International Journal of Swarm Intelligence and Evolutionary Computation

Pavone et al., Int J Swarm Intel Evol Comput 2014, 4:1
DOI: 10.4172/2090-4908.1000118

Research Article Open Access

An Immunological Algorithm for Combinatorial Optimization: the Fuel Distribution Problem as Case Study

Pavone M*, Costanza J and Cutello V

Department of Mathematics and Computer Science, University of Catania, Catania, Italy

Abstract

Routing problems are classical combinatorial optimization tasks that find much applicability in numerous industrial and real-world scenarios. One challenging variant of the routing problem is the Fuel Distribution Problem (FDP) that a transportation company must face in its everyday operations. The main activity of a transportation fuel company is restocking all its stores, i.e. petrol stations, along a geographical map, with the goal to minimizing its overall costs. In this research work we present a hybrid heuristic based on the metaphor of the immune system for solving the FDP, which basically asks to find a set of routes as shorter as possible for a fixed number of company's vehicles in order to satisfy the several received demands of customers. In particular, the presented immunological algorithm takes inspiration by the clonal selection principle, whose key features are cloning, hyper- mutation, and aging operators. Such algorithm is also characterized, in having a (i) deterministic approach based on the Depth First Search (DFS) algorithm - used in the scheme of assigning a vertex to a vehicle - and (ii) a local search operator, based on the exploration of the neighborhood. The algorithm has been tested on one real data instance, with 82 vertices, and 25 others artificial different instances, taken from DIMACS graph coloring benchmark. The experimental results presented in this work, not only prove the robustness and efficiency of the developed algorithm, but show also the goodness of the local search, and the approach based on the DFS algorithm. Both methodologies help the algorithm to better explore the complex search space.

Keywords: Immunological algorithms; Clonal selection algorithms; Memetic immune algorithms; Hybrid evolutionary algorithms; Hybrid immune algorithms; Combinatorial optimization; Routing problems

Introduction

Artificial Immune systems (AIS) represent a branch of Computational Intelligence (CI), and have been successfully applied to a wide variety of application areas, such as for instance in combinatorial and global optimization [1-3], as well as in systems and synthetic biology [4-6]. AIS are bio- inspired algorithms that take their inspiration from the natural immune system and consist of a complex network of interactions among multiple types of agents, whose aim is to detect and contrast all together the antigens, i.e. the foreign organisms that can be cause of pathologies and disease, in order to organism. All AIS algorithms that mimic the clonal selection principle form a special class called Clonal Selection Algorithms (CSA), and today represent an effective mechanism for searching and optimization [7,8]. The core operators of CSA are (i) the cloning, which triggers the growth of a new population of high affinity value, (ii) the hypermutation, which is a search procedure that leads to a faster maturation during the learning phase, and (iii) the aging, whose main purpose is to introduce diversity into the population in order to escape from local optima during the evolutionary search process.

In this research work, we present an immunological algorithm based on the clonal selection prin- ciple designed to tackle a new combinatorial optimization problem that is the Fuel Distribution problem (FDP). FDP is a classical routing problem [9] that a generic fuel transport company faces every- day. Such problem is similar but not identical to the well-known Capacitated Vehicle Routing problem (CVRP) [9], and can be seen also as a variant of the classical Multiple Container Packing problem [10-13]. The differences between FDP and CVRP are simple but very crucial as they affect both the optimization strategy to use, and the design of the instances to tackle. Given a fixed number of vehicles, each having limited capacity of fuel transportable, the main goal of FDP is to satisfy with the minimal cost all requests made by customers (i.e. fuel stations) that need a fixed quantity of fuel.

It is well known in literature that pure EAs are unsuitable and inefficient when the problem to tackle shows a complex search spaces. However, their hybridization with other techniques might be greatly help for the algorithm. All those EAs that incorporate a deterministic approach, or a local search process in order to refine the solutions, improving then the fitness function, are called Hybrid Algorithms, or better Memetic Algorithms (MAs). The strength point of MAs is the trade-off between the abilities of exploitation by the deterministic or the Randomized Local Search used, and of exploration by the EAs [4,14-18].

In this research work a clonal selection algorithm is presented for solving the FDP, which incor- porates a deterministic approach for the assignment scheme based on the (DFS) algorithm, and a local search operator based on the exploration of the neighborhood. In order to evaluate the performances of the hybrid clonal selection algorithm, hereafter called H C SA, a real data instance with 82 vertices has been used. Furthermore, in order to understand the robustness and effectiveness of the proposed algorithm we extend the experiments also on 25 different instances, taken from DIMACS graph coloring benchmark, being one of the most popular and used in literature.

The paper is organized as follow: in Sect. §2 we give a general description of the tackled problem with its features and its constraints, presenting in subsection §2.2 a formal definition from a mathematical perspective. In subsection §2.1 we describe why and what are the core differences between FDP and CVRP. In Sect. §3 we present the

***Corresponding author:** Mario Pavone, Department of Mathematics and Computer Science, University of Catania, Catania, Italy, Tel: 0039 095 738 3038; Fax: 0039 095 330094 Email: mpavone@dmi.unict.it

Received April 09, 2015; **Accepted** June 16, 2015; **Published** June 21, 2015

Citation: Pavone M, Costanza J, Cutello V (2015) An Immunological Algorithm for Combinatorial Optimization: the Fuel Distribution Problem as Case Study. Int J Swarm Intel Evol Comput 4: 118. doi: 10.4172/2090-4908.1000118

developed H C SA with its details and main features, dedicating entirely also two sections to the description of the local search designed and the deterministic approach developed for assigning one vertex to one vehicle (Sect. §4 and Sect. §5, respectively). In Sect. §6 we present the study conducted to understand the effects of each parameter with respect to all others, and how it effects on the output, using the Morris method for sensitivity analysis. The experimental results performed and comparisons made are showed and presented in Sect. §7. Finally, Sect. §8 contain the concluding remarks and the future research perspectives.

Fuel Distribution Problem

The Fuel Distribution Problem (FDP) is a real-world task that any petrol company must face every day in its business activity. FDP is a classical routing problem whose aim is restock along a geographical map a set of customers (usually petrol stations, but not only), minimizing its overall costs. Basically, FDP asks to satisfy all received requests by customers with the minimal cost for the transport company using a fixed number of vehicles, each one having a limited quantity of fuel transportable, called capacity. How to schedule the fuel distribution depends on several questions, as for instance the amount of fuel required in the overall; number of vehicles available to satisfy the demands; relationships among the demands; capacity of each vehicle; road traffic; weather conditions, because the fuel is very sensitive to temperatures (a small amount of fuel evaporates with high temperatures); and many other conditions that affect both road network and amount of the transportable fuel.

Solving such problem means to optimize several objectives subject to some constraints, such as for instance maximize the number of satisfied customers, supplying the exact demand amount by each customer; maximize the number of customers served; minimize the distance from one customer to another; minimize the evaporation along the roads; minimize the costs in each path; and so on. It is important to point out that any feasible route is built taking into account some restrictions, such as (i) the fixed number of vehicles; (ii) any vehicle is able to transport only a limited fuel amount; and (iii) the sum of the amount of a subset of customers satisfied by one vehicle must not exceed the capacity of the vehicle itself.

FDP is then very similar to the Vehicle Routing Problem (VRP), and primarily with its variant better known as Capacitated Vehicle Routing Problem (CVRP) [9]. VRP represents the class of problems in which a set of routes for a fleet of vehicles must be determined for a number of geographically dispersed cities or customers. The objective of the VRP is to deliver a set of customers with known demands on minimum-cost vehicle routes originating and terminating at a depot. CVRP, instead, is the more studied member of the family, where the uniform capacity restrictions for the vehicles are imposed. Such class of problems is closely related to two difficult combinatorial problems: (1) checking whether there exists a feasible solution is an instance of the BPP; (2) setting the vehicles capacity as infinity gets an instance of the Multiple Traveling Salesman problem [19]. Thus, due to the interplay between these two NP-complete problems, and also that the decision version of BPP is conceptually equivalent to a VRP model, the VRP instances can be extremely difficult to be solved in practice [9,19]. Although FDP and CVRP may be seen as the same problem, is important to emphasize that they are not identical due to simple differences that affect crucially the optimization strategy to use.

Differences between FDP and CVRP

In this section we present some crucial differences between FDP and CVRP highlighting and explaining why they are similar but not identical. These differences are crucial, as they affect both the optimization strategy to use for solving the problem, and the design of the instances of the problem. All definitions described below were taken by [9].

(1) In CVRP the given graph $G = (V;E)$ taken in input must be strongly connected and it is generally assumed to be complete, where the cost travel cij to go from vertex i to vertex j, with $(i, j) \in E$, is defined as the Euclidean distance between the two nodes. In this way the costs matrix obtained is symmetric and satisfies the "triangle inequality":

$$cij\_cik + ckj,\ \forall i, j, k \in V.$$

Satisfying this triangle inequality leads the search process to choose direct links rather than other paths.

In FDP instead is not known a priori what the topology of the given graph is: in general the given graph is a map (e.g. a geographical map) and therefore a sparse graph. It may be a planar graph for small maps, whilst it becomes a more dense graph for bigger ones. Anyway, it never will be represented by a complete graph. Furthermore, since in each edge is associated more than one kind of cost to satisfy (e.g. traffic, distances, weather, travel time, size of the roads), the outcome is the one to not guarantee the property of triangle inequality (Figure 1).

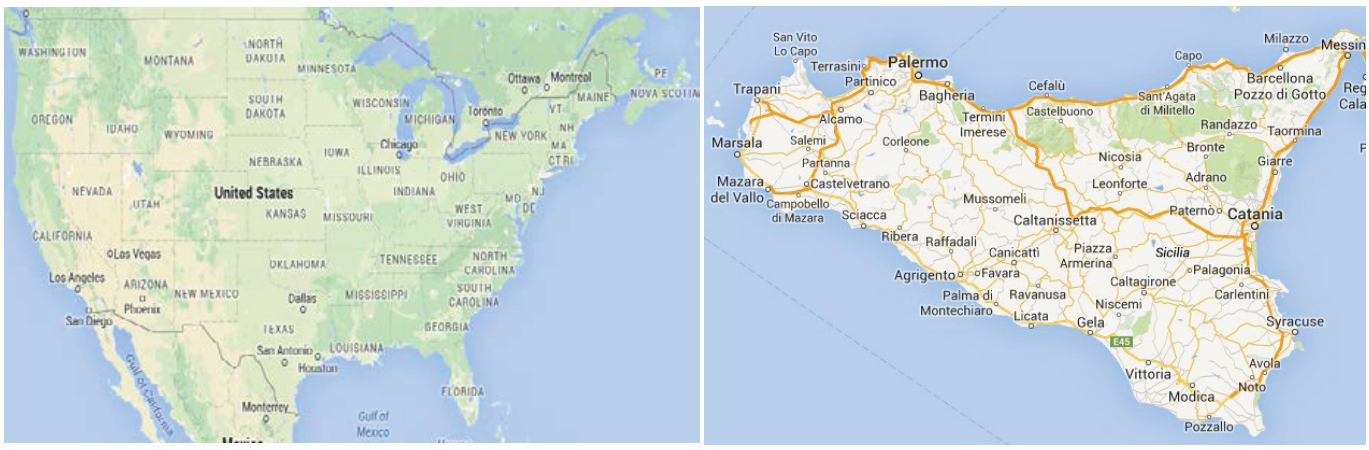

Figure 1: A geography map of United States of America, and Sicily (Italy) taken by Google maps.

(2) Because in FDP the instance is represented via a sparse graph, it becomes hard or almost impos- sible satisfy one constraint, which instead is one of important conditions for finding feasible solutions in CVRP: each customer vertex is visited by exactly one circuit. Of course, this condition is not true in FDP. Albeit each customer must be satisfied by only one vehicle, more vehicles may instead visit it. This happen because a vertex may be a link point among two or more customers, and thus a crossing among different paths. Besides, each vehicle can cross (then visit) more nodes in its road, not only because the instance isn't a complete graph - becoming almost mandatory to cross more times a given country (example figure. 1) - but also because of the several costs that affecting on the edges that might strongly affect the choice of the path to cross.

(3) Whilst in CVRP are taken into account only vehicles with same capacity, in FDP is not said. Indeed, it is possible to have vehicles with same capacity as well as with different capacities. Of course, this strongly depends by the organizational structure of a given fuel transport company.

These differences, although simple, become crucial for the design and develop of whatever algorithm, as they require different approaches and algorithmic strategies than instead the applied ones in CVRP.

Mathematical formalization as graph theoretic model

Fuel distribution problem can be formulated as follow: let be $G = (V, E)$ an undirected graph, where each node $v \in V$ represents a customer (i.e. fuel station), and the set E represents the road network, i.e. each edge $e = (u, v) \in E$ is the link between two customers' u and v. For the sake of simplification, we assume that $|V| = n$, *and* $|E| = m$.

In each vertex v is assigned a weight $q(v) \geq 0$ $(q : V \rightarrow \square)$ that indicates the fuel amount demanded by the customer v. Let T_R a constant that indicates the time needed to supply one litre of fuel in one station. At this end, the total time needed to satisfy the demand of the customer v is then given by $q(v) \times T_R$. As for the vertices, also in each edge $e \in E$ is assigned a weight $c(e) > 0$ $(c : E \rightarrow \square)$ that represents the costs on the road segment e. In this work we assume that $c(e)$ $(for\ all\ e \in E)$ includes all weights that affect the road network (see description above), i.e. the distances among stations (in km for instance); fuel's evaporation degree; temperature; traveling time for each road (hours or minutes), and many others.

Let $R = \{1, \ldots, h\}$ a set of vehicles, such that $|R| < |V|$, and let $b(r) > 0$ $(b : R \rightarrow \square)$, the limited amount of fuel transportable assigned to the vehicle $r \in R$, called capacity of r. Without loss of generality, we assume that the smallest demand is less than or equal to the smallest available capacity, as well as the largest demand is less than or equal to the largest capacity assigned

$$min_{v \in V}\{q(v)\} \leq min_{r \in R}\{b(r)\} \leq max_{v \in V}\{q(v)\} \leq max_{r \in R}\{b(r)\},$$

and besides, that the sum of all weights of vertices in V is greater than the largest capacity assigned to a vehicle in R

$$\sum_{v \in V} q(v) > max_{r \in R}\{b(r)\}.$$

The aim of FDP is to create h routes in such way to minimize the costs on each route, and maximize the satisfiability of the received demands. For simplicity then we can say that FDP asks basically to assign n vertices to h vehicles, such that (1) the overall number of the assigned vertices to vehicles must be as large as possible ($|V|$ is the optimum); and (2) the sum of the weights of the vertices assigned to each vehicle must not exceed the capacity of vehicle itself. It is worth noting as the constraint (1) means maximizing as much as possible the number of customer demands. Note that this last kind of definition is equivalent to partitioning V in h subsets, such that their union is all V, whilst their intersection is the empty set (θ).

Although the problem may be seen as a multi-objective problem, we have used instead the following single-objective function in order to evaluate each candidate solution $\vec{x} = \{x1, x2, \ldots, x_{|V|}\}$:

$$f\left(\vec{x}\right) = \frac{C_{tot}\left(\vec{x}\right)}{\sum_{r \in R}|Vr|} \times \left[1 + \left(|V| - \sum_{r \in R}|V_r|\right)^{\beta}\right]$$

(1) Where V_r is the set of all vertices assigned to the vehicle r; β is a penalty value, which assures us to give priority to those solutions able to satisfy all demands of customers; and C_{tot} is the total cost produced by the candidate solution $\vec{x}$, and it is given by

$$C_{tot}(\vec{X}) = \sum_{r \in R}\left(\sum_{x_i^r \in V_r}(q(x_i^r) \times T_R) + \sum_{e \in E_r} c(e)\right)$$

Where E_r is the subset of all edges visited by the vehicle r. Furthermore, since the candidate solution x represents a permutation of vertices from which is determined the visiting order in the graph, it follows that x_i^r is the vertex in the $i-th$ position in x assigned to the vehicle r.

HCSA – A Hybrid Clonal Selection Algorithm

Hybrid immunological algorithms are now considered a well-established technique, as these have been included/studies in [15]. The proposed hybrid immunological algorithm is based on clonal selection principle, whose main features are the operators of cloning, hyper mutation, and aging. *HCSA* is a modified and adapted version of the well-known *opt-IMMALG*, already proposed in [2,7,20].

As a typical population-based algorithm, *HCSA* works with a population of B cells, which has the purpose to defend the organism against foreign, better known as antigens (Ag). From an algorithmic perspective, an Ag is the problem to tackle, whilst the B cell represents a solution for the Ag. In particular, in our study, the Ag is an undirected graph, whose vertices are the customers[1] and edges are

1 i.e. fuel stations that have a contract with the distribution company, included the ones that haven't made demands the road connections between customers (see Sect. §2.2). Each B cell, instead, represents a permutation of vertices (i.e. a string of integers) that defines the visiting order of the customers for the assignment to a proper vehicle (if it exists) (Table 1).

In table 1 is presented the pseudocode of *HCSA*, where by $P^{(t)}$ is indicated a population of d individuals of length l = |V|. The initial population is randomly generated by a uniform distribution, which represents a subset of the search space, i.e. the starting points for the searching in the space of solutions (line 1, pseudocode). How to generate the initial population is a crucial task for EAs, as it might influence the overall performances. In traditional EAs, the initial population is generated using a random numbers distribution or chaotic sequences [21]. An interesting research work, which studied the properties of different point generators for a genetic algorithm, was presented in [22].

Miming the clonal selection principle, *HCSA* incorporates the *static cloning* operator, that clones all B cells *dup* times, producing an intermediate population $P^{(clo)}$ (line 5, pseudocode). In this work we haven't used a proportionally cloning, as it shows premature convergences. This is due to a strong pro- liferation of the best solutions found so far, which decrease the diversity in the population, and

```
HCSA(d, dup, τ_B, ρ, gen)
1.  P^(t=0) ← Init_Population(d)
2.  Evaluate_Fitness(P^(t=0))
3.  t ← 1
4.  While (¬Termination_Condition()) do
5.      P^(clo) ← Cloning (P^(t), dup)
6.      P^(hyp) ← Hypermutation(P^(clo), ρ)
7.      Evaluate_Fitness(P^(hyp));
8.      P^(LS) ← LocalSearch(P^(hyp)[best])
9.      Evaluate_Fitness P^(LS);
10.     Static_Aging (P^(t), P^(hyp), P^(LS), τ_B);
11.     P^(t+1) ← (μ + λ)-Selection P_a^(t), P_a^(hyp) and P_a^(LS);
12.     t ← t + 1;
13. end_while
```

Table 1: *Pseudo-code of Hybrid Clonal Selection Algorithm – HCSA.*

therefore not help H C SA to escape from local optima. Afterwards, each cloned B cell is undergo to the hyper- mutation operator, which mutates the cloned M times, without an explicit usage of mutation probability (line 6, pseudocode).

In *HCSA*, the number $M\left(f\left(x\right)\right)$ of mutations is determined in an inversely proportional way to the fitness value, although there exists several approaches as proposed in [23]. The *mutation rate* is determined by

$$\alpha = e^{\left(-\rho \times f\right)} \tag{3}$$

where f is the fitness function normalized in the range [0, 1], and ρ is a parameter. The Figure 2 shows the curves produced by the equation 3 at different ρ values (left plots), and the number of mutations obtained for different problem dimensions, and ρ values (right plot). For each B cell receptor, the hypermutation operator chooses randomly

$$M\left(f\left(\vec{x}\right)\right) = \lfloor(\alpha \times \ell)+1\rfloor \tag{4}$$

times two vertices u and v in $\vec{x}$, and then it swaps them having as effect to change the visiting order of the graph. At the end of the hypermutation process, we have a new population that is denoted by $P^{(hyp)}$. In order to normalize the fitness function in the range [0, 1], as proposed in [3,7], *HCSA* uses the best current fitness value decreased of an *user-defined threshold* Θ; this is due because is not known *a priori* any kind of information about global optima. In this way, *HCSA* doesn't need to have any kind of information concerning the problem; *HCSA* is a real *blind box*. In this work we have set Θ = 50%. One crucial question that might affect the performances of any immunological algorithm is what age assigns to each hypermutated clone. In order to give an enough evolutionary time for the maturation for each B cell, *HCSA* uses an *equal opportunity scheme*: when a hypermutated B cell improves the value of the fitness (called *constructive mutations*), then it will be considered to have age equal to 0; otherwise, it will maintain the same age of its parent. By this scheme we want to give an equal opportunity to each new B cell to effectively explore the search space. To refine the quality of the solutions, *HCSA* incorporates also an heuristic local search based on the exploration of the neighbourhood, where the neighbours are generated through the swapping of the vertices. This operator is described in section 4, and it is applied to the best receptor of $P^{(hyp)}$, producing a new population, called $P^{(LS)}$ (line 8, pseudocode).

After the perturbation operators, all the old B cells inside the populations $P^{(t)}$, $P^{(hyp)}$, and $P^{(LS)}$, are eliminated by the *static aging operator* (line 10, pseudocode). The parameter τ_B in input indicates the maximum number of generations, that allows to each B cell to remain into the corresponding population: when a B cell is $\tau_B + 1$ old it is erased from the own population, independently from its fitness value. This means that each B cell is allowed to remain into the population for a fixed number of generations. An exception is made only for the B cell with the best fitness value (*elitist static aging operator*). The aim of the aging operator is to produce a high diversity into the current population, and avoid premature convergences. Therefore, the aging operator plays a central role on the performances of the proposed algorithm (and AIS in general): too much diversity inside the population could produce poor solutions, as well as too small ones.

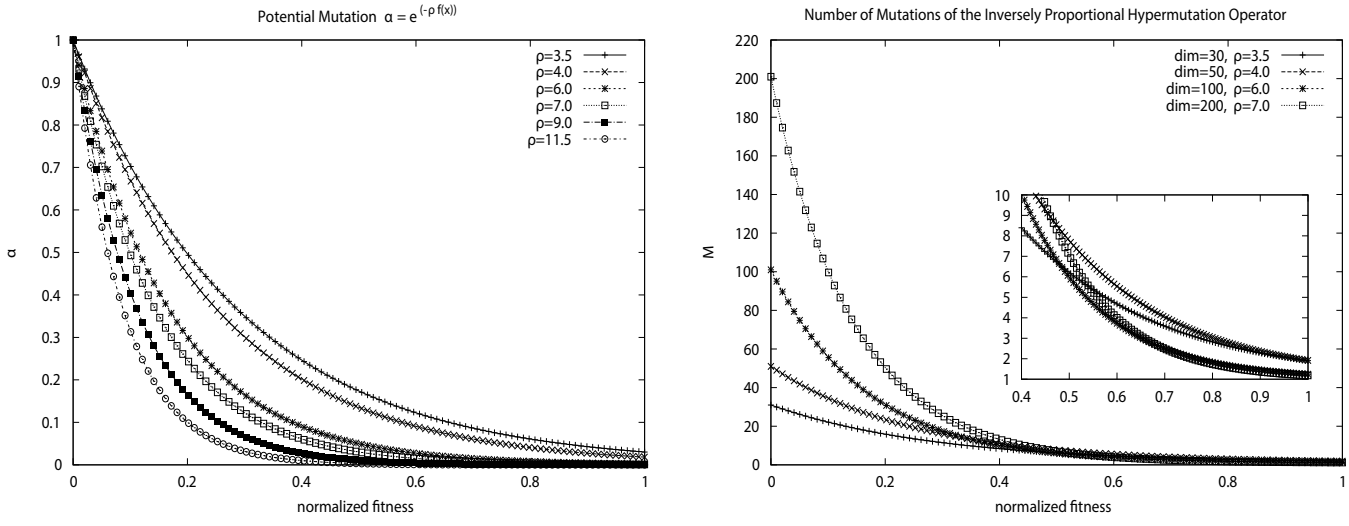

Figure 2: Potential mutation in Eq. 3 used by HCSA, and the number of mutations M (Eq. 4) obtained on several dimensions.

After the three main operators, the best survivors from the populations $P_a^{(t)}$, $P_a^{(hyp)}$ and $P_a^{(LS)}$ are selected for generate the new population $P_a^{(t+1)}$ for the next generation. Such selection happens using the $(\mu + \lambda)$-*Selection operator* (line 11, pseudocode). After the aging operator, follows the (μ + λ)-Selection operator, which generates the new population for the next generation; it selects the best *d* survivors B cell, from the populations Due the aging operator, it might occur that the survivors are less than the population size $(da < d)$; in this case the remaining $(d - da)$ new B cells will be randomly created. (Figure 3)

Finally, *Evaluate _Fitness*(P(*)) (lines 2, 7 and 9, pseudocode) computes the fitness function value of each B cell $\vec{x} \in P$ using Eq. 1, whilst *Termination Condition*() (line 4, pseudocode) is a Boolean function, which returns true if the maximum number of generations, or the maximum number of fitness function evaluations allowed, is reached; false otherwise. It's important to note that in order to minimize the costs of the routes, *HCSA* includes the *Dijkstra algorithm* [24], which computes for each vehicle the shortest path between two unconnected customers *u* and *v*, that is when $(u, v) \notin E$

Local Search

As cited in the previous section, afterwards the hypermutation operator, *HCSA* incorporates a heuristic local search in order to refine and improve the quality of the solutions. The used approach for the design of the local search was taken from [25,1], and it relies on the definition of neighbourhood, where neighbours are generated through the swapping of the vertices.

Starting from the idea that the visiting order of a graph shapes the solutions found, we define the neighbour of $\vec{x}$, all B cells $\vec{y}$ that can be obtained from $\vec{x}$ by swapping two of its elements. Because swapping all pairs of vertices is time consuming, we have used a reduced neighbourhood by a radius R_{LS}, as proposed in [1,2]: in each B cell, all vertices were swapped only with their R_{LS} nearest neighbours, to the left and to the right. Moreover, take into account the large size of the neighbourhood, we have applied the local search procedure only on the best hypermutated B cell (i.e. the best of $P^{(hyp)}$). When a single swap between two genes reduces the fitness function value (constructive mutation), then the new mutated B cell is added into the new population $P^{(LS)}$; otherwise it is not taken into account, and hence erased. The process continues until the whole neighbourhood

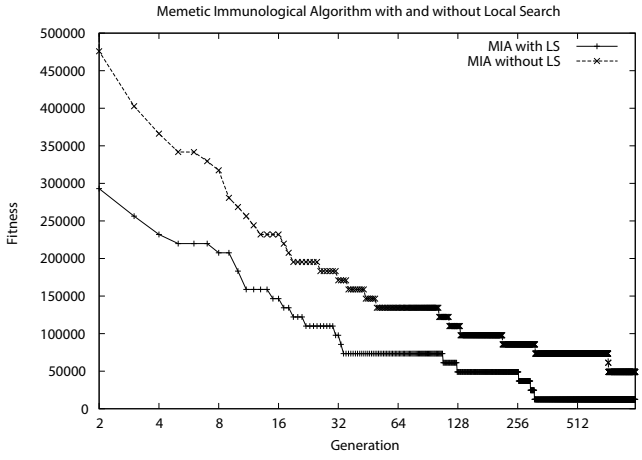

Figure 3: Best fitness curves of the hybrid clonal selection algorithm: using LS procedure versus no using LS.

with radius R_{LS} is explored. To avoid the problem to study what is the best tuning for the R_{LS} radius, *HCSA* randomly assigns a value in the range [1, (|*V*| − 1)], using a uniform distribution [25]. In this way it guarantees to swap at least two vertices.

Figure 3 shows the curves of the evolution of the best fitness using the local search procedure, or no. The experimental was made fixing the minimal values for the parameters: $d = 100$, $dup = 1$, $\tau_B = 5$, $\rho = 5.5$ and *M axGen* = 1000. From the figure is possible to see how the use of the local search helps the convergence process towards better solutions.

Heuristics for the Assignment Scheme

How to assign a vehicle to one customer or vice versa is a central point in the design of the algorithm. Because each vehicle has a maximum capacity of transportable fuel, then choosing one vertex, rather than another, can determine the satisfiability of all demands (the goal), or only some of them. To this purpose, in this work all B cells represent a vertices permutation, which determines the visit order. In details, the used heuristic scheme works as follows.

Let $\vec{x} = \{x_1, \ldots, x_n\}$ a generic B cell; $R = \{r_1, \ldots, r_h\}$ the set of the vehicles; and $b(r_i)$ the capacity of the *ith* vehicle, with $i \in [1, h]$. A vehicle r_i is randomly chosen to be assigned to the first x_1 vertex of the permutation $\vec{x}$, and afterwards decreasing the capacity of the vehicle chosen, i.e. $b_{(curr)}(r_i) = b(r_i) - q(x_1)$. For all *xj* remaining vertices, with $j = (2, \ldots, \ell = |V|)$, is possible to distinguish the following cases:

1. if exists a vertex $v \in V$ adjacent to x_j, and a vehicle $r \in R$ assigned to *v*, such that $\sum_{v \in V_r} q(v) + q(xj) \leq b(r)$, where V_r is the subset of the vertices already assigned to the vehicle *r*, then *r* is assigned to x_j. If there exist two or more vertices adjacent to x_j, with assigned different vehicles suitable to satisfy x_j, then one with higher available capacity is assigned;

2. for all vertices $v \in V$ adjacent to x_j, or not exist any $r \in R$ assigned to *v*, or if there is, it is not able to satisfy x_j. Thus, if there are one or more free vehicles, i.e. not still assigned, then one of these is randomly chosen, and assigned to $x_j$otherwise a deep search is made into the neighbourhood of x_j. If after the search, at least one vehicle was found, this is assigned to x_j, otherwise the vertex will be labelled "*not satisfied*". In this work, as search model, was used the classical "depth first search" algorithm (DFS) [24], but reduced of a radius $R_{(DFS)}$: is fixed a limit $R_{(DFS)} < |V|$ to the depth of the search into the neighbourhood. The purpose of this reduced DFS is to find a suitable vehicle not too far from x_j, in such way to have homogeneous groups. If, by the reduced dfs, we found two or more suitable vehicles, then the nearest one is assigned to the vertex x_j. For the experiments described in section 7, $R_{(DFS)}$ was fixed to 15% of *V*. Such value was chosen to avoid solutions where one vehicle must satisfy two stations too far away, for example geographically opposite.

3. not exists any vehicle able to satisfy the given vertex: i.e. for all $r \in R$ $\sum_{v \in V_r} q(v) + q(xj) \leq b(r)$, where V_r is the subset of the vertices assigned to the vehicle *r*. In this case the vertex will be labelled "*not satisfied*". Of course this occurs when the two previous steps fail.

After any assignment, the capacity of the chosen vehicle *r* is decreased: $b_{curr}(r) = b_{prev}(r) - q(x_j)$. With regard to the approach 2, if there exist more vehicles still unassigned, then one of these will be randomly chosen for the assignment, and this scheme is called "*random + dfs assignment*". Another variant may be taken into account, that we call "*dfs² assignment*": before randomly choosing a free vehicle, this variant checks the entire for a neighbourhood suitable vehicle that could be assigned; the reduced dfs, where the radius $R_{(DFS)}$ is fixed to 5% of *V*, is used. This new scheme, guarantees us the design of more

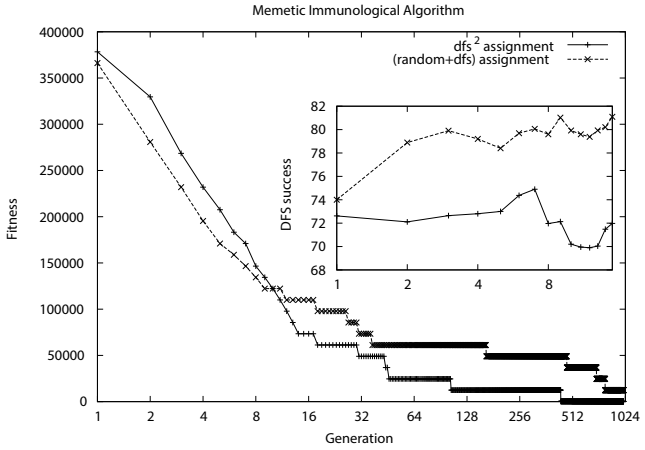

Figure 4: Curves of the best fitness for "dfs2 assignment" and "random + dfs assignment". The inset plot, instead, shows the relative curves of success of the DFS obtained.

dup	τ_B	*dfs²*	*random + dfs*	Δ
	5	**236:28** 2675:12 ± 4877:68	**12422:63** 19742:75 ± 8091:68	-12186:35
	20	**236:04** 1455:39 ± 3658:04	**12428:42** 20963:34 ± 10967:25	-12192:38
1	∞	**228:84** 1453:59 ± 3662:04	**12439:09** 20967:5 ± 5583:16	-12210:25
		235:37 7555:31 ± 5969:99	12443:66 17314:52 ± 5965:55	-12208:29
	5	**232:38** 232:93 ± 1:1	12425:92 17305:04 ± 8091:91	-12193:54
5	*20* ∞	**244:33** 13647:11 ±6563:78	12418:84 23396:64 ± 16768:86	-12174:51
		245:00 45:00 ± 0:0	**226:83** 24616:53 ± 21121:18	+18:17
	5	**225:19** 5106:2 ± 14638:93	27:32 227:32 ± 0:0	-2:13
10	20 ∞	**230:49** 1452:14 ± 3663:05	12427:99 18526:56 ± 6098:57	-12197:5

Table 2: *dfs²* versus *random + dfs*. For each experiment we show the best fitness, the mean and the standard deviation (σ). Last column Δ indicates the difference of the best fitness values. For these experiments was used the graph with $|V| = 82$. In bold face is highlighted the best fitness value for each pairs of parameter.

homogeneous groups, i.e. all vehicles satisfy only stations near to each other.

Figure 4 shows the comparisons of the curves of the best fitness obtained by the two described approaches. These curves were obtained using the following parameters: $d = 100$, $dup = 2$, $\tau B = 15$, $\rho = 5.5$, and $M\ axGen = 1000$. This figure shows how the basic idea designed in *dfs²* allows *HCSA* a better convergence towards solutions with best quality, and lower costs. The inset plot shows the curves of success of the DFS (best run) for the both approaches. Obviously, as we expected, the curves of *random + dfs are higher than dfs²*, because it develops solutions more poor, and then it needs more (Figure 4) calls to the long DFS, i.e. with radius $R_{(DFS)} = 15\%$ of $|V|$. Therefore, more calls to DFS, generates an higher number of DFS success. However, the overall percentages of the DFS success, averaged with their calls and computed on 10 independent runs, is higher in *dfs²* (73.08%) than in *random + dfs* (66.28%). In table 2, instead, is showed the comparison between the two cited approaches, varying the parameters, on the graph with 82 vertices, based on real data. This instance was used to evaluate the performances of *HCSA* not only with respect the fitness value, but also and primarily in the develop of homogeneous groups. In this table is shown the best fitness values, the mean and the standard deviation (Table 2).

(σ). Last column Δ of the table indicates the differences of the two approaches with respect the best fitness values. In bold face is highlighted the best fitness values for each pairs of parameter. The results were obtained with $d = 100$, $\rho = 2$, $M\ axGen = 100$, and 10 independent runs. The number of the vehicles h was fixed to 6, using the same capacity for all vehicles. Given the set of weights on the vertices, we can compute

$$\lambda = \frac{\sum_{i=1}^{n} q(v_i)}{h} \tag{5}$$

as an obvious lower bound (but not optimal) on the vehicle capacity needed to satisfy all required by the stations. In this work, we have fixed as capacity of the vehicles the lower bound λ increased by 0.2%; i.e. as small as possible. Inspecting such table is possible to see how *dfs²* is more suitable in finding better solutions, as well as more homogeneous. The better performances produced by *dfs²* are due because after the first generations the called number of the long DFS, i.e. the ones with radius $R_{(DFS)} = 15\%$ of $|V|$, decreases, having developed current subgroups homogeneous. Therefore, will be easier to find a suitable vehicle into the near-neighbourhood. Thanks to that, *dfs²* is more faster in computational time: the computational times are respectively *16m25919s* for *random + dfs*, and *8m17953s* for *dfs²*, obtained using the parameters $d = 1000$, $dup = 5$, $\tau_B = 5$, $\rho = 5.50$, and $gen = 200$.

For clarity we note that high values of mean and standard deviation (tables 2 and 3) are due to the penalty β value in the equation 1; when

dup	*τB*	*best*	*mean*	*σ*
		using same capacity		
1	5	229.45	231.42	1.28
	20	232.81	233.15	0.68
	∞	228.05	2670.81	4884.66
5	5	226.22	5116.51	9750.32
	20	235	5112	9754
	∞	244.64	11213.28	10125.65
10	5	229.15	2686.72	4874.28
	20	**220.92**	**6323.55**	**6102.63**
	∞	229.82	7547.7	9759.12
		using different capacity		
1	5	36830.49	53902.77	23894.93
	20	36845.67	36845.67	0.004
	∞	49017.31	57563.87	17294.56
5	5	36830.98	38051.81	3662.52
	20	36828.41	36835.69	4.29
	∞	**12436.16**	**14877.58**	**7314.36**
10	5	24639.02	40491	17291.04
	20	36825.49	41707.8	5979.59
	∞	24635.98	42929.53	23299.84

Table 3: Best solution, mean of the best solutions, and standard deviation (σ) obtained on the graph with 82 vertices, varying the parameters dup; and B For this class of experiments was fixed d = 100; MaxGen = 100; and each test was made 10 independently runs. Moreover, we have fixed = 4 for all vehicles with same capacity, and ρ = 5:5 with different capacity values. The shown results wereobtained fixing either the same capacity for all used vehicles (the lower bound _ increased by 0:2%),than with different capacity values.

one solution is not able to satisfy all customer demands then equation 1 return high values of the fitness function. Thus, high values of mean and standard deviation indicate us that the algorithm was not able to satisfy all demands in all independent runs.

Sensitivity Analysis

In sensitivity analysis methods, one can observe how a parameter affects the complexity of the instance, while all other parameters are varied simultaneously. These methods consider the interactions between parameters without depending on the stipulation of a nominal point.

The *The Morris method* [26] is a traditional model for sensitivity analysis used as a screening method for problems with high number of variables and for which function evaluations are CPU-time consuming. It is composed of individually randomized one-factor-at-a-time experiments. Each parameter may assume a discrete number of values, called levels, which are chosen within the factor range of variation.

The sensitivity measures proposed in the original work of Morris are based on what is called an elementary effect. The elementary effect of the j-th parameter is defined as

$$EE_j(p^*) = \frac{[f(p_1^*, \ldots, p_{j-1}^*, p_j^* + \Delta, p_{j+1}^*, \ldots, p_{N_p}^*) - f(p^*)]}{\Delta}$$

where Δ is a predetermined multiple of $1/(k-1)$. The distribution of elementary effects F_j is obtained by randomly sampling h points from Ψ.

For each parameter we evaluate two sensitivity measures, such as (1) μ_j an estimate of the mean of the distribution F_j, (2) and σ_j an estimate of the standard deviation of F_j. A high value of μ_j indicates a parameter with an important overall influence on the output; whilst a high value of σ_j indicates a parameter involved in interaction with other parameters or whose effect is nonlinear. In order to understand what are the parameters that affect the output, we performed the *Morris method* on a graph with 256 vertices, and 32640 edges (*queen16 16.col* – see section 9). The Morris method is a sensitivity analysis useful to understand the effects of a parameter with respect to all others, which vary simultaneously. Figure 5 shows the sensitivity analysis carried out for our objective function (equation 1). Inspecting this figure is possible to see how the vertices {36, 115, 139, 152, 172, 234} seem to be the most important, since they affect more on the objective function than the remaining vertices, whereas nodes 118, 101, and 182 are the less influential ones. (Figure 5)

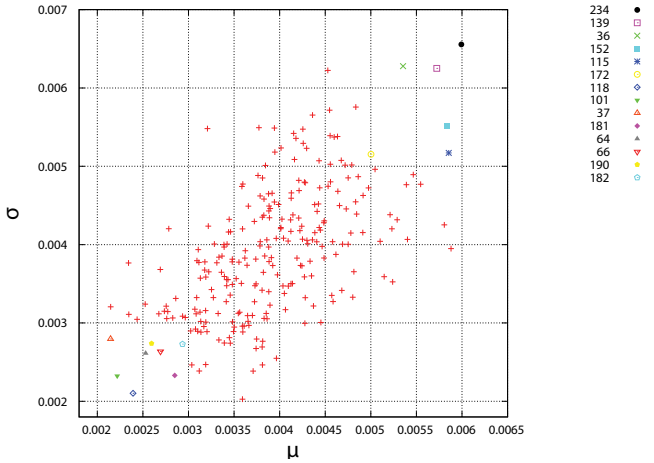

Figure 5: Normalized _ and _ of Sensitivity analysis using the Morris method. A high value of _indicates a parameter with an important overall influence on the output. A high value of _ indicates a parameter involved in interaction with other parameters or whose effect is nonlinear.

Results

To evaluate the goodness of the performance of *HCSA* and its search ability into the solutions space, we have used two different evaluation measures: (1) if *HCSA* is able to obtain good approximate solutions using the capacity of the vehicles as small as possible, and (2) the homogeneity in the assignment of the vehicles to the vertices; i.e., to avoid that a vehicle has to supply two vertices placed in opposite sites from a topological point of view. For the experiments, we have used initially a graph, based on real values, with 82 vertices. Afterward, to extend our test beds we have tested *HCSA* on several graphs, taken by the *Dimacs colouring benchmark* [27], being one of the most popular and used in literature. Therefore, in this section we report all results obtained in our experiments, describing also the experimental protocol used for each test. Of course, once experimentally proved that dfs^2 shows better performances than *random* + *dfs* with respect to the costs and the homogeneity of the solutions (see table 2), then, all results presented in this section were obtained by the dfs^2 heuristic. In table 3 we report the results obtained by *HCSA* using for all vehicles either the same quantity of the transportable fuel, and different capacity. The capacity was increased by 0.2% of the lower bound λ – Eq. 5. These experiments were made varying the parameters *dup* = {1, 5, 10}, and τ_B = {5, 20,..., ∞}, and fixing population size d = 100, ρ = 4 for the experiments where all vehicles have the same capacity, and ρ = 5.5 for all experiments with different capacity values. Moreover, the maximum number of generations was fixed to 100, and for each test was made 10 independent runs. If we give a look to the results obtained using the same capacity for all vehicles, one can see that, although the best solution is obtained with high values of *dup* (*dup* = 10), in the overall the better performances are instead obtained using smaller values; this is proved by the mean values. If we use different capacities for the vehicles, then *dup* = 5 seems instead to be the adequate setting. To simulate a real world application, we have considered the graph as road network, where each weight has been randomly generated in the range [200, 10000] (as litres) for the vertices, while for the edges in the range [5, 180] (as minutes). Moreover, we have used a small (Table 3) number of providers in order to better simulate a real application. Since in the real world is unlikely that all customers of a distribution company make demand at the same time, i.e. some node can have weight null, the random generator assigns each weight on vertices with a probability P = 50%.

Understanding the real capabilities by *HCSA* from an exploration and exploitation perspective, we have compared *HCSA* with a classical Genetic Algorithm (GA) and a deterministic algorithm based on locally optima choice strategy. For this deterministic algorithm we present three different versions: (1) starting from the vertex V1 the naive method proceeds sequentially (V_1, V_2, ..., V_n). We labelled this version as naive; (2) starting from a random vertex Vk , this method proceeds as follow (V_k , V_{k+1},..., V_n, V_1, ..., V_{k-1}). We call this second version as *DBO*; (3) the last method performs the optimal locally selection based on a permutation of the vertices (*DBP*). Table 4 presents the results obtained on this new benchmark. The table indicates the number of customers satisfied (Γ) for each algorithm, and relative best cost found. For these experiments, it has been used the following parameters d = 100, *dup* = 15, and τ_B = 15. For GA, instead, we have used the best tuning of parameters obtained after several experiments: *pop _size* = 100, P_C = 1.0, and P_m = 0.3. Furthermore, in all experiments we have used as

				HCSA		*naive*		*DBO*		*DP B*	*GA*	
Instance	$\|V\|$	$\|E\|$	*h*	Γ	*best*	Γ	*best*	Γ	*best*	Γ *best*	Γ	*best*
DSJC125.1.col	125	1472	4	125	**1018.06**	124	–	123	–	124 –	125	1056.5
DSJC125.5.col	125	7782	4	125	**977.92**	123	–	123	–	124 –	125	1010.92
DSJC125.9.col	125	13922	4	125	**978.532**	124	–	124	–	125 1167	125	1034.34
queen6 6.col	36	580	3	36	**1106.46**	35	–	35	–	35 –	36	1131.4
queen7 7.col	49	952	3	49	**1252.84**	48	–	48	–	49 1304.76	49	1274.53
queen8 8.col	64	1456	3	64	**1276.75**	63	–	63	–	64 1309.96	64	1298.4
queen8 12.col	96	2736	4	96	**1133.69**	94	–	95	–	96 1148.92	96	1146.2
queen9 9.col	81	2112	4	81	**1146.1**	80	–	80	–	80 –	81	1161.33
queen10 10.col	100	2940	4	100	**1157.4**	99	–	99	–	100 1198.01	100	1182.73
queen11 11.col	121	3960	4	121	**1082**	120	–	119	–	121 1127.98	121	1105.54
queen12 12.col	144	5192	5	144	**1252.47**	143	–	143	–	144 1296.76	144	1271.63
queen13 13.col	169	6656	5	169	**1282.47**	168	–	168	–	169 1313.1	169	1297.33
queen14 14.col	196	8372	5	196	**1283.31**	195	–	195	–	196 1307.6	196	1294.6
queen15 15.col	225	10360	6	225	**1103.09**	224	–	224	–	225 1126.5	225	1111.21
queen16 16.col	256	12640	6	256	**1326.89**	255	–	255	–	256 1345.1	256	1335
miles500.col	128	2340	4	128	**1141.23**	127	–	127	–	128 1168.028	128	16833
miles750.col	128	4226	4	128	**1110.67**	127	–	127	–	128 1150	128	1153.65
miles1000.col	128	6432	4	128	**1105.77**	126	–	127	–	128 1143.3	128	1137
miles1500.col	128	10396	4	128	**1096.96**	127	–	127	–	128 1130.4	128	1128.61
myciel5.col	47	236	3	47	**1246.87**	46	–	46	–	47 1280	47	1266.3
myciel6.col	95	755	4	95	**1140.14**	94	–	94	–	95 1147.7	95	1149.1
myciel7.col	191	2360	5	191	**1180.82**	190	–	190	–	191 1194.6	191	1193.7

Table 4: HCSA vs GA and three different versions of a deterministic algorithm. These experiments have been made using Dimacs graph colouring instances as test bed [1], being one of the most popularin literature. For each instance is showed the number of items satisfied (Γ), and relative best cost found. The experiments have been performed for 30 independent runs. We point out that if one of the algorithms is not able to satisfy all requested of the items, the relative costs have been not included in the table (_).

stop criterion a maximum number of fitness function evaluations $T_{max} = 5 \times 10^4$ for all graphs with $|V| < 100$, and $T_{max} = 5 \times 10^5$ otherwise. Besides, 30 independent runs have been performed (Table 4) for each instance. H C SA is able to satisfy all demands received, with respect deterministic algorithms. *HCSA* and GA have been able to satisfy all request received on different dimensions of the problem (from 36 to 256 vertices). However, comparing *HCSA* and GA is possible to see how the proposed algorithm is able to find better costs in all tested instances, which means that *HCSA* is able to produce more compact groups from a topological point of view. In the table 4, if one of the algorithms has not been able to satisfy all requested, the relative costs ($f(\vec{x})$) have been not included in the table and labeled as –, because the fitness value produced is high due to the penalty factor β (Eq. 1).

Conclusions and Future Work

In this research work we present an hybrid clonal selection algorithm (*HCSA*) for the *Fuel Distribution Problem*, one of the classical combinatorial optimization problem that many transportation companies must face in their everyday operations, that is to restock all their customers along a geographical map, minimizing their overall costs. Such problem is very similar but not identical to the *Capacitated Vehicle Routing Problem* – CVRP, and can be also seen as a variant of the classical *Multiple Container Packing Problem* – MCPP. We present in a proper section (Sect. §4.1) the simple differences between FDP and CVRP, which significantly effect on the optimization strategy to use, and on the design of the instances to tackle.

HCSA is based on three main operators: (i) the *cloning* that triggers the growth of a new population of high-value B cells; (ii) the *hypermutation* that can be seen as a search procedure that leads to a fast maturation; and (iii) the *aging* that causes a turn-over in the populations with the aim to generate diversity and avoid to get trapped into local optima. *HCSA* incorporates a local search heuristic in order to refine the solutions found so far, and it is based on the exploration of the neighbourhood, where the neighbours are generated through the swapping of the vertices. This heuristic is founded on the idea that the visiting order of vertices significantly affects the satisfiability or less of all demands made by customers. For proving the usefulness and improvements produced by designed local search, a study on the impact of such local search has been conducted. Such study has confirmed us how such heuristic helps *HCSA* in refine the quality of the solutions, and giving a best convergence, towards best solutions. In this research work, we propose also two variants for the assignment the customers to the most suitable vehicle, both based on the DFS algorithm. The first variant is a combination between random assignment, and DFS algorithm; whilst the second, before to choose a free vehicle for the assignment, the scheme performs a search in the neighbourhood of radius $R_{(DFS)}$. In order to limit the length of the search tree, we applied a reduced DFS based on a radius $R_{(DFS)}$; this reduction is to avoid solutions where one vehicle could be assigned to two vertices too far away. From the conducted experiments the reduced DFS seems develop good solutions, in term of quality, and homogeneity of assignments. It seems to be very promising for the future direction of our research on this problem.

To evaluate the performances of *HCSA*, we have used a graph with 82 vertices based on real data. However, to extend our experiments, and comparisons, we have also used some graphs with different topologies that were taken by the *Dimacs graph colouring benchmark*, being one of the most popular and used in literature. For this last class of experiments, we have randomly generated the weights on the vertices, and on the edges. All results were obtained using as capacity for the

vehicles the lower bound λ (Eq. 5) increased by 0.2%, i.e. as small as possible, to really understand the search ability of *HCSA* into the solutions space. Furthermore, *HCSA* has been compared with a Genetic Algorithm, and three different versions of a deterministic algorithm. Inspecting the results is possible to clearly see how *HCSA* outperforms the compared algorithms in all instances. *HCSA* is able to satisfy always all demands, as opposed to the compared algorithms, which instead fail in several instances.

Several new research directions need to be investigated: (1) study the best tuning of the radius $R_{(DFS)}$ for the reduced DFS; (2) design a better refinement operator, such to improve the convergence speed of *HCSA*, and its computational efforts; (3) and finally, perform and test HCSA taking into account a dynamical environment where the weights on the vertices and/or on the edges may change during the time, as well as vertices and /or edges turn on or turn off in the time.

Acknowledgements

The authors would like to thank all anonymous reviewers for their helpful and valuable suggestions that improve measurably the manuscript.

References

1. Cutello V, Nicosia G, Pavone M (2003) A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem In proc. of Genetic and Evolutionary Computation COnference (GECCO'03), LNCS, 2723: 171–182.
2. Cutello V,Nicosia G, Pavone M (2007). An immune algorithm with stochastic aging and kullback entropy for the chromatic number problem. Journal of Combinatorial Optimization 14(1):9–33.
3. Pavone M, Narzisi G,Nicosia G (2012). Clonal Selection - An Immunological Algorithm for Global Optimization over Continuous Spaces. Journal of Global Optimization, 53(4):769–808.
4. Hart W E, Krasnogor N, and Smith JE(2005). Recent Advances in Memetic Algorithms. In series in Studies in Fuzziness and Soft Computing, Springer, Berlin, Germany.
5. Cutello V, Nicosia G, Pavone M and Prizzi I. Protein Multiple Sequence Alignment by Hybrid Bio-Inspired Algorithms. Nucleic Acids Research, Oxford Journals, 39(6):1980–1992, 2011.
6. Cutello V, Nicosia G, Pavone M, Narzisi G(2006) Real Coded Clonal Selection Algorithm for Unconstrained Global Numerical Optimization using a Hybrid Inversely Proportional Hypermutation Operator. In proc. of 21st Annual ACM Symposium on Applied Computing (SAC'06), 2: 950–954,.
7. Cutello V, Krasnogor N, Nicosia G, Pavone M. Immune Algorithm versus Differential Evolution: A Comparative Case Study Using High Dimensional Function Optimization. in proc. of Interna- tional Conference on Adaptive and Natural Computing Algorithms (ICANNGA'07), LNCS, vol. 4431, pages 93–101.
8. Toth P, Vigo D (2014) Vehicle Routing: Problems, Methods, and Applications. MOS-SIAM Series on Optimization.
9. Raidl GR (1999). Genetic Algorithms for the Multiple Container Packing Problem. ACM SIGAPP Applied Computing Review, 782:2–31.
10. Raidl G(1999). A weight-coded genetic algorithm for the multiple container packing problem. In proc. of Annual ACM Symposium on Applied Computing (SAC'99), 291–296.
11. Fukunaga AS ,Korf AE (2007). Bin completion algorithms for multi container packing, knapsack, and covering problems. Journal of artificial intelligence research, 28:393–429
12. Soak SM., Lee S W, Yeo G T, Jeon MG (2008). An effective evolutionary algorithm for the multiple container packing problem. Progress in Natural Science, 18(3):337–344.
13. Krasnogor N, Smith JE(2005). A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Transactions on Evolutionary Computation, 9(5):474–488.
14. Krasnogor N.(2012) Memetic Algorithms. Book chapter on Handbook of Natural Computation, pp. 905–936.
15. Krasnogor N(2012)Memetic Algorithms. Book chapter on Handbook of Natural Computation, pp. 905–936.
16. Zhipeng L ,Hao J K. (2010) A Memetic Algorithm for Graph Coloring. European Journal of Operational Research, 203(1): 241–250.
17. Hao JK.(2012) Memetic Algorithms in Discrete Optimization. Book chapter on Handbook of Memetic Algorithms, Studies in Computational Intelligence, 379: 73–94.
18. Ayadi W, Hao JK (2014). A memetic algorithm for discovering negative correlation biclusters of DNA microarray data. Neurocomputing, 145:12–22.
19. Ralphs T K, Kopman L, Pulleyblank WR, Trotter(2003). On the Capacitated Vehicle Routing Problem. Mathematical Programming, 94: 343–359.
20. Cutello V, Nicosia G, Pavone M, Timmis J (2007). An Immune Algorithm for Protein Structure Pre- diction on Lattice Models. IEEE Transaction on Evolutionary Computation, 11(1):101–117.
21. Caponetto R, Fortuna L, Fazzino S, Xibilia MG(2003). Chaotic Sequences to Improve the Perfor- mance of Evolutionary Algorithms. IEEE Transaction on Evolutionary Computation, 7(3): 289–304.
22. Maaranen H, Miettinene K, Penttinen A (2007) On Initial Populations of a Generic Algorithm for Continuous Optimization Problems. Journal of Global Optimization, 37(3):405–436.
23. Cutello V, Nicosia G, Pavone M (2004). Exploring the capability of immune algorithms: a charac- terization of hypermutation operators. In proc. of 3rd international conference on artificial immune systems (ICARIS'04), LNCS, 3239:263–276, 2004.
24. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. MIT Press, 2001.
25. Cutello V, Nicosia G ,Pavone M(2004). An Immune Algorithm with Hyper-Macromutations for the Dill's 2D Hydrophobic - Hydrophilic Model. In proc. of Congress on Evolutionary Computation (CEC'04), IEEE Press,1 :1074–1080.
26. Morris MD(2001), Factorial sampling plans for preliminary computational experiments. Technometrics, 33(2):161–174.
27. Graph Coloring Instances. In http://mat.gsia.cmu.edu/COLOR/instances.html

Citation: Pavone M, Costanza J, Cutello V (2015) An Immunological Algorithm for Combinatorial Optimization: the Fuel Distribution Problem as Case Study. Int J Swarm Intel Evol Comput 4: 118. doi: 10.4172/2090-4908.1000118