Differential Evolution Algorithm for Multiple Inter-dependent Components Traveling Thief Problem

Ismail M Ali[1], Daryl Essam[2] and Kathryn Kasmarik[3]
School of Engineering and IT, University of New South Wales
Canberra, Australia
Email: [1]Ismail.Ali@student.adfa.edu.au, [2]d.essam@adfa.edu.au, [3]Kathryn.Kasmarik@adfa.edu.au

***Abstract*—Differential evolution was mainly proposed for solving optimization problems with continuous decision variables because of its Euclidean distance-based learning concept. This made it unsuitable for many binary and discrete problems. However, several studies approved the applicability of differential evolution algorithm for effectively solving such problems. In this paper, a new design of differential evolution, which incorporates mapping and repairing methods, modified mutation operator and local searches, is proposed to solve the complex multi-components traveling thief problems that are characterized by both binary and discrete parameters. Also, a novel initialization and repairing method, which enables differential evolution's operators to only evolve solutions of one component and optimally distribute/update the solutions of the other one with considering the inter-dependency between both components, is introduced. To judge the performance of the proposed algorithm, 13 strongly correlated instances of traveling thief problems have been solved and the results have been compared with those from 24 self-designed and state-of-the-art algorithms. Results demonstrated the competitive performance of the proposed algorithm in terms of the quality of obtained solutions and computational time.**

***Index Terms*—Differential evolution, Traveling thief problem, Combinatorial optimization problem, Evolutionary algorithms**

I. Introduction

Differential Evolution (DE) was first introduced in 1997 as a method that optimizes a problem with continuous decision variables by iteratively trying to improve a candidate solution [1]. DE is a stochastic population-based search technique, which is inspired by the biological model of evolution and mimicked the natural selection model. As it has a long history of successfully solving continuous optimization problems, DE is considered a powerful tool for solving such problems [2]. In its process, DE adapted three main evolutionary operators, namely: mutation, crossover and selection to direct the search towards (near-) optimal solutions. The quality of the solutions in DE is measured by a pre-defined fitness function, called the objective function, and based on the obtained fitness values, the solutions are ranked.

The Traveling Thief Problem (TTP) is an NP-hard Combinatorial Optimization Problem (COP) with both discrete and binary decision variables [3]. TTP is a recent benchmark for problems with multiple inter-dependent components. It is a combination of two well-known optimization problems: Traveling Salesman Problem (TSP) and Knapsack Problem (KP). These two components are integrated in such a way that the optimal solution for every single component (problem) does not essentially lead to obtaining the optimal solution for TTP. This means that these two components are interdependent in the sense that a solution for one component affects the quality of the solutions for other components, and hence the quality of the solutions for the whole problem. Practically, TTP reflects the complexity of real-world applications, which contain more than single NP-hard problems. This can be widely observed in many fields, such as planning, supply chain, scheduling, routing, and transportation of water tanks [4].

Several techniques have recently been developed for solving the newly introduced TTP benchmark, which is providing many test instances of the interdependent multi-components problems of TTP [3]. The first attempt to tackle TTPs was in 2014, by Polyakovskiy et al. who applied several heuristic and meta-heuristic techniques, such as simple constructive heuristic (SH) and an iterative heuristic. In their work, they incorporated the Random Local Search (RLS) with a simple Evolutionary Algorithm (EA) to repeatedly create a single new solution and record it as the best. In the next iteration, the solution is improved by comparing the newly created one with the best of the previous iterations [5]. In the same year, another optimization problem solver, called CoSolver, was introduced to solve the TTP by decomposing it into two sub-problems and solve them by separate models while maintaining the communication between them. Then it combines the obtained solutions to create the overall TTP solution [6]. Moreover, a memetic algorithm with two-stage local search and multi-complexity reduction approaches has been proposed for solving large-scale instances of TTPs in a maximum of 10 minutes of computational time [7]. Recently, an Ant Colony Optimization (ACO) algorithm with two hybridization levels is also introduced for TTPs [8]. Further investigation of TTPs has been conducted by using a meta-heuristic technique (i.e. Genetic Algorithm (GA)) with multiple local searches, such as 2-OPT, insertion and bit-flip [24]. Because of its importance in various applications and its ability to present the real-world problems with their complexity, and although many state-of-the-art techniques have been proposed, there is still a recent and high demand to find an algorithm that can solve

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

TTPs in a more efficient way and less computational time. Additionally, to our best of knowledge and according to the literature, no DE-based algorithms have been proposed for solving TTPs. So, this paper should be the first one that tackles TTPs using a DE algorithm. However, recent discrete versions of DE algorithm have successfully been applied for solving other COPs, such as motor train-sets [9], multi-skill resource-constrained project [10], TSP [11], [12] and Resource-Constrained Project Scheduling Problem (RCPSP) [13] with very promising results. Also, a binary version of DE has been proposed for solving different instances of binary KPs with outstanding performance compared with other algorithms [14]. The defined limitations, in the literature, of applying discrete versions of DE algorithms for solving COPs, such as slow convergence and becoming trapped in local optima, and the promising results, achieved by discrete DE in other problems, have formed the main motivation of this work.

In this paper, a novel design of DE, which incorporates several effective components to efficiently solve TTPs, is proposed. These components enable the proposed DE to solve TTPs in both discrete and binary domains, so it can be called Discrete-Binary DE (DB-DE). In DB-DE, the following components are incorporated: 1) a dual representation is adapted to represent every solution in both binary and discrete populations; 2) a mapping function is applied for mapping DE's continuous decision variables into discrete and binary; 3) a novel initialization and repairing function is proposed in the aim of reducing the evolutionary process of DE and hence the computational time by modifying only TSP solutions and distributing KP solutions based on them. Finally, the performance of the proposed DB-DE is compared with those of 3 DE-based and 21 state-of-the-art algorithms for solving 13 instances of TTP with different scales.

The rest of the paper is structured as follows. Section II explains the definition and mathematical model of TTPs. The proposed DE algorithm is described in Section III. The experimental results are provided in Section IV. Finally, the conclusion and future work of this study are presented in Section V.

II. Traveling Thief Problem

Traveling Thief Problem (TTP) is a class of COPs, which comprises a combination of two of the most famous optimization problems, namely TSPs and KPs.

According to its definition in [5], TTP can be formulated as a set of cities ($N = 1, ..., n$) for which a distance (d_{ij}) between any pair of cities (i.e. cities i and j, where $i, j \in N$) is known. Every city i contains a set of items ($M_i = 1, ..., k_i$) and each item k positions in i is characterized by its profit (p_{ik}) and weight (w_{ik}), thus item $I_{ik} \sim (p_{ik}, w_{ik})$. The thief had to visit each city exactly once, starting from the first city and returning to it at the end of the trip. Any item can be placed in the knapsack at any city as long as the total weight of the collected items did not exceed the maximum possible weight (W). A renting rate (R) is paid each time when an item is added. v_{max} and v_{min} denote the maximum and minimum speeds, respectively, at which the thief can move. The main objective is to find a trip that achieves the maximum profit value.

Let $y_{ik} \in 0, 1$ denotes a binary variable equals to one when item k is picked up in city i and W_i is the total weight of the collected items when the thief left that city. Therefore, the objective function for a trip ($II = (x_1, ..., x_n)$, $x_i \in N$) and packing plan ($P = (y_{21}, ..., y_{nm_i})$) is:

$$Z(II, P) = \sum_{i=1}^{n} \sum_{k=1}^{m_i} P_{ik} y_{ik} - R\left(\frac{d_{x_n x_1}}{v_{max} - vW_{x_n}} + \sum_{i=1}^{n-1} \frac{d_{x_i x_{i+1}}}{v_{max} - vW_{x_i}}\right) \quad (1)$$

where $V = \frac{v_{max} - v_{min}}{W}$ is a constant value.

III. Proposed Differential Evolution

In this section, the framework of the proposed DB-DE a new initialization and repairing method for KP population, is described. The main components of the proposed framework are shown in Fig. 1 and discussed in the following sub-sections.

A. Solution Representation and Initial Population

Initially, the TTP solution is presented by two vectors, binary and discrete. The binary vector represents the picking plan of KP that decides which item will be taken (=1) and which will not (=0), where the length of the binary vector equals the total number of items (M). The discrete vector represents the order of the cities to be visited in TSP, where its length equals the total number of cities (N). Example of both binary and discrete solution representations is shown in Fig. 2.

Fig. 2 represents a TTP solution, which can be interpreted as follows: the order of cities to be visited by a salesman is 5, 2, 1,... , 8 and will eventually return to city 5 with the following items picking plan: in city 5, no items are picked, in cities 2, 1 and N, items 2, 3 and M are respectively collected.

An initial population of TSP candidate solutions of size PS is randomly generated. Also, a matrix of the distances between all cities and each other is defined as (D), where $d_{i,j}$ is the distance between i and j cities.

B. TSP's Solutions Repairing Method

Having individuals with good qualities within the initial population is very essential to increase the DE convergence speed towards the optimal solutions and hence a significant reduction in the required computational time [11]. Consequently, the k-means clustering, previously proposed in [11], [12], is applied as a repairing method for TSP solutions. According to the conducted parametric analysis of these papers, applying k-means clustering to 10% of PS in the TSP population is recommended. In this method, the solution/path of the TSP is divided into several sub-paths, where all cities with short

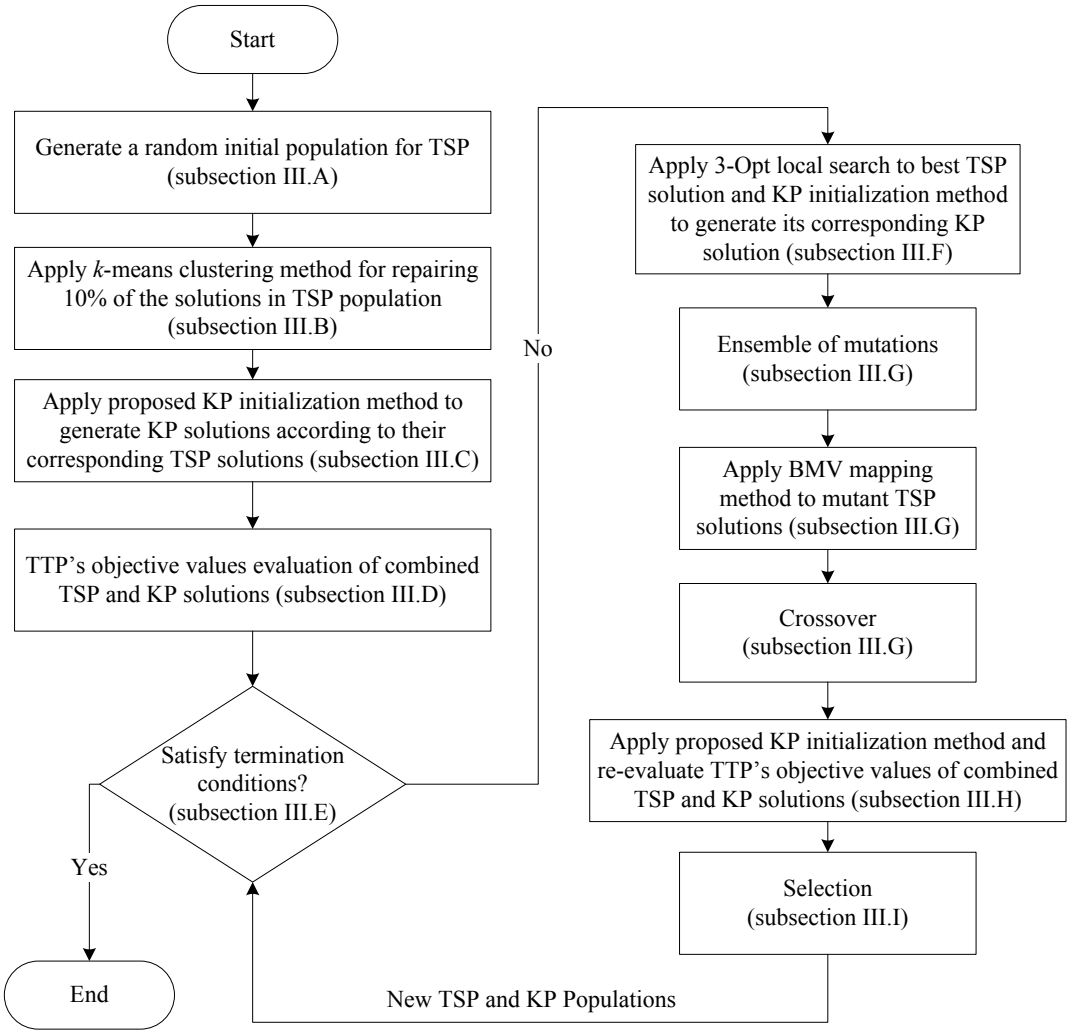

Fig. 1. Framework of proposed DE algorithm for TTPs

Items	1	2	3	…	M
KP Ind.	0	1	1	…	1

Index	1	2	3	…	N	$N+1$
TSP Ind.	5	2	1	…	8	5

Fig. 2. Individual representation of KP and TSP

distances between them are gathered to form a sub-path/group. Then, the formed sub-paths are iteratively merged to form a less number of groups, which comprises a greater number of cities. This process continues until a full path (one group) that contains all the cities, with the minimum total distances between them, is achieved.

C. KP's Solutions Initialization Method

In this step, KP individuals are randomly generated with binary values and PS size to form the KP population. In order to maintain the dependency between the TTP's components, the generated KP solutions are repaired according to their items' profits, weights and locations from the destination city (their carry distance). Location of each item in KP solution can be known by its corresponding city in TSP solution. The proposed repairing method is implemented according to Algorithm 1 through four main steps.

- *Step 1*: Generate a random KP population of PS solutions.
- *Step 2*: in each TSP solution, the distance between each city and the destination city (end of route) is measured and recorded in ($CDis$), where $CDis_i$ is the distance between city i and the destination (last city), and the

order of cities to be visited is also recorded.

- *Step 3*: For each item in KP solution, the TTP objective value, based on the item's current location from the destination, profit and weight values, is calculated by equation 1. After that, the items are sorted according to their objective values.
- *Step 4*: The KP population is re-generated/re-distributed as follows: for each KP solution, distribute the items to the cities according to the resultant order of items.

Algorithm 1 Pseudo-code of KP solution initialization and repairing method

$KP\_Pop \leftarrow$ Generate a random KP population of PS solutions **//—Step 1—**
for i = 1 : PS **do**
  $Full\_Path \leftarrow TSP\_Pop(i) + TSP\_Pop(1)$ **//Step 2-**
  for j = 1 : N **do**
    $distance \leftarrow Dis(j, j+1)$
    $CityOrder \leftarrow j$
    $Fit \leftarrow Fit(i) + distance(j)$
  end for
  for $s = 2 : N+1$ **do**
    $CDis_i \leftarrow CDis_{s-1} + distance_{s-1}$
  end for
  $DisToEnd(i) \leftarrow Fit(i) - CDis_i$
  for $item = 1 : M$ **do**
    $ItemLoc \leftarrow Cities(item)$ **//—Step 3—**
    $CityOfItem \leftarrow CityOrder(ItemLoca)$
    $ItemDtoEnd \leftarrow DisToEnd(CityOfItem)$
    $Obj(item) \leftarrow (ItemDtoEnd/(v_{max} - w_{item} \times (v_{max} - v_{min})/W))$;
    $ItemDis(item) \leftarrow values(item) - R \times t$;
    $OrderedItem \leftarrow sort(ItemDis, descend)$
  end for
  $f \leftarrow 1$; **//—Step 4—**
  $W_{remain} \leftarrow W$
  while W_{remain} ¿0 and $f < M$ **do**
    $Item \leftarrow OrderedItem(f)$
    if $(W_{remain} - w_{Item}) > 0$ **then**
      $f \leftarrow f + 1$; //continue
    end if
    $W_{remain} \leftarrow W_{remain} - w_{Item}$
    $KP\_Pop(i, Item) \leftarrow 1$
  end while
end for

D. Fitness Evaluation and Population Sorting of TTP

Once the solutions of TTP are generated and repaired, the quality of each solution (TSP and its corresponding KP solutions) is measured by calculating its fitness using equation 1. After that, the solutions in both populations are sorted according to the objective values (fitness values), where solutions with high objective values are ranked first.

E. Termination Conditions

In this study, in order to save computational time, two termination conditions are set and applied simultaneously. 1) Maximum allowed number of objective function evaluations (cfe), as the algorithm is terminated if $cfe > MaxFit$, where $MaxFit$ is the maximum number of fitness evaluations. 2) The performance's progress of the algorithm, as the algorithm is terminated if no improvement occurs in 100 consecutive iterations.

If the termination conditions of the DE evolutionary loop are not fulfilled, the following evolutionary process will continue to be applied to TSP population only. In that way, around half of the consumed computational time required to evolving solutions in both TSP and KP populations will be saved, as solutions of KP population can be directly updated by the proposed KP initialization and repairing method (subsection III-C).

F. Local Search

In order to enhance the exploitation capability of the proposed DE, the well-known 3-Opt local search [20] is applied to further explore the best obtained TSP solution (the TSP part in the best TTP solution). Then, the corresponding KP solution is updated/repaired by redistributing its items according to the exploited TSP solution (subsection III-C).

G. Mutation, Mapping Method and Crossover

In mutation, due to its outstanding performance to reproduce TSP solutions with better qualities than their parents, the ensemble of mutation strategies, previously proposed in [11], is applied. The mutation combines three frequently used mutations, namely: DE/rand/1, DE/best/2 and Trigonometric mutation and adaptively applied them based on their performances. The utilization of such mutation operator provides a great balance between exploration and exploitation processes in DE, which increase the capability of the proposed DE to quickly converge towards optimal solutions without getting stuck in a local optima.

The produced mutant vectors from the DE mutation operator usually contain real values because of its Euclidean distance-based learning concept. In order to cope with the discrete nature of TSP solutions, the Best-Matched Value (BMV) mapping method [18] is applied to map the newly produced continuous values to discrete ones with unique numbers (without repetition of cities). In BMV, the population is directed towards the optimal solution by sharing some bits/characteristics of the current best solution with the current mutant solution.

A crossover operator is applied to modify the mapped/discrete mutant vectors for obtaining better solutions. In the proposed DE, the binomial crossover [19] is used to construct trial vectors (new offspring) by taking some genes/elements from a mutant vector with probability cr and other genes from the current target vector with probability $1-cr$, as shown in equation 2.

$$\overrightarrow{u}^{j}_{i,g+1} = \begin{cases} \overrightarrow{v}^{j}_{i,g+1}, & rand(j) \leq cr \text{ or } j = a_j \\ \overrightarrow{x}^{j}_{i,g}, & \text{otherwise} \end{cases} \quad (2)$$

where $j = 1, 2, \ldots, N$ and $i = 1, 2, \ldots, PS$, with cr is the crossover possibility in the range of [0,1], $rand(j)$ is the j^{th} evaluation of a uniform random number generated within [0,1] and a_j is a randomly selected dimension, to ensure that at least one element of $\overrightarrow{u}_{i,g+1}$ is chosen from the mutant vectors defined in equation 2.

H. Fitness Re-evaluation and Sorting

After perturbing solutions of TSP population, the new produced solutions are stored in a new population. Also, the KP solutions are reproduced by redistributing their items according to the TSP solutions in the new population, using the proposed "KP solution initialization and repairing method" in subsection III-C. This process will guarantee much less computational time, as the above DE evolutionary operations are required only to deal with solutions of TSP population, whereas the KP one is updated accordingly using the repairing method.

After obtaining the updated populations of KP and TSP solutions, their fitness values are re-evaluated using equation 1. Then, they are sorted according to their objective (fitness) values, where the KP and TSP individuals with high fitness values are ranked first in both populations.

I. Selection

Finally, in the selection process, comparisons between solutions of TSP and KP in both new and old populations are conducted in terms of the TTP objective value (fitness values) achieved by the combined solution of each TSP and KP individuals. In this comparison, each TSP and KP solutions is only accepted if their TTP objective/fitness value is better than the old one. If so, the solutions from the new populations take the place of the old ones in the next generation. So, this process aims to decide the final survival TSP and KP individuals, which will form the populations of the next generation to continue the evolutionary process of the DE algorithm.

The selection also decides which mutation from the ensemble of mutations will be selected to modify the population in the next generation according to the performance of the three applied mutation operators. The selection process can be accomplished by adopting the greedy selection strategy that is shown in equation 3.

$$\overrightarrow{x}_{i,g+1} = \begin{cases} \overrightarrow{u}_{i,g+1}, & f(\overrightarrow{u}_{i,g+1}) \leq f(\overrightarrow{x}_{i,g}) \\ \overrightarrow{x}_{i,g}, & \text{otherwise} \end{cases} \quad (3)$$

IV. Experimental Results and Analysis

To demonstrate the effectiveness of our proposed DB-DE algorithm, 13 TTPs with different dimensions (N= 51, 52, 70, 76, 99, 100, 101, 105, 107, 150, 198, 280, and 783 Cities) and N-1 knapsack items, were solved. The problems are selected from the hard strongly correlated category from the online benchmark that is available on http://cs.adelaide.edu.au/~optlog/CEC2014COMP_InstancesNew/ through http://www.cec2017.org/. The algorithm was coded in MATLAB R2017b and run on a PC with 16 GB memory and i7 processor. For each problem, the algorithm was executed 30 times with 10,000 fitness evaluations in each run.

A. Parameter Tuning

The proposed DB-DE has three main parameters that may affect the quality of the obtained results: population size (PS), crossover rate (cr) and mutation rate (F). Table I presents the proposed values of these parameters. In order to determine the best combination of these parameter values, we use the Minitab statistical software to apply the Taguchi design method, which is a popular method that uses orthogonal arrays to find the best combination of parameters' values. 25 experiments with different combinations of the parameter values was implemented. For each combination of the orthogonal array, 3 instances were randomly chosen from 13 to calculate the average objective value.

TABLE I
COMBINATION OF PARAMETER VALUES

Parameters	Factors				
	1	**2**	**3**	**4**	**5**
PS	25	50	75	100	150
cr	0.1	0.3	0.5	0.7	0.9
F	0.2	0.4	0.6	0.8	1

The proposed DB-DE was run 30 runs, for each run, the best-obtained TTP objective value was recorded. The numerical results for 25 parameter combinations are presented in Table II and the trend of each parameter is plotted in Fig. 3 based on the average objective values of each combination. As the higher the objective value, the better the performance, it can be concluded from Fig. 3 that the largest value of the average objective was achieved by PS= 50, cr= 0.3 and F= 0.2. Selecting these values are making a great balance in the performance of the proposed DE, as if the value of PS was lower than 50, the population diversity will be quickly reduced, while the larger values will take more computational time to handle all the candidate solutions. The small values of cr and F make a good balance between the exploration and exploitation processes of DE. Considering the above analysis, the best combinations of parameter values are PS= 50, cr= 0.3, and F= 0.2.

TABLE II
ORTHOGONAL TABLE AND AVERAGE OBJECTIVE VALUES

Experimental number	Factors			Average objective value
	PS	cr	F	
1	25	0.1	0.2	8274
2	25	0.3	0.4	8786
3	25	0.5	0.6	6683
4	25	0.7	0.8	8065
5	25	0.9	1	7535
6	50	0.1	0.4	8132
7	50	0.3	0.6	8215
8	50	0.5	0.8	7982
9	50	0.7	1	9014
10	50	0.9	0.2	9307
11	100	0.1	0.6	8902
12	100	0.3	0.8	8780
13	100	0.5	1	7365
14	100	0.7	0.2	8921
15	100	0.9	0.4	8238
16	150	0.1	0.8	9015
17	150	0.3	1	8074
18	150	0.5	0.2	8969
19	150	0.7	0.4	6752
20	150	0.9	0.6	8160
21	200	0.1	1	7247
22	200	0.3	0.2	9212
23	200	0.5	0.4	7228
24	200	0.7	0.6	8133
25	200	0.9	0.8	9307

Fig. 3. Factor level trend of DB-DE based on average objective value

B. Comparisons

In this subsection, in order to demonstrate the efficiency of the proposed DE, several comparisons with DE-based and state-of-the-art algorithms have been conducted.

1) Self-comparisons: Several versions of DE have been implemented to evaluate the effect of each added component on the performance of DE while solving TTPs. Four DE versions were designed and are described in Table III.

TABLE III
DESCRIPTION OF THE FOUR VERSIONS OF DE

DE version	Description
DE+BMV	Standard DE and the BMV mapping method as described in Chapter [18]
DE+BMV+2Opt	DE+BMV with adapting the 2-OPT local search [21] to explore the best TSP solution
DE+BMV+3Opt	DE+BMV with adapting the 3-OPT local search 20 to explore the best TSP solution
DE+BMV+3Opt+ Repairing Method (DB-DE)	The last DE version with incorporating the proposed initialization and repairing method (subsection III-C)

Table IV shows how the performance of the DE algorithm can be enhanced by incorporating extra heuristic mechanisms, such as 2-Opt and 3-Opt local searches while solving complex COPs (i.e. TTPs). From the table, it can be noticed that the standard DE (DE+BMV) achieved negative objective values for all the tested cases of TTPs, which indicates that DE cannot minimize the total travelled distance in TSP component to be less than the total profit values of the collected items in KP part. By adding the local searches, DE exploitation capability improved and was able to achieve positive objective values for some instances of TTPs. Considering the inter-dependency between the TSP and KP components of TTP, which attained by implementing the proposed initialization and repairing method, has significantly improved the DE performance for TTP by radically increasing the obtained objective value of all TTP instances. Fig. 4 graphically presents the average objective values for 3 DE-based algorithms and DB-DE. Numerically, DB-DE can achieve better average objective values for the 13 TTPs by 94.51%, 16.38% and 9.15% percentage differences than DE+BMV, DE+BMV+2Opt and DE+BMV+3Opt, respectively.

TABLE IV
AVERAGE OBJECTIVE VALUES OF 3 DE-BASED ALGORITHMS AND DB-DE

TTPs	DE+BMV	DE+BMV+2Opt	DE+BMV+3Opt	DB-DE
eil51	-2092.86	2695.21	2849.76	4019.4
berlin52	-2943.48	2070.53	1979.25	3790.96
st70	-16678.29	722.06	1003.68	3481.09
eil76	-14013.66	472.91	1548.55	3620.88
rat99	-16334.84	5132.22	5132.22	8211.38
kroA100	-41177.98	148.8	534.74	4259.31
eil101	-23018.22	1265.47	2595.18	5183.33
lin105	-45229.86	769.13	810	3826.58
pr107	-44419.67	5988.42	5988.42	7347.93
kroA150	-71458.73	-845.17	2332.14	7907.57
d198	-101811.5	6846.46	5759.42	14792.69
a280	-216549.34	363.88	8689.42	17053.37
rat783	-1445131.8	-37561.14	-11552.66	34968.69

2) Comparisons with the state-of-the-art algorithms: In order to show the competitive performance of the proposed DB-DE algorithm, the objective values of the 13 TTPs obtained from DB-DE were compared with the maximum, minimum and mean objective values obtained from 21 algorithms drawn directly from [22]. Table V shows the average objective values

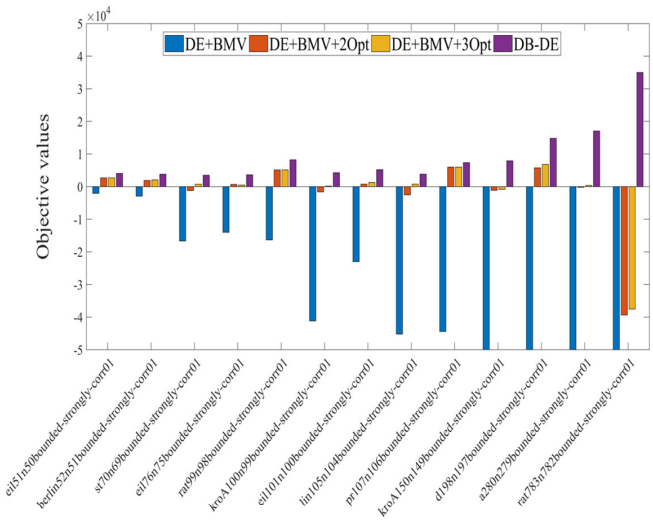

Fig. 4. Performance improvement of DE algorithm (from Standard DE to DB-DE) for 13 TTPs strongly correlated instances

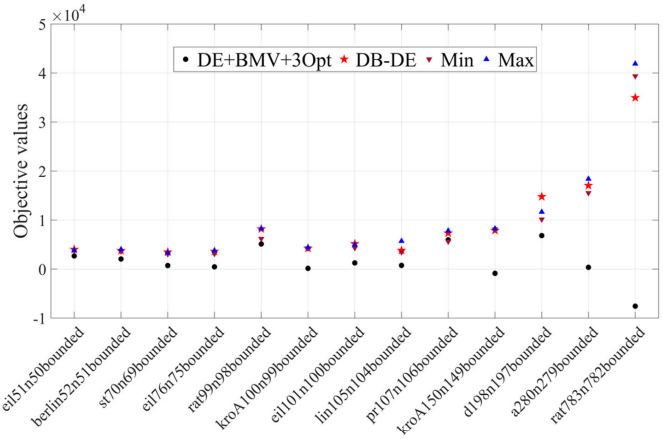

Fig. 5. Objective values obtained from DE+BMV+3Opt, DB-DE, best and worst algorithms in [22] for 13 TTPs

TABLE V
AVERAGE OBJECTIVE VALUES AND COMPUTATIONAL TIMES IN SECONDS OF DB-DE, AND THE BEST, MEAN AND WORST OF 21 ALGORITHMS IN [22] FOR 13 TTPS

TTPs	Average Objective Values				Computational Time (sec.)	
	DB-DE	Objective values of 21 algorithms			DB-DE	21 algorithms
		Min	Mean	Max		
eil51	**4019.4**	3796.66	3820.45	3844.23	**16.35**	600
berlin52	3790.96	3629.19	3822.98	**4016.77**	**17.08**	600
st70	**3481.09**	3091.14	3218.04	3344.93	**18.87**	600
eil76	**3620.88**	3134.96	3465.42	**3795.88**	**25.66**	600
rat99	**8211.38**	6203.88	7240.36	**8276.84**	**26.3**	600
kroA100	4259.31	4268.9	4352.39	**4435.88**	**26.54**	600
eil101	**5183.33**	4337.96	4669.07	5000.18	**22.29**	600
lin105	3826.58	3462.72	4588.14	**5713.56**	**22.97**	600
pr107	**7347.93**	5630.31	6725.81	**7821.3**	**25.14**	600
kroA150	7907.57	8135.86	8198.60	**8261.34**	**30.53**	600
d198	**14792.69**	10156.39	10913.23	11670.06	**61.94**	600
a280	**17053.37**	15539.35	16971.10	**18402.84**	**45.18**	600
rat783	34968.69	39366.77	40629.09	**41891.4**	**204.46**	600

and the computational times in seconds of the proposed DE (DB-DE) and the best, mean and worst objective values achieved by the 21 algorithms. In this table, the bold and underlined **values** indicate the best-obtained value, while the bold **values** indicate that the obtained value is better than the mean value. Consequently, from the table, we can notice that DB-DE obtained the best average objective values for 4 TTPs, and better values than the mean for other 4 TTPs.

Numerically, Table V demonstrates the competitive performance of the proposed DB-DE, which achieves objective values that are better or very close to those obtained from the best algorithm in [22]. For example, DB-DE obtained higher objective values than others, such in "**eil51**n50bounded", "**st70**n69bounded", "**eil101**n100bounded" and "**d198**n197bounded". On average, DB-DE achieved better objective values by 6.51% than the minimum value and is far from the maximum by 6.33%. On the other hand, the proposed DB-DE got its results in much less computational time than other algorithms, as reported in [22], the authors run all the algorithms for 10 minutes (600 seconds) as a time limit, while DB-DE spent, on average, 41.79 seconds for solving the 13 TTPs. Consequently, DB-DE could save more average computational time by 93.03% ($= \frac{600-41.79}{600} \times 100$) than other algorithms. Fig. 5 graphically presents the objective values obtained from DB-DE, DE with 3-Opt, the best and worst of 21 algorithms.

Moreover, the Friedman rank test [23] has been applied to locate the rank of the proposed DB-DE among the other 21 algorithms. The results in Table VI show that the DB-DE is ranked second in terms of the objective values and first for the computational time. Apparently, the average objective values of DB-DE can be improved by extending its run time limit up to 600 seconds.

TABLE VI
FRIEDMAN RANK TEST BASED ON THE AVERAGE OBJECTIVE VALUES AND COMPUTATIONAL TIMES

Algo.	Objective values		Computational time	
	Mean Rank	Order	Mean Rank	Order
DB-DE	2.08	2	1.00	1
Max	1.23	1	2.50	2
Mean	2.14	3	2.50	3
Min	2.69	4	2.50	4

V. CONCLUSION AND FUTURE WORK

In this paper, a new design of discrete-binary DE (DB-DE) for solving the multi-components TTPs, is proposed. Several functions have been adapted to enrich DE exploration and exploitation capabilities, such as mapping methods, repairing methods, ensemble of mutations and local searches. In DB-DE, a novel repairing method has been designed and adapted in the aim of reducing the time complexity of DE while solving complex problems, such as TTP. The proposed repairing method allows the DE evolutionary operators to only evolve

the solutions of one component (TSP-population) instead of evolving both KP and TSP solutions. At the end of each generation (before fitness evaluation), the repairing method is used to update the KP solutions, based on the locations of TSP cities, items' profits and weights. The produced KP solutions are optimally generated to fit the current visiting order of the cities and to achieve the maximum profit of the picked items. The experimental results showed that DB-DE has a very promising performance and can overcome many limitations of standard DE. Moreover, they claimed that the proposed repairing method has a great role in enabling DE to effectively solve TTPs and obtain good quality solutions, and radically reducing the computation time required by DB-DE to solve them. They also have demonstrated the applicability of DB-DE to solve complex problems with both binary and discrete decision variables (i.e. TTPs).

In the future, the performance of each component of the proposed DE will be more investigated in order to further enhance its capability to solve more complex COPs. Also, the proposed DE will be tested to solve COPs with multiple objectives.

REFERENCES

[1] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, pp. 341-359, 1997.

[2] K. R. Opara and J. Arabas, "Differential Evolution: A survey of theoretical analyses," Swarm and evolutionary computation, vol. 44, pp. 546-558, 2019.

[3] M. R. Bonyadi, Z. Michalewicz, and L. Barone, "The travelling thief problem: The first step in the transition from theoretical problems to realistic problems," in proceeding 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1037-1044.

[4] J. Stolk, I. Mann, A. Mohais, and Z. Michalewicz, "Combining vehicle routing and packing for optimal delivery schedules of water tanks," OR Insight, vol. 26, pp. 167-190, 2013.

[5] S. Polyakovskiy, M. R. Bonyadi, M. Wagner, Z. Michalewicz, and F. Neumann, "A comprehensive benchmark set and heuristics for the traveling thief problem," in proceeding Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 477-484.

[6] M. R. Bonyadi, Z. Michalewicz, M. R. Przybylek, and A. Wierzbicki, "Socially inspired algorithms for the travelling thief problem," in proceeding Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 421-428.

[7] Y. Mei, X. Li, and X. Yao, "Improving efficiency of heuristics for the large scale traveling thief problem," in proceeding Asia-Pacific Conference on Simulated Evolution and Learning, 2014, pp. 631-643.

[8] W. Zouari, I. Alaya, and M. Tagina, "A new hybrid ant colony algorithms for the traveling thief problem," in proceeding Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 95-96.

[9] C. Zhang, H. Yang, and J. Li, "Hybrid Discrete Differential Evolution Algorithm for Motor Train-Sets Scheduling," in proceeding 2018 37th Chinese Control Conference (CCC), 2018, pp. 3245-3248.

[10] P. B. Myszkowski, Ł. P. Olech, M. Laszczyk, and M. E. Skowroński, "Hybrid differential evolution and greedy algorithm (DEGR) for solving multi-skill resource-constrained project scheduling problem," Applied Soft Computing, vol. 62, pp. 1-14, 2018.

[11] I. M. Ali, D. Essam, and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," Swarm and Evolutionary Computation, vol. 52, p. 100607, 2020.

[12] I. M. Ali, D. Essam, and K. Kasmarik, "New Designs of k-means Clustering and Crossover Operator for Solving Traveling Salesman Problems using Evolutionary Algorithms," DOI: 10.5220/0007940001230130. In Proceedings of the 11th International Joint Conference on Computational Intelligence (IJCCI 2019), pages 123-130.

[13] I. M. Ali, S. M. Elsayed, T. Ray, and R. A. Sarker, "A differential evolution algorithm for solving resource constrained project scheduling problems," in proceeding Australasian Conference on Artificial Life and Computational Intelligence, 2016, pp. 209-220.

[14] I. M. Ali, D. Essam, and K. Kasmarik, "An Efficient Differential Evolution Algorithm for Solving 0–1 Knapsack Problems," in proceeding 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1-8.

[15] S. S. Reddy, "Optimal power flow using hybrid differential evolution and harmony search algorithm," International Journal of Machine Learning and Cybernetics, vol. 10, pp. 1077-1091, 2019.

[16] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," IEEE transactions on evolutionary computation, vol. 15, pp. 4-31, 2010.

[17] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," Journal of global optimization, vol. 27, pp. 105-129, 2003.

[18] I. M. Ali, D. Essam, and K. Kasmarik, "A novel differential evolution mapping technique for generic combinatorial optimization problems," Applied Soft Computing, vol. 80, pp. 297-309, 2019.

[19] Opara, K.R. and Arabas, J., 2019. Differential Evolution: A survey of theoretical analyses. Swarm and evolutionary computation, 44, pp.546-558.

[20] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," Operations research, vol. 21, pp. 498-516, 1973.

[21] C.-W. Chiang, W.-P. Lee, and J.-S. Heh, "A 2-Opt based differential evolution for global optimization," Applied Soft Computing, vol. 10, pp. 1200-1207, 2010.

[22] M. Wagner, M. Lindauer, M. Mısır, S. Nallaperuma, and F. Hutter, "A case study of algorithm selection for the traveling thief problem," Journal of Heuristics, vol. 24, pp. 295-320, 2018.

[23] Mack, G.A. and Skillings, J.H., 1980. A Friedman-type rank test for main effects in a two-factor ANOVA. Journal of the American Statistical Association, 75(372), pp.947-951.

[24] Mei, Y., Li, X. and Yao, X., 2016. On investigation of interdependence between sub-problems of the travelling thief problem. Soft Computing, 20(1), pp.157-172.