APPLICATIONS OF
MODELLING AND SIMULATION
http://arqiipubl.com/ams eISSN 2600-8084 VOL 7, 2023, 214-238

Recent Evolutionary Algorithm Variants for Combinatorial Optimization Problem

Anniza Hamdan[1,2], Sze San Nah[1*], Goh Say Leng[3], Chiew Kang Leng[1] and Tiong Wei King[1]

[1]Faculty of Computer Science & Information Technology, Universiti Malaysia Sarawak, 94300 Kota Samarahan, Sarawak, Malaysia
[2]College of Computing, Informatics and Mathematics, Universiti Teknologi MARA, Sarawak Branch, 94300 Kota Samarahan, Sarawak, Malaysia
[3]Optimization Research Group (OPT), Faculty of Computing and Informatics, Universiti Malaysia Sabah Labuan International Campus, Jalan Sungai Pagar, 87000 Labuan F.T., Malaysia

[*]Corresponding author: snsze@unimas.my

Submitted 10 October 2023, Revised 08 December 2023, Accepted 11 December 2023, Available online 13 December 2023.

Abstract: The evolutionary algorithm has been extensively used to solve a range of combinatorial optimization problems. The adaptability of evolutionary algorithm mechanisms provides diverse approaches to handle combinatorial optimization challenges. This survey paper aims to comprehensively review the recent evolutionary algorithm variants in addressing combinatorial optimization problems. Research works published from the year 2018 to 2022 are identified in terms of problem representation and evolutionary strategies adopted. The mechanisms and strategies used in evolutionary algorithms to address different types of combinatorial optimization problems are discovered. Two main aspects are used to classify the evolutionary algorithm variants: population-based and evolutionary strategies (variation and replacement). It is observed that the hybrid evolutionary algorithm is mostly applied in addressing the problems. Hybridization in evolutionary algorithm mechanisms such as initialization methods, local searches, specific design operators, and self-adaptive parameters enhance the algorithm's performance. Other metaheuristic approaches such as genetic algorithm, differential evolution algorithm, particle swarm optimization, and ant colony optimization are still preferable to address combinatorial optimization problems. Challenges and opportunities of evolutionary algorithms in combinatorial optimization problems are included for further exploration in the field of optimization research.

Keywords: Combinatorial optimization problem; Evolutionary algorithm; Hybrid mechanisms.

1. INTRODUCTION

Inspired by Darwin's theory of evolution, the evolutionary algorithm promotes a phenomenon known as the survival of the fittest [1]. A natural evolution occurs within a population of individuals that strives for survival and reproduction. In the context of problem-solving, the quality of a collection of candidate solutions determines the chance they will be kept and used to construct further solutions. There are several motivations for the growth of evolutionary algorithms such as an increasing trend towards applying parallel approaches to address complex problems demand algorithms that are both flexible and efficient [2]. Furthermore, an evolutionary algorithm can be applied to a diverse array of problems with minimal need for customization or tailoring to a specific problem. Additionally, they demonstrate the capability to yield satisfactory solutions within a reasonable time [1].

A combinatorial optimization problem (COP) is commonly viewed as a search space comprising a collection of all objects of interest including the desired solutions and defined by discrete variables (Boolean or integers) [1]. It can be very large and complex depending on the number of variables involved [3]. There are many types of COP in real-world situations, such as traveling salesman problems, nurse rostering problems, university course timetabling problems, exam timetabling problems, job shop scheduling problems, knapsack problems, shortest path orienteering problems, and many more.

As far as we are aware, there is a limited number of survey papers on the application of evolutionary algorithms on COPs. Table 1 shows the scope of these papers. Most of these papers focus on single and multi-objective problems, without any discussion of the mechanism of evolutionary algorithm. Certain operators of evolutionary algorithms are explained and reviewed. This paper aims to fill this gap with a comprehensive survey of evolutionary algorithm variants and their mechanisms. In this paper, recent variants of evolutionary algorithms and their applications on COPs are discovered. Characteristics of evolutionary algorithms are defined and used as guidance in determining the belonging of an algorithm to the family of evolutionary algorithms. Mechanisms of each evolutionary algorithm variant on specific problem presentation

of COPs are explained. Their performance in tackling these COPs is identified.

The remainder of the paper is organized as follows. In section 2, the methodologies employed for conducting the survey are explained. Section 3 provides a comprehensive definition of the evolutionary algorithm. The types of COPs covered in this paper are listed in Section 4. Section 5 reviews and summarizes the variants of the evolutionary algorithm in COPs and their mechanisms. Challenges and opportunities of the evolutionary algorithm in COPs are emphasized in Section 6. Finally, conclusions are given in Section 7.

Table 1. Summary of survey paper on the evolutionary algorithm

Reference	Title	Scope of the paper
[4]	Introductory overview: optimization using evolutionary algorithms and other metaheuristics	Optimizing environmental models by evolutionary algorithm and other metaheuristics for single and multi-objective problems
[5]	A comprehensive review on evolutionary algorithm solving multi-objective problems	Categorization of evolutionary algorithm in multi-objective optimization for different objects and purposes.
[6]	A survey on recent progress in the theory of evolutionary algorithms for discrete optimization	Review recent advancements in evolutionary algorithms theory in terms of population, crossover operator, and diversity mechanism to address stochastic and dynamic challenges.

Table 2. List of journals considered in this survey paper

Journal	No. of papers	Journal	No. of papers
Expert System with Applications	8	European Journal of Operational Research	1
Computers and Industrial Engineering	4	Evolutionary Intelligence	1
Swarm and Evolutionary Computation	4	Asia-Pacific Journal of Operational Research	1
Applied Soft Computing	3	Applied Intelligence	1
Computers and Operation Research	3	Artificial Intelligence Evolution	1
Information Science	3	IEEE Transactions on Evolutionary Computation	1
Mathematics	3	Sustainable Operations and Computers	1
Soft Computing	3	Others	3
IEEE Access	2		
Total: 53			

2. SURVEY METHODOLOGY

The aim of this survey is to review the recent applications of evolutionary algorithms on COPs. About 100 related papers, published between 2012 and 2022 are collected. As the keywords *"evolutionary algorithm"* and *"combinatorial optimization problem"* are too broad, additional related keywords are identified and used in our search such as genetic algorithm, differential evolution, particle swarm optimization, ant colony optimization, genetic programming, and evolutionary strategies. Out of the 100 papers, only 53 papers are considered for further review. These 53 papers were selected based on favored journals such as Expert System with Applications (8), Computers and Industrial Engineering (4), Swarm and Evolutionary Computation (4), Applied Soft Computing (3), Computers and Operation Research (4), Information Science (3), Mathematics (3), Soft Computing (3), IEEE Access (2) and others. Well-established journals/conferences are preferred as a precaution against predatory journals. Furthermore, these journals/conferences provide a comprehensive understanding of algorithm design and applications [6]. Table 2 shows the journal list in this survey paper.

3. EVOLUTIONARY ALGORITHM (EA)

There are two features of an EA that differentiate it from other algorithms [2]. Firstly, it operates on a population-based approach. Secondly, the individuals within the population interact and exchange information. EA is considered one of the metaheuristic approaches [4]. The term *"metaheuristic"* originated from two Greek words, *"meta"* and *"heuristic"* which gives the meaning of *"beyond"* and *"to discover"*. In the optimization view, a metaheuristic is a high-level procedure that applies multiple heuristic approaches to obtain a near-optimal solution to a COP. The mechanism of an EA is explained by the following components [1]:

- Representation (definition of individuals)
- Evaluation function (fitness function)
- Population, parent selection mechanism
- Variation operators; recombination and mutation
- Survivor selection mechanism (replacement)

Table 3. Number of papers reviewed for each type of COP

Combinatorial Optimization Problem	No. of papers
Vehicle Routing	12
Travelling Salesman	10
Job Shop Scheduling	10
Timetabling	10
Knapsack	5
Other	6
Total	53

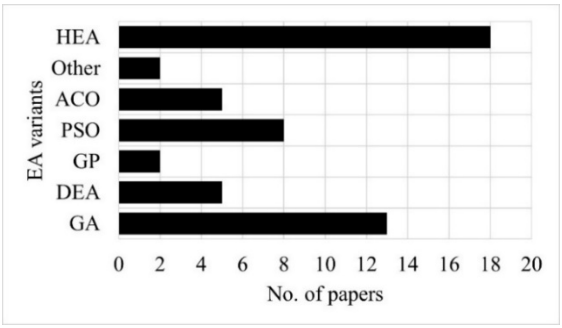

Figure 1. Variants of evolutionary algorithm

Algorithm 1: Genetic Algorithm

1 $t \leftarrow 0$
2 Initialize $(P(t=0))$
3 Evaluate$(P(t=0))$
4 **while** *termination criterion not met* **do**
5 Select parents, P_{p1}, P_{p2}
6 Crossover parents $P_{p1} and P_{p2} \rightarrow P_c(t)$
7 Mutate $P_c(t)$
8 Update population
9 **end**

Figure 2. The basic structure of GA.

4. COMBINATORIAL OPTIMIZATION PROBLEM (COP)

In addressing an optimization or modeling problem, it is necessary to identify a specific solution in a search space [1]. This search space consists of a collection of all objects of interest including the desired solution. The problem is known as a combinatorial optimization problem if the search space is defined by discrete variables, such as Booleans or integers. The computational complexity in optimization or modeling problems depends on the problem size [1]. The problem size is determined by the dimensionality of the problems subject to a number of variables. The number of possible solutions grows exponentially with the size of the problem. More time is needed to solve a huge problem. There are six types of COPs identified in this survey; Vehicle Routing, Travelling Salesman, Job Shop Scheduling, Educational Timetabling, Knapsack, and others (Traffic Light Intersection/ Shortest Path Problem). Table 3 shows the number of papers reviewed for each type of COP.

5. APPLICATION OF EAS AND COPS

In this section, multiple variants of EA used to address COPs are identified. This section describes the mechanism and performance of each variant. Figure 1 shows the application of EA variants on various COPs. From our observation, a Hybrid Evolutionary Algorithm (HEA) is the most applied variant in addressing COPs. Followed by a Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Differential Evolution Algorithm (DEA), and others. Genetic Programming (GP) is the least applied variant in addressing COPs.

5.1 Genetic Algorithm (GA)

Genetic Algorithm (GA) is a search method based on the principles of natural selection and genetics [7]. It was first introduced by Holland in 1975 to study adaptive behavior, as evidenced by the title of his early research book: *"Adaptation in Natural and Artificial Systems"* [8]. The GA encodes a solution (chromosomes) in binary string representation, which is called genes. The value of genes is called an allele [7]. To evolve good solutions in the GA, fitness proportionate selection, a low probability of mutation, and genetic recombination (crossover) are performed [8]. New generating solutions are evaluated, and the population is replaced.

The basic GA is shown in Figure 2 [7]. Line 1 indicates a counting for the generation of a population. A population is then initialized (line 2) and each chromosome in the population is evaluated by a fitness function (line 3). Two best chromosomes (known as parents) are selected (line 5). The two parents undergo crossover to produce a child (known as offspring) (line 6). The offspring is mutated (line 7). Then, the population is updated (line 8). A number of iterations are executed until a termination criterion is met.

GA is commonly applied for complicated and large-size problems due to their competitive behavior that allows the survival of solutions adapted to favorable environments [9], [10]. It exhibits the potential to increase convergence speed and enhance final solutions quality [11]. The flexibility of GA encouraged further research focuses on modifying their mechanisms such as initialization [11], [12] and genetic (crossover and mutation) operators [13], [14] to further optimize their performance.

Kobeaga *et al.* [13] proposed an EA to address an orienteering problem. An orienteering problem is similar to a traveling salesman problem or maximum collection problem. The objective was to maximize the total profit while visiting nodes in a route. A solution was represented in a sequence of nodes. An initial population was randomly generated in two steps. First, a subset of nodes was randomly chosen with probability p. Second, a route was randomly created through permutation of the nodes. A hybrid of tournament and roulette wheel selection was applied to select parents. An edge recombination crossover was implemented. A node inclusion method and Lin-Kernighan heuristic were used to improve the mutation of the route length. At some generations, tour improvement procedure (add/drop operators) was performed. The proposed algorithm was tested on 344 TSPLib instances. It was compared with three state-of-the-art algorithms (GRASP with path relinking, tabu search, and two-parameter interactive algorithm). It was competitive for low to medium-sized instances and exceptional for large-sized instances.

Plata-González *et al.* [15] proposed a GA to address a knapsack problem (KP). The KP involves selecting items with profit and weight values to maximize profit while staying within a knapsack's weight capacity. GA was used to modify the weight and profit of all items. A chromosome was represented in binary strings. A tournament selection was applied to select two fittest chromosomes. One point crossover and mutation operator were applied on the selected chromosomes to produce two offspring. Two types of tailored instances were introduced: easy-to-solve, and hard-to-solve. The proposed GA was tested on benchmark instances (Pisinger) for verification. Computational results showed that the tailored instances could improve the algorithm's performance.

Ruiz *et al.* [16] proposed a Biased Random Key of Genetic Algorithm (BRKGA) to address an open vehicle routing problem (OVRP). A solution was represented in a binary matrix. The binary matrix comprised a number of clients and routes. A random-key vector was used to initialize the solution. Each solution was randomly assigned with a value between $[0,1]$. A predecessor list and random-key vector were then used to determine the next client visited in a route. The fitness value of the solution was then evaluated. Two groups of solutions were identified: elite and non-elite individuals. An elite individual was mated with either elite or non-elite individuals in a crossover stage. An offspring was generated and went through a mutation procedure. In the mutation procedure, an offspring with a new random-key vector was created. A 3-neighborhood local search strategy was applied to improve the solution quality. Strategic oscillation was used to prevent the algorithm from being stuck in local optima. This approach was tested on three benchmark datasets (Christofides, Li *et al.*, and Golden *et al.*). The algorithm performed better than others (bumblebee mating optimization, wide solution neighborhoods metaheuristic, honeybee mating optimization, tabu search, and ant colony system). 16 out of 30 instances were improved by the proposed algorithm.

Cruz-Piris *et al.* [10] proposed a GA to address a traffic light intersection problem. A Traffic Cellular Automata (TCA) simulator was used to develop several types of intersections. A solution was represented in a sequence of bits. It comprised input of vehicles in the intersection. A value of 0 represented the absence of vehicles and 1 represented the presence of vehicles. Each solution was then evaluated based on the cost of intersection (number of vehicle conflict points). The solutions were sorted based on fitness values from low (best value) to high (worst value). The fittest solutions (without conflict) were automatically selected for the next generation. The rest of the solutions were paired randomly and combined using the crossover function. A new solution was obtained and mutated with random changes (using Gaussian distribution with mean 0 and standard deviation). A solution with conflict points was remarked as invalid and removed by the algorithm. The proposed algorithm obtained a new best solution to traffic intersection problems without any conflicts. The proposed algorithm improved a traditional solution by 9.21% to 6.98%.

Defersha and Rooyani [11] proposed a two-stage genetic algorithm (2SGA) to address a flexible job shop scheduling problem (FJSP). The objective was to determine the best sequences of job assignment to machines. A chromosome was represented in two segments: a left-hand-side (LHS) segment for job sequence and a right-hand-side (RHS) segment for machine operation sequence. In the first stage, a machine was assigned based on the job sequence in the LHS segment. The LHS segment was then combined with the RHS segment based on a shorter completion time machine. The combined segments (chromosome) consisted of three parts: job, order, and machines. In the second stage, a K-ways tournament mechanism was used to evaluate and select the best chromosomes. The selected chromosomes in a mating pool were then randomly paired. Tailored genetic operators were performed on each pair of chromosomes to produce offspring: single-point crossover, job crossover, assignment crossover, operation-swapping mutation, and assignment-altering mutation. The 2SGA was tested on several benchmark problems (Hurink, Jurisch, and Thole). It (using one CPU) performed better than other parallel genetic algorithms (using 48 CPUs). The proposed algorithm improved in terms of convergence speed and solution quality as compared to a regular GA.

Yuan *et al.* [9] proposed a co-evolutionary genetic algorithm (CGA) to address a flow shop group scheduling (FSGS) problem with job-related blocking and transportation times. The objective was to minimize job processing time. A chromosome was constructed in three processes: block mining, extraction, and recombination. Two groups were identified in the block mining process: group scheduling and job scheduling. The extraction process was used to remove redundant blocked jobs in the population. The unblocked jobs were then inserted into an empty chromosome. A classical tournament was applied to select the chromosomes with the best fitness value. A partial mapping crossover was applied to the two best chromosomes. A single-point exchange mutation was then applied to the new solutions. A coordination-oriented generic evolution operator was used to select and update the job sequences in a group schedule. The CGA was tested on actual data from the pipe-making production. It outperformed the other three methodologies (tabu search, two-level iterated greedy algorithm, and hybrid genetics and particle swarm optimization) in terms of solution effectiveness and search quality.

Park *et al.* [14] proposed a GA with a waiting strategy and rerouting indicator (RI) to address vehicle routing problems with simultaneous pickup and delivery (VRPSPD). The objective was to minimize transportation costs. A chromosome was represented in a sequence of nodes. The distance between nodes developed a route that traveled by a vehicle. Identical vehicles traveled through the nodes (customers) based on demand location, amount of product requested, and product variety index. The fitness value of each chromosome was evaluated based on the total distance traveled by the vehicles and the total amount of product transported. Two best chromosomes were selected by using a roulette wheel method. A two-point crossover was applied to these chromosomes to generate offspring. A scramble mutation operator was applied to the offspring to obtain a near-optimal solution. The waiting strategy with a rerouting indicator was implemented when new demands arrived during product delivery. The proposed algorithm was tested on simulated data. Its performance was compared with that of a CPLEX solver and other neighborhood searches (general, adjacent, and 2-segment swap). A real-life-sized instance was solved with reasonable computational time by the proposed algorithm.

Komijan *et al.* [17] proposed a GA to address a bus routing problem. The bus routing problem included factors such as gender separation, routing special students, and mixed loading into consideration. The objective was to minimize transportation costs and time. A chromosome was segmented into three parts: a sequence of bus stations, a sequence of students, and a sequence of visiting students at the same route. Two best chromosomes were selected based on their fitness value. A two-point crossover was applied to these chromosomes to generate offspring. A swap-mutation operator was used in a mutation procedure. Finally, an elitism method was applied to select the best solution. The proposed algorithm was employed on a real-world dataset of four schools in Tehran. The bus routing was improved in terms of an optimal number of stations visited and cost reduction.

Perez *et al.* [18] proposed a specific 3D chromosome in a GA to address a university timetable scheduling. The 3D chromosome consisted of a three-dimensional matrix (day, hour, group). Each chromosome's fitness was determined by subtracting the total penalties from the total preferences. The two fittest chromosomes were recombined with a single-point crossover to generate offspring. Mutation operator was carried out by exchanging professor slots, available time or day of subjects, and available time or day between two slots. The proposed algorithm was employed on a real-world dataset of the University of Isthmus, Mexico. At least one optimal solution with zero penalties was obtained.

Ghannami *et al.* [12] proposed a GA with Stratified Opposition-based Sampling (SOBS) to address the shortest path problem (SP) between pairs of nodes in a network. The SOBS method has produced a diversified initial population. A chromosome was represented in a sequence of nodes. Levenshtein distance was used to measure the chromosome length. The length of the chromosome was reduced by using an opposition-based learning (OBL) strategy. Two random chromosomes performed a one-point crossover to generate offspring. The SOBS-GA was tested on deterministic networks (fixed 20 nodes) and random networks (generated by Watts–Strogatz, Erd¨os–R´enyi, Waxman, and Barab´asi–Albert models). In comparison to a pseudo-random number generator (PRNG), the proposed algorithm generated more precise solutions in less computational time and with an improved initial population.

Abduljabbar *et al.* [19] proposed a GA to address a course timetable scheduling problem. A chromosome was represented in a binary code of 24 bits in length. The chromosomes consisted of elements such as university departments, courses, lecture halls, instructors, instructor schedules, and days. Two fittest chromosomes were selected using a roulette method. Single-point and two-point crossover operations were performed on these chromosomes. Finally, mutated chromosomes were selected to update the population. The proposed GA was tested on real-world data from the College of Computer Science, University of Technology-Baghdad. A flexible scheduling system was obtained.

Mahjoob *et al.* [20] proposed a Modified Adaptive Genetic Algorithm (MAGA) to address a multi-product and multi-period inventory routing problem (MMIRP). The MMIRP is similar to a vehicle routing problem. The objective was to minimize the number of vehicles, transportation costs, and inventory costs. A chromosome was represented in a binary matrix. The binary matrix consisted of a number of customers and periods. The modified GA was executed in two phases: construction and improvement. In the construction phase, two parents were selected based on the summation of vehicles, transportation costs, and inventory costs. The two selected parents then went through a column or row crossover of the matrix to create offspring. A mutation operator was applied to the offspring by flipping a position of two random rows or columns of the matrix. An adaptive genetic operator was used to strengthen the performance of the crossover and mutation operators. The proposed GA was tested on generated instances. The proposed methodology was better than the Cplex software and heuristic method from the literature.

Neumann *et al.* [21] proposed a GA to address a KP with stochastic profit. Two types of profit were considered: expected profit and variance of profit. The objective was to maximize the profit. KP with stochastic profit was described as profit with an uncertain probability of weights. A solution was comprised of a number of items. Each item consisted of profit and weight. It was then evaluated by using a fitness function. Two boundaries were applied in this fitness function: Chebyshev's inequality and Hoeffding's bounds. These boundaries were introduced to handle the uncertainty of weight in KP. Specific crossover and heavy tail mutation were then applied to maximize the profit. The proposed algorithm was tested on the Pisinger dataset. Better solutions were produced than other evolutionary algorithms ((1+1) EA heavy-tailed mutation and (μ+1) EA).

A total of thirteen papers on GA are reviewed: vehicle routing (4), job shop scheduling (2), timetabling (2), knapsack (2), and others (3). Several initialization strategies were introduced to enhance the quality of initial solutions; such as a block-mining-based artificial chromosome [9], biased random key [16], and stratified opposition-based sampling [12]. Adjustments to genetic operators were performed to prevent solutions from getting trapped in local optima. These modifications include strategic oscillation [16], a node inclusion method and Lin-Kernighan heuristic [13], adaptive genetic operators [20], and specific design genetic operators [9], [11], [21]. In addition, fitness evaluation was also improved by introducing two boundaries: Chebyshev's inequality and Hoeffding's bounds [21].

Algorithm 2: Differential Evolution Algorithm

1 $t \leftarrow 0$
2 Initialize population $P(t) = \{x_{1,t}, x_{2,t}, ..., x_{N,t}\}$
3 **while** *termination criterion not met* **do**
4 **for** $i = 1$ *to* N **do**
5 Select three random vectors $x^1_{i,t}, x^2_{i,t}, x^3_{i,t}$
6 $v_{i,t} = x^1_{i,t} + F * (x^2_{i,t} - x^3_{i,t})$ where F is a scaling factor
7 Perform crossover on $v_{i,t}$ and $x_{i,t}$ to obtain a target vector $u_{i,t}$
8 **if** $f(u_{i,t}) \leq f(x_{i,t})$ **then**
9 $x_{i,t+1} \leftarrow u_{i,t}$
10 **else**
11 $x_{i,t+1} \leftarrow x_{i,t}$
12 **end**
13 **end**
14 $t \leftarrow t + 1$
15 **end**

Figure 3. The basic structure of DEA

5.2 Differential Evolution Algorithm (DEA)

Differential Evolution Algorithm (DEA) is a stochastic population-based optimization algorithm. It was introduced by Ken Price and Rainer Storn in 1995 [8], [22]–[24]. Figure 3 shows the main operations involved in a DEA [25]. A population of individuals (vectors) is initialized (line 2). Three vectors are randomly selected (line 5). A differential mutation is performed on these vectors to create a mutant vector (line 6). A crossover is performed between the mutant vector, $v_{i,t}$ and a random parent vector, $x_{i,t}$ to obtain a target vector, $u_{i,t}$ (line 7). The target vector is compared with the randomly selected parent vector (line 8). The parent vector in the current population is replaced if the target vector is better (line 9). The generation count is then incremented (line 14).

DEA is generally known to successfully solve COP due to its simple structure, robustness, and ease of use [26], [27]. In classic DE, a user specifies control parameters and mutation strategies, which are kept constant during the optimization process [28]. Researchers investigating new components to enhance their capability in solving binary/discrete problems with large dimensions[26]. Additionally, ongoing research is carried out to enhance DEA performance in terms of computational time[29], convergence speed [27], and preventing trapped in local optima [26].

Ali *et al.* [26] proposed a DEA to address a binary KP. A solution was represented in a continuous vector. The length of a vector equals the total number of items. A mapping method was introduced to convert the continuous value to a binary value. A repairing method was introduced in fitness evaluation to remove items that exceed the ratio of profit to weight. A DE/rand/1 mutation strategy was then performed to produce a mutant vector. A target vector was obtained via binomial crossover between the mutant vector and parent vector. Finally, a greedy selection strategy was applied to identify elite solutions for the next generation. A diversity mechanism was used to avoid solutions stuck in local optima. The proposed DEA was tested on the KP benchmark. A performance comparison was made with other state-of-the-art algorithms (GAs with three different local searches (BFGS, Nelder-Mead, Conjugate Gradient), and GA without local search). Good solutions were obtained in a better computational time.

Hu *et al.* [29] proposed a DEA with an uncertainty handling technique (DEA_UHT) to address a stochastic reentrant job shop scheduling problem (SRJSSP). The objective was to minimize the processing time of each job by each machine. A job sequence was formed on reentrant-largest-order-value (RLOV). An active schedule was generated by allocating the jobs to machines based on the sequence. The processing time of each solution was computed by using an optimal computing budget allocation technique (OCBAT). A roulette wheel selection was applied to select the best solution. Insert-based and interchange-based operators were applied to the selected solutions as replacements for mutation and crossover operations. A poor solution was removed by a hypothesis test technique (HTT). The proposed DEA_UHT was tested on the JSSP benchmark dataset. It was faster than other methodologies (PDEA and DEA_UHT_O).

Gao *et al.* [30] proposed a DEA to address a ship-uploading scheduling problem. A solution was represented in a long array with three components: the number of incoming ships, the order of unloading ships, and the allocation of conveyors. Initial solutions were randomly generated. The addition and subtraction of arrays were carried out in the mutation stage to produce a new target vector. The crossover procedure was then applied by swapping fragments of vector components. After the crossover, infeasible solutions were possibly produced. An improvement method was then carried out to repair the infeasible solutions. The proposed DEA was tested on simulated and real-world datasets. A comparison was made with the CPLEX solver and other DE variants (ADE, CODE, JADE, JDE, and SADE). The proposed algorithm was found to be effective and efficient both on benchmark data and real-world datasets.

He *et al.* [27] proposed an adaptive multi-objective differential evolutionary algorithm (AMODE) to address an energy-efficient open shop scheduling problem (EOSSP). The objective was to minimize makespan, mean tardiness, and total energy consumption. A solution was represented in a two-layer random key. The first layer consisted of the operation sequence. The second layer consisted of the machine speed sequence. Each layer was assigned with a real number in a range of (0,10). A fuzzy correlation entropy analysis (FCEA) was used to determine the quality of each solution. Three solutions with a higher FCEA value were selected for mutation and crossover operators. Adaptive opposition-based learning (AOBL) was utilized within a mutation operator to enhance the search capability. Any infeasible solution obtained was then repaired. A crossover was performed on the operation sequence and machine speed sequence. An external archive technique was used to select the best solution. The proposed AMODE was tested on a benchmark dataset (Tailard's instances). It outperformed the three well-known algorithms (NSGA-II, NSGA-III, and MOEA/D).

Morais *et al.* [28] proposed a self-adaptive discrete differential evolution (SADE) to address a permutation flow shop (PFS) scheduling problem. Three types of SADE were introduced: (DSA-DE), DSADE-PFS1, and DSADE-PFS2. A solution was represented in a continuous vector. A largest-order value (LOV) method was used to convert the continuous vector into a permutation of a task in a discrete domain. A self-adaptive mechanism was employed to allow good control of parameters. A mutation factor and crossover rate were generated in this mechanism. Different mutation strategies were applied in each SADE. A binary crossing was operated in all types of SADE to produce a test vector. A lower-cost function of the test vector replaced a target vector. The algorithm was competitive in addressing several benchmark datasets (Carlier, Heller, Reeves, and Tailard).

A total of five papers on DEA are reviewed: job shop scheduling (4) and knapsack (1). DEA was known as a powerful tool for solving continuous optimization problems [26], [30]. A mapping method and largest order value (LOV) method were applied to handle the continuous nature [26], [28]. Several techniques were proposed in fitness evaluation to reduce computational cost such as optimal computing budget allocation technique (OCBAT), hypothesis test technique (HTT) [29], fuzzy correlation entropy analysis (FCEA) [27], and repairing method [26]. Furthermore, opposition-based learning [27] and different adaptive mechanisms were tested to have good control of parameters in DEA [28].

5.3 Genetic Programming (GP)

Introduced by Koza in 1992, Genetic Programming (GP) is perceived as the "programming of computers by means of natural selection", or "automatic evolution of computer programs" [8]. GP produces a new generation by iteratively applying genetic operations such as crossover, mutation, and reproduction. This is an extension of Genetic Algorithms in which individuals (computer programs) within a population are represented using a tree structure. In the tree structure, a node indicates a function (instructions to execute), and a link indicates an argument for each instruction. The tree's leaves are called terminals [31].

GP initializes individuals in a population by two primary methods; Full or Grow [31]. Both methods initialize individuals in a population based on a predefined maximum depth of a tree. A tree's size and shape are specified in the Full method. Variations of sizes and shapes are allowed in the Grow method.

Figure 4 shows the basic structure of GP [31]. The first population of an individual computer program is generated randomly using available functions and terminals (line 1). Iteratively, the fitness value of each individual computer program is evaluated (line 3). Two fittest computer programs are then selected to perform genetic operations (line 4). A new individual computer program is created through crossover (line 5) and mutation procedures (line 6). The best individual computer program is then updated. This process is repeated until the termination criterion is met.

In COPs, GP is commonly used to solve job shop scheduling. A tree structure in GP can be represented as a mathematical formula to determine the priority of a certain job [32]. A priority is a numerical value (integer or floating-point number) that enables the ranking of machine jobs.

Algorithm 3: Genetic Programming

```
1 Initialize a population of random computer programs
2 while termination criterion not met do
3     Evaluate individual computer program probability
4     Select two individual computer programs based on its fitness
5     Crossover is performed between two selected individuals to produce
        offspring
6     Mutation applied to the offspring
7     Update population with the best individual computer programs
8 end
```

Figure 4. The basic structure of GP

Lin *et al.* [3] proposed genetic programming hyper-heuristic (GP-HH) to address a multi-skill resource-constrained project scheduling problem (MS-RCPSP). A solution was represented in a binary tree. Each binary tree consisted of a sequence of task vectors. Ten low-level heuristics were used to develop a single task vector. A left-shifted method was then used to assign a single task vector to the available resources. The fitness value of each single task vector was then evaluated. Two best-fitted task vectors were then selected by using a roulette wheel method. A crossover was performed between these two best-fitted vectors to generate a subtree. In a mutation procedure, the subtree below a mutation point was replaced by a randomly generated tree. The new binary tree continued growth until it reached a maximum depth. GP-HH was tested on the iMOPSE benchmark dataset. It was compared with three other algorithms (hybrid ACO, GRASP, greedy_DO, and hybrid differential evolution and greedy algorithm). It was effective in handling the MS-RCPSP with respect to average cost, computational time, convergence rate, and solution precision.

Braune *et al.* [32] proposed a GP learning approach to generate priority rules for dispatching jobs in flexible shop scheduling problems. A solution was represented in a tree structure. There are two types of tree structures used: single-tree and multi-tree. The single tree structure consisted of terminals for jobs and machine decisions. In the multi-tree structure, a full or grow method was applied to generate an initial population. The fitness value (average makespan) of each tree was then evaluated. The fittest tree was selected by using a roulette wheel method, rank-based, and tournament selection. A single-point crossover was applied to generate a subtree. A standard GP mutation was carried out to select a node within a parent tree and to cut the branches. The cut branches were replaced by a new sequence of nodes. The generated subtree consisted of a dispatching rule which was used as learning information for the next generation. The proposed algorithm was tested on benchmark instances (Brandimarte and Lawrence). Better priority rules were generated.

A total of two papers on GP are reviewed. An expression tree used in GP creates a dispatching rule that helps in decision-making between tasks and resources. In [3], GP functioned as a high-level strategy to manage several low-level heuristics during the evolutionary process. Meanwhile, single-tree and multi-tree approaches were performed to minimize the makespan of the jobs machine [32].

5.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was introduced by Kennedy and Eberhart in 1995 [33]–[36]. PSO algorithm was inspired by a swarm of birds that search for food with information sharing between the individuals in the swarm. The individuals (known as particles) represent locations in a multidimensional search space. Particles move randomly in the search space to find the maximum/minimum of a given objective function. Figure 5 shows the PSO algorithm [35]. A particle is initialized by its location and velocity (line 1). Each particle's fitness value is then evaluated (line 4). The particle with the best fitness value is updated as a personal best location (line 7). The fitness value among all particles is then compared with the previous global best location. The new global best location is updated if the particle achieves a new fitness value (line 9). Each particle's velocity (line 11) and new location (line 12) are then updated. The process is repeated until a stopping criterion is met.

PSO is popular due to its computational simplicity, adaptability [33], flexibility, fast convergence, and few parameter settings [36]. It can efficiently explore different solution spaces throughout the search process and converge toward high-quality solutions [37]. Several PSO variants, modified velocity equations, neighborhood topology, and local search strategies are suggested to enhance PSO performance [38].

Algorithm 4: Particle Swarm Optimization

```
1  Initialize location and velocity of each particle
2  repeat
3      for each particle do
4          evaluate objective function f at the particles location
5      end
6      for each particle do
7          update the personal best location
8      end
9      update the global best location
10     for each particle do
11         update the velocity
12         compute new location of the particle
13     end
14 until stopping criterion met
```

Figure 5. The basic structure of PSO.

Shi *et al.* [34] proposed a discrete PSO with local search to address a split delivery routing vehicle problem (SDVRP). An integer coding method was used to represent a particle. Each particle had $6 \times nc + 1$ (nc is the number of customers) elements and comprised three parts (position of a particle, delivery amount to each customer, and velocity of a particle). An initial particle was generated by using three different methods; the save algorithm (for non-split delivery), the split sweep algorithm, and random generation (for split delivery). The position of a particle was transformed into a float number between $(0,1)$. t was then evaluated based on a fitness function. The personal best position and global best position were then updated. The updated position of a particle was changed back into an integer number form for a local search procedure. Five local search strategies were implemented (relocation, exchange, 2-opt, improvement of k-split procedure, and split exchange and split swap). Three types of velocity equations (inertia velocity equations, constriction velocity equations, and local neighborhood topology velocity equations) were applied to update the particles in the swarm. The proposed algorithm was tested on five sets of instances (Solomon, Belenguer *et al.*, Campos *et al.*, Archetti *et al.*, and Chen *et al.*). Good solutions were found for 32 instances and improved solutions were obtained for 35 instances.

Mingo Lopez *et al.* [38] proposed a hybrid particle swarm optimization with genetic operators (HPSOGO) to address a multidimensional knapsack problem. A particle was represented in a vector of real numbers. It was transformed into a vector of binary variables by using a random number generation process. The fitness value of each particle is then evaluated to determine the personal best location and global best location. Crossover and mutation procedures were performed on the personal best location and global best location. 2 random bits of personal best and 1 random bit of global best were copied to a current particle. The current particle's velocity was then mapped in a range $[0,1]$. A sigmoid function acts as a threshold that returns the value 0 or 1 to update the particle's location. HPSOGO was tested on ORLib benchmarks. Its performance was competitive with other PSO-based algorithms (PBPSO, MBPSO, CBPSO1, BPSOTVAC, CBPSOTVAC, and BPSOSIPAC) in obtaining optimal solutions.

Imran Hossain *et al.* [33] proposed a PSO with two new operations to address a university course scheduling problem (UCSP). A particle was represented in a one-dimensional matrix (consisting of a complete schedule for instructors, batches, and groups). Initial particles were generated by randomly allocating timeslots to allocated courses of all instructors. Each particle was then evaluated based on quality and violation of the instructor's preferences. The fittest solution among all particles was selected as the global best solution. The velocity of each particle was then calculated by using a swap sequence strategy. It transformed a particle solution into another known as an intermediate solution. As the forceful swap operation applied to the intermediate solution could cause conflicts, a repair mechanism was applied by randomly moving the conflicting courses to non-conflicting positions. A selective search method was then applied to determine the fittest intermediate solution. It was then updated as a final solution. The proposed PSO was tested on a real-world dataset from Khulna University of Engineering & Technology. It showed improvement in terms of quality solutions as compared to other traditional methods (genetic algorithm, harmony search, and producer-scrounger method).

Gulcu *et al.* [39] proposed a PSO to address a discrete multiple traveling salesman problem (MTSP). A particle consisted of a sub tour of each salesman. The position of each particle was evaluated by a fitness function. Two variants of PSO were introduced: APSO (metaheuristic) and HAPSO (hyper metaheuristic). In APSO, an initial particle was randomly generated. In HAPSO, Greedy Randomized Adaptive Search Procedure (GRASP) was used to generate an initial particle. 2-opt, path-relink, and swap operators were applied to obtain a particle with the best global position in both algorithms. Both APSO and HAPSO were tested on five TSP instances. HAPSO showed greater robustness compared to APSO and other algorithms in the literature (genetic algorithm and ant colony optimization).

Dahmani *et al.* [37] proposed an adaptive PSO to address a quadratic knapsack problem (KP) with a conflict graph. The objective was to maximize the total profit of items in a knapsack within the capacity limit. A constructive greedy procedure was applied to generate an initial particle. A fitness function was used to evaluate the position of each particle. The best personal position and global best position were then computed. Each particle's velocity and position were then updated. A local neighborhood optimization procedure was introduced to repair any infeasible solutions. The quality of solutions was then enhanced. Finally, a 2-opt strategy was used to refine the solution quality. The proposed algorithm was tested on benchmark instances (Shi *et al.*, Billionet and Soutif, and Yamada *et al.*). High-quality solutions were produced in average computational time as compared to other methodologies (modified descent method, neighborhood search-based metaheuristic, and GLPK solver).

Marichelvam *et al.* [40] proposed an improved particle swarm optimization (IPSO) with a two-stage procedure to address a hybrid flow shop scheduling problem (HFSSP). The objective was to minimize the weighted sum of the makespan and total flow time. A particle was represented in a sequence of jobs. In the first stage, the shortest processing time (SPT) dispatching rule was applied to arrange jobs in descending order. NEH (Nawaz, Enscore, and Ham) constructive heuristic was then used to generate initial particles. Particles were then evaluated based on their position. The personal best position and global best position were updated. Each particle's velocity and new position were then computed. In the second stage, variable neighborhood search (VNS) was applied to improve the quality of the global best solution. IPSO was tested on manufacturing industry data in Hosur, India. The algorithm's performance was superior to other methods (genetic algorithm, water flow-like algorithm, tabu search, variable neighborhood search-priori approach).

Tan *et al.* [41] proposed a hybrid particle swarm optimization with particle elimination (HPSO-PE) to address a school timetabling problem (STP). A particle was represented in a two-dimensional matrix of resources (teacher, students, classes, and rooms) and time. The current position was considered the local best solution. It was then set as the global best solution. Mutation, local, and global crossover operations were performed in the current position. A horizontal and vertical swapping was introduced to improve the effectiveness of the mutation and crossover procedures. A Hill Climbing method was used to update the local and global best position of particles. Poor particles were removed. HPSO-PE was tested on an XHSTT-2014 dataset. It was effective in addressing small and medium instances.

Thepphakorn *et al.* [36] proposed a hybrid particle swarm optimization-based timetabling (HPSOT) to address a university course timetabling problem (UCTP). Two variants of PSO were introduced: Standard particle swarm optimization (SPSO) and Maurice Clerc particle swarm optimization (MCPSO). The largest unpermitted period degree (LUPD) was applied to generate initial particles. A number of events were listed in a sequence. An event was allocated to an empty timetable to create a candidate particle (timetable). A random key technique was applied to each candidate particle. Each candidate particle was then evaluated to update the personal best position and global best position. Each particle's velocity was then updated. A repair process was adopted for any infeasible solutions found. The quality of a current particle was then compared. It was replaced if the new solution was found better. Two local search strategies were introduced to improve solution quality: Exchange operator (EO) and Insertion operator (IO). The proposed HPSOT was tested on a real-world dataset from Thailand. A comparison of five combinations EO:IO was made for both SPSO and MCPSO. Computational results showed that the $25\%: 75\%$ ratio of EO:IO configuration could generate a better total operating cost. It performed better than the original variants of PSO for all problem instances.

A total of eight papers on PSO are reviewed: vehicle routing (1), traveling salesman (1), job shop scheduling (1), timetabling (3), and knapsack (2). Several variants were introduced such as Discrete PSO [34], Metaheuristic PSO (APSO) and Hyper Metaheuristic PSO (HAPSO) [39], Adaptive PSO [37], Standard PSO (SPSO) and Maurice Clerc PSO (MCPSO) [36]. Additional genetic operators [38], [41] and local search strategies [34], [36], [37], [40]were performed to improve the solution quality in the proposed algorithms.

5.5 Ant Colony Optimization (ACO)

The double-bridge experiment by Deneubourg *et al.* [42] and Goss *et al.* [43] inspired the development of the first ACO algorithm by Dorigo *et al.* (1991) [35]. The experiment explained the behavior of ants in finding the shortest path from their nest to food sources. A trail pheromone was laid by the ants along their way to the food sources. The traces marked by the ants lead others to the food sources [35], [44]. Figure 6 shows an iterative process of an ant to create a solution (path). In line 1, pheromone values are initialized. The ants then construct a sequence of solutions (line 4). The pheromone values are then updated (line 7). It decreases (evaporation) by a certain probability value. For a good solution, the pheromone values are increased (intensification). This information is transferred from one iteration to another until a stopping criterion is met.

ACO can utilize problem information well while constructing solutions [45]. Although there is an evaporation mechanism, ACO can easily trapped in local optima [45], [46] and require long running time [46] when solving more complicated problems. Consequently, a new ACO is developed concerning optimizing ACO parameters [47], and pheromone updating strategy [46], [48].

Wang *et al.* [47] proposed a hybrid of a Symbiotic Organism Search (SOS) and ACO algorithm to address a traveling salesman problem. An initial population (a group of ants) was randomly generated. Each ant carried a probability value. The probability value was obtained via pheromone trail (α) and heuristic information (β) between two cities. The SOS method was applied to control these two parameters (α and β). They were updated via three phases: mutualism, commensalism, and parasitism. A new solution was obtained in the mutualism phase. It was then used as a reference in the commensalism phase. The reference solution was used to update the other ants. In the parasitism phase, an artificial parasite was obtained and replaced ants in the population. Adjustment of α and β balanced the pheromone guidance in finding the minimum travel distance. One local optimization strategy (combination of nearest neighbor and path reversion) was applied to improve the solution quality. The proposed SOS-ACO was tested on different TSP instances in TSPLIB. A comparison was made with the other two algorithms (ACO with local optimization, and ACO). The proposed algorithm obtained better quality solutions for large-sized instances.

Algorithm 5: Ant Colony Optimization

```
1  Initialize pheromone values
2  repeat
3      for ant k ∈ {1...m} do
4          Construct a solution
5      end
6      for all pheromone values do
7          Decrease the value by certain percentage {evaporation}
8      end
9      for all pheromone value corresponding to good solutions do
10         Increase their value {intensification}
11     end
12 until termination criterion met
```

Figure 6. The basic structure of ACO

Li *et al.* [46] proposed Saltatory Evolution Ant Colony Optimization (SEACO) to address a traveling salesman problem (TSP). A solution was represented by a pheromone matrix. The pheromone matrix consisted of a number of city nodes and the distance between cities. An initial solution was generated by randomly assigning an ant to any city node. Any unvisited city was chosen by an ant via the roulette wheel method. The process was repeated until an ant returned to the starting point. Saltatory evolution was introduced to tackle a long-running time of traditional ACO. It was carried out in three stages: path performance evaluation, near-optimal path identification rules generation, and near-optimal path identification. SEACO was tested on 24 random instances from TSPLIB. A comparison was made with a traditional ACO in terms of complexity. Optimization speed for large-sized instances was improved.

Jia *et al.* [45] proposed a Confidence-based Ant colony optimization (CBACO) to address a bi-level capacitated electric vehicle routing problem (CEVRP). In CEVRP, three kinds of vertices were considered: depot, customers, and charging stations. A max-min ant system (MMAS) was used to construct two types of routes: direct vehicle routing (CBACO-D), and giant routes that include all customers (CBACO-I). The nearest neighbor search and simple enumeration (SE) methods were used to obtain the fittest solution. Through confidence-based selection, a sub-solution (capacity feasible) was selected and refined. A recharging schedule was generated by using a SE method. It was used to repair the feasibility (electricity) of the sub-solution. This feasible solution was then evaluated, and pheromone information was updated. For CBACO-D, a 2-opt, and node-shift strategy was used to improve the solution. CBACO was tested on two benchmark datasets from IEEE WCCI2020. CBACO-D and CBACO-I routing strategies were compared. CBACO-I performed better for large instances as it promotes diversity in a population. Meanwhile, CBACO-D converged prematurely. The proposed CBACO was competitive with other methods (bilevel ACO, genetic algorithm, simulated annealing, variable neighborhood search, and iterated local search).

Zhao *et al.* [49] proposed ACO to address a multi-objective traveling salesman problem. An evolutionary experience-guided pheromone updating strategy was introduced to improve efficiency and quality of optimization. A solution was represented in a pheromone matrix. Multiple pheromone matrices were handled by several groups of ants. Each group has its own pheromone and heuristic matrix. The shortest pheromone trail was measured in an Euclidean distance. Intragroup evolutionary information-guided pheromone updating (IGPU) was used to update the pheromone matrix and to avoid the solution from being stuck at local optima. A learning automata (LA) with adaptive strategies was performed to update the pheromone matrix for each group. The proposed algorithm was tested on nine multiobjective benchmarks of TSP. Its performance was competitive as compared to other methods (multiobjective EA using decomposition and ant colony, bicriterion ant, multiobjective ACO, multiobjective ACS, and population-based ACO for multiobjective).

Thiruvady *et al.* [48] proposed a surrogate-assisted ant colony optimization (SACO) to address resource-constrained job scheduling with uncertainty. Two new techniques were introduced: fine-tuned local search and global updates on the pheromone trail. A solution was represented in a permutation of jobs. It was then mapped to a schedule by using greedy heuristics. A pheromone trail was weighted by a job position. The local pheromone trail was updated every time a job was selected and added to a schedule. Permutation swapping and β-sampling procedures were applied in a local search procedure to achieve the best local trail. The β-sampling procedure moves the subsequences of job permutation to different parts of the pheromone trail. A promising solution was identified by calculating the distance between the two trail solutions. The proposed algorithm was tested on ORLib instances. The proposed SACO outperformed traditional ACO in terms of computational time and uncertainty levels.

A total of five papers on ACO are reviewed: vehicle routing (1), traveling salesman (3), and job shop scheduling (1). Symbiotic Organism Search (SOS) was incorporated into ACO to optimize parameters [47]. Pheromone updating strategies were introduced to avoid solutions stuck in local optima, such as Intragroup evolutionary information-guided pheromone updating [49]. Besides, local search was applied to refine the solutions: a combination of nearest neighbor and path reversion [47]. Saltatory evaluation ACO was introduced to tackle the long execution time of ACO [46].

5.6 Other Approaches

5.6.1 Artificial Bee Colony

Artificial Bee Colony (ABC) algorithm was invented by Karaboga in 2005 [50]. It is simulating the behavior of honeybees in searching for food sources. The ABC algorithm comprises three key elements: a food source, employed bees, and unemployed bees (scout and onlooker). There are four stages involved; population initialization, employed bee, onlooker, and scout. In the initialization phase, a scout bee initializes a population of food sources (solutions). In the employed bee phase, the information on a food source (amount of nectar) is evaluated. An employed bee then shares the information with an onlooker bee that is waiting in the hive.

In the onlooker bee phase, an onlooker bee probabilistically chooses a rich nectar food source (good quality solution) based on the information shared by the employed bee. A good solution is stored in memory. A poor food source is abandoned. In the scout bee phase, an employed bee with an abandoned food source becomes a scout bee. The scout bee then randomly populates a new solution. This process continues until the best solution is obtained or a termination criterion is met.

ABC algorithm was originally tested with a set of benchmark numerical test functions [50]. It is able to produce very good results at low computational costs in optimization problems. Its successful performance as compared to other well-known evolutionary algorithms (GA, PSO, DE, and ACO) has motivated researchers to extend the use of this algorithm to other areas.

Zhu *et al.* [51] proposed an ABC algorithm to address a school timetabling problem (STP). In a food source initialization, educators were allocated to a scheduled timetable based on their availabilities, preferences, and expertise. Employed bee was then allocated to each of the scheduled timetables. The scheduled timetable was then evaluated. In the onlooker phase, a parameter called trail was used as a boundary to limit the exploration of food sources. The food source was abandoned once the limit had been reached. The employed bee with an abandoned food source then became a scout bee and attempted to find a new food source. A swapping method was applied in a virtual search space (VSS) to improve the bee's ability to search for

a new food source. The proposed ABC was tested on a randomly generated dataset. A comparison was made between ABC and constraint programming (CP). Satisfactory solutions for large-sized instances were obtained.

5.6.2 Cuckoo Search

Cuckoo Search (CS) algorithm was proposed by Yang and Deb in 2009 [52], [53]. It is a nature-inspired metaheuristic algorithm based on the brood parasitic behavior of some cuckoo species that lay eggs in another bird's nest (host nest) [52], [54]. Figure 7 shows the basic structure of CS. In the initial phase, a population of n host nests is generated. Cuckoos are attracted by these initial host nests. They lay their eggs in these host nests randomly (Levy flights). The quality of a host nest is then compared with that of another random host nest. If the random host nest is better, it will replace the old one. A fraction (p_α) of each nest is calculated. The worst nests are abandoned and new ones are constructed. These nests are then evaluated and nests with good quality are kept. The good-quality nests are then ranked and the current best nest is identified. This process iterates until it reaches a maximum generation or stopping criterion. In an early comparison between CS algorithm with GA, PSO, and other conventional algorithms, the CS algorithm showed superior performance in solving many optimization problems [54]. CS algorithm exhibits a good balance of intensive local search and an efficient exploration of the whole search space. In a survey of CS variants and applications [52], it shows advantages in terms of fewer parameters as compared to other algorithms, and ease of hybridization with other optimization algorithms.

Zhang *et al.* [53] proposed a random walked discrete cuckoo search (RW-DCS) algorithm to address a traveling salesman problem (TSP). A discrete cuckoo search was applied to maintain the diversity of a population. An initialization of paths (host nests) was performed by using a roulette wheel method. A starting city in a path was randomly selected. The next city added to the path was selected by a probability based on its distance from the previous city. The quality of a path was evaluated based on its length. The shorter path gives a higher-quality solution. The original Levy flights were replaced by a local adjustment operator (2-opt) and discrete random walk. The fitness of a new path was then calculated. If the fitness of a new path was higher than the original path, the current path was updated as a globally optimal solution. The proposed RW-DCS was tested on a TSPLIB dataset. Its performance was superior in terms of convergence speed and population diversity as compared to other methods (Discrete Cuckoo Search, Discrete Bat Algorithm, and Discrete Sin-Cosine Algorithm).

5.7 Hybrid Evolutionary Algorithm (HEA)

The EA-based hybrid approach is robust in practice and is a rapidly growing research area with great potential [1]. Memetic Algorithm (MA) is one of the most popular EA-based hybrids. MA improves the algorithm performance by incorporating a local search in a Genetic Algorithm. On the other hand, other hybrids could incorporate parallel computation, apply local search strategies on initialization and variation mechanisms, and implement a diversification strategy on population.

5.7.1 HEA (Memetic Algorithm)

Nesmachnow *et al.* [55] proposed a parallel evolutionary algorithm (EA) to address a traffic light synchronization problem. The objective was to improve the average speed of buses and other vehicles. A master-slave model was applied in the parallel EA. In a master phase, a seeded initialization procedure was applied to generate initial candidate solutions (traffic light configurations). These configurations were passed to the slave for simulation (using SUMO) and evaluation. The evaluated configurations were returned to the master for optimization. A one-point crossover was applied to recombine the two fittest configurations. A Gaussian mutation and random modification methods improved the recombined configuration. The best configuration was then determined by using a standard tournament selection. The proposed EA was tested on the actual data of the Bus Rapid Transit system, in Uruguay. Improvements were shown in the bus's average speed (up to 15.4%) and other vehicles (by 24.8%).

Algorithm 6: Cuckoo Search

```
t → 0
Initialize population of n host nests, X_i(i = 1, 2, 3...n)
while (t < MaxGeneration) or (stopping conditions are not met) do
    Get a cuckoo randomly by Levy flight
    Evaluate its fitness, F_i
    Choose a random nest (say, j)
    if F_i > F_j then
        replace j by new solution
    end
    Abandon a fraction (Pα) of worst nests and build new nests
    Keep the best solutions
    Rank the solutions and find the current best
end
```

Figure 7. The basic structure of Cuckoo Search

Rezaeipanah *et al.* [56] proposed an Improved Parallel Genetic Algorithm with Local Search (IPGALS) to address a university course timetabling problem (UCTP). A chromosome was represented in a two-dimensional vector of gene sequences (event time and event location). Initial population was generated randomly and evaluated. A roulette wheel method was used to select two fittest chromosomes. The two fittest chromosomes were then recombined with three crossover operations (uniform, onepoint, and heuristic) to generate offspring. Three mutation operators (local, global, and swap) were performed to refine the offspring. A distance of feasibility (DF) criterion was measured in an improvement operator to reduce a violation of hard constraints. Finally, an elitism operator was used to determine the best offspring. The proposed IPGALS was tested on BenPaechter and ITC-2007 Track 2 benchmark instances. It produced superior solutions compared to other techniques (Improved GA, Improved GA with local search).

Altıntaş *et al.* [57] proposed a self-adaptive MA to address an exam timetabling problem (ETP). ETP involves an assignment of a set of exams to a certain list of periods (subject to some constraints). Multiple constructive low-level heuristics were applied to generate initial solutions (first population). Evaluation of solutions involved penalizing unassigned exams and calculating a number of conflicts. Two fittest solutions were selected (based on historical performance) for crossover operations to obtain offspring. The offspring were refined by using two mutation operators. A Hill Climbing algorithm was then applied to the refined offspring by moving each exam to a period with the lowest penalty. The offspring replaced the worst solution in the population. The proposed algorithm was tested on instances obtained from two universities in Turkey. A comparison of classroom utilization rates was carried out. Self-adapted parameters used in the proposed algorithm demonstrated excellent performance, resulting in improved classroom utilization rates.

Wang *et al.* [58] proposed an adaptive MA to address dynamic electric vehicle routing problems (DEVRP) with time-varying demands. A chromosome was represented in a sequence of customer nodes. In an initialization step, chromosomes were generated randomly and evaluated. The two fittest chromosomes were selected (using a binary tournament) based on the minimum travel distance. Order crossover (CO) and 2-opt mutation operator were then performed to generate offspring. Two different inverse-based adaptive local search operators (Single inverse (SI) and Multiple inverse (MI)) were used to refine the offspring. An immigrant scheme was then used to replace the worst chromosome in the population. The proposed adaptive MA was tested on VRPLIB instances. The proposed algorithm with the adaptive local search (SI) performed better than other algorithms (genetic algorithm and variants of memetic algorithm).

Liu *et al.* [59] proposed a memetic search with efficient local search and extended neighborhood (MATE) in addressing vehicle routing with simultaneous pickup-delivery and time windows. A chromosome was represented in a set of vehicle routes. Each route consisted of a sequence of customer nodes. Residual capacity and radical surcharge (RCRS) heuristics were applied in the initialization stage. Customers were assigned to a route with minimum cost until it reached the RCRS criterion (number of remaining vehicles, and distance of the customer to the depot). In a crossover phase, two parents were randomly selected. An offspring inherits a random route from each parent. A route-assembly-regret-insertion (RARI) method was applied to insert unassigned customers into the offspring. Move operators (local search) with step size changes were then applied to improve the quality of the offspring. A removal-and-insertion operator was employed to obtain the best solution. MATE was tested on a Solomon benchmark dataset. A comparison was made with other algorithms (genetic algorithm, parallel simulated annealing, lexicographic-based two-stage, and adaptive large neighborhood search-path relinking). The proposed MATE algorithm achieved new best solutions in 12 of 65 instances.

Jiang *et al.* [60] proposed a relevance matrix evolutionary algorithm (RMEA) to address a capacitated vehicle routing problem (CVRP). A CVRP contained N customers. A chromosome was represented in a $N \times N$ relevance matrix. The relevance matrix was evaluated based on three aspects: surrounding customers, the maximum distance between all customers, and local information of elite individuals. The two fittest chromosomes were selected for a crossover stage. Three-step crossover and swap-mutation operations were performed on the two fittest chromosomes to generate offspring. The relevance matrix was then updated for the next generation. A diversity preservation strategy was triggered if the best solution was not found in certain generations. Several local operators (remove, swap, and replace) were employed in this strategy to refine the quality of solutions. The proposed RMEA was tested on three CVRP benchmarks (E, CMT, and Golden). It outperformed (fast convergence speed) eight state-of-the-art heuristic methods (genetic algorithm, adaptive variable neighborhood search algorithm, swarm-based constructive optimization algorithm, intelligent water drops algorithm, large neighborhood search algorithm, and ant colony optimization algorithm, ant colony optimization-simulated annealing, and greedy randomized adaptive search procedure).

Al-Taani *et al.* [61] proposed an MA to address a traveling salesman problem (TSP). A solution was represented in a sum of all tours' costs performed by a number of salesmen. Initial tours were selected randomly. A tournament selection method was applied to choose the fittest solution. One-point and two-point strategies were applied to the two fittest solutions in a crossover stage. In the mutation stage, a swapping strategy was then applied to the generated offspring. A local search procedure (hill climbing algorithm) was used to improve each individual's fitness and accelerate the convergence speed. The proposed MA was tested on TSPLIB instances. The algorithm performance was competitive when compared to other methods (ACO, Adaptive PSO, Hybrid PSO).

Afsar *et al.* [62] proposed a memetic evolutionary algorithm (MEA) to address a bi-objective fuzzy job scheduling problem (FJSP). The objective was to minimize makespan and energy consumption. A solution was represented in a sequence of jobs processing time (on all machines). A non-dominated sorting genetic algorithm (NSGA-II) was employed to generate random individuals. Parents were selected by using a lower selective pressure method (randomly paired individuals in a population). Each pair of individuals was then recombined (to generate offspring). The makespan of offspring was improved by using Tabu search. A heuristic procedure was used to reduce the non-processing energy (NPE) of each offspring. NPE was defined as the energy consumed while the machine is in an idle state. Finally, Mixed integer programming (MIP) solver was carried out to further improve these non-dominated solutions. The job starting time of each solution was modified by the MIP solver to

ensure the optimality of NPE. The proposed MEA was tested on twelve FJSP instances. Optimal solutions were obtained.

Nikfarjam *et al.* [63] proposed a bi-level EA to address a traveling thief problem (TTP). The objective was to maximize the structural diversity of a set of solutions. A TTP is a combination of a traveling salesman problem and a knapsack problem. A solution was represented in a set of tours. Each tour consisted of a sequence of cities (with different capacities of items for knapsack). An initial population was randomly generated. Two fittest tours were then selected for edges assembly crossover (EAX) to generate a new tour. The new tour was mutated by a 2-opt operator. A bit-flip mutation was used to generate the new packing list (items with more profits) for the new tour. In a local search, two KP operators (Dynamic programming or alternatively $(1+1)$) were used to determine an optimal packing list for the tour. The new tour with a higher profit of packing list replaced the current solution. The proposed algorithm was tested on TTP benchmark instances. The use of KP operators improved population diversity and produced an efficient set of solutions.

Calvete *et al.* [64] proposed a hybrid EA to address a bus routing problem. The objective was to minimize both the routing cost and the total walking distance of individuals to a pickup point. A chromosome was represented in a set of bus routes. Each route consisted of a sequence of pickup points. Initial solutions were randomly generated and evaluated. Two fittest chromosomes were selected and recombined by two crossover operations (uniform and one-point) to generate offspring. Offspring was then mutated and went through four local search procedures. Finally, a non-dominated sorting genetic algorithm (NSGAII) was applied to arrange the quality of solutions based on their non-domination rank. This strategy identified the best members within the population for the next generation. The proposed algorithm was tested on a set of instances (Schittekat) from literature. The new chromosome representation improved the algorithm's performance.

5.7.2 HEA (Other Hybrid)

Zhang *et al.* [65] proposed a hybrid of Evolutionary Scatter Search with Particle Swarm Optimization (ESS-PSO) to address a vehicle routing problem with a time window (VRPTW). A solution was represented in a sequence of customer nodes. Initial solutions were generated by using an ESS diversification method with two types of Push Forward Insertion Heuristics (PUSH). Two reference sets (minimum travel distance and least number of vehicles) were obtained by using an ESS reference set update method. An ESS subset generation method was then performed to select the fittest solutions (one subset solution and two subset solutions). For one subset solution, a GA operator was performed by applying a route-exchange crossover and customer-extracted mutation. Two new mutated solutions were obtained. For two subset solutions, a new operator called route +/- was performed to generate a new solution. The new solutions obtained were then used as initial particles in a route-segment PSO. One particle was selected as a "starting solution" and its velocity was updated based on the guidance solution (reference sets from ESS) and current solution. Two learning strategies (a path-relinking method and a selection method) were applied to update better solutions. The proposed ESS-PSO was tested on a Solomon benchmark. It produced good solutions as compared to other methodologies (Local genetic algorithm, multi-objective evolutionary algorithm, Tabu Search-Artificial bee colony).

Maskooki *et al.* [66] proposed a Customized Genetic algorithm (CGA) with dynamic programming (DP) to address a moving-target traveling salesman problem (MTSP). The MTSP is similar to a traveling salesman problem. The objective was to maximize the number of ships in a route and to minimize total travel distance within a working day with two shifts (8-16 hours). A solution was represented in a layered graph (discrete time slot). Each layer consisted of nodes (number of ships visited) and a route of surveillance boats. A dynamic programming operator was applied to generate initial solutions. Two criteria (the number of visited boats and the travel distance) were evaluated to select two fittest solutions (parents). A modified crossover and two mutation operators (replacement and insertion) were performed on the parents to obtain two offspring. Infeasible solutions were fixed. The proposed CGA was tested on a real-world dataset of a surveillance boat in the Baltic Sea. It was more competitive in solving large-sized instances than an ILP solver.

Lu *et al.* [67] proposed a Hybrid Evolutionary Algorithm (HEA) to address a covering salesman problem (CSP). The CSP is similar to a traveling salesman problem. A solution was represented in a sequence of nodes traveled by a salesman. Initial solutions were generated by using a greedy constructive procedure and refined by a two-phase tabu search. In the first phase, two operators (insertion and deletion) were used to search for infeasible or redundant solutions. Drop and interchange operators were then used in the second phase to refine the solutions. Two solutions were then selected randomly to perform a solution reconstruction strategy (crossover procedure). A generated offspring was then improved by a destroy-and-repair mutation procedure. The two-phase tabu search was applied again in a local refinement procedure. The proposed HEA was tested on CSP benchmark instances. It produced good solutions for small and medium-sized instances as compared to other methodologies (local search algorithm, integer linear programming-based heuristic algorithm, hybrid ant colony optimization algorithm, multi-start iterated local search algorithm, parallel variable neighborhood search algorithm, ABC, and GA).

Dofadar *et al.* [68] proposed a Hybrid Evolutionary Approach to address a university course allocation problem (UCAP). The hybrid evolutionary approach combined a Local Repair Algorithm (LRA) and a Modified Genetic Algorithm (MGA). A solution was represented in three segments (course index, classroom type, and faculty index). The LRA was used to generate initial solutions. For each solution, the switching tolerance method was applied to reallocate the course and faculty index. Each solution was then evaluated. A solution with a maximum score was then updated (as an intermediate solution) and used in MGA. Course index and faculty index were then recombined in a crossover procedure. If a current score was not updated, the switching tolerance method was activated. A mutation procedure was performed by modifying the faculty and course elements in the solution. The proposed algorithm was tested on a real-world dataset Department of Computer Science & Engineering of BRAC University. It was compared with other methodologies (genetic algorithm, memetic algorithm, stochastic hill climbing, simulated annealing, and tabu search. Cost-efficient and less time-consuming solutions were obtained.

Ilhan [69] proposed an Improved Simulated Annealing with Crossover Operator (ISA-CO) to address a capacitated vehicle routing problem (CVRP). A solution was represented in an integer string (number of customers and number of vehicles). An initial solution was created by randomly picking a starting point. Customers were continuously added to the solution until it

reached the capacity limit. Local search operators (swap, scramble, insertion, and reversion) were applied to improve the initial solutions. Two solutions were then selected based on a predefined rate. A partially mapped crossover (PMX) and order crossover (CO) were performed on the solutions to obtain new solutions. The new solutions were then refined. The proposed ISA-CO was tested on 91 benchmark instances (Augerat *et al.*, Christofides and Eilon, Christofides *et al.*). It performed better than other methodologies (enhanced savings calculation, differential evolution algorithm with local search, hybrid large neighborhood search algorithm, unsupervised fuzzy clustering, artificial bee colony, hybrid firefly algorithm).

Arik [70] proposed a population-based Tabu search (TS_{POP}) with evolutionary strategies to address a permutation flow shop scheduling (PFSS) problem. A solution was represented by a sequence of job orders. A profile fitting procedure (based on the idle time of machines) was applied to generate initial solutions. The solutions were evaluated and arranged in ascending order (based on makespan value). Local search procedures (insertion, swapping, double swapping) were applied to improve the quality of each solution. If the solutions were trapped in local optima, two-point crossover, and swap-mutation procedures were carried out to refine the solutions. The best solution was then updated. The proposed TS_{POP} was tested on Taillard's instances. It produced good quality solutions as compared to others (discrete differential algorithm, and iterated greedy).

Agrawal *et al.* [71] proposed a two-stage evolutionary algorithm to address a multi-objective Euclidean traveling salesman problem (ETSP). The two stages were identified as heuristic local search HLS and the second stage of heuristic local search (SHLS). Both stages consisted of identical steps, including 1) initialization, 2) non-dominated ranking, 3) guided crossover, 4) guided mutation, and 5) elitism. A solution (chromosome) was represented in a sequence of cities traveled by a salesman. In the HLS, each chromosome was formed by randomly picking a starting city and extended by using a nearest-neighbor strategy. The fitness of each chromosome was then evaluated and sorted by using a non-dominated ranking method. A guided crossover was used to generate offspring. The quality of the offspring was then improved by a 2-opt mutation operator. The offspring were used as seed solutions in the SHLS. However, the initialization step in SHLS applied a random weight for distances of cities. The steps of HLS were then repeated. Finally, an elitism method was applied to select the best solution. The proposed algorithm was tested on four TSP benchmark datasets. It showed great performances as compared to others (classic GA, PSO, ACO, and hybrid PSO-ACO).

Lu *et al.* [72] proposed a hybrid evolutionary approach (HEA) to address a minimum spanning tree problem (MSTP). A solution was represented in a set of vertices. A randomized construction procedure was applied to generate initial solutions. The initial solutions were then improved by a tabu search procedure. Two parents were randomly selected. A backbone-based crossover was performed on the parents to generate offspring. A destroy-and-repair procedure was used in a mutation operator to generate multiple distinct offspring. An iterated tabu search (ITS) procedure with threshold exploration was then carried out to search for the best solution. The proposed HEA was tested on benchmark instances from OR-library. It produced good quality solutions in competitive computational times as compared to others (large-scale neighborhood search algorithm, ACO, enhanced second order algorithm, relaxation adaptive memory programming algorithm, and biased random-key genetic algorithm).

A total of eighteen papers on HEA are reviewed: memetic algorithms (10) and other hybrids (8). Additional local search [60], [63], [70]was incorporated in the proposed algorithms to improve the quality of solutions and increase convergence speed, such as hill-climbing [57], different inverse-based adaptive local search [58], move operator [59], and tabu search[62], [72]. Modifications on genetic operators (crossover and mutation) were made to enhance the balance of exploration and exploitation in algorithms, such as a route-assembly-regret-insertion (RARI) method [59], modified crossover and mutation (replacement and insertion) [66]backbone-based crossover and destroy-and-repair procedure [72]. Furthermore, selection strategies were applied to obtain the best solution; removal-and-insertion operator [59] a non-dominated sorting genetic algorithm (NSGAII) [64], and learning strategies (a path relinking method and selection method) [65].

5.8 Summary of EA Variants and Their Applications

In this section, EA variants and their applications in COP are summarized in Table 4. We can observe that HEA is a mostly applied variant in tackling COP. Out of 18 papers applying HEA, most of the COP covered is a vehicle routing problem. Table 5 presents the summary of the methodology used in each variant for specific COP, organized by year and variant to highlight the trends.

Table 4. EA variants and their applications

COP	GA	DEA	GP	PSO	ACO	Other	HEA
Vehicle Routing	4	-	-	1	1	-	6
Travelling Salesman	-	-	-	1	3	1	5
Job Shop Scheduling	2	4	1	1	1	-	1
Timetabling	2	-	-	3	-	1	4
Knapsack	2	1	-	2	-	-	-
Other	3	-	1	-	-	-	2
Total	13	5	2	8	5	2	18

Table 5. Summary table of EA variants and their applications

Variant	Year	Methodology	COP	Reference
GA	2018	Edge Recombination Crossover, novel method for node inclusion, and Lin-Kernighan heuristic.	Other (Orienteering)	[13]
	2019	GA with tailored instances.	Knapsack	[15]
	2019	Biased random key with GA (BRKGA) and codification predecessor.	Open vehicle routing	[16]
	2019	Cellular Automata and Genetic Algorithm.	Other (Traffic light intersection)	[10]
	2020	Two-stage GA.	Flexible Job Shop Scheduling	[11]
	2020	A block-mining-based artificial chromosome construction strategy.	Group Scheduling	[9]
	2021	GA with waiting strategy (WS) and rerouting indicator (RI).	Vehicle routing simultaneous pickup and delivery	[14]
	2021	GA with penalty operator.	Bus routing	[17]
	2021	Specific 3D chromosomes.	University Timetable Scheduling	[18]
	2021	Stratified opposition-based sampling (SOBS) in GA.	Other (Shortest Path)	[12]
	2022	GA with flexible multi-solutions timetable.	Timetable Schedule	[19]
	2022	Two phases modified adaptive GA: construction and improvement.	Inventory Routing	[20]
	2022	Heavy tail mutations and specific problem crossover.	Knapsack	[21]
DEA	2021	Incorporates new components to solve the standard issue of DE.	Knapsack	[26]
	2022	DEA with uncertainty handling techniques (OCBAT & HTT).	Stochastic Reentrant Job Shop Scheduling	[29]
	2022	Redefined Addition & Subtraction for Crossover & Mutation Purposes.	Other (Ship-Unloading Scheduling)	[30]
	2022	Adaptive multi-objective DEA.	Open shop scheduling	[27]
	2022	Self-adaptive mechanism with different mutation operators.	Permutation Flow Shop Scheduling (PFSS)	[28]
GP	2020	Hyper-heuristic GP.	Multi-skill resource-constrained Project Scheduling	[3]
	2022	Two types of decision trees generate new priority rules.	Flexible Job Scheduling	[32]
PSO	2018	Five steps local search with k-split and split exchange procedures.	Split Delivery Vehicle Routing	[34]
	2018	Hybrid PSO with genetic operators for binary variables.	Multidimensional Knapsack	[38]
	2019	Forceful swap operation with repair mechanism and selective search.	University Course Scheduling	[33]
	2019	Two PSO-based algorithms; APSO and HAPSO.	Multiple Travelling Salesman	[39]
	2020	PSO with local neighborhood optimization procedure.	Knapsack	[37]
	2020	Two-stage PSO with Variable neighborhood search (VNS).	Hybrid Flow Shop	[40]
	2021	PSO with Particle Elimination.	School timetabling	[41]

	2021	Hybrid PSO with two variants: SPSO & MCPSO.	University course timetabling	[36]
ACO	2021	ACO with Symbiotic Organism Search (SOS).	Travelling Salesman	[47]
	2022	Saltatory Evolution ACO.	Travelling Salesman	[46]
	2022	Confidence-based selection ACO.	Capacitated Electric Vehicle Routing	[45]
	2022	An evolutionary experience-guided pheromone updating strategy and novel Intragroup Information-Guided Approach.	Multi-objective TSP	[49]
	2022	Surrogate-assisted ACO.	Resource Constrained Job Scheduling with Uncertainty	[48]
Other approach	2021	ABC algorithm with Virtual Search Space (VSS).	School Timetabling	[51]
	2022	Random walked discrete cuckoo search.	Traveling Salesman	[53]
HEA (MA)	2019	Parallel EA with master-slave model.	Other (Traffic light intersection)	[55]
	2021	Parallel GA with Local Search and Distance of feasibility (DF) criterion.	University course timetabling	[56]
	2021	Self-Adaptive MA.	Exam Timetabling	[57]
	2021	MA with two local search operators (Single and Multiple Insertion).	Dynamic Electric Vehicle Routing	[58]
	2021	MA with efficient local search and extended neighborhood (MATE).	Vehicle Routing with Simultaneous Pickup-Delivery and Time Windows	[59]
	2022	Relevance Matrix Evolutionary Algorithm (RMEA) with diversity preservation strategy.	Capacitate Vehicle Routing	[60]
	2022	A hybrid of GA and Hill Climbing algorithm (HCA).	Multiple Traveling Salesman	[61]
	2022	MA with tabu search and heuristic procedure.	Job Shop Scheduling	[62]
	2022	Bi-level EA with two KP operators.	Other (Travelling Thief)	[63]
	2022	Hybrid EA with four local search procedures.	Bus Routing Design	[64]
HEA (Other)	2018	A hybrid of Evolutionary Scatter Search (ESS) and PSO.	Vehicle routing problem with time windows	[65]
	2021	Customized GA with dynamic programming.	Moving-Target Traveling Salesman	[66]
	2021	Two-phase tabu search with the Lin-Kernighan method.	Covering Salesman	[67]
	2021	Local Repair Algorithm (LRA) and Modified Genetic Algorithm (MGA).	Course Timetabling	[68]
	2021	Improved simulated annealing (for initial solutions) with crossover.	Capacitated Vehicle Routing	[69]
	2021	Population-based tabu search with evolutionary strategies.	Permutation Flow Shop Scheduling (PFSS)	[70]
	2021	Two stages EA with heuristic local search (HLS).	Multi-objective Euclidian TSP	[71]
	2022	Hybrid EA with adaptive tabu search.	Other (Minimum Spanning Tree)	[72]

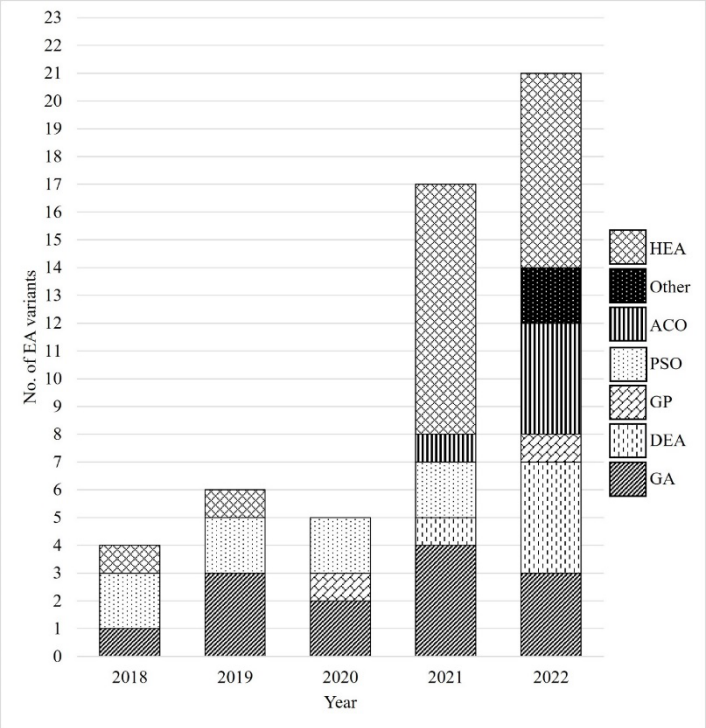

Figure 8. Variants of EA and its application per year

In addition, Figure 8 shows the application of the EA variant in COP domains from 2018 until 2022. The application of EA variants for COPs has demonstrated an increasing trend over these years. We can observe that the total number of HEAs (18) between 2021 and 2022 demonstrates the adaptability of EA in tackling COPs.

6. CHALLENGES AND OPPORTUNITIES OF EA IN COPS

6.1 Diversified Initial Solutions versus Computational Time

The nature of each COP brings different objectives, variables, and constraints. The flexibility of EA in optimizing various COPs influences its overall performance and quality. A high number of variables and problem dimensionality increase the complexity of a COP. EA may struggle to explore the exponential growth in the problem's search space within a reasonable time. Diversified solutions could mitigate suboptimal solutions obtained from inefficient exploration in EA. Premature or slow convergence occurs when suboptimal solutions get trapped in local optima [68]. Initial solutions with better diversity and fitness give a better chance of finding good solutions in EA [12], [69]. Table 6 shows several initialization methods proposed. These methods were proven to give an intensification search by promoting diversity and avoiding premature convergence [62], [65].

Diversified initial solutions provide a reasonable computational time for small-size instances but pose more significant challenges when dealing with large-size instances. Large instances require more computational time due to parameter tunings and additional steps in initialization methods [13]. Excessive exploration in EA hinders an algorithm's capacity to achieve superior solutions within a small number of generations. The proposed initialization methods are still controllable in terms of exploration [12], [59]. Figure 9 shows that the HEA variant mostly applies the initialization method in their mechanism.

Table 6. Initialization methods in EA mechanisms

Variant	COP	Initialization method	Reference
GA	Other (Orienteering)	Random number generation with d2d parameter	[13]
	Other (Shortest path)	Stratified opposition-based sampling (SOBS)	[12]
DEA	Ship scheduling	Random number generation	[30]
PSO	Knapsack	Greedy constructive	[37]
	Job Shop Scheduling	SPT dispatching rule and NEH algorithm	[40]
	Timetabling	Largest unpermitted period degree (LUPD)	[36]
ACO	Vehicle Routing	Nearest neighbor and simple enumeration	[45]
HEA	Job Shop Scheduling	Non-dominated sorting genetic algorithm (NSGA-II)	[62]

	Vehicle Routing	Evolutionary scatter search (ESS) and Push Forward Insertion Heuristics (PUSH).	[65]
	Vehicle Routing	Residual capacity and radical surcharge (RCRS) heuristics	[59]
	Traveling Salesman	Greedy constructive and two-phase tabu search	[67]
	Vehicle Routing	Improved by local search operators (swap, scramble, insertion, and reversion)	[69]
	Job Shop Scheduling	Profile fitting	[70]
	Timetabling	Local Repair Algorithm (LRA)	[68]
	Other (Minimum spanning tree)	Improved by tabu search.	[72]

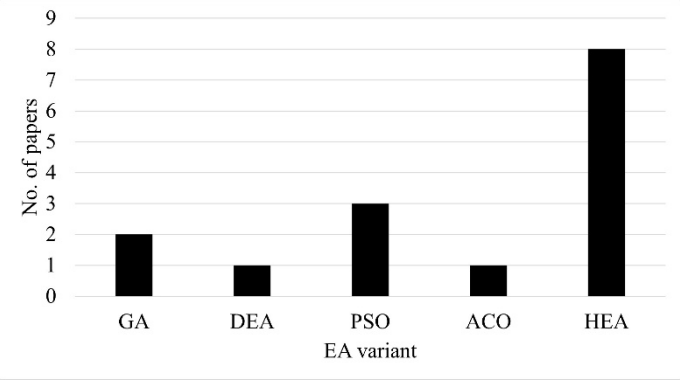

Figure 9. Number of papers applied the initialization method in EA mechanisms

6.2 Balancing Exploration and Exploitation

Balancing exploration and exploitation in EA is crucial to achieving optimal performance. It can be achieved by measuring the convergence speed and computational time. Excessive exploration may lead to slow convergence and require more computational time [12]. Meanwhile, excessive exploitation can result in premature convergence, giving poor solutions [45]. Incorporating local search could improve the exploitation in EA [59], [61], [67]. In optimization, "local search" refers to a heuristic mechanism that is focused on finding a solution within its local neighborhood that is as good or better than others. Such a solution is termed a local optimum [1],[2]. Table 7 shows local search strategies applied in the EA mechanism. These strategies were applied to refine solutions. They improved the convergence speed by promoting new neighborhoods for the current best solutions [34], [70].

Figure 10 shows the maximum number of iterations applied at which the convergence occurs. The number of iterations in each EA variant is determined by improved solutions obtained from applied local search strategies [45], [58]. Improved solutions compared with near-optimal solutions in terms of objective function value, the best solution obtained [40], a hypothesis test model [46], a smaller optimality gap [70], and learning strategies applied [34], [58]. For instance, an adaptive selection strategy applied in [34] destroyed the structure of last generation solution and caused slow convergence speed. In addition, several factors are still being considered to explore different neighborhoods and escape local optima, such as problem representation, size of instances, and initial solutions [34], [45]. Developing appropriate local search strategies is essential to avoid unnecessary local moves [58].

HEA variant shows the most applied local search in their EA mechanism as shown in Figure 11. In the future, the integration of more efficient local search strategies could be implemented to improve both the overall search process and the quality of solutions [58], [59], [61].

Table 7. Local search strategies in EA mechanisms

COP	Local search strategies (Refine solutions)	Variant	Number of local searches	Maximum number of iterations	Reference
Job Shop Scheduling	Variable neighborhood search (VNS)	PSO	1	20	[40]
	Insertion, swapping, double swapping	HEA	3	1000	[70]
	Tabu Search and heuristic procedure	HEA	1	30	[62]

Other (Thief Traveling)	KP operators and Dynamic programming	HEA	2	10000	[63]
Traveling Salesman	Saltatory evolution (path performance evaluation, near-optimal path identification rules generation, and near-optimal path identification.)	ACO	3	200	[46]
	Hill climbing	HEA	1	1000	[61]
Vehicle Routing	5 local strategies were used in genetic operators (relocation, exchange, k-split, split exchange, split swap, and 2-opt)	PSO	5	15000	[34]
	Confidence-based selection, 2-opt, and node-shift strategy	ACO	3	30	[45]
	Single inverse and multiple inverse.	HEA	2	30	[58]
	Move operators (local search) with step size changes	HEA	1	50	[59]
	A diversity preservation strategy: local operators (remove, swap, and replace)	HEA	3	100	[60]

6.3 Specific Design Operator to Meet the Type of COP

Choosing the right operators, such as selection, crossover, and mutation, is crucial for effectively balancing exploration and exploitation in EA. The operators are specifically constructed to align with the distinct characteristics and demands of the problem at hand. Collaboratively, they enhance the quality of solutions through consecutive generations [64]. Table 8 shows operators introduced to replace the crossover, mutation, and selection mechanism in EA.

The specific design operators employed to substitute the mutation and crossover operators in EA for a more compact search [29]. Furthermore, the selection operators applied could avoid repeating the evaluation of poor solutions [29], hence reducing the computational cost [27].

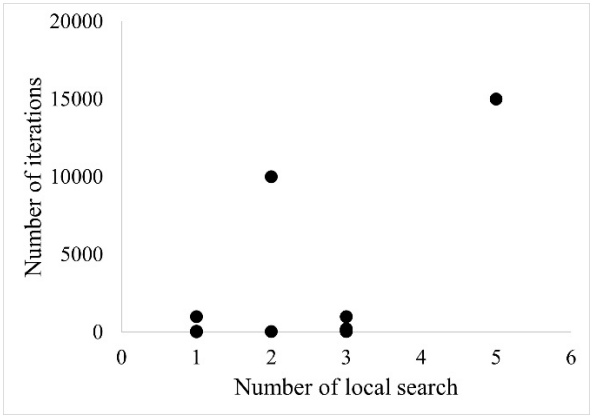

Figure 10. Number of iterations based on local search applied in EA mechanisms

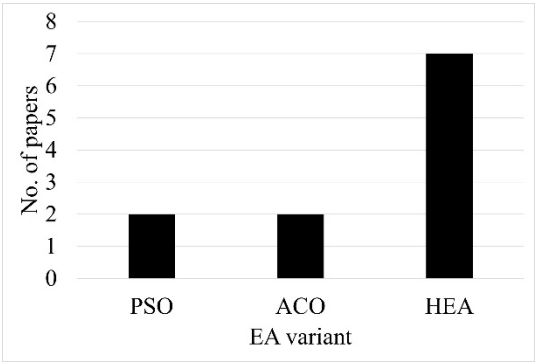

Figure 11. Number of papers applied local search in EA mechanisms

Table 8. Specific crossover, mutation, and selection in EA mechanisms

Variant	COP	Crossover	Mutation	Selection	Reference
GA	Knapsack	Specific crossover operator	Heavy-tailed	Standard selection	[21]
DEA	Job Shop Scheduling	Interchange-based operator	Insert-based operator	Hypothesis test technique (HTT)	[29]
	Job Shop Scheduling	Standard crossover	Adaptive opposition-based learning applied	External archive technique	[27]
PSO	Timetabling	Swap sequence strategy	Repair mechanism	Selective search	[33]
	Timetabling	Horizontal and vertical swapping of matrices		Particles elimination	[41]
ACO	Traveling Salesman	Intragroup evolutionary information-guided pheromone updating (IGPU)		Learning automata (LA) with adaptive strategies	[49]
HEA	Vehicle Routing	Random crossover	Route-assembly-regret-insertion (RARI) method	A removal-and-insertion operator	[59]
	Vehicle Routing	Uniform and one-point	Mutated using four local search procedures	NSGAII	[64]
	Vehicle Routing	route-exchange crossover, customer-extracted mutation, and route +/- operator.		Two learning strategies (a path-relinking method and a selection method)	[65]
	Timetabling	MGA crossover	MGA mutation	Switching tolerance	[68]
	Minimum spanning tree	Backbone-based crossover	Destroy-and-repair procedure	Iterated tabu search (ITS) procedure with threshold exploration	[72]

6.4 Population Diversity versus Parameter Tuning

Population diversity tends to decline over the evolutionary process. It is primarily attributed to the exploitation aspect of an algorithm due to the selection of higher-fitness individuals for the next generations. Multiple studies have recommended introducing variation and adjusting relevant parameters as potential solutions to address this issue [47], [71]. However, a large number of parameters gives more combinatorial space for parameter setting and requires more time to manually manage the parameter tuning. Figure 12 shows the average computational time of an algorithm for different numbers of parameters used in the self-adaptive strategies. Note that the computational time for different strategies may vary depending on the specific type of COP under investigation.

A self-adaptive algorithm was introduced in EA to control the parameter setting [28] and to align with the problem representation [20]. Table 9 shows the application of self-adaptive parameters in EA. Adaptive crossover and mutation operators were created by learning experiences of individuals' fitness values [58] These operators use real-time feedback to control the selection mechanism of local optimization algorithms [20], [57].

In summary, diversified initial solutions give more exploration space in EA. However, excessive exploration could lead to slow convergence and require more computational time. Applying local search, designing specific operators for different types of COP, and employing self-adaptive strategies could improve the algorithm's performance while achieving better solutions. Across these modifications, HEA was observed to apply more local search and specific design operators (see Table 10). In the PSO variant, both studies achieved less than 0.0001 p-values as compared to other methods in terms of best solutions [37], [40]. Meanwhile, two variants of confidence-based selection ACO had shown different effects on the diversified initial solutions and local search applied in terms of convergence speed and solution quality [45].

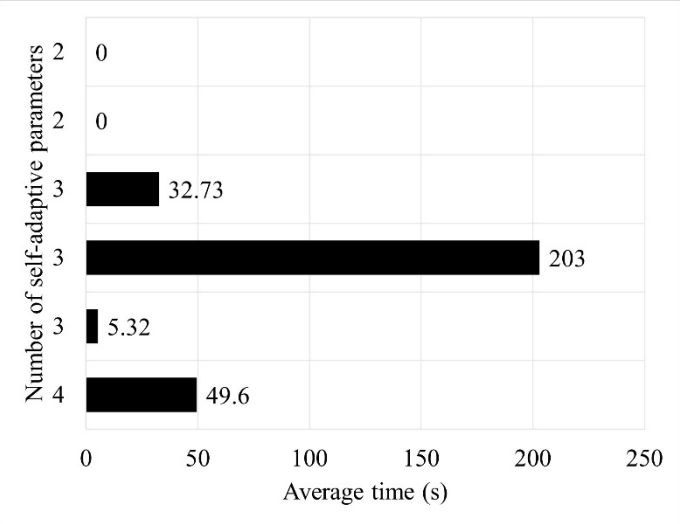

Figure 12. Average time (in seconds) for different numbers of parameters applied in EA mechanisms

Table 9. Self-adaptive parameters in EA

Variant	COP	Self-adaptive strategies	Parameters	Number of instances	Average Time (s)	Reference
GA	Vehicle Routing	Adaptive crossover and mutation operators	4	96	49.6	[20]
DEA	Job Shop Scheduling	Three types of Self Adaptive DEA introduced in mutation	3	110	5.32	[28]
	Ship scheduling	Semi-adaptive parameters	3	25	209	[30]
PSO	Knapsack	Adaptive parameter swarm intelligence	3	45	32.73	[37]
ACO	Traveling Salesman	Parameters controlled by the Symbiotic Organisms Search (SOS) method	2	10	N/A	[47]
HEA	Vehicle Routing	Adaptive local search: Single inverse and multiple inverse.	2	1	N/A	[58]

Table 10. EA variants and applications of initialization method, local search, specific operators, and self-adaptive parameters

Reference	Initialization method	Local search	Specific Operators	Self-adaptive parameters	Variant
[65]	/	-	/	-	HEA
[40]	/	/	-	-	PSO
[37]	/	-	-	/	PSO
[59]	/	/	/	-	HEA
[45]	/	/	-	-	ACO
[68]	/	-	/	-	HEA
[62]	/	/	-	-	HEA
[72]	/	-	/	-	HEA
[30]	/	-	-	/	DEA

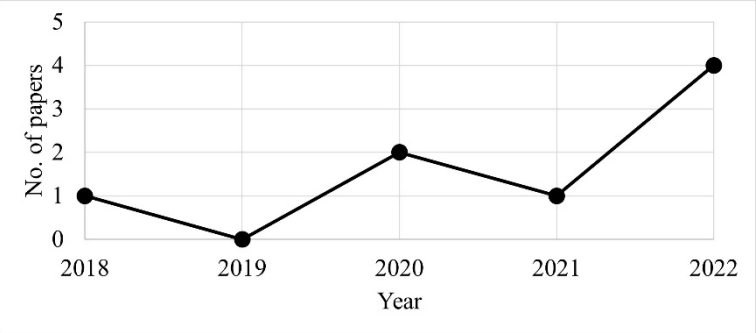

Figure 13. The trend of hybridization strategies (initialization method, local search, specific operators, self-adaptive parameters) in EA mechanisms

From Figure 13, the application of hybridization strategies in EA increased since 2018. Adopting strategies such as initialization method, local search, specific design operators, and self-adaptive parameters has shown that the HEA variant is highly competitive in tackling COP. Better performance is achieved in terms of convergence speed [65]. The use of local search (such as move operator with step changes) is effective in obtaining high-quality solutions [46]. In the future, different hybridizations of optimization EA can be explored to obtain a more robust performance of HEA [68].

7. CONCLUSION

An evolutionary algorithm (EA) is a population-based and metaheuristic approach that complies with five mechanisms: initialization, evaluation, selection, variation, and replacement. It can be applied to a diverse array of problems such as COP with minimal need for tailoring. This paper surveys 53 applications of EA in six types of COPs: vehicle routing problem, knapsack problem, job scheduling problem, timetabling problem, knapsack problem, and others. HEA variant has shown an increasing trend since 2018. Adoption of hybridization strategies such as initialization method, local search, specific design operator, and self-adaptive parameter tuning in EA mechanisms improve the algorithm performance in terms of convergence speed and solution quality. Challenges and opportunities of these achievements are discussed to provide improvement in the future.

ACKNOWLEDGEMENT AND FUNDING

The authors receive no financial support for the research, authorship, and publication of this article.

DECLARATION OF CONFLICTING INTERESTS

The authors declare no potential conflicts of interest with respect to the research and publication of this article.

REFERENCES

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.
[2] E. K. Burke and G. Kendall, *Search Methodologies*. Boston, MA: Springer US, 2014.
[3] J. Lin, L. Zhu and K. Gao, A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem, *Expert Systems with Applications*, 140, 2020, 112915.
[4] H. R. Maier, S. Razavi, Z. Kapelan, L. S. Matott, J. Kasprzyk and B. A. Tolson, Introductory overview: Optimization using evolutionary algorithms and other metaheuristics, *Environmental Modelling & Software*, 114, 2019, 195-213.
[5] Y. Qu, Z. Ma, A. Clausen and B. N. Jorgensen, A comprehensive review on evolutionary algorithm solving multi-objective problems, *22nd IEEE International Conference on Industrial Technology (ICIT)*, Valencia, Spain, 2021.
[6] B. Doerr and F. Neumann, A survey on recent progress in the theory of evolutionary algorithms for discrete optimization, *ACM Transactions on Evolutionary Learning*, 1(4), 2021, 1-43.
[7] K. Sastry, D. E. Goldberg and G. Kendall, Genetic algorithms, in *Search Methodologies*, Boston, MA: Springer US, 2014, 93-117
[8] A. E. Eiben and J. E. Smith, Popular evolutionary algorithm variants, in *Introduction to Evolutionary Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, 99-116.
[9] S. Yuan, T. Li and B. Wang, A co-evolutionary genetic algorithm for the two-machine flow shop group scheduling problem with job-related blocking and transportation times, *Expert Systems with Applications,* 152, 2020, 113360.
[10] L. Cruz-Piris, M. A. Lopez-Carmona and I. Marsa-Maestre, Automated optimization of intersections using a genetic algorithm, *IEEE Access*, 7, 2019, 15452-15468.
[11] F. M. Defersha and D. Rooyani, An efficient two-stage genetic algorithm for a flexible job-shop scheduling problem with sequence dependent attached/detached setup, machine release date and lag-time, *Computers & Industrial Engineering,* 147, 2020, 106605.
[12] A. Ghannami, J. Li, A. Hawbani and A. Al-Dubai, Stratified opposition-based initialization for variable-length chromosome shortest path problem evolutionary algorithms, *Expert Systems with Applications,* 170, 2021, 114525.

[13] G. Kobeaga, M. Merino and J. A. Lozano, An efficient evolutionary algorithm for the orienteering problem, *Computers & Operations Research*, 90, 2018, 42-59.
[14] H. Park, D. Son, B. Koo and B. Jeong, Waiting strategy for the vehicle routing problem with simultaneous pickup and delivery using genetic algorithm, *Expert Systems with Applications*, 165, 2021, 113959.
[15] L. F. Plata-González, I. Amaya, J. C. Ortiz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín and C. A. Coello, Evolutionary-based tailoring of synthetic instances for the Knapsack problem, *Soft Computing*, 23(23), 2019, 12711-12728.
[16] E. Ruiz, V. Soto-Mendoza, A. E. Ruiz Barbosa and R. Reyes, Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm, *Computers & Industrial Engineering*, 133, 2019, 207-219.
[17] A. R. Komijan, P. Ghasemi, K. Khalili-Damghani and F. HashemiYazdi, A new school bus routing problem considering gender separation, special students and mix loading: a genetic algorithm approach, *Journal of Optimization in Industrial Engineering,* 14(2), 2021, 23-39.
[18] E. C. Pérez, O. M. Rios, D. P. Bautista, S. S. Sanchez and F. A. Acevedo, A genetic algorithm solution for scheduling problem, *XVII International Engineering Congress (CONIIN)*, Queretaro, Mexico, 2021.
[19] I. A. Abduljabbar and S. M. Abdullah, An evolutionary algorithm for solving academic courses timetable scheduling problem, *Baghdad Science Journal*, 19(2), 2022, 398-408.
[20] M. Mahjoob, S. S. Fazeli, S. Milanlouei, L. S. Tavassoli and M. Mirmozaffari, A modified adaptive genetic algorithm for multi-product multi-period inventory routing problem, *Sustainable Operations and Computers*, 3, 2022, 1-9.
[21] A. Neumann, Y. Xie and F. Neumann, Evolutionary algorithms for limiting the effect of uncertainty for the knapsack problem with stochastic profits*, Lecture Notes in Computer Science*, 198, 2022, 294-307.
[22] Z. Zeng, M. Zhang, T. Chen and Z. Hong, A new selection operator for differential evolution algorithm, *Knowledge-Based Systems*, 226, 2021, 107150.
[23] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song and J. Xu, An improved differential evolution algorithm and its application in optimization problem, *Soft Computing*, 25(7), 2021, 5277-5298.
[24] Q. -K. Pan, P. N. Suganthan, L. Wang, L. Gao and R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, *Computers & Operations Research*, 38(1), 2021, 394-408.
[25] K. R. Opara and J. Arabas, Differential evolution: A survey of theoretical analyses, *Swarm and Evolutionary Computation*, 44, 2019, 546-558.
[26] I. M. Ali, D. Essam and K. Kasmarik, Novel binary differential evolution algorithm for knapsack problems, *Information Sciences*, 542, 2021, 177-194.
[27] L. He, Y. Cao, W. Li, J. Cao and L. Zhong, Optimization of energy-efficient open shop scheduling with an adaptive multi-objective differential evolution algorithm, *Applied Soft Computing*, 118, 2022, 108459.
[28] M. de F. Morais, M. H. D. M. Ribeiro, R. G. da Silva, V. C. Mariani and L. dos S. Coelho, Discrete differential evolution metaheuristics for permutation flow shop scheduling problems, *Computers & Industrial Engineering*, 166, 2022, 107956.
[29] R. Hu, X. Wu, B. Qian, J. Mao and H. Jin, Differential evolution algorithm combined with uncertainty handling techniques for stochastic reentrant job shop scheduling problem, *Complexity*, 2022, e9924163.
[30] Z. Gao, M. Zhang and L. Zhang, Ship-unloading scheduling optimization with differential evolution, *Information Sciences,* 591, 2022, 88-102.
[31] W. B. Langdon, R. Poli, N. F. McPhee and J. R. Koza, Genetic programming: an introduction and tutorial, with a survey of techniques and applications, *Computational Intelligence: A Compendium*, 115, 2008, 927-1028.
[32] R. Braune, F. Benda, K. F. Doerner and R. F. Hartl, A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems, *International Journal of Production Economics*, 243, 2022, 108342.
[33] Sk. Imran Hossain, M. A. H. Akhand, M. I. R. Shuvo, N. Siddique and H. Adeli, Optimization of university course scheduling problem using particle swarm optimization with selective search, *Expert Systems with Applications*, 127, 2019, 9-24.
[34] J. Shi, J. Zhang, K. Wang and X. Fang, Particle swarm optimization for split delivery vehicle routing problem, *Asia-Pacific Journal of Operational Research*, 35, 2018, 1840006.
[35] D. Merkle and M. Middendorf, Swarm Intelligence, *Springer eBooks*, 2013, 213-242.
[36] T. Thepphakorn, S. Sooncharoen and P. Pongcharoen, Particle swarm optimisation variants and its hybridisation ratios for generating cost-effective educational course timetables, *SN Computer Science*, 2, 2021, 4.
[37] I. Dahmani, M. Hifi, T. Saadi and L. Yousef, A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict Graphs, *Expert Systems with Applications*, 148, 2020, 113224.
[38] L. F. Mingo López, N. Gómez Blas A. Arteta Albert, Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations, *Soft Computing*, 22(8), 2017, 2567-2582.
[39] S. D. Gulcu and H. K. Ornek, Solution of multiple travelling salesman problem using particle swarm optimization based algorithms, *International Journal of Intelligent Systems and Applications in Engineering,* 7(2), 2019, 72-82.
[40] M. K. Marichelvam, M. Geetha and Ö. Tosun, An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors – A case study, *Computers & Operations Research*, 114, 2020, 104812.
[41] J. S. Tan, S. L. Goh, S. Sura, G. Kendall, and N. R. Sabar, Hybrid particle swarm optimization with particle elimination for the high school timetabling problem, *Evolutionary Intelligence*, 1, 2021, 1915-1930.
[42] J. -L. Deneubourg, S. Aron, S. Goss and J. M. Pasteels, The self-organizing exploratory pattern of the argentine ant, *Journal of Insect Behavior*, 3(2), 1990, 159-168.

[43] S. Goss, S. Aron, J. L. Deneubourg and J. M. Pasteels, Self-organized shortcuts in the Argentine ant, *Naturwissenschaften*, 76(12), 1989, 579-581.
[44] M. Dorigo and C. Blum, Ant colony optimization theory: A survey, *Theoretical Computer Science*, 344(2-3), 2005, 243-278.
[45] Y. -H. Jia, Y. Mei and M. Zhang, Confidence-based ant colony optimization for capacitated electric vehicle routing problem with comparison of different encoding schemes, *IEEE Transactions on Evolutionary Computation,* 26(6), 2022, 19-1408.
[46] S. Li, Y. Wei, X. Liu, H. Zhu and Z. Yu, A New fast ant colony optimization algorithm: the saltatory evolution ant colony optimization algorithm, Mathematics, 10(6), 2022, 925.
[47] Y. Wang and Z. Han, Ant colony optimization for traveling salesman problem based on parameters optimization, *Applied Soft Computing*, 107, 2021, 107439.
[48] D. Thiruvady, S. Nguyen, F. Shiri, N. Zaidi and X. Li, Surrogate-assisted population based ACO for resource constrained job scheduling with uncertainty, *Swarm and Evolutionary Computation*, 69, 2022, 101029.
[49] H. Zhao and C. Zhang, An ant colony optimization algorithm with evolutionary experience-guided pheromone updating strategies for multi-objective optimization, *Expert Systems with Applications*, 201, 2022, 117151.
[50] D. Karaboga, B. Gorkemli, C. Ozturk and N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, *Artificial Intelligence Review*, 42(1), 2012, 21-57.
[51] K. Zhu, L. D. Li and M. Li, School Timetabling Optimisation using artificial bee colony algorithm based on a virtual searching space method, *Mathematics*, 10(1), 2021, 73.
[52] M. Shehab, A. T. Khader and M. A. Al-Betar, A survey on applications and variants of the cuckoo search algorithm, *Applied Soft Computing*, 61, 2017, 1041-1059.
[53] Z. Zhang and J. Yang, A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization, *Computers & Industrial Engineering*, 169, 2022, 108157.
[54] X. -S. Yang and S. Deb, Cuckoo search: recent advances and applications*, Neural Computing and Applications,* 24(1), 2013, 169-174.
[55] S. Nesmachnow et al., Traffic lights synchronization for bus rapid transit using a parallel evolutionary algorithm, *International Journal of Transportation Science and Technology*, 8(1), 2019, 53-67.
[56] A. Rezaeipanah, S. S. Matoori and G. Ahmadi, A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search, *Applied Intelligence*, 51, 2021, 67-92.
[57] T. Yiğit and C. Altıntaş, Analysing the effects of classroom utilisation with a self-generating multimeme memetic algorithm for the exam timetabling problem, *Advances in Artificial Intelligence Research (AAIR)*, 1(2), 2021, 43-51.
[58] N. Wang, Y. Sun and H. Wang, An adaptive memetic algorithm for dynamic electric vehicle routing problem with time-varying demands, *Mathematical Problems in Engineering*, 2021, 1-10.
[59] S. Liu, K. Tang and X. Yao, Memetic search for vehicle routing with simultaneous pickup-delivery and time windows, *Swarm and Evolutionary Computation*, 66, 2021, 100927.
[60] H. Jiang, M. Lu, Y. Tian, J. Qiu and X. Zhang, An evolutionary algorithm for solving capacitated vehicle routing problems by using local information, *Applied Soft Computing*, 117, 2022, 108431.
[61] A. T. Al- Taani and L. M. Al-Afifi, Solving the multiple traveling salesman problem using memetic algorithm, *Artificial Intelligence Evolution*, (1), 2022, 27-40.
[62] S. Afsar, J. J. Palacios, J. Puente, C. R. Vela and I. González-Rodríguez, Multi-objective enhanced memetic algorithm for green job shop scheduling with uncertain times, *Swarm and Evolutionary Computation*, 68, 2022, 101016.
[63] A. Nikfarjam, A. Neumann and F. Neumann, Evolutionary diversity optimisation for the traveling thief problem, *Proceedings of the Genetic and Evolutionary Computation Conference*, Boston, Massachusetts, 2022.
[64] H. I. Calvete, C. Galé and J. A. Iranzo, Approaching the Pareto front in a biobjective bus route design problem dealing with routing cost and individuals' walking distance by using a novel evolutionary algorithm, *Mathematics*, 10(9), 2022, 1390.
[65] J. Zhang, F. Yang and X. Weng, An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows, *IEEE Access*, 6, 2018, 63468-63485.
[66] A. Maskooki, K. Deb and M. Kallio, A customized genetic algorithm for bi-objective routing in a dynamic network, *European Journal of Operational Research*, 297(2), 2022, 615-629.
[67] Y. Lu, U. Benlic and Q. Wu, A highly effective hybrid evolutionary algorithm for the covering salesman problem, *Information Sciences,* 564, 2021, 144-162.
[68] D. F. Dofadar, R. H. Khan, S. Hasan, T. A. Taj, A. Shakil and M. Majumdar, A hybrid evolutionary approach to solve university course allocation problem, *2021 International Conference on Artificial Intelligence and Blockchain Technology*, Beijing, China, 2021.
[69] İ. İlhan, An improved simulated annealing algorithm with crossover operator for capacitated vehicle routing problem, *Swarm and Evolutionary Computation*, 64, 2021, 100911.
[70] O. A. Arık, Population-based Tabu search with evolutionary strategies for permutation flow shop scheduling problems under effects of position-dependent learning and linear deterioration, *Soft Computing*, 25(2), 2020, 1501-1518.
[71] A. Agrawal, N. Ghune, S. Prakash and M. Ramteke, Evolutionary algorithm hybridized with local search and intelligent seeding for solving multi-objective Euclidian TSP, *Expert Systems with Applications*, 181, 2021, 115192.
[72] Y. Lu, U. Benlic and Q. Wu, A hybrid evolutionary algorithm for the capacitated minimum spanning tree problem, *Computers & Operations Research*, 144, 2022, 105799.