Imperialist Competitive Algorithm with Independence and Constrained Assimilation

Ivars Dzalbs
College of Engineering, Design and Physical Sciences
Brunel University London
London, United Kingdom

Tatiana Kalganova
College of Engineering, Design and Physical Sciences
Brunel University London
London, United Kingdom

Ian Dear
College of Engineering, Design and Physical Sciences
Brunel University London
London, United Kingdom

Abstract— **This work proposes an improved Imperialist Competitive Algorithm (ICA) based algorithm for solving constrained combinatorial problems, called ICA with Independence and Constrained Assimilation (ICAwICA). The proposed algorithm introduces the concept of colony independence – a free will to choose between classic ICA assimilation to the empire's imperialist or any other imperialist in the population. Furthermore, a constrained assimilation process has been implemented that combines classical ICA assimilation and revolution operators, while maintaining population diversity. In order to evaluate the performance and generalisation aspects of the proposed approach, two different kinds of combinatorial benchmark problems were selected – subset selection and routing, Multiple Knapsack Problem (MKP) and Multiple Depot Vehicle Routing Problem (MDVRP), respectively. The algorithm showed definite improvement over classic ICA and outperformed most of the competition on both types of problems across multiple instances, indicating the generic, universal nature of the ICAwICA. Moreover, it ranked 2nd among the recently published algorithms that are customised to the specific problem with the use of problem-specific operators, while the proposed algorithm had no such operators.**

Keywords— combinatorial optimisation, multidimensional knapsack problem (MKP), multi depot vehicle routing problem (MDVRP), imperialist competitive algorithm (ICA), meta-heuristics

I. Introduction

Optimisation is the process of finding the best solution among a pool of possible solutions. Optimisation is applied to a wide range of engineering, economic and even social systems to minimise cost or maximise profits. There is no single optimisation technique that can be efficiently applied across all optimisation problems. Hence several optimisation methods have been developed for different kinds of optimisation problems [1]. A metaheuristic is one of such methods; it offers a near-optimal solution in less compute time than exact methods [2]. Metaheuristics include algorithms such as Genetic Algorithm (GA) [3], Ant Colony Optimization (ACO) [4], Particle Swarm Optimization (PSO) [5], more recently Imperialist Competitive Algorithm (ICA) [6], etc.

Imperialist Competitive Algorithm (ICA), intensively researched during the last decade, is a subset of metaheuristic algorithms that are modelled based on geopolitical behaviour. It can also be classified as a social Darwinism that follows evolutionary computing principles. Atashpaz-Gargari and Lucas first proposed the ICA in [6] for solving continuous cost functions and since the algorithm has generated interest amongst many researchers. Its application can be found in various engineering disciplines – scheduling, assembly line balancing, facility layout optimisation, computer engineering and other areas of industrial engineering [7]. Most recently, applications such as prediction [8][9], clustering [10] and encryption [11] have emerged, demonstrating the versatility and wide application areas of the algorithm.

There have been various attempts on improving the standard ICA search performance. For example, authors in [12] proposed an adaptive ICA (AICA) that uses a probabilistic model based on colony positions to escape local optimum. Similarly, [13] improved the convergence speed of the algorithm by adding additional value to an unfeasible solution, based on its distance from the relative imperialist. Both [14] and [15] enhanced ICA by implementing an attraction and repulsion concept during the search for better solutions. Less researched area is the use of local search in ICA. Local search has been used to improve convergence on other metaheuristics, such as in Ant Colony System [4] by local pheromone update rules, or small swarm division in PSO [16]. The standard ICA does not implement any form of local search and therefore, may get stuck in local optima before converging to the global best solution [17]. Only few approaches for solving this problem have been proposed in the literature, such as simulated annealing-like processes in [18], where the local search process is applied for machine-selection part and the operation-sequence part in flexible job-shop problem (FJSP). The 2-opt is another popular local-search operator for routing problems, such as Travelling Salesman Problem (TSP). For example, work in [19] uses 2-opt with ICA to improve the imperialists. For continuous optimization problems, local search operator such as *random line search* has been explored in [20], where authors applied the problem-specific local search for the imperialist solutions.

However, many of these local search implementations rely on problem-specific operators or assimilation. These operators exploit the underlying problem dynamics and are an effective way to improve the convergence. Although some can be transferrable across similar class problems, they are rarely generic enough to be applied for a wide range of problems. For example, a 2-opt local search would be of no use for a knapsack problem. In attempt to overcome this issue, paper proposes a modified ICA, where the local search process is performed in terms of both an Independence operator and a Constrained Assimilation (ICAwICA). Compared to existing ICA local search approaches, ICAwICA proposes a more generic implementation that does not require problem-specific operators.

Thus, in this paper, we present a more generic algorithm with a local search that expands on the classic ICA, with the use of novel Independence operator and Constrained Assimilation, called ICAwICA. The contributions can be summarized into the following to aspects:

- A novel generic ICA is proposed, where the standard assimilation and revolution process is replaced with constrained assimilation and the novel independence operator used for local search.
- The performance of the ICAwICA algorithm is comprehensively evaluated via well-known Multiple Knapsack Problem (MKP) and Multi Depot Vehicle Routing Problem (MDVRP) benchmark instances. The experiment results demonstrate the superiority over classic ICA and universality of the local search.

The rest of the paper is structured as follows: first, both MKP and MDVRP are introduced and formalised. Next, in section two, standard ICA, as well as proposed ICAwICA, is presented with example applications. Experimental setup, benchmarks used, and comparative results to the state-of-the-art approaches for both MKP and MDVRP are described in section three. And finally, the last section concludes the paper.

A. Mulktiple Knapsack Problem (MKP)

The Multidimensional Knapsack Problem (MKP) is a well-known constrained optimisation problem, that has multiple real-world engineering applications, such as cutting stock [21], distributed computing resource allocation [22], cargo loading [23], satellite management [24], project selection [25] and capital budgeting [26]. The MKP is an extension of the 0-1 knapsack problem, where items have weight vectors in multiple dimensions. The goal is to maximise the total profit by putting items into knapsacks while satisfying weight capacity constraints across all dimensions. MKP is formulated in (1) [27].

$$\begin{aligned} &\max && \sum_{y=1}^{n} p_y s_y && \\ &\text{subject to:} && \sum_{y=1}^{n} w_{zy} s_y \le W_z && \forall z \in \{1,\dots,m\} \\ & && s_y \in \{0,1\} && \forall y \in \{1,\dots,n\} \end{aligned} \tag{1}$$

where every item y in the list of n items ($y = 1 \ldots n$) has a profit p_y and weight w_{zy} associated with an m-dimensional weight vector ($z = 1 \ldots m$), that tries to satisfy a capacity constraint W_z in that dimension. Variable s_y indicates whether the item is selected and included in the solution. Capacities, weights and profits are assumed to be positive.

Being an NP-hard problem with practical applications, many different approaches have been proposed for solving MKP, which can be divided into two groups – exact, deterministic, single-solution based algorithms and stochastic population/meta-heuristic based algorithms, with this paper focusing on the latter approach.

Comprehensive literature review of solving MKPs was provided by [28], and a more recent MKP overview by [29] summarises algorithms used for solving MKP. This paper focuses on the state-of-the-art population and meta-heuristic algorithms used for solving MKP instances, such as ant colony optimisation [18][19], various types of genetic algorithms [32][33][34], evolutionary algorithms [35][36] [37], variations of particle swarm optimisation algorithm [38][39], binary harmony search [40], binary cuckoo search algorithm [41], whale optimisation algorithm [42] and alike. Most of the research in population-based algorithms focus on small MKP instances with $n \le 100$, while only a few explore large instances with $n = 500$ and above. This paper tries to cover both small and large instances of MKP for a comprehensive study.

B. Multi Depot Vehicle Routing Problem (MDVRP)

The Vehicle Routing Problem (VRP), first described in 1959 [43], is an extension of the Traveling Salesman Problem (TSP) [44]. Compared to TSP, where an agent has only to visit all cities once, VRP introduces demands for each customer or stop. Demands need to be satisfied by routing vehicles such that they start and finish their paths at the same depot. Many real-life problems can be modelled as a form of VRP, for example, picking up and delivering mail, packages or any other goods or services. Due to the wide range of practical applications, many variations of VRP have since been explored. For instance, capacitated VRP introduces capacity constraints on the vehicles; VRP with Time Windows (VRPTW) requires delivery to happen within a specific time window; VRP with maximum vehicle distance constraints (DVRP) and many others [45].

A common VRP derivation is the Multi-Depot Vehicle Routing Problem (MDVRP). MDVRP is an extension of classical VRP by the introduction of multiple depots. Vehicles in the MDVRP are subject to capacity constraints (how much cargo can be carried on board) and the maximum duration for the route before the vehicle needs to return to the original depot. The MDVRP resembles a lot of everyday transportation, logistics and distribution problems and, therefore, has been a common research area [46]. Furthermore, the MDVRP is also an NP-hard combinatorial optimisation problem; thus, optimal solutions are hard to find [47]. Although exact algorithms for solving these classes of problems exist, they are limited to small problem instances [48]. A wide range of metaheuristics and population-based algorithms have been used [46] to solve larger instances of the MDVRP.

The main aim of the MDVRP is to route a fleet of vehicles from multiple depots to multiple customers requiring goods or services.. Fig. 1 shows an example of a simple MDVRP solution with ten customers (as circles) and two depots (as rectangles). Although there exist multi-objective approaches for solving MDVRP [49], the most common goal is to minimise the total cost.

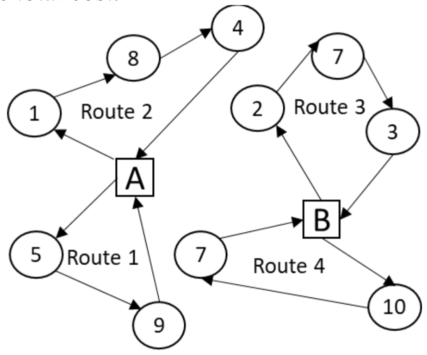


Fig. 1. Example of an MDVRP with ten customers (as circles) and two depots (A and B as rectangles)

The MDVRP can be formalised in a mathematical model based on [50] and [51]. Given a direct graph $G = (S, E)$ where $S = C \cup D$ is a set of customers $C = \{C_1, C_2, \dots, C_N\}$ and depots $D = \{D_1, D_2, \dots, D_M\}$ and E is a set of edges between all the nodes in the graph. In a fully connected graph, every edge E_{ij} between nodes S_i and S_j $(i \neq j)$ has associated positive cost c_{ij} - distance or time, for example. Each customer has a positive demand d_i ($i \in C$). Furthermore, there is also a fleet of K identical vehicles available at each depot $D_k \epsilon D$ (that are not allowed to exceed capacity Q_{max} and duration R_{max}). The goal is to minimise the total cost across all vehicles (2).

$$min \sum_{i \in S} \sum_{j \in S} c_{ij} x_{ij} \tag{2}$$

where x_{ij} equals to 1 if i comes after j in the customer sequence on any route of all vehicles and 0 otherwise. The problem is subject to the following constraints a) each vehicle route starts and ends at the same depot; b) the total demand on each route does not exceed vehicle capacity Q_{max}; c) the maximum route duration R_{max} is not exceeded; e) each customer is served by exactly one vehicle.

Since the first formulation in [43], many exact and heuristic algorithms have been explored for vehicle routing problems. Most notably, [52] proposed a heuristic approach based on the cost savings algorithm that has since been used in some form in many other algorithms [53]. Another popular heuristics approach was introduced in [54] that allowed problems divided into sub-problems based on vehicles and then solved separately, combining results into a single solution afterwards. Although heuristic approaches such as integer programming [55] and variable neighbourhood search [56] have the potential to find optimal solutions every time, they generally do not scale well with the problem size and are limited to smaller MDVRP instances or are very time-consuming [48].

Meta-heuristic algorithms offer a stochastic approach for solving highly complex combinatorial problems with near-optimal or optimal solutions. They have been a growing interest in many areas [57], and MDVRP is no exception. A recent survey of metaheuristic algorithms [46] suggests that two of the most common algorithms used for solving MDVRP are Ant Colony Optimization (ACO) and Genetic Algorithm (GA). However, other algorithms like Particle Swarm Optimization (PSO) [58] and Ant Lion Optimization (ALO) [59] have also been successfully applied. GA is a nature-inspired algorithm that is based on the natural selection process. A comprehensive summary of methods and approaches used for solving MDVRP with GA is presented in [45]. ACO is another popular approach for solving VRP class problems as it mimics ants travelling and searching for food while creating paths for other ants to follow. Many implementations of ACO for MDVRP exist in the literature; the most recent work includes [60] who applied the ACO algorithm for fresh seafood delivery routing problems.

II. The Imperialist Competitive Algorithm with Independence and Constrained Assimilation

The following section introduces the classic Imperialist Competitive Algorithm (ICA) and the novel ICA with Independence and Constrained Assimilation (ICAwICA) algorithm. It discusses the changes and advantages of constrained assimilation. Finally, ICAwICA application to two different example problems is considered.

A. Classic ICA

Like many other population algorithms, ICA starts its search by generating a random initial population where each individual of the population represents a country. Countries within ICA can be thought of as chromosomes in a genetic algorithm. The initial population is separated into multiple groups (so-called empires). Most influential countries become imperialist within the empire and weakest - their colonies. Each colony within empire moves closer to their imperialist in the form of assimilation operator. In order to provide diversity amongst countries, a revolution operator (mutation in GA) is implemented. If at any point a colony becomes stronger than its imperialist, then the two countries are swapped, such that imperialist is the strongest country in the empire. The search follows an iterative process, where after each iteration, the weakest colony within the weakest empire is assigned to one of the stronger empires – following the imperialist competition process. An empire is eliminated once it contains no more colonies. The search usually continues until the termination criteria are met. Ideally, the search is terminated once all empires are eliminated and only one, the best, empire remaining.

B. ICAwICA

The proposed ICAwICA follows the classic ICA [6] principles for both empire initialisation and empire competition; however, assimilation and revolution operators are replaced with a constrained assimilation and repair mechanism. Furthermore, in the classic ICA, each colony within an empire is moving closer to the imperialist within that empire. In contrast, in ICAwICA all colonies are given a free choice to move closer to any of the imperialists of other empires (independence), as long as it improves the country's well-being (associated cost). Therefore, at each iteration, a colony k has a probability based on a uniform distribution ($rand$) of either move closer to their own empire's imperialist or to move closer to any other imperialist j, determined by $independanceRate$ (0-1.0). Moreover, this process is repeated $N_{localIter}$ times for each colony to explore more search space around its position in the form of local search. Pseudocode of the ICAwICA is shown in Fig. 4. The flowchart for both classic ICA and ICAwICA is shown in Fig. 2, with red indicating the changes.

C. Constrained Assimilation

Classic ICA was first developed for continuous math's problem with simple assimilation processes [6], ICA has since been applied to multiple binary problems, such as feature selection [61][62], content-based-image retrieval (CBIR) [63] and single-dimensional 0-1 knapsack problems [64]. However, binary assimilation approaches cannot always be extended to other discrete, non-binary problems.

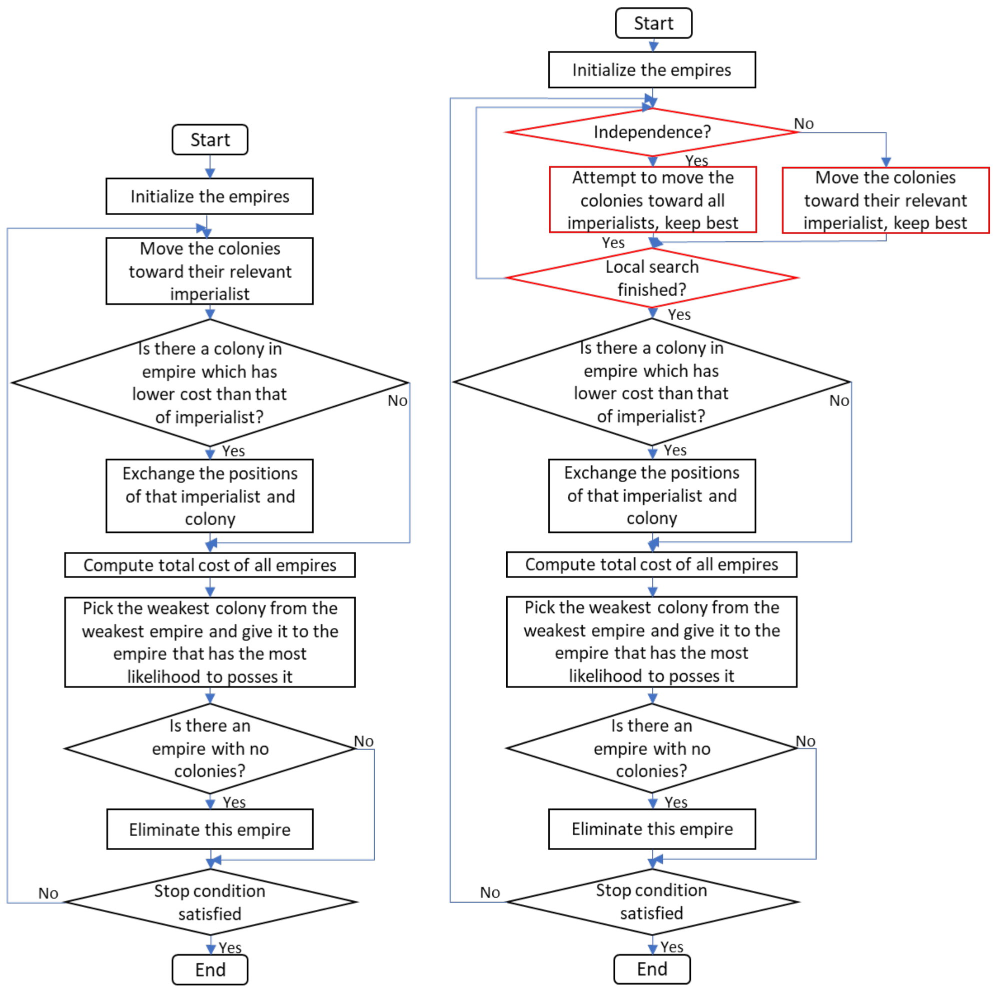
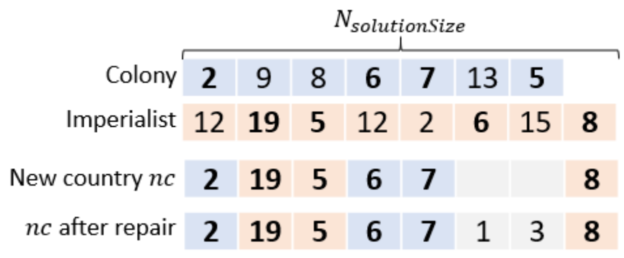

Start
Initialize the empires
Move the colonies toward their relevant imperialist
Is there a colony in empire which has lower cost than that of imperialist?
Yes
No
Exchange the positions of that imperialist and colony
Compute total cost of all empires
Pick the weakest colony from the weakest empire and give it to the empire that has the most likelihood to posses it
Is there an empire with no colonies?
Yes
No
Eliminate this empire
Stop condition satisfied
No
Yes
End
Start
Initialize the empires
Independence?
No
Yes
Attempt to move the colonies toward all imperialists, keep best
Move the colonies toward their relevant imperialist, keep best
Yes
Local search finished?
Yes
Is there a colony in empire which has lower cost than that of imperialist?
Yes
No
Exchange the positions of that imperialist and colony
Compute total cost of all empires
Pick the weakest colony from the weakest empire and give it to the empire that has the most likelihood to posses it
Is there an empire with no colonies?
Yes
No
Eliminate this empire
Stop condition satisfied
No
Yes
End

Fig. 2. Flowchart of classic ICA [6] (to the left) and the proposed ICAwICA (on the right), with red indicating the changes.

Furthermore, most ICA discrete assimilation implementations follow simple genetic-algorithm-like crossover operations, where the chromosomes are expected to be of equal size [65] [66]. The proposed Constrained Assimilation (CA) process does not require equal chromosome/solution size and is extendable to other constrained discrete problems. CA exploits the fact that two solutions cannot always be merged without violating constraints. Therefore, CA builds a new incomplete solution from the two donor solutions/countries (colony and imperialist) according to the assimilation rate and finishes the solution by a repair mechanism.

There are multiple ways to implement the solution repair mechanism - based on heuristics, existing solution population, sequence-based [67] etc. The most straightforward repair mechanism is - scanning through all possible entries and trying to add them to the solution without violating constraints (used in this paper). Furthermore, this incomplete solution repair enables diversity without an explicit revolution operator like classic ICA. Although more computationally expensive than simple assimilation, this approach has potential for broad applications and generalisation, as it does not depend on two solutions having the same size nor problem-specific assimilation or repair mechanism. Furthermore, the generated solutions with CA is always within constraints and does not require any penalty cost definition at evaluation.

A CA example is provided in Fig. 3 where both colony and imperialist are assimilated, with bold integer values corresponding to solution entries (item indices in MKP case, or depo indices in MDVRP case) that are passed to the new country, determined by assimilation rate. In this simple example, a 50% assimilation rate of $N_{solutionSize}$ is used to build the new country. Due to constraints, not all solution entries can be added to the new country and hence the solution is in an incomplete state. The repair process iterates over all possible solution entries and fills the gaps while complying with constraints. Let us consider in detail the assimilation process shown in Fig. 3. The colony solution is shown in blue and the imperialist in yellow, with the newly generated country nc, new entries (index 1 and index 3) were introduced to the solution after repair that were not in any of the donor countries



	$N_{solutionSize}$							
Colony	**2**	9	8	**6**	**7**	13	5	
Imperialist	12	**19**	**5**	12	2	6	15	**8**
New country nc	**2**	**19**	**5**	**6**	**7**			**8**
nc after repair	**2**	**19**	**5**	**6**	**7**	1	3	**8**

Fig. 3. Imperialist and colony constrained assimilation process with solution repair. With integer values corresponding to solution entries (item indices in MKP case or depo indices in MDVRP case).

```
1. Initialize ICA parameters.
2. Create the population randomly.
3. Initialize empires:
  for $i = 1$ to $N_{population}$
        Compute the cost function $C_i$;
        Sort the computed cost $C_i$ in descending order for the entire population;
        Select $N_{imperialists}$ out of $N_{population}$;
        Normalize the cost of each imperialist $C_n$;
        Calculate the normalized power of each imperialist $P_n$;
        Assign remaining countries $N_{countries}$ to the imperialists;
  end loop
do
   4. Assimilation and local search process for ICAwICA:
     for k $= 1$ to $N_{colonies}$
       for $l = 1$ to $N_{localIter}$
         if $rand < independanceRate$
            for $j = 1$ to $N_{imperialists}$
               assimilate colony $k$ closer to $j$
               if cost for new position is less than original position
                  keep assimilated position
               else
                  discard and move back to original position
               endif
            end loop
         else
           assimilate colony $k$ closer to empire's Imperialist
         endif
       end loop
     end loop
     for $j = 1$ to $N_{imperialists}$
        if the cost of any colony is less than cost of imperialist
             exchange the position of the colony and imperialist;
        endif
     end loop
   Pick the weakest colony (colonies) from the weakest empire and
   assign it to the empire with highest probability to possess it;
  5. Elimination process:
     If there is imperialist with no colonies
        eliminate the imperialist;
     endif
while stopping condition not met;
```

Fig. 4. The pseudocode for new assimilation and local search method for ICAwICA

D. *ICAwICA solution encoding for MKP and MDVRP*

The ICAwICA is generic and does not rely on any specific solution structure or problem-specific assimilation operators and, therefore, can be applied to various kinds of discrete optimisation problems. We explore two different types of combinatorial problems – a subset selection problem in MKP and a routing problem in MDVRP. In the MKP case, each element in the solution represents an item index that has been added in the knapsacks. Thus, the performance of the solution is evaluated by iterating over all entries and matching indices to the item profits.

For the MDVRP, first, customer-depot relationships are encoded as a country. Each country is represented as a vector of the size of the number of customers, where each customer is assigned a depot index. An example of new country creation via assimilation for the MDVRP is shown in Fig 5, where the initial colony has encoded the following grouping: Customer 2 and 8 will be routed from Depot 1; Customers 1, 3 and 6 will be routed from Depot 2; Customers 5,7,9 and 10 will be routed from Depot 3, and finally, Customer 4 will be routed from Depot 4. Each time a new country is created as part of the ICAwICA assimilation process, capacity constraints are considered such that the total demand for all customers assigned to the depot does not exceed the maximum capacity available across all vehicles to the given depot.

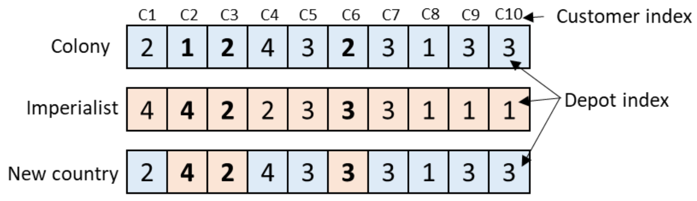


Fig 5. Customer assignment to depots in MDVRP using ICAwICA assimilation. Where C1-C10 are customer indices and the encoded integers are depot indices that are assigned to a given customer, with bold representing assimilated changes.

Furthermore, the example in Fig 5 also shows an assimilation process for the colony and imperialist; considers ten customers that are grouped into four depots. Bold type represents assimilated changes. For example, Customer 2 (C2) demand was previously supplied by Depot 1 but now is supplied by Depot 4. Similarly, Customer 6 (C6) demand was previously supplied by Depot 2 but now is supplied by Depot 3.

Finally, solution performance is evaluated by first grouping all depot indices in the solution, then constructing routes based on the sequence it was added to the solution (from left to right). Thus, in the example in Fig 5, the new country solution would be Depot 1 supplying customer 8, Depot 2 supplying customers 1 and 3, Depot 3 supplying customer sequence 5-6-7-9-10, and finally, Depot 4 supplying customers 2 and 4.

III. EXPERIMENTS

In this section, the proposed ICAwICA algorithm performance is compared to classic ICA. Next, the dynamics of independence operator are analysed. Finally, extensive computational experiments on classical MKP and MDVRP benchmark instances are conducted and compared to the current state-of-the-art algorithms.

A. *Benchmark instances*

Multidimensional knapsack problem instances were chosen because of their availability, ease of implementation and the frequent use as benchmarks across the research community. ICAwICA was tested across 41 accessible benchmark instances, all available from the compiled library in [68].

The simplest benchmarks used in this paper are derived from the WEISH dataset, containing 30 problems with the number of items ranging from 30 to 90 and with five knapsacks each. Furthermore, to explore the performance of the proposed algorithm across a range of datasets, large MKP instances, generated by Glover and Kochenberger (GK) [69], were also selected. The GK dataset contains 11 instances with the number of items ranging from 100 to 2500 with 15 to 100 knapsacks each and provides a broad spectrum of complexity.

Moreover, the ICAwICA was also tested on the 23 Cordeau's MDVRP benchmark instances obtained from [70]. The benchmark dataset offers a wide range of complexity, from the number of customers ranging from 50 to 360 and the number of depots from 2 to 9; and specifies the current Best-Known Solution (BKS).

TABLE 1. COMPARISON OF BEST AND AVERAGE SCORES BETWEEN CLASSIC ICA AND ICAWICA ACROSS SIX TEST PROBLEM INSTANCES. AVERAGE AND BEST OUT OF 10 RUNS WITH STANDARD DEVIATION (STD), BKS – BEST KNOWN SOLUTION.

Dataset	BKS	Classic ICA [6]			ICAwICA		
		Average	Best	Std	Average	Best	Std
MKP-gk01	3766	3753.8	3766	8.11	**3766.0**	**3766**	0.00
MKP-gk03	5656	5631.5	5638	5.12	**5649.2**	**5650**	0.90
MKP-gk06	7680	7629.7	7639	8.16	**7669.7**	**7671**	1.19
MDVRP-p01	576.87	587.20	580.70	8.92	**576.87**	**576.87**	0.00
MDVRP-p03	641.19	658.10	645.16	7.55	**655.29**	**641.19**	3.25
MDVRP-p06	876.5	893.80	885.84	10.83	**887.71**	**876.50**	3.93

B. Experimental setup

The proposed ICAwICA algorithm was implemented in C++ using the Visual Studio 2019 (v142) compiler. The computation was performed on a workstation with AMD Threadripper 2990WX processor (3.0 GHz, 64GB RAM), running Windows 10 Pro operating system.

Like classic ICA, ICAwICA also has multiple algorithmic hyper-parameters that were empirically set and are as follows for all tested instances unless specified otherwise:

MKP - total number of countries $N_{population}$ is set to 4096 for all instances with the number of items $n < 500$ and value of 512 for all instances with $n \geq 500$. Out of all countries, 40% are initialised as imperialists $N_{imperialists}$. Local iterations $N_{localIter}$ are set to 3. Assimilation rate β set to 0.5; coefficient associated with an average power of empire's colonies ξ set to 0.05; $independanceRate$ set to 0.7 (70% probability of independence). Due to constrained computing resources, limited time and a large problem set, termination criteria of stagnation were implemented, where the search terminates if no improvement has been made to the best solution for ε number of iterations. For problem instances with $n < 500$, ε is set to $0.1n$, and for MKP instances with $n \geq 500$, $\varepsilon = n$.

MDVRP - the total number of countries $N_{population}$ is set to 4096 for all instances. Out of all countries, 40% are initialised as imperialists $N_{imperialists}$. Local iterations $N_{localIter}$ are set to 16. Assimilation rate β set to 0.05; coefficient associated with an average power of empire's colonies ξ set to 0.05; $independanceRate$ set to 0.7 (70% probability of independence). Finally, stagnation iterations ε set to 10.

Due to the stochastic nature of the algorithm, 30 independent runs were computed for each problem instance. Best and average solution performance, as well as the average time in seconds $t_{avg}(s)$ (average time in minutes $t_{avg}(m)$) required to reach such performance value, were recorded for all problem instances.

C. Comparison to classic ICA

Novel ICAwICA was first compared to classic ICA based on [6]. Three problem instances from both MKP (gk01, gk03, gk06) and MDVRP (p01, p03, p06) were selected for comparison, and the results are summarised in Table 1.

Results show a significant improvement in the best scores obtained - ICAwICA reaching best-known solution (BKS) in four out of six instances, while classic ICA only once. Furthermore, average scores are consistently higher, and the standard deviation suggests that ICAwICA results are also more consistent. It is worth noting that MKP objective is to maximise profit, while MDVRP is to minimise the total route cost. Therefore, the average error gap (3) against the best-known solution is used for easier comparisons and are summarised in Fig. 6. The average error for ICAwICA is consistently smaller than classic ICA across all six test instances.

$$Average\ error\ gap\ (\%) = \frac{1}{n}\sum_{i=1}^{n}\frac{o_i - p_i}{o_i} * 100\% \qquad (3)$$

where o_i is the optimal score for instance i, and p_i – achieved best or average score on the instance.

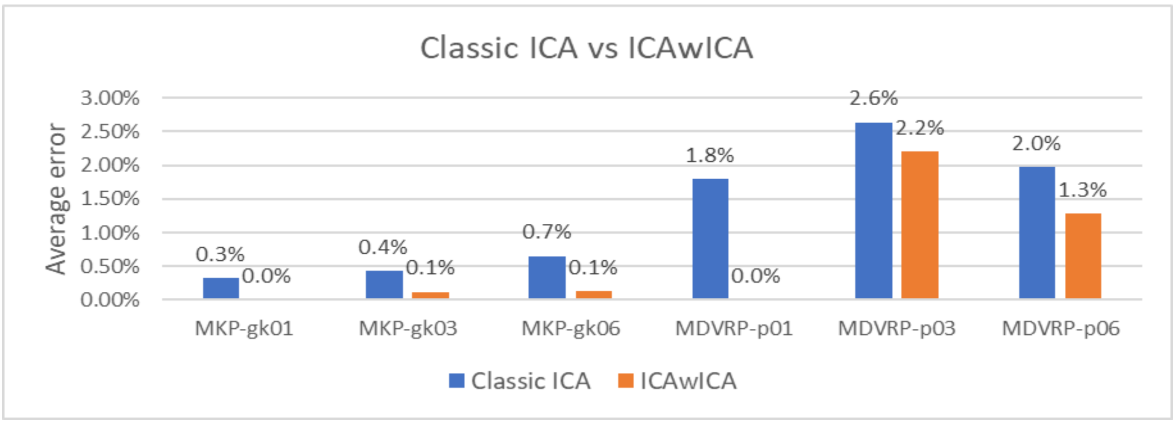


Fig. 6. Comparison between Classic ICA [6] and ICAwICA for six test problem instances. Expressed as average error percentage to the best know solution. The graph demonstrates ICAwICA achieves average erorr of 0.62% while Classic ICA achieves 1.3%, relative improvement of over two times.

TABLE 2. SENSITIVITY ANALYSIS OF INDEPENDENCE RATE AS AN AVERAGE ERROR PERCENT GAP FOR SIX TEST PROBLEM INSTANCES. WITH 0 REPRESENTING ICA WITH NO INDEPENDENCE OPERATOR, $t_{avg}(s)$ REPRESENTING THE AVERAGE TIME IN SECONDS TO CONVERGE TO THE BEST SOLUTION, BKS – BEST KNOWN SOLUTION

		Independence rate					
Dataset	**BKS**	**0**	**0.2**	**0.4**	**0.6**	**0.8**	**1**
MKP-gk01	3766	3.02%	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
MKP-gk03	5656	2.75%	0.13%	**0.12%**	**0.12%**	**0.12%**	**0.12%**
MKP-gk06	7680	2.36%	0.34%	0.20%	0.14%	**0.12%**	**0.13%**
MDVRP-p01	576.87	4.79%	0.61%	0.04%	**0.00%**	**0.00%**	**0.00%**
MDVRP-p03	641.19	10.98%	3.07%	2.67%	**2.12%**	**2.12%**	2.17%
MDVRP-p06	876.5	7.79%	2.72%	1.53%	**1.25%**	1.36%	1.46%
Average error		5.28%	1.14%	0.76%	**0.61%**	**0.62%**	0.65%
$t_{avg}(s)$		40	570	836	1033	1315	1524

D. Sensitivity analysis of independence rate

The newly implemented mechanism of colony independence was tested by altering the $independanceRate$ parameter from 0 to 1, with 0.2 increments and the average error gap (3) as well as execution time $t_{avg}(s)$ recorded. The experimental results are summarised in Table 2.

Results in Table 2 show a definite improvement in the introduction of the Independence operator within ICA. Compared to ICA with no independence (independence rate of 0) and ICA with independence rate higher than 0, the average error across all test instances reduced by a factor of 4.6 (5.28% and 1.14% respectively). However, there is also a time penalty associated with doing the extra work of assimilating to all imperialists compared to a single imperialist, with an average time to reach the final solution increasing from seconds to minutes. The best average error was achieved with the Independence rate between 0.6 and 0.8, and therefore independence rate at 0.7 was adopted for use throughout all further experiments.

E. Comparison to the state-of-the-art metaheuristics for MKP

To evaluate performance of the proposed ICAwICA algorithm, 12 state-of-the-art population-based/heuristic algorithms were compared across 41 common MKP instances.

First, a comparison was performed on simple WEISH instances, where most algorithms in the literature can achieve the optimum solution. Therefore, performance is measured in terms of the success rate (how many times the algorithm was able to achieve optimum) or in terms of the average error percentage error (3) across all instances. For the comparison, the six best-performing algorithms were selected from the literature, which includes Ant Colony Optimization with Dynamic impact (ACOwD) described in [31], Improved Whale Optimization Algorithm (IWOA) [42], two variations of binary differential search TE-BDS and TR-BDS proposed in [71], and two implementations of Particle Swarm Optimization (PSO) with self-adaptive check and repair - SACRO-CBPSOTVAC and SACRO-BPSOTVAC [39].

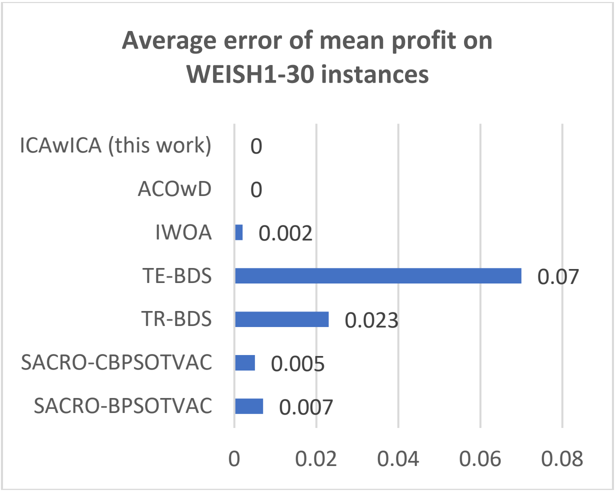


Fig. 7. The average error of the mean profit across all WEISH (1-30) instances. Average of 30 independent runs.

Results in Fig. 7 show that all compared algorithms can reach the optimal solution in most cases. However, only 2 of them ICAwICA and ACOwD can do it consistently across 30 runs with 100% success rate. ICAwICA achieved the optimal solution every time (100% success rate), at the first iteration, and on average took 1.5 seconds.

Next, large Glover and Kochenberger (GK) instances were solved and compared to eight heuristic algorithms from the literature in terms of average error percent (3) gap against best-known profit (BKS) from the literature. Compared algorithms include ACOwD, IWOA, Two-phase tabu-evolutionary algorithm (TPTEA) [35], harmony search based algorithm NBHS2 proposed in [40], evolutionary algorithm with logic gates LGEA [36], shuffled complex evolution algorithm SCEcr [37], hyper-heuristic inspired CF-LAS [72] and BCSA – binary cuckoo search algorithm [41]. Table 3 is colour coded from red (worst average error %) to the best average error percent, in green, for each problem instance with dashes (-) representing scores that were not available. Compared to 8 other algorithms in the literature, ICAwICA shows competitive results, coming in second place for gk01-gk09 and in top three for gk10 and in fourth place for the largest gk11 instance. The best achieved error percentage along with the average time $t_{avg}(s)$ and standard deviation (Std) have been included for reference. The proposed algorithm performs well on medium to large MKP instances, however, struggles on very large instances (gk11). Further investigation needs to be conducted to improve performance on the most complex benchmarks.

TABLE 3. ALGORITHM COMPARISON ACROSS LARGE GLOVER AND KOCHENBERGER (GK) KNAPSACK INSTANCES. RESULTS ARE EXPRESSED AS AVERAGE ERROR PERCENTAGE GAP % AGAINST BEST-KNOWN PROFIT. COLOUR CODED FROM THE BEST GAP (GREEN) TO WORST GAP (RED) FOR ANY GIVEN DATASET. WITH DASH (-) REPRESENTING RESULTS THAT ARE NOT AVAILABLE. BKS – BEST KNOWN SOLUTION, STD – STANDARD DEVIATION OF THE ABSOLUTE VALUE.

Data set	Problem size (n x m)	BKS	ACOwD [31]	NBHS2 [40]	IWOA [42]	LGEA [36]	TPTEA [35]	SCEcr [37]	CF-LAS [72]	BCSA [41]	ICAwICA (this work) Average	Best	Std	$t_{avg}(s)$
gk01	100x15	3766	0.14%	0.29%	0.68%	0.66%	0.00%	0.76%	0.31%	0.23%	0.00%	0.00%	0.00	16.8
gk02	100x25	3958	0.05%	0.30%	-	0.55%	0.00%	1.06%	0.36%	0.27%	0.05%	0.03%	0.99	19.4
gk03	150x25	5656	0.26%	0.55%	0.85%	0.97%	0.06%	0.91%	0.37%	0.17%	0.12%	0.11%	0.90	62.5
gk04	150x50	5767	0.17%	0.45%	0.89%	1.02%	0.01%	1.48%	0.45%	0.15%	0.07%	0.05%	0.93	84.4
gk05	200x25	7561	0.21%	0.44%	0.94%	1.32%	0.01%	0.73%	0.24%	0.18%	0.09%	0.04%	1.54	145.7
gk06	200x50	7680	0.26%	0.52%	0.77%	1.05%	0.08%	1.14%	0.46%	3.54%	0.13%	0.12%	1.19	247.7
gk07	500x25	19221	0.20%	0.26%	1.09%	1.08%	0.04%	0.46%	0.13%	0.70%	0.11%	0.07%	5.89	280.3
gk08	500x50	18806	0.22%	0.56%	0.85%	-	0.06%	0.67%	0.20%	0.77%	0.12%	0.08%	2.98	357.8
gk09	1500x25	58091	0.18%	0.27%	1.54%	1.08%	0.02%	1.78%	1.77%	0.98%	0.14%	0.09%	14.61	1611.0
gk10	1500x50	57295	0.20%	0.54%	0.80%	1.01%	0.04%	0.36%	0.10%	-	0.18%	0.12%	13.67	2219.1
gk11	2500x100	95238	0.32%	0.64%	1.07%	1.13%	0.07%	0.30%	0.09%	-	0.31%	0.24%	61.54	7200.6
		Average	0.20%	0.44%	0.95%	0.99%	0.04%	0.88%	0.41%	0.78%	0.12%	0.09%	9.48	1113.2

TABLE 4. BEST SOLUTION OBTAINED BY ICAWICA COMPARED TO OTHER ALGORITHMS IN THE LITERATURE ACROSS CORDEAU'S MDVRP BENCHMARK INSTANCES AND THE BEST-KNOWN SOLUTION (BKS). THE BEST SCORES REPRESENTED IN BOLD, N REPRESENTING THE NUMBER OF CUSTOMERS, M – THE NUMBER OF DEPOTS. AVERAGE ERROR PERCENTAGE CALCULATED USING BKS AS A REFERENCE, $t_{avg}(m)$ – AVERAGE TIME TO CONVERGE TO A SOLUTION, IN MINUTES, STD – STANDARD DEVIATION

Data set	N	M	BKS	CoES, 2016 [73]	IACO, 2017 [60]	TSH, 2019 [74]	ACO+, 2020 [51]	ICAwICA (this work) Best	Average	Std	$t_{avg}(m)$
p01	50	4	**576.87**	**576.87**	**576.87**	**576.87**	**576.87**	**576.87**	576.87	0.00	4.2
p02	50	4	**473.53**	**473.87**	**473.53**	**473.53**	**473.53**	**473.53**	481.24	3.00	6.2
p03	75	5	**641.19**	**641.19**	**641.19**	**641.19**	**641.19**	**641.19**	655.29	3.25	7.9
p04	100	2	1001.59	1007.40	**1001.49**	1008.47	1003.52	1006.66	1015.11	3.97	12.4
p05	100	2	**750.03**	**750.11**	750.26	758.87	751.90	753.40	789.15	4.39	20.3
p06	100	3	**876.50**	**876.50**	**876.50**	881.76	881.60	**876.50**	887.71	3.93	14.7
p07	100	4	885.80	888.41	885.69	896.96	**884.66**	895.53	916.79	8.12	11.5
p08	249	2	**4420.94**	4445.37	4482.44	4430.36	4428.00	**4420.94**	4493.66	17.87	65.2
p09	249	3	3900.22	3895.70	3912.23	3971.59	**3897.33**	3900.22	3975.29	23.52	67.6
p10	249	4	3663.02	3666.35	3663.00	3779.10	**3657.03**	3666.35	3696.71	10.88	82.2
p11	249	5	3554.18	3569.68	3648.95	3652.01	**3549.99**	3554.18	3604.88	22.75	71.0
p12	80	2	**1318.95**	**1318.95**	**1318.95**	**1318.95**	**1318.95**	**1318.95**	1359.49	4.88	10.0
p13	80	2	**1318.95**	**1318.95**	**1318.95**	**1318.95**	**1318.95**	**1318.95**	1320.79	0.82	8.9
p14	80	2	**1360.12**	**1360.12**	1365.68	1365.69	**1360.12**	1365.68	1394.01	6.71	6.7
p15	160	4	**2505.42**	2526.06	2505.29	2552.79	**2505.42**	2565.67	2644.14	6.13	25.5
p16	160	4	**2572.23**	**2572.23**	2587.87	**2572.23**	**2572.23**	**2572.23**	2577.66	1.6	16.0
p17	160	4	2709.09	2709.09	**2708.99**	2731.37	2709.09	**2709.09**	2742.93	5.51	12.3
p18	240	6	**3702.85**	3771.35	3781.04	3802.29	3710.49	3710.49	3756.70	20.83	73.2
p19	240	6	**3827.06**	**3827.06**	**3827.06**	3831.71	**3827.06**	**3827.06**	3857.36	5.21	42.3
p20	240	6	**4058.07**	**4058.07**	**4058.07**	4097.06	4091.78	**4058.07**	4134.88	21.06	73.6
p21	360	9	**5474.84**	5608.26	**5474.84**	5617.53	5505.39	5495.54	5564.61	24.90	81.9
p22	360	9	**5702.16**	5702.16	**5702.06**	5706.81	5702.16	5702.16	5753.71	25.14	86.0
p23	360	9	**6095.46**	6129.99	**6095.46**	6145.58	6140.53	6145.58	6205.46	24.05	83.7
			Average error gap	0.33%	0.33%	0.96%	0.13%	0.28%			

E. Comparison to the state-of-the-art metaheuristics for MDVRP

The ICAwICA algorithm was next evaluated for the MDVRP compared to other state-of-the-art approaches. Although there have been many algorithms applied to the MDVRP, the most recent approaches in literature were selected and are summarised in Table 4. A cooperative coevolutionary algorithm called CoES [73], Improved Ant Colony Optimization (IACO) [60], Tabu Search Heuristic (TSH) in [74], as well as hybrid Ant Colony with simulated annealing and local search algorithm called ACO+ [51] were selected for the comparison. The ICAwICA algorithm was also compared to the best-known solutions (BKS) in [70]; it is worth mentioning that these solutions are outdated as better results are reported in the literature. Nevertheless, the best-known solutions of [70] are included for reference.

Compared with other algorithms in Table 4, ICAwICA was able to obtain the same best score in 11 out of 23 instances and outperformed the four rival algorithms on p08 instance. On average error percentage in respect to BKS, ICAwICA fell short compared to ACO+ (0.13% vs 0.28% error), however, outperformed other compared approaches.

IV. CONCLUSIONS AND FUTURE WORK

This work proposes a novel generic Imperialist Competitive Algorithm (ICA) based algorithm for solving constrained combinatorial problems called ICA with Independence and Constrained Assimilation (ICAwICA). The algorithm implements a new Independence operator for ICA, where each of the colonies has a free will to choose between assimilating to its imperialist or any other imperialist in the population. Besides, a generic constrained assimilation process is proposed as part of the local search. The constrained assimilation exploits the fact that two solutions cannot be merged without violating constraints. Furthermore, it combines the classic ICA assimilation and revolution operators in one, in a generic manner.

To evaluate the performance and versatility of the ICAwICA algorithm, two different kinds of combinatorial benchmark problems were selected – subset selection and routing, Multiple Knapsack Problem (MKP) and Multiple Depot Vehicle Routing Problem (MDVRP), respectively. First, the ICAwICA was compared to classic ICA, and results showed a definite improvement in all benchmark test instances. Next, the sensitivity of Independence operator was performed, analysis shows that independence probability of greater than zero, improves the results at the expense of computing time. Finally, the ICAwICA was compared to the current state-of-the-art population-based algorithms for both MKP and MDVRP. The proposed algorithm outperformed the majority of the competition on both types of problems across multiple instances, indicating the generic, universal nature of the ICAwICA.

The proposed algorithm can be improved in multiple ways. First, instead of simply iterating over all possibilities as part of repair mechanism, a more efficient selection process based on heuristics can be explored. Similarly, the proposed independence operator is slow as it is required to assimilate to all imperialists, smarter selection of top imperialists can be implemented. Furthermore, although this paper focuses on solving MKP and MDVRP, other discrete constrained optimisation problems can be explored.

REFERENCES

[1] S. S. Rao, Engineering Optimization, 5th ed. Hoboken: John Wiley & Sons, Ltd, 2019.

[2] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey," Int. J. Comput. Eng. Res. Trends, vol. 351, no. 5, pp. 2349–7084, 2015.

[3] M. J. Fogel, L. J., Owens, A. J., & Walsh, Artificial intelligence through simulated evolution. John Wiley & Sons, 1966.

[4] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53–66, Apr. 1997, doi: 10.1109/4235.585892.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, vol. 4, no. 2, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[6] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," 2007 IEEE Congr. Evol. Comput. CEC 2007, pp. 4661–4667, 2007, doi: 10.1109/CEC.2007.4425083.

[7] S. Hosseini and A. Al Khaled, "A survey on the Imperialist Competitive Algorithm metaheuristic: Implementation in engineering domain and directions for future research," Appl. Soft Comput. J., vol. 24, pp. 1078–1094, 2014, doi: 10.1016/j.asoc.2014.08.024.

[8] Q. Fang, H. Nguyen, X.-N. Bui, and T. Nguyen-Thoi, "Prediction of Blast-Induced Ground Vibration in Open-Pit Mines Using a New Technique Based on Imperialist Competitive Algorithm and M5Rules," Nat. Resour. Res., vol. 29, no. 2, pp. 791–806, Apr. 2020, doi: 10.1007/s11053-019-09577-3.

[9] B. Tashayo, K. Behzadafshar, M. Soltani Tehrani, H. Afkhami Banayem, M. H. Hashemi, and S. S. Taghavi Nezhad, "Feasibility of imperialist competitive algorithm to predict the surface settlement induced by tunneling," Eng. Comput., vol. 35, no. 3, pp. 917–923, Jul. 2019, doi: 10.1007/s00366-018-0641-3.

[10] Z. Aliniya and S. A. Mirroshandel, "A novel combinatorial merge-split approach for automatic clustering using imperialist competitive algorithm," Expert Syst. Appl., vol. 117, pp. 243–266, 2019, doi: 10.1016/j.eswa.2018.09.050.

[11] Z. Aliniya and S. A. Mirroshandel, "A novel combinatorial merge-split approach for automatic clustering using imperialist competitive algorithm," Expert Syst. Appl., vol. 117, no. January 2018, pp. 243–266, Mar. 2019, doi: 10.1016/j.eswa.2018.09.050.

[12] M. Abdechiri, H. Bahrami, and K. Faez, "Adaptive Imperialist Competitive Algorithm (AICA)," Proc. 9th IEEE Int. Conf. Cogn. Informatics, ICCI 2010, pp. 940–945, 2010, doi: 10.1109/COGINF.2010.5599776.

[13] M. R. Maheri and M. Talezadeh, "An Enhanced Imperialist Competitive Algorithm for optimum design of skeletal structures," Swarm Evol. Comput., vol. 40, no. November, pp. 24–36, 2018, doi: 10.1016/j.swevo.2017.12.001.

[14] L. D. Afonso, V. C. Mariani, and L. Dos Santos Coelho, "Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization," Expert Syst. Appl., vol. 40, no. 9, pp. 3794–3802, 2013, doi: 10.1016/j.eswa.2012.12.093.

[15] A. Rabiee, M. Sadeghi, and J. Aghaei, "Modified imperialist competitive algorithm for environmental constrained energy management of microgrids," J. Clean. Prod., vol. 202, pp. 273–292, 2018, doi: 10.1016/j.jclepro.2018.08.129.

[16] J. J. Liang and P. N. Suganthan, "Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search," in 2005 IEEE Congress on Evolutionary Computation, 2005, vol. 1, no. May 2014, pp. 522–528, doi: 10.1109/CEC.2005.1554727.

[17] M. Li, D. Lei, and H. Xiong, "An imperialist competitive algorithm with the diversified operators for many-objective scheduling in flexible job shop," IEEE Access, vol. 7, pp. 29553–29562, 2019, doi: 10.1109/ACCESS.2019.2895348.

[18] S. Karimi, Z. Ardalan, B. Naderi, and M. Mohammadi, "Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm," Appl. Math. Model., vol. 41, pp. 667–682, 2017, doi: 10.1016/j.apm.2016.09.022.

[19] Z. Ardalan, S. Karimi, O. Poursabzi, and B. Naderi, "A novel imperialist competitive algorithm for generalized traveling salesman problems," Appl. Soft Comput. J., vol. 26, pp. 546–555, 2015, doi: 10.1016/j.asoc.2014.08.033.

[20] J. L. Lin, H. C. Chuan, Y. H. Tsai, and C. W. Cho, "Improving imperialist competitive algorithm with local search for global optimization," Proc. - Asia Model. Symp. 2013 7th Asia Int. Conf. Math. Model. Comput. Simulation, AMS 2013, pp. 61–64, 2013, doi: 10.1109/AMS.2013.14.

[21] R. E. G. P. C. Gilmore, "The Theory and Computation of Knapsack Functions," Oper. Res., vol. 14, no. 6, pp. 1045–1074, 1966.

[22] H. P. B. Gavish, "Allocation of databases and processors in a distributed data processing," in Management of Distributed Data Processing, J. Akola, Ed. Amsterdam: North-Holland, 1982, pp. 215–231.

[23] W. Shish, "A branch & bound method for the multiconstraint zero-one knapsack problem," J. Oper. Res. Soc., vol. 30, pp. 369–378, 1979.

[24] M. Vasquez and J. K. Hao, "A 'logic-constrained' knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite," Comput. Optim. Appl., vol. 20, no. 2, pp. 137–157, 2001, doi: 10.1023/A:1011203002719.

[25] C. C. Petersen, "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R&D Projects," Manage. Sci., vol. 13, no. 9, pp. 609–772, 1967.

[26] D. N. N. H. Martin Weingartner, "Methods for the Solution of the Multidimensional 0/1 Knapsack Problem," Oper. Res., vol. 15, no. 1, pp. 83–103, 1967.

[27] T. Setzer and S. M. Blanc, "Empirical orthogonal constraint generation for Multidimensional 0/1 Knapsack Problems," Eur. J. Oper. Res., vol. 282, no. 1, pp. 58–70, 2020, doi: 10.1016/j.ejor.2019.09.016.

[28] A. Fréville, "The multidimensional 0-1 knapsack problem: An overview," Eur. J. Oper. Res., vol. 155, no. 1, pp. 1–21, 2004, doi: 10.1016/S0377-2217(03)00274-1.

[29] M. J. Varnamkhasti, "Overview of the Algorithms for Solving the Multidimensional Knapsack Problems," Adv. Sudies Biol., vol. 4, no. 1, pp. 37–47, 2012.

[30] M. Kong, P. Tian, and Y. Kao, "A new ant colony optimization algorithm for the multidimensional Knapsack problem," Comput. Oper. Res., vol. 35, no. 8, pp. 2672–2683, 2008, doi: 10.1016/j.cor.2006.12.029.

[31] J. Skackauskas, T. Kalganova, I. Dear, and M. Janakram, "Dynamic Impact for Ant Colony Optimization algorithm," Feb. 2020.

[32] A. Rezoug, M. Bader-El-Den, and D. Boughaci, "Knowledge-based Genetic Algorithm for the 0-1 Multidimensional Knapsack Problem," 2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc., no. 2, pp. 2030–2037, 2017, doi: 10.1109/CEC.2017.7969550.

[33] A. Rezoug, M. Bader-El-Den, and D. Boughaci, "Guided genetic algorithm for the multidimensional knapsack problem," Memetic Comput., vol. 10, no. 1, pp. 29–42, 2018, doi: 10.1007/s12293-017-0232-7.

[34] I. K. Gupta, "A hybrid GA-GSA algorithm to solve multidimensional knapsack problem," Proc. 4th IEEE Int. Conf. Recent Adv. Inf. Technol. RAIT 2018, pp. 1–6, 2018, doi: 10.1109/RAIT.2018.8389069.

[35] X. Lai, J. K. Hao, F. Glover, and Z. Lü, "A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem," Inf. Sci. (Ny)., vol. 436–437, pp. 282–301, 2018, doi: 10.1016/j.ins.2018.01.026.

[36] A. A. Ferjani and N. Liouane, "Logic gate-based evolutionary algorithm for the multidimensional knapsack problem," 2017 Int. Conf. Control. Autom. Diagnosis, ICCAD 2017, pp. 164–168, 2017, doi: 10.1109/CADIAG.2017.8075650.

[37] M. Daniel Valadao Baroni and F. M. Varejao, "A shuffled complex evolution algorithm for the multidimensional knapsack problem using core concept," 2016 IEEE Congr. Evol. Comput. CEC 2016, pp. 2718–2723, 2016, doi: 10.1109/CEC.2016.7744131.

[38] B. Haddar, M. Khemakhem, S. Hanafi, and C. Wilbaut, "A hybrid quantum particle swarm optimization for the Multidimensional Knapsack Problem," Eng. Appl. Artif. Intell., vol. 55, pp. 1–13, 2016, doi: 10.1016/j.engappai.2016.05.006.

[39] M. Chih, "Self-adaptive check and repair operator-based particle swarm optimization for the multidimensional knapsack problem," Appl. Soft Comput. J., vol. 26, pp. 378–389, 2015, doi: 10.1016/j.asoc.2014.10.030.

[40] X. Kong, L. Gao, H. Ouyang, and S. Li, "Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm," Comput. Oper. Res., vol. 63, pp. 7–22, 2015, doi: 10.1016/j.cor.2015.04.018.

[41] K. K. Bhattacharjee and S. P. Sarmah, "Modified swarm intelligence based techniques for the knapsack problem," Appl. Intell., vol. 46, no. 1, pp. 158–179, 2017, doi: 10.1007/s10489-016-0822-y.

[42] M. Abdel-Basset, D. El-Shahat, and A. K. Sangaiah, "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0–1 knapsack problem," Int. J. Mach. Learn. Cybern., vol. 10, no. 3, pp. 495–514, 2019, doi: 10.1007/s13042-017-0731-3.

[43] G. B. Dantzig and J. H. Ramser, "The Truck Dispatching Problem," Manage. Sci., vol. 6, no. 1, pp. 80–91, Oct. 1959, doi: 10.1287/mnsc.6.1.80.

[44] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, "Erratum: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization," J. Oper. Res. Soc., vol. 37, no. 6, pp. 655–655, Jun. 1986, doi: 10.1057/jors.1986.117.

[45] S. Karakatič and V. Podgorelec, "A survey of genetic algorithms for solving multi depot vehicle routing problem," Appl. Soft Comput. J., vol. 27, pp. 519–532, 2015, doi: 10.1016/j.asoc.2014.11.005.

[46] S. Samsuddin, M. S. Othman, and L. M. Yusuf, "a Review of Single and Population-Based Metaheuristic Algorithms Solving Multi Depot Vehicle Routing Problem," Int. J. Softw. Eng. Comput. Syst., vol. 4, no. 2, pp. 80–93, 2018, doi: 10.15282/ijsecs.4.2.2018.6.0050.

[47] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freem. 1979.

[48] N. Sharma and M. Monika, "A Literature Survey on Multi Depot Vehicle Routing Problem," IJSRD -International J. Sci. Res. Dev., vol. 3, no. 04online, pp. 2321–613, 2015.

[49] L. F. Galindres-Guancha, E. M. Toro-Ocampo, and R. A. Gallego-Rendón, "Multi-objective MDVRP solution considering route balance and cost using the ILS metaheuristic," Int. J. Ind. Eng. Comput., vol. 9, no. 1, pp. 33–46, 2018, doi: 10.5267/j.ijiec.2017.5.002.

[50] J. Renaud, G. Laporte, and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," Comput. Oper. Res., vol. 23, no. 3, pp. 229–235, 1996, doi: 10.1016/0305-0548(95)O0026-P.

[51] P. Stodola, "Hybrid ant colony optimization algorithm applied to the multi-depot vehicle routing problem," Nat. Comput., vol. 6, 2020, doi: 10.1007/s11047-020-09783-6.

[52] G. Clarke and J. W. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Oper. Res., vol. 12, no. 4, pp. 568–581, Aug. 1964, doi: 10.1287/opre.12.4.568.

[53] P. Surekha and S. Sumathi, "Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms," World Appl. Program., no. 13, pp. 118–131, 2011.

[54] B. E. Gillett and J. G. Johnson, "Multi-terminal vehicle-dispatch algorithm," Omega, vol. 4, no. 6, pp. 711–718, 1976, doi: 10.1016/0305-0483(76)90097-9.

[55] D. Gulczynski, B. Golden, and E. Wasil, "The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results," Comput. Ind. Eng., vol. 61, no. 3, pp. 794–804, 2011, doi: 10.1016/j.cie.2011.05.012.

[56] A. Imran, "A Variable Neighborhood Search-Based Heuristic for the Multi-Depot Vehicle Routing Problem," J. Tek. Ind., vol. 15, no. 2, pp. 95–102, Dec. 2013, doi: 10.9744/jti.15.2.95-102.

[57] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," Comput. Ind. Eng., vol. 137, no. September, p. 106040, Nov. 2019, doi: 10.1016/j.cie.2019.106040.

[58] Y. M. Shen and R. M. Chen, "Optimal multi-depot location decision using particle swarm optimization," Adv. Mech. Eng., vol. 9, no. 8, pp. 1–15, 2017, doi: 10.1177/1687814017717663.

[59] S. B. Sarathi Barma, J. Dutta, and A. Mukherjee, "A 2-opt guided discrete antlion optimization algorithm for multi-depot vehicle routing problem," Decis. Mak. Appl. Manag. Eng., vol. 2, no. 2, pp. 112–125, 2019, doi: 10.31181/dmame1902089b.

[60] B. Yao, C. Chen, X. Song, and X. Yang, "Fresh seafood delivery routing problem using an improved ant colony optimization," Ann. Oper. Res., vol. 273, no. 1–2, pp. 163–186, 2019, doi: 10.1007/s10479-017-2531-2.

[61] S. J.MousaviRad, F. Akhlaghian Tab, and K. Mollazade, "Application of Imperialist Competitive Algorithm for Feature Selection: A Case Study on Bulk Rice Classification," Int. J. Comput. Appl., vol. 40, no. 16, pp. 41–48, 2012, doi: 10.5120/5068-7485.

[62] D. S. Huang, K. Han, and A. Hussain, "Improved Binary Imperialist Competition Algorithm for Feature Selection from Gene Expression Data," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9773, p. V, 2016, doi: 10.1007/978-3-319-42297-8.

[63] M. Mirhosseini and H. Nezamabadi-pour, "BICA: a binary imperialist competitive algorithm and its application in CBIR systems," Int. J. Mach. Learn. Cybern., vol. 9, no. 12, pp. 2043–2057, 2018, doi: 10.1007/s13042-017-0686-4.

[64] S. Nozarian, H. Soltanpoor, and M. Vafaei, "A Binary Model on the Basis of Imperialist Competitive Algorithm in Order to Solve the Problem of Knapsack 1-0," Proc. third Int. Conf. Serv. Emerg. Mark. 2012, vol. 34, pp. 67–71, 2012.

[65] S. Xu, Y. Wang, and A. Huang, "Application of imperialist competitive algorithm on solving the traveling salesman problem," Algorithms, vol. 7, no. 2, pp. 229–242, 2014, doi: 10.3390/a7020229.

[66] S. H. Mirhoseini, S. M. Hosseini, M. Ghanbari, and M. Ahmadi, "A new improved adaptive imperialist competitive algorithm to solve the reconfiguration problem of distribution systems for loss reduction and voltage profile improvement," Int. J. Electr. Power Energy Syst., vol. 55, pp. 128–143, 2014, doi: 10.1016/j.ijepes.2013.08.028.

[67] S. S. Ray, S. Bandyopadhyay, and S. K. Pal, "Genetic operators for combinatorial optimization in TSP and microarray gene ordering," Appl. Intell., vol. 26, no. 3, pp. 183–195, 2007, doi: 10.1007/s10489-006-0018-y.

[68] J. H. Drake, "Benchmark instances for the Multidimensional Knapsack Problem," 20215. [Online]. Available: https://www.researchgate.net/publication/271198281_Benchmark_instances_for_the_Multidimensional_Knapsack_Problem.

[69] G. A. K. Glover, Fred, "Critical event tabu search for multidimensional knapsack problems," Meta-Heuristics.

[70] "Multiple Depot VRP Instances," University of Malaga, Spain. [Online]. Available: http://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-instances/.

[71] J. Liu, C. Wu, J. Cao, X. Wang, and K. L. Teo, "A Binary differential search algorithm for the 0–1 multidimensional knapsack problem," Appl. Math. Model., vol. 40, no. 23–24, pp. 9788–9805, 2016, doi: 10.1016/j.apm.2016.06.002.

[72] J. H. Drake, E. Özcan, and E. K. Burke, "A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional Knapsack problem," Evol. Comput., vol. 24, no. 1, pp. 113–141, 2016, doi: 10.1162/EVCO_a_00145.

[73] F. B. De Oliveira, R. Enayatifar, H. J. Sadaei, F. G. Guimarães, and J. Y. Potvin, "A cooperative coevolutionary algorithm for the Multi-Depot Vehicle Routing Problem," Expert Syst. Appl., vol. 43, pp. 117–130, 2016, doi: 10.1016/j.eswa.2015.08.030.

[74] M. E. H. Sadati, D. Aksen, and N. Aras, "The r-interdiction selective multi-depot vehicle routing problem," Int. Trans. Oper. Res., vol. 27, no. 2, pp. 835–866, 2020, doi: 10.1111/itor.12669.