We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,300

Open access books available

172,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Search Algorithms on Logistic and Manufacturing Problems

Gladys Bonilla-Enriquez

and Santiago-Omar Caballero-Morales

Abstract

The supply chain comprehensively considers problems with different levels of complexity. Nowadays, design of distribution networks and production scheduling are some of the most complex problems in logistics. It is widely known that large problems cannot be solved through exact methods. Also, specific optimization software is frequently needed. To overcome this situation, the development and application of search algorithms have been proposed to obtain approximate solutions to large problems within reasonable time. In this context, the present chapter describes the development of Genetic Algorithms (an evolutionary search algorithm) for vehicle routing, product selection, and production scheduling problems within the supply chain. These algorithms were evaluated by using well-known test instances. The advances of this work provide the general discussions associated to designing these search algorithms for logistics problems.

Keywords: vehicle routing problem, knapsack problem, flow-shop Scheduling, local-search Algorithms, genetic algorithms

1. Introduction

According to the Council of Supply Chain Management Professionals (CSCMP), logistics is defined as the process of planning, implementing and controlling all operations and information flow for the efficient and effective transportation and storage of goods or services from a point of origin to a point of consumption. As presented in **Figure 1**, many operations are involved in a logistics network, and manufacturing is a crucial operation to transform inbound goods (e.g., raw materials) into outbound goods (e.g., end products, sub-assemblies, work-in-process, etc.) throughout this network.

Due to the complexity of these operations, where many of them involve problems of NP-hard computational complexity, research and improvement efforts require the use of advanced of quantitative and qualitative strategies and tools. Among these, the use of Search Algorithms such as meta-heuristics has been proposed to solve to near-optimality large NP-hard problems within reasonable time [1].

As presented in **Figure 1**, transportation is needed for the efficient flow of goods throughout the supply chain (SC). Thus, the analysis and solution of routing problems are the first set of problems to be addressed in this chapter.

Then, manufacturing planning is needed to achieve the required quantities of sub-assemblies and end-products to supply the customers (or even other suppliers)

Figure 1.
General example of a logistics network.

in time through the SC. Thus, production planning problems are the second set of problems to be addressed in this chapter. Note that both sets are mutually important and dependent for the appropriate performance of the SC.

While there are many search algorithms or meta-heuristic approaches to solve these problems, this chapter addresses the specific configuration settings to apply Genetic Algorithms (GA) to solve both sets of problems. As the solutions have different representations (i.e., permutations, binary chains, real numbers), having a common algorithmic base can lead to a better understanding for successful implementation for other problems and contexts.

GA are based on the principle of natural selection of “survival of the fittest” where individuals within a population compete between each other for vital resources (i.e., food, shelter, etc.) and/or to attract mates for reproduction. Due to this selection mechanism, it is expected that poorly performing individuals have less chance to survive in contrast to the most adapted or “fit” individuals which are more likely to reproduce, inheriting their good characteristics to their offspring to make them better and more adapted to their environment [2].

Figure 2 presents the general structure and main elements of a GA. This meta-heuristic is population-based. Thus, it works by continuously improving on a set of solutions by using reproduction operators which facilitate the search mechanisms for the solution space of the problem. This set, known as the population, consists of N feasible solutions which are evaluated through a fitness function (i.e., the total distance equation, or objective function, to determine the total cost associated to each solution). Then, the solutions with the best fitness values become candidates for reproduction to (hopefully) inherit their best features to new solutions and improve the overall population in the next generation (iteration). It is expected that after X generations the mean fitness of the population converges to a local optimum.

Within this context, the present chapter addresses the different representations of candidate solutions, fitness functions, and reproduction operators, for the application of GA to solve the following sets of problems:

- Routing Planning (Section X.2): Traveling Salesman Problem (TSP) and Capacitated Vehicle Routing Problem (CVRP).

Figure 2.
General structure and main elements of a GA.

- Production and Selection of the Most Profitable Goods (Section X.3): Economic Lot Problem with Multiple Items and Knapsack Problem.
- Production Scheduling (Section X.4): Permutation Flow-Shop Scheduling Problem.

This chapter ends with a discussion of the results and the practical implications of the future work (Section X.5).

2. Genetic algorithm for route planning problems

2.1 Traveling salesman problem

The Traveling Salesman Problem (TSP) represents the scenario of a salesperson who must visit each place within a set of cities or towns. This must be performed with the following considerations: the salesperson starts and ends the whole journey at a single location (i.e., the main office) and must visit each place only once [3]. Although this is the basic understanding of the TSP, the main feature of finding a single route, or sequence of minimum distance or cost, is shared by other real-world applications such as vehicle routing [4], production planning [5], service time [6], and design of computer networks [7]. **Figure 3** presents an overview of the TSP model with $n = 12$ cities.

Note that each single route that complies with the previous restrictions represents a candidate solution, and there are as much as $n!$ candidate solutions if brute search were to be considered as solving method to find the optimal or best solution. Just for the example presented in **Figure 3**, there are up to $12! = 479'001,600$ or

Figure 3.
Example of a feasible TSP solution with $n = 12$ cities.

479.00e+006 feasible solutions to visit all 12 cities. This number increases exponentially as n increases linearly. Thus, if just a single city is added to the TSP problem, the number of feasible solutions can increase to $13! = 6.23\text{e}+009$.

This leads to a problem with an infinite solution space if large sets of cities are considered. This classifies the TSP as an NP-hard problem, which is very difficult to solve within reasonable time, even with the most advanced computational systems. Thus, different meta-heuristics have been developed to provide fast near-to-optimal solutions. Among these meta-heuristics the following can be mentioned [8]: Nearest Neighbor (NN), Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Tree Physiology Optimization (TPO).

As presented in [8] GA and SA are among the most suitable heuristics, achieving error gaps from best known solutions within the 10% mark for small ($n < 100$), moderate ($100 < n < 150$) and large ($150 < n < 450$) TSP instances. However, within the context of TSP solutions, it is always recommended to test the solving methods with very large instances (i.e., $n > 500$) to corroborate their performance.

Thus, the developed GA considers TSP instances with $n \approx 1000$. For this purpose, the GA considers the structure presented in **Figure 2** with the settings and reproduction operators presented in **Table 1** and described in **Figure 4**.

Parameter	Setting
Generations (Iterations)	1000
Fitness Function	Symmetric Euclidean Distance of TSP Route
Population Size	According to the size of the TSP
Selection	Tournament
Reproduction Operators:	
Crossover	Partially-Matched Crossover (PMX)
Mutation	Swap, Inversion

Table 1.
GA settings for the TSP.

Figure 4.
Partially-matched crossover, and swap/inversion mutation operators for the TSP.

Implementation of the GA was performed in MATLAB with an Intel Core i7–5500 CPU at 2.40 GHz and 8GB RAM. Testing was performed with a set of TSP instances from the TSPLIB95 database [9]. The details of these instances, including the GA’s population size N used for each case, are presented in **Table 2**. The results of the tests can be observed in **Figure 5** and **Figure 6**.

As presented in **Figure 5**, the mean error gap through all instances begins to decrease as the selection and reproduction mechanisms of the GA start to operate on the initial and updated populations. By the 300th generation the mean error gap decreases under the 10% mark to finally reach an approximate of 7% by the 1000th generation. This corroborates the performance reported in [8].

Finally, **Figure 6** presents the performance of the GA based on the size of the test instances (n). With the settings reported in **Table 1**, as n increases, the GA takes more time to converge to a local optimum which, in some cases, it is slightly over the 10% mark. Also, the size of the population (N) must be increased to improve the search performance.

Based on these findings, particularly for the TSP with $n \approx 1000$, the following recommendations can be made:

- Diversification of solutions depends of the size of the population (N) and the TSP (n). Because N is the only controllable parameter, it is important to find an appropriate balance between it and n because a large N can increase the computational memory load of the algorithm which is already affected by n .
- A larger number of generations should be considered for large TSP problems. This because convergence may get slower due to n , independently of the reproduction or selection operators, or the size of the population.
- Integration with other heuristics or meta-heuristics can improve on the initial population or some of the search operators, and thus, on the convergence of the GA through all generations. This process, called hybridization, has led to obtain very suitable results for large TSP instances [10].

As an example of hybridization, **Figures 5** and **6** present the performance of the revised GA (hybrid-GA) with a much smaller N ($= 50$ for all instances) and a Greedy algorithm to improve four offspring (two by crossover, one by flip mutation, one by swap mutation) which are included within the updated population. This increases the speed of the GA, reaching the 10% by the 100th generation, with

a final mean error gap of 5% by the 1000th generation. Also, improvement of the large instances ($n > 500$) is observed, achieving error gaps under the 10% mark.

<i>N</i>	Size of the TSP (<i>n</i>)	Name of the Instance	<i>N</i>	Size of the TSP (<i>n</i>)	Name of the Instance
200	51	eil51.tsp	200	198	d198.tsp
200	52	berlin52.tsp	300	200	kroA200.tsp
200	70	st70.tsp	300	200	kroB200.tsp
200	76	eil76.tsp	300	225	ts225.tsp
200	76	pr76.tsp	300	225	tsp225.tsp
200	99	rat99.tsp	300	226	pr226.tsp
200	100	kroA100.tsp	300	262	gil262.tsp
200	100	kroB100.tsp	300	264	pr264.tsp
200	100	kroC100.tsp	300	280	a280.tsp
200	100	kroD100.tsp	300	299	pr299.tsp
200	100	kroE100.tsp	300	318	lin318.tsp
200	100	rd100.tsp	500	400	rd400.tsp
200	105	lin105.tsp	500	417	fl417.tsp
200	107	pr107.tsp	500	439	pr439.tsp
200	124	pr124.tsp	500	442	pcb442.tsp
200	127	bier127.tsp	500	493	d493.tsp
200	130	ch130.tsp	500	574	u574.tsp
200	136	pr136.tsp	500	575	rat575.tsp
200	144	pr144.tsp	800	654	p654.tsp
200	150	ch150.tsp	800	657	d657.tsp
200	150	kroA150.tsp	800	724	u724.tsp
200	150	kroB150.tsp	1500	783	rat783.tsp
200	152	pr152.tsp	1500	1002	pr1002.tsp
200	159	u159.tsp	1500	1060	u1060.tsp
200	195	rat195.tsp	1500	1084	vm1084.tsp

Table 2.
TSPLIB instances for GA testing.

2.2 Capacitated vehicle routing problem

The Capacitated Vehicle Routing Problem (CVRP) represents an extension on the TSP. As shown in **Figure 7**, the CVRP determines a set of routes that start and end at a specific place or location (e.g., a distribution center). These routes must visit or serve a finite number of locations and meet their demand requirements. Each route must be served by a single vehicle (e.g., a salesperson) with finite capacity, and only one vehicle can serve a location. Thus, the CVRP can be understood as a variant of the multiple-TSP with capacity restrictions [11].

As in the case of the TSP, the CVRP is a combinatorial problem of NP-hard complexity which cannot be solved within a reasonable polynomial time [12]. Due to this, the CVRP has been addressed by different meta-heuristics such as Tabu -Search (TS) [13, 14], GA [15], SA [16, 17], and Particle Swarm Optimization (PSO) [18].

Figure 5.
Mean error gap across all TSP test instances with (a) the GA, and (b) the revised hybrid-GA.

Figure 6.
Error gap across all TSP test instances with (a) the GA, and (b) the revised hybrid-GA.

Figure 7.
Example of a feasible CVRP solution with $n = 12$ cities and 3 routes.

For this case, the GA presented in **Figure 2** was modified to solve the CVRP. The GA and its configuration settings are presented in **Figure 8** and **Table 3** respectively. Note that the reproduction operators remain the same as considered for the TSP. Testing was performed with a set of instances from the CVRPLIB database [19, 20]. **Table 4** presents the details of the selected instances.

As presented in **Figure 9**, the mean error gap reaches the 10% mark by the 200th generation, with an approximate of 8.5% by the 1000th generation. In contrast to the patterns observed in **Figure 6**, in **Figure 10** there is not a clear relationship between the size of the instance (n) and the error gap. Thus, there are large instances with very small error gaps (approximately 6%) and medium instances with large error gaps (over 10%). This however is expected because there are more tasks to be performed on the CVRP such as route segmenting and capacity restriction compliance. This leads to frequently consider GAs for small CVRP instances ($n < 200$) [15, 21].

Figure 8.
Modified structure of the GA for the CVRP.

Parameter	Setting
Generations (Iterations)	1000
Fitness Function	Symmetric Euclidean Distance of CVRP Routes
Population Size	$N = 100$
Selection	Tournament
Reproduction Operators:	
Crossover	Partially-Matched Crossover (PMX)
Mutation	Swap, Inversion

Table 3.
GA settings for the CVRP.

Size of the CVRP (n)	Number of CVRP Routes	Name of the Instance	Size of the CVRP (n)	Number of CVRP Routes	Name of the Instance
100	25	X-n101-k25	335	84	X-n336-k84
105	14	X-n106-k14	343	43	X-n344-k43
109	13	X-n110-k13	350	40	X-n351-k40
114	10	X-n115-k10	358	29	X-n359-k29
119	6	X-n120-k6	366	17	X-n367-k17
124	30	X-n125-k30	375	94	X-n376-k94
128	18	X-n129-k18	383	52	X-n384-k52
133	13	X-n134-k13	392	38	X-n393-k38
138	10	X-n139-k10	400	29	X-n401-k29
142	7	X-n143-k7	410	19	X-n411-k19
147	46	X-n148-k46	419	130	X-n420-k130
152	22	X-n153-k22	428	61	X-n429-k61
156	13	X-n157-k13	438	37	X-n439-k37
161	11	X-n162-k11	448	29	X-n449-k29
166	10	X-n167-k10	458	26	X-n459-k26
171	51	X-n172-k51	468	138	X-n469-k138
175	26	X-n176-k26	479	70	X-n480-k70
180	23	X-n181-k23	490	59	X-n491-k59
185	15	X-n186-k15	501	39	X-n502-k39
189	8	X-n190-k8	512	21	X-n513-k21
194	51	X-n195-k51	523	137	X-n524-k153
199	36	X-n200-k36	535	96	X-n536-k96
203	19	X-n204-k19	547	50	X-n548-k50
208	16	X-n209-k16	560	42	X-n561-k42
213	11	X-n214-k11	572	30	X-n573-k30
218	73	X-n219-k73	585	159	X-n586-k159
222	34	X-n223-k34	598	92	X-n599-k92
227	23	X-n228-k23	612	62	X-n613-k62
232	16	X-n233-k16	626	43	X-n627-k43
236	14	X-n237-k14	640	35	X-n641-k35
241	48	X-n242-k48	654	131	X-n655-k131
246	47	X-n247-k50	669	126	X-n670-k130
250	28	X-n251-k28	684	75	X-n685-k75
255	16	X-n256-k16	700	44	X-n701-k44
260	13	X-n261-k13	715	35	X-n716-k35
265	58	X-n266-k58	732	159	X-n733-k159
269	35	X-n270-k35	748	98	X-n749-k98
274	28	X-n275-k28	765	71	X-n766-k71
279	17	X-n280-k17	782	48	X-n783-k48
283	15	X-n284-k15	800	40	X-n801-k40
288	60	X-n289-k60	818	171	X-n819-k171
293	50	X-n294-k50	836	142	X-n837-k142
297	31	X-n298-k31	855	95	X-n856-k95

Size of the CVRP (n)	Number of CVRP Routes	Name of the Instance	Size of the CVRP (n)	Number of CVRP Routes	Name of the Instance
302	21	X-n303-k21	875	59	X-n876-k59
307	13	X-n308-k13	894	37	X-n895-k37
312	71	X-n313-k71	915	207	X-n916-k207
316	53	X-n317-k53	935	151	X-n936-k151
321	28	X-n322-k28	956	87	X-n957-k87
326	20	X-n327-k20	978	58	X-n979-k58
330	15	X-n331-k15	1000	43	X-n1001-k43

Table 4.
CVRPLIB instances for GA testing.

Figure 9.
Mean error gap across all CVRP test instances with the GA.

Figure 10.
Error gap across all CVRP test instances with the GA.

Based on these findings, particularly for the CVRP with $n \approx 1000$, the following recommendations can be made:

- Due to the size of the population and the additional tasks, faster processes are needed for diversification of solutions. In example, Tabu Search (TS)

uses small sets of candidate solutions (neighbors) through the consideration of movements (or moves). Also, convergence to a local optimum can be minimized by forbidding certain moves (e.g., make them *tabu*) which would make the algorithm to revisit a region within the solution space. This is an advantage when compared to GA, which requires full-candidate solution populations, and avoidance of previously obtained solutions may require additional tasks.

- Hybridization can improve the convergence and overall search performance of near-optimal solutions. In example, implementing a *tabu* mechanism on the population can reduce the rate of previously visited solutions (same solutions) and even dynamically reduce the size of the population.
- Initial convergence of the GA, and overall initial performance, may benefit from an initial population with very suitable solutions. However, this may restrict the diversification of solutions through later generations.

3. Genetic algorithm for production and selection of goods

3.1 Economic lot quantity with multiple items

In manufacturing, an important aspect is the supply of resources such as raw materials, sub-assemblies, end/final products, etc. The availability of these resources must comply with time and cost restrictions.

Within this aspect, the Economic Lot Quantity (EOQ) models are aimed to estimate the lot size Q which minimizes operational costs associated to inventory management. In general, Q minimizes the following cost function:

$$T = \left(\frac{D}{Q} \right) C_o + \left(\frac{Q}{2} \right) C_h. \quad (1)$$

Where C_o is the ordering cost per lot, C_h is the holding cost per unit of product, and D is the cumulative demand through a planning horizon [22]. As presented in **Figure 11**, Q can also be understood as the lot size that equals the total order cost with the total holding cost through a planning horizon (and this leads to minimize T):

$$\left(\frac{D}{Q} \right) C_o = \left(\frac{Q}{2} \right) C_h. \quad (2)$$

Note that Eq. (2) leads to define:

$$Q = \sqrt{\frac{2DC_o}{C_h}}. \quad (3)$$

Which computes the optimal value for Q . Now, if N items with independent orders are considered, then:

$$T = \sum_{i=1}^N \left(\left(\frac{D_i}{Q_i} \right) C_{oi} + \left(\frac{Q_i}{2} \right) C_{hi} \right). \quad (4)$$

Figure 11.
Inventory management costs associated to the EOQ model.

Under the assumption of independence, Q_i can be optimally computed by using Eq. (3) for each item [22]. Thus, for the present case, the GA is only developed to verify its efficiency to solve the EOQ to optimality with a large N .

The GA follows the standard structure presented in **Figure 2**. As the solution consists of a set of Q_i values, the restrictions associated to permutations (such as in the case of TSP/CVRP) are not present. Thus, a simpler crossover operator can be used.

Figure 12 presents an overview of the linear crossover operator used for the GA. On the other hand, **Table 5** presents the configuration settings of the GA.

The average results for different randomly generated sets of N products are presented in **Figure 13**. As this is a simpler problem than both, the TSP and the CVRP, optimality can be reached within 100–200 generations. Note that it is always recommended to select an exact method if it is available and results can be obtained within very reasonable time.

Figure 12.
Linear crossover operator for the multiple-item EOQ ($\alpha = 0.5$).

Parameter	Setting
Generations (Iterations)	2000
Fitness Function	Total Inventory Management Cost (<i>T</i>)
Population Size	<i>N</i> = 1000
Selection	Tournament
Reproduction Operators:	
Crossover	Linear Crossover
Mutation	Swap, Inversion

Table 5.
GA settings for the multiple-item EOQ.

Figure 13.
Mean error gap across all multiple-item EOQ with the GA.

3.2 Knapsack problem

The Backpack or Knapsack Problem (KP) is a binary multicriteria problem of NP-hard computational complexity and it is frequently considered as a strategy to select items to maximize profits without affecting capacity restrictions [23, 24].

The KP can be mathematically formulated as a vector of binary variables x_j where $x_j = 1$ if the item j is selected, and $x_j = 0$ otherwise. Then, if p_j is a measure of importance (in this case, profit) for an item j , w_j represents the size of said item, and cv is the size of the backpack, the problem refers to the selection of the quantity of all elements whose binary vectors x_j satisfy the following restrictions [24]:

$$\sum_{j=1}^n w_j x_j \leq cv \tag{5}$$

$$x_j \in \{0,1\}, j = 1, \dots, n \tag{6}$$

Figure 14.
Uniform crossover operator for the KP.

Parameter	Setting
Generations (Iterations)	100
Fitness Function	Total Profit of Selected Goods (T)
Population Size	$N = 1000$
Selection	Tournament
Reproduction Operators:	
Crossover	Uniform Crossover
Mutation	Swap, Inversion

Table 6.
GA settings for the KP.

Figure 15.
Mean objective function values across all KP instances with the GA.

These must contribute to maximize the following objective function:

$$T = \sum_{j=1}^n p_j x_j \tag{7}$$

The KP also can be extended to consider more restrictions. In example, if cv is the volumetric capacity of the backpack, cz can be added to include its weight capacity. Thus, if w_j represents the volume of the item j , z_j can be used to represent its weight, leading to the following restriction:

$$\sum_{j=1}^n z_j x_j \leq cz \tag{8}$$

Figure 14 presents an overview of the reproduction operator for the GA considered to solve a large KP instance. Note that, due to the binary nature of the decision variable, the crossover and mutation operators can be implemented faster. Then, the configuration settings of the GA are reviewed in **Table 6**.

Based on the instance reported in [24], six random test instances with $N = 250$ items were generated. **Figure 15** presents the mean results for these instances. Error gap assessment was performed with the optimization software Lingo. This led to an error gap of 4.0% which is consistent with the results reported in [24].

4. Genetic algorithm for production scheduling problems

This chapter ends with an application of GA for solving one of the most useful models for manufacturing planning. This model, known as the Permutation Flow-Shop Scheduling Problem (PFSP), consists of finding the optimal sequence of N -jobs to be processed on M -machines [25]. The optimal sequence of jobs is the one that minimizes the make-span of the N -jobs through the M -machines, thus, minimizing the completion time of the last job on the last machine. Note that this sequencing implies two important restrictions: (a) no job can be started on the following machine until it is finished in the previous machine; and (b) a job cannot be started on a machine if it is busy processing another job. As consequence, this is one of the main strategies to reduce idle and waiting times within a workshop [26].

For illustration purposes, **Figure 16** shows an example of a solution for a 5-jobs (a, b, c, d, e) and 3-machines (1, 2, 3) PFSP. Note that each job may take different processing times depending of the assigned machine, and the established sequence remains the same for all machines. Thus, the established sequence has a direct effect on the completion time or *makespan*.

Thus, the information (i.e., processing times) of a PFSP with N -jobs and M -machines is frequently presented as shown in **Table 7**. As in the case of the TSP/CVRP models, the PFSP is also of NP-hard computational complexity, thus, meta-heuristic methods are frequently considered to solve it within reasonable time.

Figure 16.
Example of a 5-jobs, 3-machines PFSP.

Job	Processing Times				
	P_1	P_2	P_3	...	P_M
1	P_{11}	P_{12}	P_{13}	...	P_{1M}
2	P_{21}	P_{22}	P_{23}	...	P_{2M}
...	...	...	...	...	...
N	P_{N1}	P_{N2}	P_{N3}	...	P_{NM}

Table 7.
Data of the PFSP.

Figure 17.
Mean error gap across 30 randomly selected 20×20 PFLP test instances with the GA.

As it is a permutation-based problem, the structure and settings considered for the TSP GA (see **Figure 2** and **Table 1**) were considered for the PFSP with 500 generations. For testing purposes, the library and best results reported in [27] for 30 randomly selected 20-jobs, 20-machines PFSP instances were considered. The results are presented in **Figure 17**.

As observed, the mean error gap reaches the 10% mark at the beginning of the GA, with a final mean error gap of 0.005% by the 500th generation. Thus, the GA can provide near-optimal results for the PFSP.

5. Conclusions and future work

In this chapter the basic elements of a GA were reviewed to describe its application for different logistics and manufacturing problems. The routing problems, beyond the transportation context, can be applied on machine maintenance schemes or material changing services within production plants to minimize operational times. Also, they can be applied to improve the material flow through the warehouse, which is a main facility within the SC. Operations such as order-picking and bin-shelving can be optimized by modeling them as TSP instances [28].

On the other hand, the KP for selection of items is a problem shared with other contexts such as waste reduction in cutting processes, selection of investments and portfolios, decisions for capital budgeting and asset-backed securitization [29]. The

PFSP has been also extended on other fields such as in scheduling of quality control tasks on different machines [30].

Thus, the relevance of solving these combinatorial problems, particularly those of large scale, is very important due to their impact in other science and industrial fields.

Within the search algorithms, the GA can provide very suitable results for these problems. However, as presented in Sections X.2, X.3., and X.4, final performance depends of the type of problem. While the GA can achieve mean error gaps under the 10% mark for TSP/CVRP, for the PFSP the GA can achieve near optimal results under the 1% mark.

These results were supported by extensive experiments which were performed with well-known test databases or libraries. In practice, these experiments also provide important feedback to consider alternative meta-heuristics or develop hybrid approaches for improvement of performance.

This is because, as reviewed, a single meta-heuristic or search algorithm may not be enough to solve all problems if near-optimality is required. In this case, hybridization between different methods have improved on the search mechanisms of meta-heuristics, either deterministic or stochastic. Also, the integration with mathematical programming (which implies an exact solving method) has provided innovative proposals to solve NP-hard problems [31].

Future work is extensive on this field because:

- better solving methods are required due to the presence of increasingly complex combinatorial problems;
- advanced mathematical modeling is required to reduce the complexity of NP-hard problems and thus, make them more suitable to optimization through meta-heuristics or exact methods such as Branch & Bound;
- automatic decision models require the use of Big Data Analysis which, to some extend, depends of meta-heuristic methods.

Thus, as a concluding remark, it can be stated that any advance on these algorithms can impact on different fields. Just to mention an important field within the current industry, meta-heuristics are playing an important role on the implementation of dynamic decision models within Industry/Manufacturing 4.0 systems. Within this context, recent works have reported the application and improvement of these search algorithms for cost-efficient deployment of computing systems in logistics centers [32], dynamic CVRP [33], and development of Digital-Twin platforms [34].

Conflict of interest

The author declares no conflict of interest.

IntechOpen

Author details

Gladys Bonilla-Enriquez¹ and Santiago-Omar Caballero-Morales^{2*}

1 Technological Institute of Puebla, Puebla, Mexico

2 Autonomous People's University of the State of Puebla, Puebla, Mexico

*Address all correspondence to: santiagoomar.caballero@upaep.mx

IntechOpen

© 2021 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

References

- [1] Simchi-Levi D, Chen X, Bramel J. *The Logic of Logistics Theory, Algorithms, and Applications for Logistics Management*. 3rd ed. New Delhi, India: Springer Science+Business Media; 2014. 447 p. DOI: 10.1007/978-1-4614-9149-1
- [2] Sivanandam SN, Deepa SN. *Introduction to Genetic Algorithms*. 1st ed. Berlin: Springer; 2008. 442 p. DOI: 10.1007/978-3-540-73190-0
- [3] Singh-Juneja S, Saraswat P, Singh K, Sharma J, Majumdar R, Chowdhary S. TravellingSalesmanProblemOptimization Using Genetic Algorithm. In: *Proceedings of the Amity International Conference on Artificial Intelligence (AICAI 2019)*; 4-6 February 2019; Dubai, United Arab Emirates, United Arab Emirates: IEEE; 2019. p. 264-268
- [4] Crama Y, van de Klundert J, Spieksma FCR. Production planning problems in printed circuit board assembly. *Discrete Appl Math*. 2002;**123**:339-361
- [5] Bertsimas DJ, Simchi-Levi D. A new generation of vehicle routing research: robust algorithms, addressing uncertainty. *Oper Res*. 1996;**44**(2):286-304
- [6] Tsung-Sheng C, Yat-Wah W, Wei TO. A stochastic dynamic travelling salesman problem with hard time windows. *Eur J Oper Res*. 2009;**198**:749-759
- [7] Bharati TP, Kalshetty YR. A hybrid method to solve travelling salesman problem. *IJIRCCE*. 2016;**4**(8):15148-15152
- [8] Halim AH, Ismail I. Combinatorial Optimization: Comparison of Heuristic Algorithms in Travelling Salesman Problem. *Arch Computat Methods Eng*. 2019;**26**:367-380. DOI: 10.1007/s11831-017-9247-y
- [9] Reinelt G. TSPLIB 95 [Internet]. 1995. Available from: <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/> [Accessed: 2021-01-20]
- [10] Nguyen HD, Yoshihara I, Yamamori K, Yasunaga M. Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS*. 2007;**37**(1):92-99. DOI: 10.1109/TSMCB.2006.880136
- [11] Toth P, Vigol D. Models, relaxations and exact approaches for the capacitated vehicle routing problem. *Discrete Applied Mathematics*. 2002;**123**(1-3):487-512. DOI: 10.1016/S0166-218X(01)00351-1
- [12] Zhang DZ, Lee CKM. An Improved Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem. In: *Proceedings of the 2015 IEEE International Conference on Systems, Man, and Cybernetics*; 9-12 October 2015; Kowloon, China: IEEE; 2015. p. 2124-2128. DOI: 10.1109/SMC.2015.371
- [13] Kwon YJ, Kim JG, Seo J, Lee DH, Kim DS. A Tabu Search Algorithm using the Voronoi Diagram for the Capacitated Vehicle Routing Problem. In: *Proceedings of the 5th International Conference on Computational Science and Applications (ICCSA 2007)*; 26-29 August; Kuala Lumpur, Malaysia: IEEE; 2007. p. 480-485. DOI: 10.1109/ICCSA.2007.11
- [14] Jin J, Crainic TG, Lokketangen A. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. *European Journal of Operational Research*. 2012;**222**(3):441-451. DOI: 10.1016/j.ejor.2012.05.025
- [15] Nazif H, Lee L. Optimised crossover genetic algorithm for capacitated vehicle

routing problem. *Applied Mathematical Modelling*. 2012;**36**:2110-2117. DOI: 10.1016/j.apm.2011.08.010

[16] Mari F, Mahmudy WF, Santoso PB. An Improved Simulated Annealing for the Capacitated Vehicle Routing Problem (CVRP). *Jurnal Ilmiah Kursor*. 2018;**9**(3):119-128. DOI: 10.28961/kursor.v9i3.178

[17] Ilhan I. A population based simulated annealing algorithm for capacitated vehicle routing problem. *Turkish Journal of Electrical Engineering & Computer Sciences*. 2020;**28**:1217-1235

[18] Khaddar-Bakhayt AG, Al-Sattar HA, Abbas IT. Solving CVRP by Using Two-stage (DPSOTS) Algorithm. *Global Journal of Pure and Applied Mathematics*. 2017;**13**(6):1553-1567

[19] Uchoa E, Pecin D, Pessoa A, Poggi M, Vidal P, Subramanian A. New Benchmark Instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*. 2017;**257**(3):845-858. DOI: 10.1016/j.ejor.2016.08.012

[20] Uchoa E, Pecin D, Pessoa A, Poggi, M, Subramanian A, Vidal T. CVRPLIB [Internet]. 2014. Available from: <http://vrp.atd-lab.inf.puc-rio.br/index.php/en/> [Accessed: 2021-01-20]

[21] Baker BM, Ayeche MA. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*. 2003;**30**(5):787-800. DOI: 10.1016/S0305-0548(02)00051-5

[22] Sanchez-Sierra ST, Caballero-Morales SO, Sanchez-Partida D, Martinez-Flores JL. Facility Location Model with Inventory Transportation and Management Costs. *Acta Logistica*. 2018;**5**(3):79-86

[23] Martello S, Toth P. *Knapsack Problems: Algorithms and Computer Implementations*. 1st ed. England: John Wiley & Sons Ltd; 1990. 308 p

[24] Bonilla-Enriquez G, Sanchez-Partida D, Caballero-Morales SO. Algoritmo genetico para el problema logistico de asignacion de la mochila (Knapsack Problem). *Research in Computing Science*. 2017;**137**:157-168

[25] Singhal E, Singh S, Dayma A. An Improved Heuristic for Permutation Flow Shop Scheduling (NEH ALGORITHM). *International Journal of Computational Engineering Research*. 2012;**2**(6):30-36

[26] Rosas-Gonzalez A, Clemente-Guerrero DM, Caballero-Morales SO, Flores-Juan JC. Machines Permutation Flow-Shop Scheduling Problem with Break-Down Times. *International Journal of Computer Applications*. 2013;**83**(1):1-6. DOI: 10.5120/14409-2488

[27] Watson JP, Barbusescu L, Whitley DL, Howe AE. Contrasting structured and random permutation flow-shop scheduling problems: Search space topology and algorithm performance. *INFORMS Journal on Computing*. 2002;**14**(2):98-123

[28] Bartholdi JJ, Hackman ST. *Warehouse & Distribution Science*. Release 0.96. Atlanta, Georgia, USA: The Supply Chain and Logistics Institute; 2014. 323 p

[29] Kellerer H, Pferschy U, Pisinger D. *Knapsack Problems*. 1st ed. Germany: Springer-Verlag Berlin Heidelberg; 2004. 548 p. DOI: 10.1007/978-3-540-24777-7

[30] Erseven G, Akgün G, Karakaş A, Yarıkan G, Yücel Ö, Öner A. An Application of Permutation Flowshop Scheduling Problem in Quality Control Processes. In: *Proceedings of the International Symposium for Production Research (ISPR 2018)*; 28-31 August; Vienna, Austria: Springer; 2018. p. 849-860. DOI: 10.1007/978-3-319-92267-6_68

[31] Boschetti MA, Maniezzo V, Roffilli M, Röhlér AB. Matheuristics: Optimization, Simulation and Control. In: Proceedings of the International Workshop on Hybrid Metaheuristics (HM 2009); 16-18 January; Concepcion, Chile: Springer Verlag; 2009. p. 171-177. DOI: 10.1007/978-3-642-04918-7_13

[32] Li CC, Yang JW. Cost-Efficient Deployment of Fog Computing Systems at Logistics Centers in Industry 4.0. IEEE Transactions on Industrial Informatics. 2018;**14**(10):4603-4611. DOI: 10.1109/TII.2018.2827920

[33] Abdirad M, Krishnan K, Gupta D. A two-stage metaheuristic algorithm for the dynamic vehicle routing problem in Industry 4.0 approach. Journal of Management Analytics. 2020. DOI: 10.1080/23270012.2020.1811166

[34] Balderas D, Ortiz A, Mendez E, Ponce P, Molina A. Empowering Digital Twin for Industry 4.0 using metaheuristic optimization algorithms: case study PCB drilling optimization. The International Journal of Advanced Manufacturing Technology. 2021. DOI: 10.1007/s00170-021-06649-8