Particle Swarm Optimization solving Delivery Problem

Knapsack

MM. Nekhubvi
1 Academy of Computer Science and Software Engineering
University of Johannesburg, South Africa
215000028@student.uj.ac.za

***Abstract*—Discrete/valued optimization issues have been effectively solved using particle swarm optimization techniques. A well-known discrete optimization problem is the multidimensional knapsack problem. Numerous methods, such as heuristic methods and mathematical programming, can be used to solve the various version of the knapsack problem. The particle swarm optimization technique is proposed to solve the multidimensional knapsack problem. The proposed methods experiment shows a comprehensive set of benchmark problems solution and demonstrates the competitiveness of the compared methods presented to the state-of-the-art heuristic methods. Examining the findings for this class's combinatorial problems points to the promise of a straightforward PSO.**

Keywords-Particle Swarm Optimization, knapsack, heuristic methods, multidimension knapsack.

I. INTRODUCTION

The quest for the optimal configuration or grouping of setting to meet some objective criterion has issues with major practical application [1]. These issues are frequently raised to as optimization issues. The knapsack problem is well known optimization issue with several real-world industrial, transportation, and logistical application [2] and in this paper we will examine the delivery vehicle in logistic carrying boxes and the boxes need to be optimized and only the important boxes can be delivered. The delivery knapsack problem can be simple and broadly described as follows: choose a few boxes from a group of boxes with related costs and weight of the box to fit in a vehicle while maximizing the total of the weight without exceeding the vehicle capacity [3]. There are, however, a number of variants to solve the problem including:

- Multidimensional knapsack problem: more than one vehicle is available
- Single knapsack problem: all boxes must be put in a single vehicle.
- Bounded knapsack problem: the are limited boxes available to be selected.
- Multiple-choice knapsack problem: boxes are grouped into subset and only one box may be chosen from each subset.

This paper is going try and solve the Multidimension knapsack problem. the family of NP-hard1 problem includes all the knapsack problems. Although knapsack problems are NP-Hard problems, many huge cases may be handled in matter of time. This is because of the various approaches for solving problems, including both heuristic algorithms and accurate algorithms that have been researched throughout the number of years [3].

For optimization problems, and particularly for the NP complete multidimension knapsack problems, several domain-specific and problem-independent heuristics have been created. The primary goal is to examine the development of a computational methods that is efficient in terms of processing speed and solution quality [4]. Therefore, the application of delivery problem proposes to solve the problem using the Particle Swarm Optimization (PSO) which was proposed by Kennedy and Eberhart [5]. PSO has been effectively used to solve a number of issues including those that involving pattern recognition, scheduling, image processing, classification, and mobile robots, among others [6]. PSO is based in the modeling of the movement of individual fish schools or birds and their aggregate comportment as a swarm. The efficacy of PSO has been demonstrated through applications to several nonlinear optimization issues [6]. PSO often employs the penalty function approach to resolve restricted issues, which reduces the limited problem to unconstrained problem by pushing the objective purpose despite poor taming.

To achieve great search efficiency, PSO combine international and local searches. Within the problem space, a population of random particles with random locations and velocity initialize the algorithm [7]. PSO then updates succeeding generations to look for optima. The two best values are used to update each particle throughout each loop.

In this paper, we will examine the constraint optimization problem. The problem will be a logistic vehicle, and we must determine the quantity of each box to be packed in the company vehicle to be delivered given a set of boxes, each with weight and amount each box worth/price so that the total weight is as close to or less than a what a vehicle can carry and the total amount of all the boxes is as high as feasible. The paper will be separated into sections. Section I will discuss the problem formulated and what we are trying to solve. Section II

will look at the work that other researchers have done. In the third section, we will start looking at the optimization approach used for the implementation of the PSO of the system. Then in the fourth section, we look at the result/performance of the experimental made for the system. Lastly, the last section is the conclusion of the paper.

II. PROBLEMS: FORMULATION AND INSTANCES

A. Problem statement

The problem is known as the multiple knapsacks with assignment restrictions, a variation of the well-researched multiple knapsack problem, which is a broadening of the single-knapsack problem. The multiple knapsack problem deals with items that are constrained to knapsacks [8]. The challenge is to maximize the allotted weight for each pack while considering assignment constraints. Mathematically, the problem can be formulated as:

$$maximize \quad f(x) = \sum_{i=1}^{n} P_i x_i \qquad (1)$$

$$Subject\ to \quad \sum_{i=1}^{n} w_{ij}\, x_i \le C_j \quad \forall j = 1,2,\dots,m, \quad (2)$$

$$w_{ij} \ge 0, C_j \ge 0, \quad (3)$$

$$x_i \in \{0,1\}, \quad i = 1,2,\dots,n, \quad (4)$$

where M= {1, 2,..., m} is the collection of knapsack restriction with capacities in form of bi (i∈M), and N= {1, 2,...., n} is the set of the boxes. Each boxes costs a certain number of resources w(i,j) and generates Cj unit of profit for every knapsack i∈M. There are often minimal restrictions relative to the number of variables i.e., ⩽ m ⩽ n and all the Multidimension knapsack problem coefficients are non-negative number (c∈Nn, a∈Nm×n, b∈Nm) [9].

As mentioned before, the knapsack problem we are trying to solve is a delivery vehicle where we with boxes with weight and the value of the boxes, and we are trying to maximize the weight with the value to be picked for delivery. The optimization of these problems is to choose the correct box that will fit the vehicle's capacity. How can we solve the problem?

The solution to the problem is to develop a model that will minimize the weight and maximize the value of each box that needs to be delivered. The paper will make use of one of the mentioned members of the PSO to solve our delivery problem.

PSO is used to solve many problems in fields of engineering and computer science and is gaining the popularity as one of the powerful global optimization tools and competing with the population-based search algorithms like Evolutionary Algorithms [16]. Swarm produces a very smart search behaviours using error and collaborative trail. Swarm has subset members that are called particle, and they represent a solution of the problem to be solved. The particle in the swarm draws on both its own experience and that of its closest neighborhood (fitness). And each particle is assigned a fitness value [15]. These particle travel across the search space at a predetermined velocity in pursuit of the best solution. Particle has memory to keep track of the best position is has obtained in the processing stage.

In this paper we are going to look at the how we can implement the PSO algorithm to solve the knapsack problem. Next, we look at the related work that has been done.

B. The problem solution to be solved.

III. LITERATURE REVIEW

In this section, we examine the experiment that is researched in solving the knapsack and different approaches used in the evolutionary algorithms. Primarily we will examine the MKP since that is what the paper is all about and what we are trying to solve. First, we will look at how the knapsack works and how it was developed, and then how the problem is solved.

A. knapsack problem

It is widely considered that solving the knapsack problem and a nondeterministic polynomial-time complete combinatorial problem will be computationally challenging in general [10]. The knapsack problem is derived from the difficulty faced by someone confined by a fixed-size knapsack and who must load it with the most valuable stuff. When choosing among a group of non-divisible projects or jobs within the confines of a specific budget or time limit, the problem frequently arises in resource allocation [3]. Early works on the knapsack problem data back to 1897 [11], and the subject has been researched for over a century. The term "knapsack problem," which alludes to the everyday challenge of packing the most important of useful goods without overstuffing the luggage, first appeared in the early works of mathematician Tobias Dantzig (1884-1956) [12]. Back then, the problem was solved without a computer or calculator, and some problems looked impossible to solve or could take years to solve. Still, nowadays, the knapsack problem can be solved using computer PSO in just a few seconds. Now let's look at how the problem is solved using Particle Swarm Optimization.

B. chance constrain optimization problem.

A subset of stochastic optimization is chance-constrained optimization [13]. They are typically difficult to solve unless a special case like linear optimization or minimum spanning tree exists. The approach has been expanded to a class of chance-constrained optimization problems where the objective function is quasi-convex [14], and chance-constrained shortest path problems have been examined [15]. The n items with weight and profit, as well as the bound B of the knapsack, serve as the input in the chance-constrained knapsack problem [16]. The weight is random variable {w1, . . ., wn}, and variances $(\sigma^2,\dots,\sigma^{2n})$ for each item that needs to be selected. The profit of things is shown by the deterministic symbols {p1, ..., pn}. The solution's search space is {0,1}^n, meaning that items i is selected if and only if xi =1, 1 ≤ i ≤ n. the weight W(x) =

$\sum_{j=1}^{n} w_j x_j$ expected weight E(x) = $\sum_{j=1}^{n} a_j x_j$ and variance of weight is Varw(X) = $\sum_{j=1}^{n} \sigma^2{}_j x_j$ [17]. The profit formulation is shown in the problem statement section. The goal of this problem is to choose a subset of items where the profit is maximized under the conditions of equation 2, the chance restriction. A solution must breach the bound of constraint B with a probability of at most in order to violate the chance constraint. For our investigations, we assume that the item weights are independent of one another and selected in accordance with a predetermined distribution that enables the use of various probabilistic tools [18]. The next subsection investigates how they formulate equations to solve knapsack.

C. Optimization approaches for particle swarm optimization

As already said that the PSO is a heuristic method for optimization, stimulated in the characteristic of social agent found in nature. The population-based computations are used in the model. Particles or agents effect change according to their state (position) in the multidimensional search space of the issue, according to their personal knowledge and the impact of the neighboring particles. PSO initializes particles by randomly producing many workable solutions that are vectorially specified, much like other evolutionary approaches [18]. Using an n-dimension search space as starting point the *ith* viable solution, or *ith* particle is traits by an n-dimension vector Xi = (xi1, xi2, . . ., xin). The individual is expressed as, which corresponds to the ith particle's ideal location. The best particle in the swarm which corresponds to global optimum, is designated as Xj = (xj1,xj2, . . ., xjn). The *ith* particles velocity is presented by symbol Vi = (vi1, vi2, . . ., vin). The swarm's evolution process involves the following actions, with the superscript k signifying the number of iterations [19]:

$$V_{ij}(t+1) = W_{ij}(t) + c_1 r_{1j}[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}[Y_{ij}(t) - x_{ij}(t)] \quad (3)$$

$$y_i(t+1) = y_i(t) + x_i(t+1) \quad (4)$$

Where j = 1,2, . . ., n; and i= 1,2, . . ., N this is the size of the particle swarm; inertia weight is w, and c1, c2 are both positive constants and r1, and r2 are both random numbers, uniformly distributed in [0,1].

Although the PSO methods finds the best solution more quickly than the evolutionary algorithms it has issues with premature convergence and inadequate tuning. Due to their low inertia weight, particles have a tendency to settle into local optimums in an early stage of implementation [20]. The best solution can be undiscovered in the latter stage due to the enormous weak fine-tuning and inertia weight influence of the particles [21]. Below is the summary of the particle swarm optimization algorithm's whole computing process.

- STEP 1. Initialize: set up the settings and population with arbitrary position and velocity.
- STEP 2. Evaluation: calculate each particle's fitness value (the desired objective function).
- STEP 3. Find the *pbest:* if the particle I fitness value is higher than its best fitness value in history, then particle I present fitness value will become its new pbest.
- STEP 4. Find the *gbest*: set gbest as the current value if any updated pbest superior to the current gbest.
- STEP 5. Update the velocity and position: According to the equation (3) and (4), modifying the velocity and go to the next position.
- STEP 6. Stopping condition: stop if the necessary number of iterations or CPU time has been obtained, otherwise return to step 2 [22].

The PSO was introduced to solve various continuous nonlinear functions and has drawback in PSO is applying it on real world continuous nature. The disadvantage were resolved by Kennedy and Eberhart [5] by implementing the discrete binary version of PSO. Particle are represented by the binary solution and the velocity altered to change the probability for each binary dimension to take a rate of one.

D. *Optimization Approaches Multidimension knapsack optimization.*

The MKP issue limited combinatorial optimization. The most typical PSO method for resolving limited optimization issues is the application of a penalty function [23]. By pushing the restriction and creating a single objective, the restricted issue is changed into an unconstrained one, which is then reduced using an unconstrained optimization technique. A good solution for avoiding and managing restrictions the issue of selecting the proper parameters for the penalty function is Den's parameter-less method [24]. Deb's approach was originally put out and used in genetic algorithms and more recently used in the placing fine-tuning strategy using PSO.

Resource allocation and cutting stock loading are different applications of MKP. Though computationally difficult due to the problem's NP-hard, it known to exist. Numerous solution methods have been put out for the MKP, one of the most researched combinatorial optimization issues [20]. The existing MKP algorithms can be divided into heuristic and precise algorithms. The branch-and-bound technique and hybrid approaches that combine the branch-and-bound technique and other technique are foundation of representation of the exact algorithms [5].

IV. EXPERIMENT

The section of the paper we will discuss the methodologies of solving the Delivery problem using the Particle Swarm Optimization, which is the probabilistic and global optimization algorithm in nature since it contains the random processes. The steps that are used to implement the system are mentioned above and some of the equations that are used in the implementation of the delivery problem.

A.Dataset

The dataset that is used to test the are placed in the files. The files contain the name of boxes in latter together with the weight of the boxes and the value of each box. Also, the

vehicle capacity is also included. Below the table will present how the data is placed in the files.

Table 1. Dataset

***		
Capacity of vehicle (int) = 150		
A	10	50
B	15	30
C	51	20
D	60	30

The table above is the demonstration of how the data going to be processed is represented. The alphabet represents the name of the boxes, the next column on the right represents the weight of box and the next after that then represents the value of the boxes.

B. function to optimize cost

The system first changes the values of the of boxes to be maximized to a function so the PSO can be able to understand. In the case of the system built, three function were implemented first for the function of the weight then value of the box function. Then when all a created then the system combined the weight and value function to one which maximize the functions we attempt to solve.

C. particle class

First, the implementation initializes all the variables that will be needed to process. Calculated the fitness of the boxes using the cost function that were mentioned above. Then calculated the updates of velocity and position. In the velocity we initialize the coefficients of previous velocity w, c1 cognitive constant and c2 social constant to a very small values to get the best solution. Then position is updated based on the new velocity updates.

D. Implementation of PSO

Here we initialize the number of dimensions to be global to use in the particle. We established swam then begin to optimize the boxes based on the fitness and which one has the best position as a group. Then we get the result of the problem we are solving, and we set the iteration to be 50 and have 100 particles. Then print the graph total profit vs the capacity of the vehicle.

V. RESULTS

In this section we will demonstrate all the results found in the program from a different Dataset and different parameters mentioned above. The compare all the results to see how efficiently the system perform. The instruction is simple the user of the system will have to run the program then select the file they need to process. First, we show the presentation of the initialization of the starting position. The table below will show the starting initialization of starting position.

Table 2. starting position bounds of file, A

Name of box	Lower bound	Upper bound
A	0	10
B	0	3
C	0	5
D	0	2
E	0	3

Then now we will represent the iteration of process image to show how the iteration works in the system in figure 1.

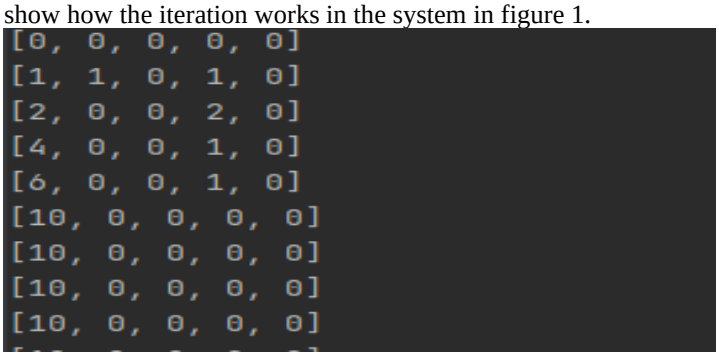

Figure 1. the iteration of file A process

The picture show that the solution was found in the early stage of the process as it shows that the 10 is A and starting to show multiple time in the solution. The overall result show that is A in the figure below.

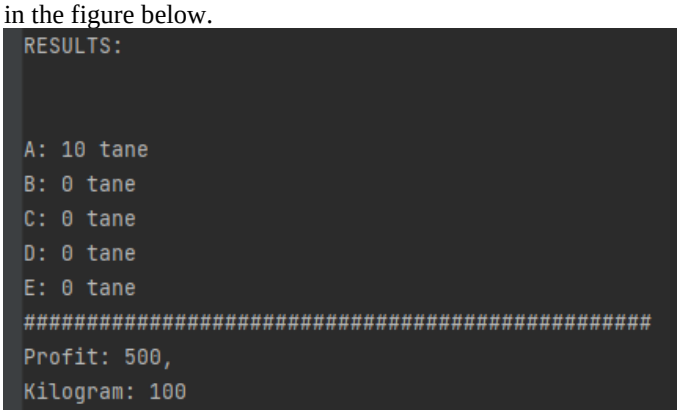

Figure 2. Results

As mentioned above in the iteration of A results show that box A is selected. It is different solution for other files and the system can process multiple datasets in a file and gives different results depending on which of the boxes has the best fitness. The figure represents the total profit vs the capacity.

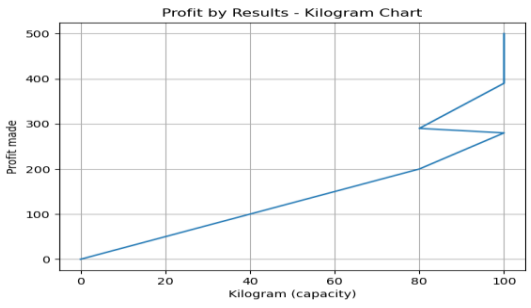

Figure 3. profit vs capacity

Then below is a table comparing different file datasets to show how the system works.

Table 3. comparison of files.

FILE NAME	SIZE	RESULTS	PROFIT
B.TXT	5	B , D	845
C.TXT	15	A, D	700
D.TXT	15	C	400
E.TXT	20	P	600
EE.TXT	20	H	1800

The comparison above shows the result of solving one problem at time but some of the data given to the system were to test how the system will perform given multiple to solve at once. The table below will illustrate the results.

Table 4. simultaneously problem.

File I.txt	Size	Results	profit
Problem 1	20	Q, H	1800
Problem 2	20	P, H, G	1800
Problem 3	15	P	1800

The result above can be represented as a graph below as.

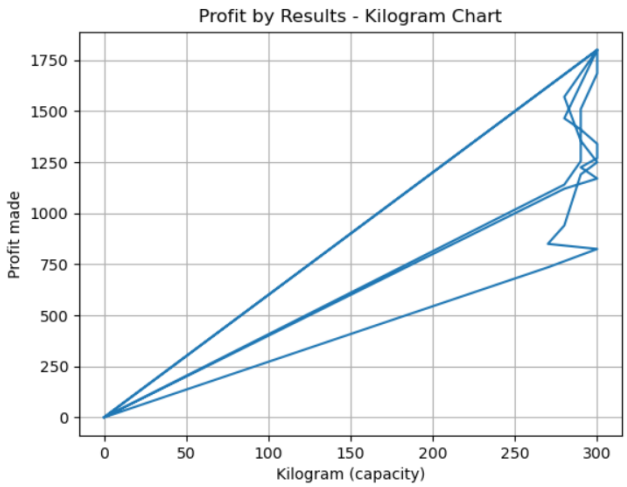

Figure 3. file I.txt

CONCLUSION

In the paper we proposed to solve the delivery knapsack problem using the particle swarm optimization to find the optimal solution. In the knapsack problem we were looking at the weight of the boxes and values of those boxes to be selected to delivered. The algorithm proposed was the gbest PSO to solve the problem. This is discrete problem of multidimension knapsack problem (MKP). The system was created with multiple datasets so to see how the system will perform and compared to each other to see how it work. The experiment shows significant outcome when choosing the boxes to be delivered and we used the iteration of 50 as our termination condition and through the results we can see that it finds the solution earlier before the termination condition reaches.

The system makes use of the particle class where we calculate the fitness of the boxes every time we iterate. Then we updated the velocity and the position of the particles before we right the PSO algorithm. The development makes use of very small values of constant inertia weight, cognitive and social constants to update the velocity. The results of the problem are displayed in tables and graphs to check the solution or which box is chosen among the other to be delivered. In the upcoming work I would like to try solve this problem using the Ant agent and compare all the algorithms of optimization to see which of the three including the EC algorithm perform better.

REFERENCES

[1] Hembecker, F., Lopes, H.S. and Godoy, W., 2007, April. Particle swarm optimization for the multidimensional knapsack problem. In International conference on adaptive and natural computing algorithms (pp. 358-365). Springer, Berlin, Heidelberg.

[2] Martello, S. and Toth, P., 1990. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc...

[3] Bansal, J.C. and Deep, K., 2012. A modified binary particle swarm optimization for knapsack problems. Applied Mathematics and Computation, 218(22), pp.11042-11061..

[4] Chu, P.C. and Beasley, J.E., 1998. A genetic algorithm for the multidimensional knapsack problem. Journal of heuristics, 4(1), pp.63-86.

[5] Eberhart, R.C., Shi, Y. and Kennedy, J., 2001. Swarm intelligence. Elsevier..

[6] Eberhart, R.C. and Shi, Y., 2001, May. Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546) (Vol. 1, pp. 94-100). IEEE..

[7] Chih, M., 2015. Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem. Applied Soft Computing, 26, pp.378-389.

[8] Lust, T., 2010. New metaheuristics for solving MOCO problems: application to the knapsack problem, the traveling salesman problem, and IMRT optimization. Faculté Polytechnique de Mons. pp 23-31

[9] Haddar, B., Khemakhem, M., Hanafi, S. and Wilbaut, C., 2016. A hybrid quantum particle swarm optimization for the multidimensional knapsack problem. Engineering Applications of Artificial Intelligence, 55, pp.1-13.

[10] R. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks," in IEEE Transactions on Information Theory, vol. 24, no. 5, pp. 525-530, September 1978, doi: 10.1109/TIT.1978.1055927.

[11] Mathews, G.B., 1896. On the partition of numbers. Proceedings of the London Mathematical Society, 1(1), pp.486-490.

[12] Dantzig, T. and Mazur, J., 2007. Number: The language of science (Masterpiece Science ed.).

[13] Shapiro, A., Dentcheva, D. and Ruszczynski, A., 2021. Lectures on stochastic programming: modeling and theory. Society for Industrial and Applied Mathematics.

[14] Nikolova, E., 2010. Approximation algorithms for reliable stochastic combinatorial optimization. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (pp. 338-351). Springer, Berlin, Heidelberg.

[15] Banks, A., Vincent, J. and Anyakoha, C., 2008. A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Natural Computing, 7(1), pp.109-124..

[16] Eberhart, R. and Kennedy, J., 1995, October. A new optimizer using particle swarm theory. In MHS'95. Proceedings of the sixth international symposium on micro machine and human science (pp. 39-43). Ieee.

[17] Klopfenstein, O. and Nace, D., 2008. A robust approach to the chance-constrained knapsack problem. Operations Research Letters, 36(5), pp.628-632.

[18] Li, F.F., Shoemaker, C.A., Qiu, J. and Wei, J.H., 2015. Hierarchical multi-reservoir optimization modeling for real-world complexity with application to the Three Gorges system. Environmental Modelling & Software, 69, pp.319-329.

[19] 孔爱良, 梁硕, 李春来, 梁志峰 and 陈艳, 2017. Optimizing micro-grid operation based on improved PSO. Journal of Hohai University (Natural Sciences), 45(6), pp.550-555.

[20] Shi, Y. and Eberhart, R., 1998, May. A modified particle swarm optimizer. In 1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360) (pp. 69-73). IEEE.

[21] Chen, H.T., Wang, W.C., Chen, X.N. and Qiu, L., 2020. Multi-objective reservoir operation using particle swarm optimization with adaptive random inertia weights. Water Science and Engineering, 13(2), pp.136-144.

[22] Chih, M., 2018. Three pseudo-utility ratio-inspired particle swarm optimization with local search for multidimensional knapsack problem. Swarm and evolutionary computation, 39, pp.279-296.

[23] Yang, J.M., Chen, Y.P., Horng, J.T. and Kao, C.Y., 1997, April. Applying family competition to evolution strategies for constrained optimization. In International conference on evolutionary programming (pp. 201-211). Springer, Berlin, Heidelberg.

[24] Varadarajan, M. and Swarup, K.S., 2008. Differential evolution approach for optimal reactive power dispatch. Applied soft computing, 8(4), pp.1549-1561.