Computational Design and Engineering*, 7(6), 830-863. doi: https://doi.org/10.1093/jcde/qwaa060

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-aided Design*, 43(3), 303-315. doi: https://doi.org/10.1016/j.cad.2010.12.015

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179(13), 2232-2248. doi: https://doi.org/10.1016/j.ins.2009.03.004

Singh, A., & Kumar, A. (2021). Applications of nature-inspired meta-heuristic algorithms: A survey. *International Journal of Advanced Intelligence Paradigms*, 20(3-4), 388-417. doi: https://doi.org/10.1504/ijaip.2021.119026

Suyanto, S., Wibowo, A. T., Al Faraby, S., Saadah, S., & Rismala, R. (2021). Evolutionary rao algorithm. *Journal of Computational Science*, 53, 101368. doi: https://doi.org/10.1016/j.jocs.2021.101368

Talbi, E. G. (2009). *Metaheuristics: From design to implementation*. Hoboken, USA: John Wiley & Sons.

TSPLIB website. (2022). Retrieved from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Wei, W., Ouyang, H., Wu, W., Li, S., & Zou, D. (2022). A behavior-selection based Rao algorithm and its applications to power system economic load dispatch problems. *Applied Intelligence*, 52(10), 11966–11999. doi: https://doi.org/10.1007/s10489-021-02849-7

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Proceedings of the 5th International Symposium on Stochastic Algorithms - Foundations and Applications*, 169-178, Sapporo, Japan.

Yang, X. S., & Deb, S. (2009). Cuckoo search via Lévy flights. *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC)*, 210-214, Coimbatore, India.