A Hybrid Clonal Selection Algorithm for the Vehicle Routing Problem with Stochastic Demands

Yannis Marinakis[1], Magdalene Marinaki[1] and Athanasios Migdalas[2]

[1] School of Production Engineering and Management, Technical University of Crete, 73100 Chania, Greece, marinakis@ergasya.tuc.gr,magda@dssl.tuc.gr
[2] Department of Civil Engineering, Aristotle University of Thessalonike, 54124 Thessalonike, Greece, samig@civil.auth.gr
and Industrial Logistics, Luleå Technical University, 97187 Luleå, Sweden, athmig@ltu.se

Abstract. The Clonal Selection Algorithm is the most known algorithm inspired from the Artificial Immune Systems and used effectively in optimization problems. In this paper, this nature inspired algorithm is used in a hybrid scheme with other metaheuristic algorithms for successfully solving the Vehicle Routing Problem with Stochastic Demands (VRPSD). More precisely, for the solution of this problem, the Hybrid Clonal Selection Algorithm (HCSA) is proposed which combines a Clonal Selection Algorithm (CSA), a Variable Neighborhood Search (VNS), and an Iterated Local Search (ILS) algorithm. The effectiveness of the original Clonal Selection Algorithm for this NP-hard problem is improved by using ILS as a hypermutation operator and VNS as a receptor editing operator. The algorithm is tested on a set of 40 benchmark instances from the literature and ten new best solutions are found. Comparisons of the proposed algorithm with several algorithms from the literature (two versions of the Particle Swarm Optimization algorithm, a Differential Evolution algorithm and a Genetic Algorithm) are also reported.

Keywords: Clonal Selection Algorithm, Variable Neighborhood Search, Iterated Local Search, Vehicle Routing Problem with Stochastic Demands.

1 Introduction

Artificial Immune Systems (AIS) [7, 9] are inspired by the workings of the natural immune system and take ideas from it in order to use them for constructing computational models to solve real-world problems. The natural immune system has a number of capabilities, including the ability of distinguishing between self and foreign/non-self, the ability of recognizing and destroying a number of pathogens, the ability of maintaining a memory of previous invaders and the ability of protecting the organism from misbehaving cells in the body [3]. Taking into account the distinct features of the natural immune system, the AIS algorithms are classified into three categories [3] which are the Positive/Negative Selection algorithm [14], the Clonal Expansion and Selection algorithm [10, 11] and the Network Algorithms [28]. For information about natural and artificial immune systems please see [3, 7–9, 12, 13, 27].

One of the main algorithms based on clonal selection theory is the CLONALG algorithm [10]. In [10], the clonal selection algorithm is used for a binary character recognition task, for a multi-modal optimization task and for solving a 30 cities instance of the Traveling Salesman Problem. The clonal selection algorithm has been used for many real world problems, for example in [33] an improved clonal selection algorithm is used for solving the traveling salesman problem, in [18] an adaptive clonal selection algorithm is used for the edge linking of images, in [32] a clonal selection based memetic algorithm is proposed for solving job shop scheduling problems, in [6, 20] a clonal selection algorithm is used for solving the Vehicle Routing Problem, in [29, 30] a clonal selection algorithm is used for the solution of facility layout problems, in [25] a clonal selection principle is used to solve the constrained economic load dispatch (ELD) problem, in [5] a clonal selection principle is used for the automatic detection of multiple circular shapes from complicated and noisy images with no consideration of the conventional Hough transform principles and in [15] a Baldwinian clonal selection algorithm is proposed to deal with optimization problems.

This paper presents a novel approach to solve the Vehicle Routing Problem with Stochastic Demands using a hybridized version of the Clonal Selection Algorithm (CSA) with the Variable Neighborhood Search (VNS) [17] and the Iterated Local Search (ILS) [19] algorithms. These are utilized within the CSA in the role of the receptor editing operator and the hypermutation operator respectively. The use of these two algorithms is intended to improve the exploration and the exploitation abilities of the hybrid CSA by employing two very powerful metaheuristics within an evolutionary framework provided by the CSA.

The Vehicle Routing Problem with Stochastic Demands (VRPSD) is a well known NP-hard problem. In this problem, the customers demands are known only when the vehicle arrives to them. This is in contrast to the Capacitated Vehicle Routing Problem (CVRP) where all the customer demands are known beforehand. In a VRPSD an a priori tour [1] is performed, i.e. a vehicle with finite capacity leaves from the depot with full load, visits each customer exactly once and returns to the depot. However, returns from nodes, which are stochastic points [26], to the depot are also included in the final route when the vehicle needs replenishment. Based on the demand of the next customer in the a priori tour, the vehicle can either proceed to the depot for restocking or it can go to the next customer. Many times although the expected demand of the customer is less than the load of the vehicle, a *preventive restocking* is chosen. This means that the vehicle returns to the depot for replenishment in order to avoid the risk of visiting the next customer without having sufficient load. Concerning the solution of the VRPSD, a number of algorithms have been proposed in the literature [1, 2, 4, 16, 21–23].

The rest of the paper is organized as follows: Section 2 presents detailed derivation of the proposed algorithm. Section 3 the computational results obtained are given and commented. Section 4 concludes the paper and discusses some future research directions.

2 Hybrid Clonal Selection Algorithm for the Vehicle Routing Problem with Stochastic Demands

In this section, the proposed algorithm for the solution of the VRPSD, the Hybrid Clonal Selection Algorithm (HCSA), is presented in detail. The Clonal Selection Algorithm (CSA) [11] is inspired by the clonal selection and affinity maturation process of B cells. B cells in the natural immune system are a type of lymphocytes, where lymphocytes are a type of leukocytes (white blood cells) that are responsible for identifying and killing pathogens [3] once the immune system has detected their presence, where pathogens are foreign bodies including viruses, bacteria, multi-cellular parasites, and fungi [3]. In a clonal selection algorithm, a large quantity of antibodies is created, where the antibodies in the natural immune system are glycoproteins (protein+carbohydrate) secreted into the blood in response to an antigenic stimulus that neutralise the antigen by binding specifically to it [3]. The antibodies will bind strongly to a specific antigen, where antigens are foreign molecules expressed by a pathogen that trigger an immune system response [3].

In the algorithm, an antibody corresponds to a solution while an antigen represents the optimization problem. The degree of binding between the antibody and the antigen (affinity) represents the objective function to be optimized. The objective is to start from an initial population of solutions (antibodies) and using the algorithm iteratively, to improve the quality of the solutions in the population. A variant of this algorithm is to be developed subsequently.

Initially, we have to choose the population (NP) of the antibodies. Each antibody is randomly placed in the n-dimensional space as a candidate solution (in the VRPSD n corresponds to the number of nodes). Every solution in the HCSA is mapped using the path representation of the tour, that is the specific sequence of the nodes. Afterwords, the fitness (expected length of the a priori tour) of each antibody is calculated using the following equations [1]:

$$f_j(q) = Minimum\{f_j^p(q), f_j^r(q)\}, \tag{1}$$

where

$$\begin{aligned} f_j^p(q) = d_{j,j+1} + \sum_{k \le q} f_{j+1}(q-k)p_{j+1,k} + \\ \sum_{k>q}[2d_{j+1,0} + f_{j+1}(q+Q-k)]p_{j+1,k}, \end{aligned} \tag{2}$$

and

$$f_j^r(q) = d_{j,0} + d_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)p_{j+1,k} \tag{3}$$

with boundary condition:

$$f_n(q) = d_{n,0}, q \in L_n, \tag{4}$$

where $G = (V, A, D)$ is a complete graph, $V = \{0, 1, ..., n\}$ is the set of nodes with node 0 being the depot, $A = \{(i, j) : i, j \in V, i \neq j\}$ is the set of arcs, $D = \{d_{ij} : i, j \in V, i \neq j\}$ is the travel distance between node i and j, Q is the vehicle's capacity and $\xi_i,\ i = 1, ..., n$ are the customer demands. The demand follows a discrete probability distribution $p_{ik} = Prob(\xi_i = k), k = 0, 1, 2, ..., K \leq Q$ and it does not exceed Q, $s = (0, 1, ..., n)$ is an a priori tour, q is the vehicle's remaining load after serving customer j, $f_j(q)$ is the expected cost from the customer j onward ($f_0(q)$ is the expected cost of the a priori tour), $f_j^p(q)$ is the expected cost of the route when the vehicle does not return to the depot but goes to the next customer and $f_j^r(q)$ is the expected cost when the vehicle returns to the depot for preventive restocking. Due to the random behavior of customer demands, a *route failure* may occur, i.e. the final demand of any route may exceed the actual vehicle capacity. In order to avoid a route failure, a threshold value may be chosen [31] such that if the residual load after servicing a customer is greater than or equal to this value, then the next customer is visited. However, if the residual load is lower than the threshold value, a return to the depot for preventive restocking is chosen.

From the initial population NP, the best F solutions are selected. The selected antibodies are cloned and mutated to construct new candidate population of antibodies. The F antibodies generate F_c clones proportional to their fitness function (affinities). The fittest antibodies create larger number of clones. The number of cloned antibodies is given from the following equation:

$$F_c = \sum_{i=1}^{F} round\frac{\beta F}{i}, \tag{5}$$

where β is a multiplying factor. Subsequently, the hypermutation operator is applied.

In this paper, two different hypermutation operators are used. The first one is the classic hypermutation operator (to be described below) while the second one is a metaheuristic, an Iterated Local Search algorithm (see section 2.2).

The first mutation operator selects the bits of the clone to be mutated randomly. Initially, a mutation operator number (Cr) which controls the fraction of bits to be mutated is selected. The value of Cr is compared to the value returned by a random number generator $rand_i(0, 1)$. If the random number is less or equal to Cr, then, the corresponding bit is to be mutated, otherwise the corresponding bit remains unchanged. Hence, the choice of the Cr-value is critical since if its chosen value is close to or equal to 1, then, most of the bits of the clone are mutated. However, if the chosen value is close to 0, then, almost none of the bits are mutated. In the classic CSA, the hypermutation rate of clones is inversely proportional to their fitness function (antigenic affinity). The higher is the affinity, the smaller is the mutation rate. In the proposed algorithm, in addition to the previous procedure, the clones produced from a common antibody have different hypermutation rates. This change is introduced in order to impose diversity of the clones.

Finally, a receptor editing step is applied. The receptor editing is used as a diversification phase of the algorithm. Two different receptor editing operators are used: One is the classic receptor editing operator (to be described in the following) while the other is metaheuristic algorithm based on the Variable Neighborhood Search (see section 2.1). Hence, if a clone gets stuck at a local optimum, the receptor editing phase of the algorithm introduces the possibility of escaping from the local optimum. When the receptor editing is called, then a search of unexplored places in the solution space is performed by reversing bits of the clone.

In contrast to the classic Clonal Selection Algorithm, the proposed algorithm applies either a hypermutation operator or a receptor editing operator to a clone. This is an attempt to generate the largest possible number of different clones. In order to decide whether a receptor editing operator or a hypermutation operator should be applied to a cloned antibody, a maturate operator number (Mr) is selected. The value of Mr is compared to the value of a random number generator $rand_i(0,1)$. If the random number is less or equal to the Mr for the corresponding clone, a hypermutation operator is applied, otherwise a receptor editing operator is applied.

The population of the clones is evaluated and the best among them (S) are selected in order to be added to the original population. Finally, new randomly generated solutions (R) are created that replace the worst antibodies of the population (NP) according to a specified percentage. In the next generation, the antibodies that belong to the subset NP survive and all the other clones or antibodies (subsets F, F_c, S and R) are removed. A pseudocode of the proposed algorithm is as follows:

```
**Algorithm HCSA**
Definition of parameters used in the main phase of the algorithm
    Definition of the maximum number of iterations
    Definition of the maximum number of antibodies ($NP$)
    Definition of numbers $R$ and $S$
    Definition of the number of $Cr$, $Mr$ and $\beta$
Initialization Phase
    Generation of the initial population ($NP$) of the antibodies
    Calculation of the fitness function of each antibody
    Sorting of the antibodies according to their fitness' functions
Main Phase
do while the maximum number of iterations has not been reached
```

```
    Selection of a subset $F$ of the solutions from $NP$
    For each member of $F$ create a set of clones $F_c$
    **for** each clone
        **if** $rand(0,1) \leq Mr$
            Call the hypermutation operator
        **else**
            Call the receptor editing operator
        **endif**
    **endfor**
    Calculation of the fitness function of each clone
    Sorting of the clones according to their fitness' functions
    Selection of a subset $S$ of the clones from $F_c$
    Generation of random solutions $R$
    Calculation of the fitness function of each random solution
    Replacement of the worst members of the $NP$ with better solutions
        from $S$ and $R$
    Survival of the $P$ subset into the next iteration
    Removal of all the other subsets ($F$, $F_c$, $S$ and $R$) from the population
**enddo**
**return** The best antibody (best solution found)
```

2.1 Variable Neighborhood Search

A Variable Neighborhood Search (VNS) [17] algorithm is applied in order to optimize the antibodies. The basic idea of the method is the successive search in a number of different neighborhoods of a solution. The search is applied either in a random or in a more systematic manner and aims at escaping from a local minimum in one neighborhood by switching to a another neighborhood. In this paper, the VNS algorithm is utilized in the following way. Initially, a number of local search algorithms based on different neighborhoods are listed. In the case of the Vehicle Routing Problem with Stochastic Demands, such algorithms are the 2-opt, 1-0 insert, 2-0 insert, 1-1 interchange, and 2-2 interchange.

In order to keep the complexity of the algorithm manageable, only one local search combination is applied to each antibody per iteration. Hence, a VNS operator number C_{VNS} determines which local search algorithm is selected. The value of C_{VNS} is compared to the output of a random number generator, $rand_i(0,1)$. Given the list of local search algorithms, if the random number is less than or equal to the C_{VNS}, then, the first algorithm is used. If the random number is less than or equal to the $2 * C_{VNS}$, then, the second algorithm is used, and so on. In this implementation, C_{VNS} is set equal to 0.1. Since a combination of local search algorithms is preferable to a simple local search, the list of selectable algorithms is enlarged with several combinations (2-opt and 1-1 interchange, 2-opt and 1-0 insert, 1-0 insert and 2-2 interchange, 2-0 insert and 1-1 interchange and, finally, 2-opt, 1-1 interchange, 1-0 insert, 2-2 interchange and 2-0 insert) which are added to the five simple local search algorithms listed previously.

2.2 Iterated Local Search

The purpose of the **Iterated Local Search (ILS)** algorithm is to improve previously produced local optima [19, 24, 27]. In this method, a perturbation is applied to each solution in order to produce new current solution. The perturbation can be thought as a large random move in the solution space.

There are a number of different ways to perform this perturbation. Thus, either a static perturbation can be applied where the length of the perturbation is fixed or a dynamic perturbation can be applied where the length of the perturbation is determined dynamically without using any memory of previous perturbations. Finally, an adaptive perturbation can be applied where the length of the perturbation is adapted during the iterations based on memory of previous good perturbations. The ILS can be thought of as an hypermutation operator as it changes every clone in a different way.

All three kinds of perturbations are applied in order to improve the exploration and exploitation abilities of the hybrid algorithm. In the static case, the length is fixed but the indices that it is applied upon vary. In the dynamic case, each clone is different as the length of the applied perturbation is not fixed. Finally, in the adaptive case, a number of good previous perturbations are used.

As an example, suppose that we have the following antibody (route):

1 2 3 4 5 6 7 8 9 10.

In order to produce 5 clones we can use the static version in which an initial perturbation length of, say, 5 is selected. Subsequently two indices, say, 4 and 10 are selected. The 5 nodes (customers) of the given route between the indices 4 and 10 are then perturbed. Thus, one possible new clone is then

1 2 3 4 9 7 8 5 6 10.

Yet another clone could be:

1 2 3 4 7 9 6 8 5 10.

If different indices are selected, for instance, the indices 2 and 8, then, a completely different clone is produced:

1 2 7 5 3 6 4 8 9 10.

In the dynamic version, not only the indices but also the perturbation length changes. Thus, if the length is 3 and the indices are 2 and 6, then, one possible clone is

1 2 5 3 4 6 7 8 9 10,

and yet another is

1 2 4 5 3 6 7 8 9 10.

3 Results and Discussion

Two different approaches are mainly used in the literature in order to deal with the route failure in the VRPSD. Both approaches have as a goal the minimization of the expected cost. In one approach [4, 16], vehicles follow their assigned routes until a route failure occurs. Then a replenishment is performed at the depot and subsequently the vehicle returns to the customer where the route failure occurred and resumes the servicing. In this approach, a set of vehicles can be used. The second approach uses a "preventive restocking strategy" [1, 21, 31]. In this approach, in order to avoid the route failure, a threshold value is used. If the residual load after servicing a customer is greater than or equal to this threshold value, then the vehicle proceeds to the next customer, otherwise it returns to the depot for replenishment. In this case, only one vehicle is used.

Considering these issues, the proposed hybrid is tested for the benchmark in [4, 16, 34]. Note, however, that the obtained results are not be directly comparable to those of [4, 16] due to different handling of route failure. We use the same transformation of the customers demands as the one proposed by [4, 16] and therefore we assume customer demands to be independent Poisson random variables with the mean demand for each customer equal to the deterministic value of the demand given in the corresponding VRP problem.

Table 1. Parameters for all algorithms

	PSO	CENTPSO	DE	GA	HCSA
particles/individuals/antibodies (NP)	80	80	200	150	100
iterations	3500	3500	3500	3500	3500
F	-	-	-	-	$0.25NP$
R	-	-	-	-	$0.1NP$
S	-	-	-	-	$0.2NP$
ls_{iter}	100	100	100	100	100
$c_{1,min} = c_{2,min}$	2	2	-	-	-
$c_{1,max} = c_{2,max}$	5	5	-	-	-
w_1	-	0.8	-	-	-
w_2	-	0.9	-	-	-
u_{bound}	4	4	-	-	-
l_{bound}	-4	-4	-	-	-
Cr	-	-	0.8	-	0.7
Mr	-	-	0.8	-	0.6
β	-	-	0.5	-	0.4
Probability of crossover	-	-	-	0.8	-
Probability of mutation	-	-	-	0.2	-

Four algorithms are compared to the proposed HCSA. All algorithms were implemented in modern Fortran and compiled with the Lahey f95 compiler. The chosen algorithm parameters are presented in Table 1. The first column of the table lists the names of the parameters. All algorithms share the same number of iterations and the same number of local search iterations. In Table 1, whenever the symbol $-$ appears in a column, the parameter is not used by the corresponding algorithm. The parameters for Particle Swarm Optimization (PSO), Combinatorial Expanding Neighborhood Topology Particle Swarm Optimization (CENTPSO), Differential Evolution (DE) and Genetic Algorithm (GA) are borrowed from the papers [21–23]. We have followed the similar to these methodology for the choice of parameter values for the HCSA. Thus, many different alternative values were tested and those finally selected gave the best results w.r.t. solution quality and the computational time needed in order to obtain it. Once the final parameter values were selected, 10 different runs were performed for each problem instance.

In Table 2, the results of the proposed hybrid algorithm for a set of forty benchmark instances [4, 16] are presented. The first column of Table 2 gives the names of the instances, which, in turn, include the number of nodes and the number of vehicles. For example, the instance name A-n32-k5 specifies 32 nodes and 5 vehicles. The number of nodes is in the range from 16 to 60. The second column lists the capacity of the vehicles in each problem instance. Column 3 lists the Best Known Solutions (BKS) from the literature. These solutions are based on the preventive restocking approach for the route failure. Column 4 lists the best cost obtained (S), column 5 the obtained solution quality (ω), column 6 gives the average (av) cost over the 10 runs, column 7 presents the standard deviation (stdev), column 8 the variance (var), column 9 gives the median cost (md) and, finally, column 10 gives the average CPU time (in seconds) for these runs. The solution quality is given in terms of the relative deviation from the best known solution (BKS). Hence for the HCSA we have:

$$\omega = \frac{(c_{HCSA} - c_{BKS})}{c_{BKS}}\% \qquad (6)$$

where c_{HCSA} denotes the cost of the solution found by the HCSA and c_{BKS} is the cost of the best known solution for the specific problem instance. As it can be seen, the proposed algorithm gives in 10 instances new best solutions for the Vehicle Routing Problem with Stochastic Demands. The improvement in the quality of the solutions to these instances is between 0.03% and 1.01%. The average deviation of the best known solution is 0.83%. In sixteen out of forty instances the quality of the solutions is less than 1%, in seven is between 1% to 2% and in seven instances is more than 2% with the worst quality equal to 4.07%. The algorithm in all runs gave very good results with small differences between them as the variance is in the range [0.11, 1.25], with the average variance being equal to 0.61, and as the standard deviation is in the range [0.33, 1.12], with the average standard deviation equal to 0.77. In seven instances, the average values over the ten runs are less than the corresponding best known solution which demonstrates the effectiveness of the proposed algorithm. For these instances, the quality has improved by 0.02% up to 0.73%. For the rest of the instances, the average quality over the ten runs is between 0.01% and 4.15%, with the average quality being between 0% and 1% for seventeen instances, between 1% and 2% for eight instances and more than 2% for eight instances.

As it would be interesting in addition to the results of the proposed algorithm (HCSA) to present results of how each one of the modifications of the

Table 2. Results of the proposed algorithm using the second approach of dealing with the issue of the route failure occurrence

Instance	Q	BKS	S	ω	av	stdev	var	md	CPU (sec)
A-n32-k5	100	820.44	822.93	0.30	823.80	0.69	0.48	823.82	215.8
A-n33-k5	100	684.2	689.44	0.77	690.38	0.90	0.81	690.16	238.2
A-n33-k6	100	762.39	763.07	0.09	763.94	0.85	0.71	763.63	241.5
A-n34-k5	100	788.7	798.39	1.23	799.29	0.84	0.70	799.19	242.4
A-n36-k5	100	826.21	822.71	-0.42	823.43	0.80	0.63	823.20	245.8
A-n37-k5	100	693.18	700.89	1.11	701.78	0.87	0.75	701.50	249.2
A-n37-k6	100	995.22	1007.3	1.21	1008.26	0.91	0.82	1008.07	248.1
A-n38-k5	100	752.2	752.98	0.10	753.88	1.00	1.00	753.59	251.4
A-n39-k5	100	853.002	859.53	0.77	860.54	0.67	0.45	860.71	263.5
A-n39-k6	100	845.25	850.17	0.58	851.04	0.70	0.48	850.97	259.8
A-n44-k6	100	977.0056	1016.8	4.07	1017.56	0.61	0.37	1017.57	264.5
A-n45-k6	100	988.12	1007.4	1.95	1008.37	0.72	0.53	1008.41	268.4
A-n45-k7	100	1175.45	1186.4	0.93	1187.31	0.83	0.69	1187.10	269.2
A-n46-k7	100	976.84	977.19	0.04	978.19	0.70	0.49	978.23	275.8
A-n48-k7	100	1132.15	1145	1.14	1145.99	0.78	0.60	1146.04	301.5
A-n53-k7	100	1094.2	1098.15	0.36	1099.07	0.96	0.93	1098.56	317.8
A-n54-k7	100	1217.2	1255.7	3.16	1256.39	0.51	0.26	1256.49	324.5
A-n55-k9	100	1118.4	1149.1	2.74	1150.10	0.80	0.64	1150.19	329.8
A-n60-k9	100	1436.5	1479.8	3.01	1480.59	0.78	0.60	1480.53	405.1
E-n22-k4	6000	378.56	376.02	-0.67	376.99	1.12	1.25	376.43	875.5
E-n33-k4	8000	847.38	848.95	0.19	849.41	0.33	0.11	849.43	918.7
E-n51-k5	160	544.86	549.25	0.81	550.13	0.91	0.84	549.67	457.2
P-n16-k8	35	441.98	437.51	-1.01	438.54	0.86	0.74	438.72	85.6
P-n19-k2	160	207.46	213.05	2.69	213.93	0.73	0.54	213.92	142.5
P-n20-k2	160	224.25	226.29	0.91	227.34	0.85	0.72	227.12	157.8
P-n21-k2	160	218.13	216.46	-0.77	217.37	0.74	0.55	217.20	162.5
P-n22-k2	160	223.06	225.67	1.17	226.60	0.76	0.58	226.45	165.8
P-n22-k8	3000	586.91	586.71	-0.03	587.47	0.89	0.80	587.10	358.2
P-n23-k8	40	532.82	532.12	-0.13	532.86	0.52	0.27	532.69	97.8
P-n40-k5	140	471.11	468.67	-0.52	469.59	0.87	0.75	469.41	314.2
P-n45-k5	150	527.9	527.27	-0.12	528.20	0.77	0.59	528.06	358.4
P-n50-k10	100	724.6	736.69	1.67	737.74	0.87	0.76	737.46	347.5
P-n50-k7	150	570.94	575.12	0.73	575.68	0.41	0.17	575.68	398.5
P-n50-k8	120	654.87	660.92	0.92	661.69	0.78	0.61	661.43	372.5
P-n51-k10	80	773.48	789.39	2.06	790.28	0.78	0.61	790.39	205.2
P-n55-k10	115	720.67	726.22	0.77	727.18	0.81	0.65	727.15	298.5
P-n55-k15	70	999.94	1002	0.21	1002.77	0.90	0.81	1002.51	194.2
P-n55-k7	170	587.95	587.14	-0.14	587.94	0.53	0.28	588.04	265.5
P-n60-k10	120	772.86	790.62	2.30	791.56	0.73	0.54	791.20	278.5
P-n60-k15	80	1012.9	1005.9	-0.69	1006.63	0.52	0.27	1006.62	215.5

initial version of the Clonal Selection Algorithm (CSA) affects the proposed algorithm, we present in Table 3 the results of the classic CSA, of a CSA with ILS and of a CSA with VNS. As it can be observed the results of the proposed

Table 3. Comparison of the proposed algorithm with three other versions of CSA

	BKS	CSA		CSA-ILS		CSA-VNS		HCSA	
		S	ω	S	ω	S	ω	S	ω
A-n32-k5	820.44	851.15	3.74	824.69	0.52	825.11	0.57	822.93	0.30
A-n33-k5	684.20	703.82	2.87	689.93	0.84	691.18	1.02	689.44	0.77
A-n33-k6	762.39	785.25	3.00	763.99	0.21	763.47	0.14	763.07	0.09
A-n34-k5	788.70	819.37	3.89	799.32	1.35	799.24	1.34	798.39	1.23
A-n36-k5	826.21	855.15	3.50	822.92	-0.40	822.89	-0.40	822.71	-0.42
A-n37-k5	693.18	707.35	2.04	701.91	1.26	703.16	1.44	700.89	1.11
A-n37-k6	995.22	1028.15	3.31	1008.12	1.30	1007.54	1.24	1007.30	1.21
A-n38-k5	752.20	772.15	2.65	753.12	0.12	755.39	0.42	752.98	0.10
A-n39-k5	853.00	868.25	1.79	860.06	0.83	861.04	0.94	859.53	0.77
A-n39-k6	845.25	869.35	2.85	851.99	0.80	850.90	0.67	850.17	0.58
A-n44-k6	977.01	1024.19	4.83	1017.60	4.15	1017.27	4.12	1016.80	4.07
A-n45-k6	988.12	1019.36	3.16	1007.45	1.96	1009.89	2.20	1007.40	1.95
A-n45-k7	1175.45	1259.25	7.13	1186.94	0.98	1186.67	0.95	1186.40	0.93
A-n46-k7	976.84	1001.19	2.49	977.28	0.04	977.76	0.09	977.19	0.04
A-n48-k7	1132.15	1185.14	4.68	1147.57	1.36	1146.03	1.23	1145.00	1.14
A-n53-k7	1094.20	1117.19	2.10	1098.70	0.41	1099.96	0.53	1098.15	0.36
A-n54-k7	1217.20	1284.36	5.52	1256.07	3.19	1256.88	3.26	1255.70	3.16
A-n55-k9	1118.40	1177.26	5.26	1149.30	2.76	1151.28	2.94	1149.10	2.74
A-n60-k9	1436.50	1527.43	6.33	1479.82	3.02	1480.71	3.08	1479.80	3.01
E-n22-k4	378.56	408.57	7.93	376.11	-0.65	378.50	-0.02	376.02	-0.67
E-n33-k4	847.38	850.15	0.33	849.67	0.27	849.18	0.21	848.95	0.19
E-n51-k5	544.86	551.15	1.15	549.40	0.83	549.58	0.87	549.25	0.81
P-n16-k8	441.98	511.75	15.78	439.25	-0.62	438.06	-0.89	437.51	-1.01
P-n19-k2	207.46	223.22	7.59	214.14	3.22	214.66	3.47	213.05	2.69
P-n20-k2	224.25	232.15	3.52	227.07	1.26	228.86	2.05	226.29	0.91
P-n21-k2	218.13	218.92	0.36	218.42	0.13	218.24	0.05	216.46	-0.77
P-n22-k2	223.06	230.26	3.23	226.01	1.32	227.90	2.17	225.67	1.17
P-n22-k8	586.91	675.15	15.04	587.09	0.03	589.54	0.45	586.71	-0.03
P-n23-k8	532.82	608.25	14.16	532.65	-0.03	533.79	0.18	532.12	-0.13
P-n40-k5	471.11	471.41	0.06	469.53	-0.34	469.76	-0.29	468.67	-0.52
P-n45-k5	527.90	532.15	0.81	527.32	-0.11	527.70	-0.04	527.27	-0.12
P-n50-k10	724.60	758.92	4.74	737.32	1.76	737.00	1.71	736.69	1.67
P-n50-k7	570.94	581.42	1.83	575.89	0.87	575.59	0.81	575.12	0.73
P-n50-k8	654.87	668.25	2.04	661.51	1.01	662.21	1.12	660.92	0.92
P-n51-k10	773.48	807.13	4.35	790.82	2.24	789.59	2.08	789.39	2.06
P-n55-k10	720.67	742.22	2.99	726.84	0.86	727.83	0.99	726.22	0.77
P-n55-k15	999.94	1065.35	6.54	1002.75	0.28	1002.93	0.30	1002.00	0.21
P-n55-k7	587.95	588.49	0.09	587.18	-0.13	588.41	0.08	587.14	-0.14
P-n60-k10	772.86	802.31	3.81	792.73	2.57	792.19	2.50	790.62	2.30
P-n60-k15	1012.90	1082.12	6.83	1007.44	-0.54	1006.30	-0.65	1005.90	-0.69

algorithm are better than the results of the other three implementations. The results of the initial version of the CSA are inferior than the results of all the other implementations. These results prove that the addition of each one of the modifications is very important for the effectiveness of the algorithm and each one helps the algorithm to improve its results. The next question that we have to answer is which of the two modifications helps the algorithm more. As it can be observed none of them outperforms clearly from the other. The CSA-ILS gives better results from the CSA-VNS in 24 instances (worst in 16 instances). The average quality of the CSA is 4.36%, of the CSA-ILS is 0.97%, of the CSA-VNS is 1.07% while the average quality of the proposed algorithm is 0.84%. Thus, the inclusion of the two modifications is necessary for the effectiveness of the proposed algorithm.

Table 4 presents a comparison of HCSA with four other evolutionary optimization algorithms: a constriction Particle Swarm Optimization (PSO) algorithm [21], a Differential Evolution (DE) algorithm [23], a Genetic Algorithm (GA) [23], and the Combinatorial Expanding Neighborhood Topology Particle Swarm Optimization (CENTPSO) [22]. The first column of the table, as before, gives the instance name. Columns two and three present the best cost over ten runs and the quality of the solutions produced by the PSO algorithm. Columns four and five give the corresponding values for the GA. Similarly, columns six and seven present the results obtained by the DE algorithm while columns eight and nine are dedicated to the CENTPSO algorithm. Finally, the last two columns show the results obtained by the new HCSA.

The new hybrid algorithm finds, as previously noticed, new best solutions in ten out of forty instances. The average quality of the obtained solutions is 0.18% for the CENTPSO, 0.83% for the HCSA, 0.83% for the PSO, 1.22% for the DE, and 2.02% for the GA. Compared to the CENTPSO the proposed algorithm finds better solutions in fourteen instances, compared to the PSO the proposed algorithm finds better solutions in twenty instances, compared to the DE the proposed algorithm finds better solutions in twenty six instances and, finally, compared to the GA the proposed algorithm finds better solutions in thirty three instances. The proposed algorithm outperforms both DE and GA on all instances. The performance of the new HCSA is similar to that PSO as both algorithms produce better than the best known solutions for twenty instances and both correspond to an average quality of 0.83%. The results of the HCSA are slightly inferior than those of the CENTPSO algorithm as the proposed hybrid finds 10 new best solutions but CENTPSO maintains 23 of the 30 best solutions it had previously obtained for the same instances.

4 Conclusions

In this paper, a new hybridized algorithm based on Clonal Selection Algorithm for the solution of the Vehicle Routing Problem with Stochastic Demands has been proposed. The purpose of this hybridization has been to improve the effectiveness of the original Clonal Selection Algorithm for this NP-hard problem. Hence, the Iterated Local Search algorithm was introduced as a hypermutation operator and the Variable Neighborhood Search algorithm as a receptor edit-

Table 4. Comparison of the proposed algorithm with other evolutionary algorithms from the literature

		PSO		GA		DE		CENTPSO		HCSA	
Instance	BKS	S	ω	S	ω	S	ω	S	ω	S	ω
A-n32-k5	820.44	821.65	0.15	836.07	1.91	820.5	0.01	820.44	0.00	822.93	0.30
A-n33-k5	684.2	687.04	0.42	693.4	1.34	684.2	0.00	687.04	0.42	689.44	0.77
A-n33-k6	762.39	769.62	0.95	762.4	0.00	762.6	0.03	762.39	0.00	763.07	0.09
A-n34-k5	788.7	789.88	0.15	812.3	2.99	788.7	0.00	789.88	0.15	798.39	1.23
A-n36-k5	826.21	836.05	1.19	833.3	0.86	835.1	1.08	826.21	0.00	822.71	-0.42
A-n37-k5	693.18	693.18	0.00	707.65	2.09	702	1.27	693.18	0.00	700.89	1.11
A-n37-k6	995.22	999.72	0.45	1018	2.29	1008.2	1.30	995.22	0.00	1007.3	1.21
A-n38-k5	752.2	756.56	0.58	755.5	0.44	752.2	0.00	755.2	0.40	752.98	0.10
A-n39-k5	853.002	853.08	0.01	858.7	0.67	862.6	1.13	853.002	0.00	859.53	0.77
A-n39-k6	845.25	847.92	0.32	867.12	2.59	845.7	0.05	845.25	0.00	850.17	0.58
A-n44-k6	977.0056	978.83	0.19	1005.9	2.96	980.6	0.37	977.0056	0.00	1016.8	4.07
A-n45-k6	988.12	997.41	0.94	1007.9	2.00	996.86	0.88	988.12	0.00	1007.4	1.95
A-n45-k7	1175.45	1175.45	0.00	1239.4	5.44	1213.1	3.20	1175.45	0.00	1186.4	0.93
A-n46-k7	976.84	984.98	0.83	976.84	0.00	979.7	0.29	978.23	0.14	977.19	0.04
A-n48-k7	1132.15	1132.15	0.00	1182.3	4.43	1146.7	1.29	1132.15	0.00	1145	1.14
A-n53-k7	1094.2	1096.6	0.22	1117.8	2.16	1100.2	0.55	1094.2	0.00	1098.15	0.36
A-n54-k7	1217.2	1223.23	0.50	1283.9	5.48	1279.5	5.12	1217.2	0.00	1255.7	3.16
A-n55-k9	1118.4	1124.3	0.53	1168.1	4.44	1150.9	2.91	1118.4	0.00	1149.1	2.74
A-n60-k9	1436.5	1454.15	1.23	1517.25	5.62	1483.2	3.25	1436.5	0.00	1479.8	3.01
E-n22-k4	378.56	390.99	3.28	385.12	1.73	379.16	0.16	378.56	0.00	376.02	-0.67
E-n33-k4	847.38	847.38	0.00	849.35	0.23	848.25	0.10	847.38	0.00	848.95	0.19
E-n51-k5	544.86	544.86	0.00	550.15	0.97	549.18	0.79	544.86	0.00	549.25	0.81
P-n16-k8	441.98	455.21	2.99	443.98	0.45	444.55	0.58	441.98	0.00	437.51	-1.01
P-n19-k2	207.46	213.51	2.92	216.66	4.43	215.04	3.65	207.46	0.00	213.05	2.69
P-n20-k2	224.25	226.79	1.13	225.89	0.73	224.25	0.00	226.13	0.84	226.29	0.91
P-n21-k2	218.13	218.13	0.00	218.38	0.11	218.52	0.18	218.13	0.00	216.46	-0.77
P-n22-k2	223.06	229.45	2.86	223.06	0.00	229.45	2.86	229.45	2.86	225.67	1.17
P-n22-k8	586.91	590.72	0.65	587.32	0.07	589.89	0.51	586.91	0.00	586.71	-0.03
P-n23-k8	532.82	536.34	0.66	536.07	0.61	545.26	2.33	532.82	0.00	532.12	-0.13
P-n40-k5	471.11	471.24	0.03	471.11	0.00	472.15	0.22	471.24	0.03	468.67	-0.52
P-n45-k5	527.9	530.52	0.50	531.29	0.64	527.9	0.00	529.16	0.24	527.27	-0.12
P-n50-k10	724.6	739.51	2.06	755.15	4.22	724.6	0.00	738.94	1.98	736.69	1.67
P-n50-k7	570.94	570.94	0.00	580.34	1.65	575.92	0.87	570.94	0.00	575.12	0.73
P-n50-k8	654.87	659.19	0.66	658	0.48	664.02	1.40	654.87	0.00	660.92	0.92
P-n51-k10	773.48	795.43	2.84	805.8	4.18	789.04	2.01	773.48	0.00	789.39	2.06
P-n55-k10	720.67	737.87	2.39	742.4	3.02	730.15	1.32	720.67	0.00	726.22	0.77
P-n55-k15	999.94	1008.6	0.87	1002.6	0.27	1016.4	1.65	999.94	0.00	1002	0.21
P-n55-k7	587.95	587.95	0.00	588.34	0.07	588.47	0.09	587.95	0.00	587.14	-0.14
P-n60-k10	772.86	772.86	0.00	803.18	3.92	790.55	2.29	772.86	0.00	790.62	2.30
P-n60-k15	1012.9	1021.58	0.86	1068.6	5.50	1067.6	5.40	1012.9	0.00	1005.9	-0.69

ing operator. The resulting hybrid algorithm was tested on a set of benchmark instances and gave new best solutions in ten out of forty instances with an average quality equal to 0.83%. The experimentation has also showed that the new hybrid is competitive with other evolutionary algorithms and that it may even outperform them w.r.t. solution quality. Indeed, the new hybrid found 10 new best solutions and showed better results on all benchmark instances than the classic versions of Differential Evolution and Genetic Algorithms. The new hybrid performs equally to the constriction PSO and it is only slightly inferior than a more involved version of PSO, the Combinatorial Expanding Neighborhood Topology Particle Swarm Optimization. Our future research will be focused on the application of this algorithm to other difficult routing problems.

References

1. Bianchi, L., Birattari, M., Manfrin, M., Mastrolilli, M., Paquete, L., Rossi-Doria, O., Schiavinotto, T. (2006). Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modeling and Algorithms*, 5(1), 91-110.
2. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2), 239-287.
3. Brabazon., A., ONeill, M. (2006). *Biologically Inspired Algorithms for Financial Modeling*, Natural Computing Series, Springer-Verlag, Berlin.
4. Christiansen, C.H., Lysgaard, J. (2007). A branch-and-price algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research Letters*, 35, 773-781.
5. Cuevas, E., Osuna-Enciso, V., Wario, F., Zaldívar, D., Pérez-Cisneros, M. (2012). Automatic multiple circle detection based on artificial immune systems. *Expert Systems with Applications*, 39, 713-722.
6. Dabrowski, J. (2008). Clonal Selection Algorithm for Vehicle Routing. *Proceedings of the 2008 1st International Conference on Information Technology, IT 2008* 19-21 May 2008, Gdansk, Poland.
7. Dasgupta, D. (Ed.) (1998). *Artificial Immune Systems and their Application*, Springer, Heidelberg.
8. Dasgupta, D., Niño, L.F. (2009). *Immunological Computation: Theory and Applications*, CRC Press, Taylor and Francis Group, USA.
9. De Castro, L.N., Timmis, J. (2002). *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, Heidelberg.
10. De Castro, L.N., Von Zuben, F.J. (2000). The clonal selection algorithm with engineering applications. *Workshop on Artificial Immune Systems and Their Applications (GECCO00)*, Las Vegas, NV, 3637.
11. De Castro, L.N, Von Zuben, F.J. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3), 239-251.
12. Engelbrecht, A.P. (2007). *Computational Intelligence: An Introduction*, Second Edition, John Wiley and Sons, England.
13. Flower, D., Timmis, J. (Eds.)(2007). *In Silico Immunology* , Springer, USA.
14. Forrest, S., Perelson, A. Allen, L., Cherukuri, R. (1994). Self-nonself discrimination in a computer, *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press: Los Alamitos, California, 1994, 202-212.

15. Gong, M., Jiao, L., Zhang, L. (2010). Baldwinian learning in clonal selection algorithm for optimization. *Information Sciences*, 180, 1218-1236.
16. Goodson, J.C., Ohlmann, J.W., Thomas, B.W. (2012). Cyclic-order neighborhoods with application to the vehicle routing problem with stochastic demand. *European Journal of Operational Research*, 217, 312-323.
17. Hansen, P., Mladenovic, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130, 449-467.
18. Li, F., Gao, S., Wang, W., Tang, Z. (2009). An Adaptive Clonal Selection Algorithm for Edge Linking Problem. *IJCSNS International Journal of Computer Science and Network Security*, 9(7), 57-65,
19. Lourenco, H. R., Martin, O., Stützle, T. (2002). Iterated Local Search. *Handbook of Metaheuristics. Vol. 57 of Operations Research and Management Science*, Kluwer Academic Publishers, 321-353.
20. Ma, J., Shi, G., Gao, L. (2009). An Improved immune clonal selection algorithm and its applications for VRP. *Proceedings of the IEEE International Conference on Automation and Logistics Shenyang*, August 2009, China.
21. Marinakis, Y., Iordanidou, G.R., Marinaki, M. (2013). Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands. *Applied Soft Computing*, 13, 1693-1704.
22. Marinakis, Y., Marinaki, M. (2013). Combinatorial Expanding Neighborhood Topology Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands. *GECCO: 2013, Genetic and Evolutionary Computation Conference*, 6-10 July 2013, Amsterdam, The Netherlands.
23. Marinakis, Y., Marinaki, M., Spanou, P. (2013). A Memetic differential evolution algorithm for vehicle routing problem with stochastic demands and customers, *(submitted)*.
24. Martin, O., Otto, S. W., Felten, E. W. (1991). Large-Step Markov Chains for the Traveling Salesman Problem. *Complex Systems*, 5(3), 299-326.
25. Panigrahi, B.K., Yadav, S.R., Agrawal, S., Tiwari, M.K. (2007). A clonal algorithm to solve economic load dispatch. *Electric Power Systems Research*, 77, 1381-1389.
26. Stewart, W.R., Golden, B.L. (1983). Stochastic vehicle routing: A comprehensive approach. *European Journal of Operational Research*, 14, 371-385.
27. Talbi, E.-G. (2009). *Metaheuristics : From Design to Implementation*, John Wiley and Sons, USA.
28. Timmis, J., Neal, M. (2000). A Resource Limited Artificial Immune System for Data Analysis. *Research and Development in Intelligent Systems*, Springer, Cambridge, UK, 14, 1932.
29. Ulutas, B.H., Islier, A.A. (2009). A clonal selection algorithm for dynamic facility layout problems *Journal of Manufacturing Systems*, 28, 123-131.
30. Ulutas, B.H., Kulturel-Konak, S. (2011). An artificial immune system based algorithm to solve unequal area facility layout problem. *Expert Systems with Applications*, doi:10.1016/j.eswa.2011.11.046.
31. Yang, W.H., Mathur, K., Ballou, R.H. (2000). Stochastic vehicle routing problem with restocking. *Transportation Science*, 34, 99-112.
32. Yang, J.-H., Sun, L., Lee, H.P., Qian, Y., Liang, Y.-C. (2008). Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering*, 5, 111-119.
33. Zhu, Y., Gao, S., Dai, H., Li, F., Tang, Z. (2007). Improved Clonal Algorithm and Its Application to Traveling Salesman Problem. *IJCSNS International Journal of Computer Science and Network Security*, 7(8), 109-113.
34. http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/Vrp-All.tgz