Viral Systems: a new bio-inspired optimisation approach

Pablo Cortés*, José M. García, Jesús Muñuzuri and Luis Onieva

Seville University

Ingeniería Organización.

Escuela Superior Ingenieros, Camino de los Descubrimientos s/n.

Sevilla 41092. SPAIN

Abstract.- The paper presents a new approach to deal with combinatorial problems. It makes use of a biological analogy inspired by the performance of viruses. The replication mechanism, as well as the hosts' infection processes is used to generate a metaheuristic that allows the obtention of valuable results. The Viral System (VS) theoretical context is described and it is applied to a library of medium-to-large-sized cases of the Steiner problem for which the optimal solution is known. The method is compared with the metaheuristics that have provided the best results for the Steiner problem. The Viral System provides better solutions than genetic algorithms and certain tabu search approaches. For the most sophisticated tabu search approaches (the best metaheuristic approximations to the Steiner problem solution) VS provides solutions of similar quality.

Keywords: virus, optimisation, metaheuristic, artificial intelligence, Steiner tree

1 Introduction

In real life, most problems are combinatorial. For this type of problem, the available algorithms usually present weaknesses. The mathematically rigorous algorithms (as branching or cutting techniques) tend to be too slow while ad hoc heuristics often produce poor solutions. Since the last decades of the past century, metaheuristics and Artificial Intelligence (AI) have been trying to deal with these difficulties. This is the case of Simulated Annealing (see Kirkpatrick [1] and Rumelhart and McClelland [2] for the first references), Genetic Algorithms (firstly described by Holland [3], Holland et al. [4] and Goldberg [5]), Tabu Search (defined by Glover [6] and [7]) or more recently the use of Agent Technology (see Van Dyke [8], and Jennings et al. [9] for a survey including wide bibliography in the field). The use of agents to deal with optimisation problems is a new and incipient framework. The Network of Excellence "AgentLink" funded by the European Commission under its Fifth Framework Information Society Technologies programme has edited a report describing the current state-

* Corresponding author. Fax: +34 95 448 73 29. E-mail: pca@esi.us.es

of-the-art of agent technologies and identifying trends and challenges that will need to be addressed over the next 10 years to progress in the field and realise the benefits (Luck et al. [10]).

Recently, Artificial Immune Systems (AIS) have arisen. They are computer algorithms inspired by the principles and processes of the vertebrates immune system. The algorithms typically exploit the immune system's characteristics of learning and memory to solve a problem. They are coupled to artificial intelligence and closely related to genetic algorithms. Processes simulated in AIS include pattern recognition, hypermutation and clonal selection for B-cells, negative selection of T-cells, affinity maturation and immune network theory. AIS began in the eighties with Farmer et al. [11].

In this paper, we propose the use of a biological analogy based on viral infection. Not much research has been carried out using viral analogies. We have been able to find very few papers proposing the use of viruses , all of them as part of genetic algorithms. For instance, Kubota et al. [12] propose them as part of a specific operator in genetic algorithms, and Saito [13] has described the use of genetic algorithms which make use of a virus evolutionary theory (GAV), and an algorithm based on the conception of horizontal evolution caused by virus infections. GAV is carried out by attacking a chromosome by a number of viruses, and having the genes of the chromosome recombined by the attack. The infection is allowed when the evaluation value goes up, but it falls into local minima easily. In order to escape from these local minima, an infection which makes the evaluation value worse in a small rate under small probability is allowed as well. All these approaches do not fit with our definition of Viral System as a new metaheuristic.

Here, we introduce a new optimisation approach that makes use of several ideas from multi-agent systems (MAS), as well as from other well-known AI approaches. In our proposal, we consider that the viruses are part of a general infection, where each virus tries to behave to its own benefit, but resulting in the benefit of the Viral System.

A generic optimisation problem can be defined as (1), and its complexity is typically NP-complete (Garey and Johnson [14]).

$$\text{Min}\left\{ f(x) : g_i(x) \le 0,\ \forall i = 1, \cdots, n \right\} \qquad (1)$$

Where x is the set of feasible solutions of the problem, $f(x)$ is the objective function, and $g_i(x) \le 0$ are the problem constraints. The objective function and the constraints are not necessarily linear, and the variables could be integer or continuous.

Based on this problem, the next section of the paper follows with the description of the biological analogy that we will use to solve the problem in section 2, the description of the approach in section 3, and the results for a collection of trial problems in section 4. Finally, we review the main aspects in the conclusion section.

2 Biological analogy description

2.1 Generic definition of virus

Viruses are intracellular parasites shaped by nucleic acids, such as DNA or RNA, and proteins. The protein generates a capsule, called a capsid, where the nucleic acid is located. The capsid plus the nucleic acid shape the nucleus-capsid, which defines the virus. Figure 1 depicts different types of viruses.

FIGURE 1

Each type of virus has different capabilities with respect to infecting cells, including the mechanisms of replication and spread/inoculation, and strategies to weaken the immune response of the host. Furthermore, each one of the different classes of viruses is able to be lodged in hosts that can be of animal, vegetal or bacterial type. Hence, they can represent unicellular or multi-cellular organisms.

2.2 Virus replication mechanism

Although the replication mechanism of viruses depends on the type of virus, we have considered the replication mechanism for phages, which are viruses that infect bacteria. The phage life cycle (see figure 2) follows the following process:

1. The virus is adhered to the border of the bacterium. Subsequently the virus manages to penetrate the border being injected inside this one, (a) and (b) in figure 2.
2. The infected cell stops the production of its proteins, beginning to produce the phage proteins. So, it starts to replicate copies of the virus nucleus-capsids, (c) and (d) in figure 2.
3. After replicating a number of nucleus-capsids, the bacterium border is broken, and the new viruses are released, (e) in figure 2, which can infect other close cells, (f) in figure 2.

FIGURE 2

The life cycle of the viruses can be developed in one step, through this lytic replication, or it can include more than one step. Some viruses are capable of lodging in cells giving rise to the lysogenic replication. In this case the process (see figure 3) is as follows:

1. The virus infects the host cell, being lodged in its genome, (a) and (b) in figure 3.
2. The virus remains hidden inside the cell during a while until it is activated by any cause, for example ultraviolet irradiation or X-rays, (c) in figure 3.

3. The replication of cells altered, with proteins from the virus, starts.

FIGURE 3

The virus lodged in the genome of the host cell can cause alterations in the cell genome, (e) in figure 3. Occasionally certain genes participating in growth processes as the protooncogenes can give birth to tumour cells. Note that the virus enters the cell and then, the infected cell (bacteria for phagocyte case) is known as the organism's host.

2.3 Antigenic properties in certain viruses and infection end

Some viruses have the property of leading an antigenic response in the infected organism. In these situations an immune response is originated causing the creation of antibodies.

The end of the virus infection can be reached in two ways: the organism beats the virus implying the host recovery, or the virus beats the defence capabilities of the organism and the host death takes place.

2.4 Natural optimization in Viral Systems and its computational equivalency

In nature, viruses are active organisms that reach their major success when infecting individuals with a low level of health (note that the objective function $f(x)$ represents the cell's health).

The viral infection acts in an efficient way by means of developing a continuous process in time searching for individuals, or bacteria in the case of phagocyte infections, that are "suitable for infection", precisely those with a lower level of health (or low value of $f(x)$in computational terms). This way, viruses optimize their objective, and infecting less healthy individuals (poorer solutions) viruses are propagated through a major size of the population. In fact, in some occasions when the virus infection capabilities decrease, the virus using the cell genome (solution encoding) mutates to a different form with the intention of achieving higher success in the infection process, see figure 3. This describes a continuous process of the infection's systematic progress lodging on unhealthy cells that can lead the organism up to two different ends. The organism dies (it corresponds to reaching the optimum or beating the gap) or the organism resists the infection according to the virus isolation case (it corresponds to reaching the maximum number of iterations). In fact, the concept is close to reality because "more successful epidemics" are those where a larger number of deaths took place. For example, this was the case of the Black Death killing between a third and two thirds of Europe's population in the mid-late 14th century, the Spanish Flu Pandemic causing between 50 million and 100 million people worldwide killed in 1918 and 1919, or more recently the African Ebola with mortality rates ranging from 50% to 90%. So, following this data the death of the organism would lead to the objective (optimum) of a "successful infection".

3 Viral System description

3.1 Viral System components

The use of natural analogies is quite common when dealing with optimization in NP-complete problems. In this line, multiagent systems (MAS) have contributed a large number of methods taken from the natural world. This is the case of ants for path planning or brood sorting, termites for nest building, wasps for task differentiation, birds and fish for flocking, and wolves for surrounding preys (see Van Dyke [8] for an in-depth explanation). Here, we present viruses infecting organisms.

Our Viral System (VS) is an approach capable of dealing with any optimisation problem. To present the VS we follow the Van Dyke suggestion.

Each VS is defined by three components: a set of viruses, an organism and an interaction between them:

$$VS = <Virus, Organism, Interaction>$$

The *Virus* component of the VS is a set consisting of single viruses:

$$Virus = \{ Virus_1, Virus_2 , \dots , Virus_n \}$$

Each virus is defined in four components:

$$Virus_i = <State_i, Input_i, Output_i, Process_i>$$

Where each component means:

- $State_i$ characterizes the virus. It defines the cell infected by the virus. It is typically the mathematical encoding of the solution in computational terms, which we also call *genome*.

  A concrete virus, $Virus_i$, can produce the infection of a cell of the organism providing a host. Additionally, the evolution of the residence time of the virus inside the cell can be defined by the number of nucleus-capsids replicated for the lytic replication (NR) or the number of iterations for the lysogenic replication (IT). So, the three-tuple genome-NR-IT defines the $State_i$ for the $Virus_i$.

- $Input_i$ identifies the information that the virus can collect from the organism. This information is always collected in the proximity of the virus. $Input_i$ represents the input's interaction with the organism (organism's information → virus). It corresponds to the neighbourhood of the cell in computational terms.

- $Output_i$ identifies the actions that the virus can take. $Output_i$ represents the output's interaction with the organism (virus → organism). It corresponds to the selection mechanism of the type of virus replication in computational terms.

- $Process_i$ represents the autonomous behaviour of the virus, changing the $State_i$. It corresponds to the replication operator process in computational terms.

The *Organism* component of the VS is defined by two components:

$$Organism = < State_o, Process_o >$$

Where each component means:

- $State_o$ characterizes the organism state in each instant. It consists of the clinical picture and the lowest healthy cell (the best solution found of the optimization problem).

  The set of feasible solutions in a specific space $\Re^n$ is given by the problem constraints (2).

$$K = \{x : g_i(x) \leq 0,\ \forall i = 1, \cdots, n\} \quad (2)$$

  Where each one of the feasible solutions of the problem (2), $x \in K$, has been called a cell. The *genome* is the mathematical encoding of each cell or feasible solution. When a virus infects a cell, this cell enters the population of infected cells. The total amount of infected cells constitutes the infected part of *K* for each time instant, and it is named "*clinical picture*". It contains the overall information of the infection needed by the algorithm in each instant, *t*.

  Thus, the clinical picture consists of every three-tuple genome-NR-IT defining the $State_i$ of each $Virus_i$. In the same way, the overall clinical picture plus the lowest value of $f(x)$ defines the *Organism* $State_o$. Figure 4 depicts the *State* concept for the organism and the viruses.

FIGURE 4

- $Process_o$ represents the autonomous behaviour of the organism that tries to protect itself from the infection threat, consisting of antigen liberation. Medically, an antigen is any substance that elicits an immune response. The antigens generate an immune response by means of antibodies trying to fight the virus infection. The computational mission of the antigens is to liberate space in the population of infected cells (clinical picture), trying to maintain free record memory in the clinical picture to incorporate new infected cells (new feasible solutions). Thus, due to the antigens' activity, infected cells (in the clinical picture) can be recovered (removed) and cells in the organism that could be infected are not infected due to this antigenic substance. In computational terms, the antigens can be viewed as destructive agents, as described in Talukdar et al. [15].

3.2 Viral System Interaction

The *Interaction* component of the VS is conditioned by the *Input* and *Output* actions that lead to a *Process* of every virus and the corresponding *Organism* response. A $Virus_i$ process implies a resulting change in the organism, and the same applies for an *Organism*'s process. The interaction is the union of both actions.

3.2.1 Virus input sensor: Neighbourhood identification

The input sensor of each virus, $Input_i$, collects information from the organism. The sensors map the genome of the cell and detect the set of cells close to the infected one. This set is named the neighbourhood of the feasible solution x, $V(x)$. The neighbourhood depends on the shape of the constraints of the problem, $g_i(x)$.

3.2.2 Virus output ejector: replication type selection

The ejector, $Output_i$, selects the type of evolution of the virus. We consider two options: the lytic replication if the life cycle of the virus is executed in only one step; and the lysogenic replication if the life cycle of the virus is executed in two steps. So, in order to select the type of replication, we consider lytic replication (with probability p_{lt}) or lysogenic replication (with probability p_{lg}), where $p_{lt} + p_{lg} = 1$.

3.2.3 Process: Lytic replication

For the lytic replication case we have considered two different evolutions for different types of infection: a parsimonious infection, selecting the cells to be infected; and a devastating infection where a massive number of cells is infected. An example of the first case is the HIV virus, which through a step-by-step evolution destroys the immune system during a process that can take years. On the contrary, an example of the second case is the Ebola virus with a rapid and massive infection that very often produces the death of the patient in a few days.

3.2.3.1 Virus process in massive infection

The lytic replication starts only after a specific number of nucleus-capsids have been replicated. So, each time instant (iteration t) a number of virus replications (NR) takes place. The number of replications per iteration is calculated as function of a binomial variable, Z, adding its value to the total NR.

After a specific number of nucleus-capsids has been replicated inside the cell (LNR), the bacterium border is broken, liberating the lodged viruses. All these viruses are active and prepared to infect new cells. The value of LNR depends on the cell's health conditions. So a healthy cell (with high value of $f(x)$) will have low probability of getting infected, and therefore the value of LNR will be higher. In the opposite it will have a lower value of LNR. The following equation (3) shows the calculation procedure for LNR in a cell x:

$$\mathrm{LNR}_{\text{cell-}x} = \mathrm{LNR}^0 \cdot \left(\frac{f(x) - f(\hat{x})}{f(\hat{x})} \right)$$

where $\hat{x}$ is the cell that produces the best known result of the problem (in terms of $f(x)$) and x is the infected cell being analysed. LNR^0 is the initial value for LNR (3)

The number of nucleus-capsids replicated each iteration can be approximated by a Binomial distribution given by the maximum level of nucleus-capsids replicated, LNR, and the single probability of one replication, p_r,: Z = Bin (LNR , p_r).

Once the distribution has been stated, we can calculate the probability of replicating exactly z nucleus-capsids, P(Z=z), as well as the average, E(Z), and variance, Var(Z), equations (4-6).

$$\mathrm{P}(\mathrm{Z} = z) = \binom{\mathrm{LNR}}{z} p_r^z \cdot (1 - p_r)^{\mathrm{LNR}-z} \quad (4)$$

$$\mathrm{E}(\mathrm{Z}) = p_r \cdot \mathrm{LNR} \quad (5)$$

$$\mathrm{Var}(\mathrm{Z}) = p_r \cdot (1 - p_r) \mathrm{LNR} \quad (6)$$

Once the bacterium border is broken liberating the viruses, each one of the viruses has a probability, p_i, of infecting other new cells of the neighbourhood.

Let the neighbourhood cardinality of the feasible solution, x, be $|V(x)|$. Then, we can define Y as a binomial random variable representing the cells infected by the virus in the neighbourhood. The parameters of the binomial distribution are Y = Bin ($|V(x)|$, p_i).

In this situation, the probability of infecting exactly y nucleus-capsids, P(Y=y), as well as the expected number of new infected cells, E(Y), and the variance, Var(Y), will be (7-9):

$$\mathrm{P}(\mathrm{Y} = y) = \binom{|V(x)|}{y} p_i^y \cdot (1 - p_i)^{|V(x)|-y} \quad (7)$$

$$\mathrm{E}(\mathrm{Y}) = p_i \cdot |V(x)| \quad (8)$$

$$\mathrm{Var}(\mathrm{Y}) = p_i \cdot (1 - p_i) |V(x)| \quad (9)$$

Under these conditions, the growth of the total number of infected cells follows a geometric progression and there is no apparent limit to the growth of the population infected. So it must be bounded by another

phenomenon in order to maintain the VS bounded-in-computation (this is described in the *Organism Process* section as the antigenic response).

3.2.3.2 Organism process in massive infection

The organism responds to the viral infection by releasing antigens. The antigens generate an immune response by means of antibodies. The *Organism Process* is defined by the antigenic capacity of the organism, which depends on the $State_o$ and leads to a new situation of the *Organism State*.

Each one of the infected cells in the clinical picture has a probability of developing antibodies against the infection. The probability of generating antibodies for any cell of the infected population can be given by a Bernoulli probability distribution.

Therefore, the probability of generating antibodies is p_{an}: $A(x) = Ber(p_{an})$, where x is an infected cell. Hence, the total population of infected cells generating antibodies is characterized by a Binomial distribution of parameters: the size of the population infected, n, and the probability of generating antibodies, p_{an}: $A(population) = Bin(n, p_{an})$.

On the other hand, we know that when a virus reaches the LNR limit, the bacterium border is broken and new viruses try to infect new cells in their neighbourhoods. Under these conditions, we must compute the antigenic capacity for every cell in the neighbourhood of an active virus. It is estimated as a Bernoulli probability distribution given by the probability of generating antibodies, p_{an}: $A(x') = Ber(p_{an}) : x' \in V(x)$. Therefore, the total number of cells in the neighbourhood with antibodies will follow a Binomial probability distribution given by the total size of the neighbourhood for all the active viruses, $|V(x)|$, and the probability of generating antibodies, p_{an}: $A = Bin(|V(x)|, p_{an})$.

Then, the probability of finding exactly a immune cells, $P(A=a)$, as well as the expected value of immune cells, $E(A)$, and the variance, $Var(A)$, is given by (10-12).

$$P(A = a) = \binom{|V(x)|}{a} p_{an}^{a} \cdot (1 - p_{an})^{V-a} \quad (10)$$

$$E(A) = p_{an} \cdot |V(x)| \quad (11)$$

$$Var(A) = p_{an} \cdot (1 - p_{an}) \cdot |V(x)| \quad (12)$$

For this massive infection case, we have to guarantee two opposed tendencies. Firstly, we have to allow enough capability for the infection expansion (infecting as many cells in the neighbourhood as possible). Secondly, we have to keep the infection evolution bounded in order to have enough gaps in the clinical picture to guarantee

the incorporation of the new infected cells. For example, in the case of many cells being infected and not having many available gaps in the clinical picture, it would not be possible to incorporate these new infected cells.

The steady state of the clinical picture includes cells with different numbers of replicated nucleus-capsids. This number will vary from 0 (for the most recent cells infected) to LNR for those infected cells on the point of breaking their borders for liberating the cultivated viruses.

A Markovian Process defines the evolution of the clinical picture. The following Table 1 gives the transition law between the states, that is, the transition probability matrix, P.

TABLE 1

Once one cell reaches the state LNR the border is broken and the viruses are liberated. In the next iteration the cell is erased from the clinical picture leaving its gap for new infected cells.

Let $\pi = (\pi_0, \pi_1, ..., \pi_{\text{LNR}})$ be the probability of any cell of having 0, 1, … , LNR nucleus-capsids replicated.

The, equations (13-15) are satisfied in steady state.

$$\pi = \pi \cdot \text{P} \qquad (13)$$

$$\pi_j = \frac{1}{1-p_0}\left(\sum_{k=0}^{j-1} p_{j-k} \cdot \pi_k\right) \qquad \forall j = 1, 2, ..., \text{LNR} \qquad (14)$$

$$\pi_0 + \pi_1 + ... + \pi_{\text{LNR}} = 1 \qquad (15)$$

Therefore, the probability of having exactly LNR replicated nucleus-capsids, π_{LNR}, can be calculated by solving the (13-15) previous equations' system. With *n* cells in the clinical picture, the expected number of cells with exactly LNR nucleus-capsids is given by $n \cdot \pi_{\text{LNR}}$.

The viruses cultivated inside each one of these cells will be capable of infecting new neighbour cells. Let the cell x_k be an infected cell that has broken its border. Then, $\text{P}(\text{Y} = y \mid x_k) = \binom{|V(x_k)|}{y} p_i^y \cdot (1-p_i)^{|V(x_k)|-y}$ is the probability of infecting exactly *y* cells of its neighbourhood.

The expected number of new cells infected in the neighbourhood of x_k is given by $\text{E}(\text{Y} \mid \text{x}_\text{k}) = p_i \cdot |V(x_k)|$.

Being $n \cdot \pi_{\text{LNR}}$ the expected number of cells with exactly LNR nucleus-capsids and assuming independent events, the total expected number of new cells infected in the organism could be estimated as $\sum_{k=1}^{n \cdot \pi_{\text{LNR}}} p_i \cdot |V(x_k)|$.

Since $\sum_{k=1}^{n\cdot\pi_{\text{LNR}}} p_i \cdot |V(x_k)|$ is normally higher than $n \cdot \pi_{\text{LNR}}$, the number of gaps in the clinical picture will be lower than the total expected number of new cells infected in the organism. However in order to control the infection, the Organism responds to it by means of antibody generation.

As we have stated before, at each iteration every cell has a probability equal to p_{an} of generating antibodies and become immune to the virus infection.

The expected number of cells in the clinical picture having developed antibodies will be $(n - n \cdot \pi_{\text{LNR}}) \cdot p_{an}$ and the expected number of cells in the neighbourhood of x_k with antibodies will be $(p_i \cdot |V(x_k)|) \cdot p_{an}$.

Therefore, the expected number of empty gaps in the clinical picture will be given by (16) and the expected number of new cells infected by (17).

$$n \cdot \pi_{\text{LNR}} + (n - n \cdot \pi_{\text{LNR}}) \cdot p_{an} \qquad (16)$$

$$(1 - p_{an}) \cdot \left(\sum_{k=1}^{n\cdot\pi_{\text{LNR}}} p_i \cdot |V(x_k)| \right) \qquad (17)$$

Equation (16) should be higher than equation (17) in order to fix an adequate value for p_{an}. So, considering $|\overline{V(x)}|$ as the average neighbourhood size for the concrete problem and solving the equation we can state (18) as an approximate value for p_{an}.

$$p_{an} > \frac{n \cdot \pi_{\text{LNR}} \cdot (p_i \cdot |\overline{V(x)}| - 1)}{n \cdot \pi_{\text{LNR}} \cdot (p_i \cdot |\overline{V(x)}| - 1) + n} \qquad (18)$$

Figure 5 defines the algorithm evolution for the massive infection case. The initial state is depicted by the clinical picture on the left hand side. The viruses reaching the level of nucleus-capsids (LNR) break the border and start infecting new cells in their neighbourhoods. This phenomenon corresponds to the virus process formed by the input sensors and output ejectors. The Organism process is characterized by the antigenic response by liberating space in the clinical picture and by creating antibodies in cells in the viruses' neighbourhoods. Finally, the interaction (right-hand side of the figure) defines the new clinical picture, with new infected cells lodging viruses.

FIGURE 5

3.2.3.3 Virus process in selective infection

After replicating nucleus-capsids, according to equations (4)-(6), and once the number of nucleus-capsids surpasses the limit given by LNR, the border of the cell is broken and the viruses are liberated. For this case, one single cell is selected to be infected. In order to do so, the neighbourhood is evaluated and one of the least healthy cells is selected, configuring the new host to expand the infection. This type of infection can be computationally considered a step-by-step method.

3.2.3.4 Organism process in selective infection

In this case, the virus selects a cell with a low value of $f(x)$ in the neighbourhood. However, the virus will not be able to infect those cells that have developed antigens.

A higher value of $f(x)$ implies a healthy cell and therefore this cell will have a higher probability of developing an antigenic response. On the contrary, a cell with a low value of $f(x)$ will imply an unhealthy cell with a lower probability of developing an antigenic response.

In order to represent this phenomenon, we use of a hypergeometric function. The cell with an inverse objective function evaluation, $\frac{1}{f(x)}$, in ranking position-i, has a probability of generating antibodies, $p_{an}(x)$, that is given by $q(1\text{-}q)^i$, being q the probability of generating antibodies for the worst individual. Finally, a residual probability remains, which is added to the worst individual.

Then, if the probability of generating antibodies for the case of cell x is $p_{an}(x)$, $A(x)$ is defined as a Bernoulli random variable: $A(x) = \text{Ber}(p_{an}(x))$.

If cell x generates antibodies, the cell is not infected and it is therefore not included in the new clinical picture. For recording this clinical picture we use the original cell (that was infected by the virus and that reached the LNR limit) and we initiate a lysogenic cycle for that cell.

Figure 6 defines the algorithm evolution for the selective infection case. The initial state is on the left-hand side: the virus process starts with viruses breaking the border and starting the infection of new cells in their neighbourhoods. Each virus selects the most promising cell, which is the least healthy cell. The Organism process is characterized by the probability of antigenic response in the least healthy cell. Those cells developing antibodies are not infected. Finally, the interaction (right hand side of the figure) defines the new clinical picture, with new infected cells lodging viruses. The cells generating antibodies follow a new lysogenic replication.

FIGURE 6

3.2.4 Process: Lysogenic replication

3.2.4.1 Virus Process

If the life cycle of the virus is executed in two steps it is called lysogenic replication. In this situation, the virus remains hidden inside the cell until an external cause activates the virus. We consider that the activation of the lysogenic replication can happen after a limit of iterations has passed (LIT). As for the calculation of LNR, the value of LIT depends on the cell's health conditions, so a healthy cell (high value of $f(x)$) will have a low probability of getting infected, id est. the value of LIT will be higher. On the contrary, it will have a lower value of LIT. The following equation (19) shows the calculus procedure for LIT in a cell x:

$$\mathrm{LIT}_{\text{cell-}x} = \mathrm{LIT}^0 \cdot \left(\frac{f(x) - f(\tilde{x})}{f(\tilde{x})} \right) \qquad (19)$$

where LIT^0 is the initial value for LIT

Once the virus has been activated, it produces alterations in the cell's genome. It is equivalent to a genome mutation process in the mathematical programming encoding of the feasible solution.

3.2.4.2 Organism Process

The lysogenic interaction is described as the substitution of the new genome-modified cell by the old one. It is quite similar to a mutation process in several types of evolutionary algorithms.

Finally, for both type of replications (lysogenic and lytic), one last control is applied to the *Interaction* between the *Organism* and the *Virus*. As much for the lytic replication as for the lysogenic replication, a duplication control is carried out in order to avoid visits to cells previously explored and recorded in the clinical picture.

3.3 Viral System algorithm

3.3.1 Initialisation

The algorithm initialisation provides the first *State* for the *Organism* and the *Virus*.

1. Inoculate n viruses over n cells. This gives the first clinical picture with the first set of feasible solutions.
2. Define type of virus infection: selective or massive infection.

3.3.2 Steady state

1. In case of massive infection: initiate the *Organism Process*: antigenic response of the clinical picture.
2. The *Output* ejector of each virus determines if any *Virus Process* can start. And in positive case, the type of cycle to be initiated for each cell: lytic or lysogenic replication.
3. In case of lytic replication.

3.1. All the infected cells start to replicate nucleus-capsids. This updates the NR parameter for the $Virus_i$ lodged in the cell.

3.2. Those cells with NR greater than LNR represent active viruses.

3.2.1. The *Input* sensor determines the neighbourhood of the cell infected, $V(x)$.

3.2.2. Initiate *Virus Process*:

3.2.2.1. Case of massive infection: Y-A cells of the neighbourhood are infected, and must be incorporated into the clinical picture. If there is not enough free space in the population, it will randomly erase the necessary cells from the Y-A selected cells.

3.2.2.2. Case of selective infection: One only cell from the neighbourhood is selected according to the selective selection. The antigenic response of such cell is evaluated as a Bernoulli process (A). In case of antigenic response a lysogenic replication is initiated.

4. In case of lysogenic replication. If the number of iterations is greater than LIT.

4.1. The *Input* sensor maps the cell genome

4.2. Initiate *Virus Process*: the virus mutates the cell genome encoding.

4.3. The new mutated cell substitutes the previous one in the clinical picture.

5. Update the best solution reached in the *Organism State*.

3.3.3 Ending

The VS ending is achieved in two ways: the collapse and death of the organism, or the isolation of the virus.

Collapse and death of the organism

Computationally, the death of the organism can be reached when the difference between the best found solution and a known lower bound is smaller than a stated gap. There exist certain lower bounds known for several NP-problems. Nevertheless, a lower bound could always be calculated by means of the linear or Lagrangian relaxation for problems with a linear objective function and linear constraints. In case of knowing the optimum of the problem, the gap can be set equal to zero. This is a common case when dealing with trial problem collections.

When the difference between the lower bound (LB) and the best found solution is below a *gap*, we consider that the organism has collapsed (20), and call the end of the VS.

$$gap = \frac{|\, f(x^*) - \mathrm{LB}\,|}{\mathrm{LB}} \qquad (20)$$

Virus isolation

After a number of iterations have been produced (Nmax), we consider that the viral infection cannot evolve further.

When this criterion is used together with the previous one, the situation denotes that the gap is not reached, and the virus does not create a serious infection in the organism. Under this condition, the organism would have survived the virus infection.

3.4 Selective infection case pseudocode

The following exhaustive pseudocode corresponds to the selective infection case. The massive infection case will follow the main procedures, but it will replace the procedure 'Get_Best_Neighbourhood_Solution' by the specific massive procedure corresponding to the selection of a set of cells to be infected as previously described in section 3.3.

```
Procedure Virus_System(N_max , clinical_size , p_lt , p_i , p_an , LNR , LIT)
  CP = ∅ {Clinical Picture}
  iterations = 0
  Get_Initial_Clinical_Picture(CP , clinical_size , p_lt)
  Do
    iterations = iterations + 1
    For c = 1 to clinical_size
      If Replicat_Type(CP(c)) = 'Lytic' Then
        Lytic_Replication (c , LNR)
      Else
        Lysogenic_Replication(c)
      End If
    Next
  Loop Until iterations= N_max or Check_Gap(CP) = True
End Procedure

Procedure Get_Initial_Clinical_Picture(CP , clinical_size , p_lt)
  For i = 1 to clinical_size
    {Get randomly a feasible solution}
    CP(i)= Get_Rnd_Feasible_Solution()
    {Assign randomly a replication type, Lytic or Lysogenic}
    Replicat_Type(CP(i)) = Get_Rnd_Replication _Type (p_lt)
  Next
End Procedure

Procedure Lytic_Replication (c)
{Get the number of replicated nucleus-capsids}
  Z = Get_Rnd_Binomial_Probability(LNR , p_r)
  z = Get_Value_Binomial(Z , LNR , p_r)
  CP(c).NR = CP(c).NR + z
```

```
  If CP(c).NR > CP(c).LNR Then
    Interaction_Virus(c)
  End If
End Procedure

Procedure Interaction_Virus(c)
  Get_ Neighbourhood (c , Vc)
  {Get the probability of generating antibodies}
  A = Get_Rnd_Binomial_Probability(|Vc| , pan)
  a = Get_Value_Binomial(A, |Vc|, pan)

  {Get the probability of infection in the neighbourhood}
  Y = Get_Rnd_Binomial_Probability(|Vc| , pi)
  y = Get_Value_Binomial(Y , |Vc| , pi)
  Get_Best_Neighbourhood_Solution(Vc , y , a , cNEW)
  c= cNEW
  Replicat_Type(CP(c)) = Get_Rnd_Replication_Type (pit)
End Procedure
Procedure Get_Best_Neighbourhood_Solution(Vc , y , a)
  j = Get_Rnd_Starting_Evaluation(|Vc|)
  i = 0
  Do
    i = i + 1
    Solution = Get_Solution(Vc , j)
    if health(cNEW ) < health(Solution) then
      cNEW = Solution
    end if
    j = j+1
  Loop Until i = y-a
End Procedure

Procedure Lysogenic(c)
  CP(c).IT = CP(c).IT + 1
  If CP(c).IT > CP(c).LIT Then
    Mutation(c, cNEW)
    c = cNEW
    Replicat_Type(CP(c)) = Get_Rnd_Replication _Type(PLytic)
  End If
    End Procedure
```

4 An application example for Viral Systems: the Steiner problem

We have used the Steiner problem, a well-known NP-Complete problem, to test the efficiency of our Viral System described above.

The Steiner problem can be stated as follows:

- Given a non-directed graph $G = (N,A)$ with $|N|$ nodes and $|A|$ arcs with costs c_{ij} $\forall(i,j)\in A$; and a subset $T\subseteq N$ with $|T|$ nodes called terminals or targets, with the rest of the nodes in N called Steiner nodes,

- find a network $G_T \subseteq G$ joining all the terminal nodes in T at minimum cost. This network can include some of the Steiner nodes but does not have to include all the Steiner nodes.

The Steiner problem has been widely dealt with in the scientific literature. There exists a wide range of heuristics giving good approximations (e.g. Takahashi and Matsuyama [16], and Beasley [17]), as well as other well-proven direct methods like branch and bound, branch and cut, cutting plane algorithms, etc. (e.g. Wong [18], Chopra et al. [19], and Grötschel et al. [20]). Also, metaheuristics have been applied to the Steiner problem successfully. Gendreau et al. [21] for Tabu Search, as well as the Genetic Algorithms from Esbensen [22], and Voss and Gutenschwager [23] are three of the most notable approaches.

4.1 VS characterization for the Steiner problem

The ***Organism state*** is depicted by the clinical picture representing the infected part of the Steiner problem hull, K, containing all the feasible solutions. A common formulation (21) for K is the coverage one, e.g. Koch et al. [24].

$$
\begin{aligned}
K: \quad & X\left(\delta(W)\right) \geq 1,\ \forall W \subset N,\ W \cap T \neq \varnothing,\ \left((N \setminus W) \cap T \neq \varnothing\right) \\
& 0 \leq x_{ij} \leq 1,\ \forall (i,j) \in A \\
& x \text{ integer}
\end{aligned}
\qquad (21)
$$

Where $\delta(X)$ denotes the cut induced by $X \subseteq N$, that is, the set of arcs with one node in W and one in its complement, and $X(F) = \sum_{(i,j) \in F} x_{ij},\ \forall F \subseteq A$. It is easy to see that there is a one-to-one correspondence between Steiner trees in $G = (N,A)$ and $\{0,1\}$ vectors satisfying K.

We represent the ***genome*** of the cells by a bit string of size equal to $|N|$ in which each bit position i corresponds to the node i in the graph. A 1 means that the node i is connected, while the bit is set to 0 otherwise. As all the terminals must be in the Steiner tree, it is sufficient to use a bit string of size $|N\text{-}T|$ including only the Steiner nodes belonging to the Steiner tree.

Under these conditions, a Steiner tree can be constructed by means of a minimum spanning tree (MST) that contains all the terminal nodes (set T), the subset of Steiner nodes in the bit string fixed to 1 and, perhaps, some artificial arcs if the set is disconnected. In case of necessity of introduction of artificial arcs due to the disconnection of the tree, there will be diverse possibilities. We make use of the graph construction mechanisms described in Gendreau et al. [21].

Once we have stated the cell genome we can define the ***Virus state***. The three-tuple formed by the genome of each cell infected plus the number of replicated nucleus-capsids (in the case of lytic replication) or the number of generations (in the case of lysogenic replication) defines the virus state.

The entire infected cell population, which is the clinical picture, and the best solution complete the *Organism* and therefore the *Virus* state.

The ***Output ejectors*** of the *Virus* component of the VS are clearly defined by the type of replication. On the contrary, the ***Input sensors*** must be carefully stated. In fact, a key decision in every VS is to state an adequate cell neighbourhood for the virus in the lytic replication process and a genome alteration process for the lysogenic replication.

In the case of the Steiner problem, for the lysogenic replication and given a feasible solution $x \in K$, the genome alteration is made by flipping a bit in the string.

For the lytic replication and given a feasible solution $x \in K$, the neighbourhood of such a solution consists of the set of bit strings that can be obtained by the removal or the addition of a single Steiner node from/to the current cell encoding. In order to be efficient, the new MSTs must be found by manipulating a rooted tree data structure carefully. Gendreau et al. [21] describes the procedure in-depth.

Finally, the *Virus* and *Organism* components are completed by the specification of the *Process*. The ***Organism Process*** consists of the antigenic response and it is mainly determined by the determination of the parameter p_{an} , as was stated in section 3.2.3.4. The ***Virus Process*** consists of the determination of the type of replication that is conditioned by the parameters p_{lt} and p_{lg}. Additionally, the *Virus Process* depends on the parameters of replication, p_r, infection, p_i, and the limits LNR^0 and LIT^0.

Due to the special encoding for the Steiner problem solutions the neighbourhood size is constant and equal to the number of Steiner nodes. It must be noted that the neighbourhood is set by changing the value of each bit from 0 to 1 and vice versa.

The ***Interaction*** takes place after the selection of the *Virus Process*. It is basically determined by the random evolution of the viral infection and the antigenic capacity of response.

4.2 Specific pseudocode procedures for the Steiner problem

The general pseudocode functions and procedures were described in section 3.4. We have included here the more specific procedures for the Steiner problem. They are mainly the neighbourhood characterization and the Steiner-oriented mutation for the lysogenic replication. Such procedures will have to be specifically oriented to the problem being analysed. The following pseudocode describes them.

```
Procedure Get_ Neighbourhood (c, V_c)
  V_c = ∅ {Neighbourhood}
  For i = 1 to NumSteiners
    If i∈Solution(c) then
      Neighbour_NEW = Extract_from_Solution(c, i)
    Else
      Neighbour_NEW = Introduce_in_the_Solution(c, i)
    End if
    V_c = V_c + Neighbour_NEW
  Next
End Procedure

Procedure Get_Mutation_Solution(c, c_NEW)
  SNode = Get_Rnd_SteinerNode(NumSteiners)
  If SNode∈Solution(c) then
    c_NEW = Extract_from_Solution(c, SNode)
  Else
    c_NEW = Introduce_in_the_Solution(c, SNode)
  End if
End Procedure
```

4.3 Computational results

In order to evaluate the quality of the VS solutions, we have compared the VS results with the optimal solutions of a test problems' collection of the OR-Library (series C, D and E, each one of them including 20 problems). Steiner problems series C consists of trials with 500 nodes, a number of arcs varying from 625 to 12,500, and terminals from 5 to 250, while series D consists of problems with 1,000 nodes, arcs varying from 1,250 to 25,000, and terminals from 5 to 500; and finally series E includes trials of 2,500 nodes, arcs varying from 3,125 to 62,500, and terminals from 5 to 1,250. The collection is accessible in the Web page http://people.brunel.ac.uk/~mastjjb/jeb/info.html. For the statistical calculations, due to the large size of the problems, we were forced to replace the binomial probabilistic distribution by its corresponding normal approximation.

Firstly, Steiner OR-Library problems were pre-processed according to the graph reduction rules (Winter [25]) that reduce the graph size. The graph characteristics are defined by the number of nodes, terminals and arcs. One of the parameters that determine the difficulty of obtaining the optimum is the percentage of terminals with respect to the total number of nodes, as Cortés et al. [26] have previously stated, although other factors like the total number of nodes and arcs also affect the computing time. The argument is based on the fact that problems with a low density of terminals or with a high density of terminals have a small feasible region; however problems with a medium density of terminals have a large feasible region.

Using this rationale, we divided the problems into three groups: group 1 for problems with less than 15% of terminal nodes; group 2 for problems with a percentage of terminal nodes between 15% and 30%; and group 3 for problems with more or equal than 30% of terminal nodes.

According to the special characteristics of the Steiner problem we realised that the selective infection provided a more adequate approach. We used an experimental design based on a fractional factorial design at two levels in order to analyse the impact of the VS parameters on solution quality. The fractional factorial design is a factorial experiment in which only an adequately chosen fraction of the treatment combinations required for the complete factorial experiment is selected to be run. In fact, for a number of factors k in a full factorial design, 2^k runs must be computed, which can quickly become very large. The solution to this problem is to use only a fraction of the runs specified by the full factorial design. The advantages of choosing fractional factorial designs for 2-level experiments are the desirable properties of being both balanced and orthogonal. See, for example, Montgomery [27] for an in-depth explanation on fractional factorial design.

The main two-level parameters of the VS according to the selective infection were encoded for the fractional factorial design as:

- ITER: number of iterations. Fractional factorial design levels: 50,000 (+) / 10,000 (-)
- POB: clinical picture. Fractional factorial design levels: 100 (+) / 50 (-)
- PLITI: probability for the lytic cycle. Fractional factorial design levels: 0.7 (+) / 0.3 (-)
- LNR: Limit of iterations for the lytic cycle (number of replications inside the cell to be broken). Fractional factorial design levels: 20 (+) / 10 (-)
- LIT: Limit of iterations for the lysogenic cycle. Fractional factorial design levels: 20 (+) / 10 (-)
- PZ: Probability of replicating z nucleus-capsids. Fractional factorial design levels: 0.7 (+) / 0.3 (-)

The experimental design was done with the Stein-C problems from the OR-Library. The fractional factorial design at two levels generated a total number of 2^6 parameter settings that were executed 4 times in order to reduce the random component. This gave a total amount of 256 trial parameter settings for each Stein-C problem. Since there are 20 test problems in the Stein-C set, we solved 5,120 instances. We grouped the parameter settings in the Stein-C set into three different categories according to the percentage of terminals (a critical parameter when solving the Steiner problem). Thus, Stein-C group 1 includes parameter settings with percentage of terminals under 15%, Stein-C group 2 parameter settings with percentage of terminals between 15% and 30%, and Stein-C group 3 for parameter settings with percentage of terminals higher than 30%. Hence, the group 1 is formed by steinc1, c2, c6, c7, c11, c12, c16, and c17 cases (giving 2,048 instances), while the

group 2 is formed by steinc8, c9, c13, c14, c18, and c19 cases (giving 1,536 instances), and the group 3 is formed by steinc3, c4, c5, c10, c15, and c20 cases (giving 1,536 instances).

We used Limdep econometric software in order to adjust the mode of the experimental design. In accordance with the selected codification, the variance analysis coincides with the minimum square error regression analysis. The results summarised for the three groups are shown below in Table 2,

TABLE 2

We have used italics to highlight the significant parameters for each specific group. Group 1 shows that the number of iterations (ITER), the clinical picture size or population (POB), the lytic replication probability (PLITI), and the limit of iterations for the lysogenic cycle (LIT) are the most relevant parameters in the experimental design. Also, the interactions with a t-ratio higher than 1 or lower than -1 must be taken into account. For example, the independent analysis of parameter ITER will recommend a low value of the parameter, whereas in the case of parameter POB a high value is recommended; however, a positive combination of both parameters ITER and POB (which is also called the combined effect or the cocktail) is recommended in order to obtain a better fitting. This situation corresponds to a contribution '+ +' or '- -' for ITER_POB. We have to take into account all these possible combinations in order to select the best overall combination of parameters (the final selection is shown in Table 3). Finally, we highlight that we obtained a very good adjusted R-squared equal to 92.1% for the data of this group.

Group 2 shows that the number of iterations (ITER), the clinical picture sizes (POB), the lytic replication probability (PLITI), and the limit of iterations for the lysogenic cycle (LIT) are the most relevant parameters in the experimental design. Also, the interactions with a t-ratio greater than 1 or lower than -1 must be taken into account. We obtained a very good adjusted R-squared equal to 93.1%.

Finally, for group 3, the number of iterations (ITER), the clinical picture size (POB), the lytic replication probability (PLITI), the limit of iterations for the lytic cycle, and the limit of iterations for the lysogenic cycle (LIT) are the most relevant parameters in the experimental design. Also, the interactions with a t-ratio greater than 1 or lower than -1 were taken into account. We obtained a very good adjusted R-squared equal to 98.4%.

Table 3 describes the best parameter selection for each group of Steiner problems. The selection of parameters was done by calculating the average error for each possible combination of sequences.

TABLE 3

It is important to note that the VS efficiency is non-dependent on the probability of generating a great or low number of nucleus-capsids (parameter PZ) with respect to the Steiner problem case, so its performance showed

non-dependency from this parameter. The rest of parameters can affect in different ways the objective of reducing the error of the method.

After executing the whole fractional factorial design at two levels we obtained the optimum for all the Stein-C set problems. Then we used the parameters from Table 3 to solve the more complex Stein-D and Stein-E collections. First, we executed four times the VS with the first set of parameters, and then we executed four times the VS with the second set of parameters. Table 4, Table 5 and Table 6 show the results (in error percentage with respect to the optimum) for the Stein-C, Stein-D and Stein-E problems and their comparison with the Tabu Search approach from Gendreau et al. [21] (their P-Tabu and F-Tabu methods), which is the best approach for the Steiner problem in terms of solution quality. The tables include the best Genetic Algorithm approaches that have solved the Steiner problem, which are due to Esbensen [22] (GA-E) and Voss and Gutenschwager [23] (GA-V). Finally, we also show the results obtained for the Minimum Path Heuristic (MPH) by Takahashi et al. [16] and their results reported by Gendreau et al [21]. This collection of methods provides an adequate basis of comparison in order to test the effectiveness of our VS.

TABLE 4, TABLE 5, TABLE 6

According to the results of Table 4, Table 5 and Table 6 (a total of 60 problems), VS was the best approach in 48 times and beat clearly the GA-E (22 times), GA-V (9 times), P-Tabu (40 times) and MPH (25 times) approaches. Only F-Tabu showed better performance, being the best approach 51 times. However, VS provided a better solution for the C9, C14, D3, E3, E12, E15 and E19 problems. VS provided very valuable results taking into account that F-Tabu was processed after selecting the 100 best different trees found by the MPH algorithm, after executing the P-Tabu approach as an initial search and reprocessing it into the final Full Tabu Steiner (F-Tabu). So the quality of the F-Tabu results is very high but it is also very much conditioned by the very good seed that was provided (MPH). Furthermore, it is remarkable that MPH obtains better results than special oriented genetic algorithms such as GA-E and GA-V; it gives an idea of the seed quality for F-Tabu. Undoubtedly it is due to the heuristic being executed 100 times in order to obtain the best possible seed to be supplied to the P-Tabu and F-Tabu methods. On the contrary, we apply VS directly to the reduced graph without pre-processing it with any special previous heuristic as MPH (or the others that F-Tabu uses). Nevertheless, our initial clinical picture (the seed of our method) is wholly random-generated. We did not use a good seed provided by a good heuristic because we were interested in observing the quality of the VS evolution to the final solution, more than on outperforming previous heuristics. However, we realized that without searching for a

good seed we were obtaining results equivalent (in quality terms) to the best Steiner approach: the F-Tabu algorithm.

We also have to state that when the problem was fully executed for the overall set of parameters with all the possible combinations, that is, what we did with the Stein-C, VS always obtained the optimum of the problem. Nevertheless, when we reduced the possible combinations of parameters (focusing them on Table 3) the results quality was maintained on a very high standard, but several (very few) times we were outperformed by F-Tabu, due to all our previous explanations (mainly the preparation of the seed by means of other previously executed heuristics).

With respect to the time consumption, we have omitted the CPU times in Tables 7, 8 and 9 as they cannot be compared with other algorithms whose tests were run in a different computer. However, we can estimate the order of time consumption by the algorithm's complexity, given in (22).

$$time \sim O\left(ITER \cdot NumSteiners \cdot Ngraph^2\right) \qquad (22)$$

where ITER is the maximum number of iterations, NumSteiners the number of Steiner nodes in the graph and Ngraph the total number of nodes in the graph.

The evaluation of the neighbourhood of a cell is the process that requires the largest amount of computational time in the algorithm, and this process is the same used in the F-Tabu algorithm to reach the next solution in the search procedure. Therefore, the worst case would be to run lytic replications in all iterations of the algorithm. For that case, the degree of complexity would be equal to the number of neighbourhood solutions evaluated multiplied by the complexity of the evaluation process. The number of evaluations is obtained as the number of iterations multiplied by the number of Steiner nodes. The evaluation process is the calculation of a minimum spanning tree, which is $O(Ngraph^2)$.

Furthermore, many combinations of VS parameters were analysed in our experimentation. Hence, we just indicate in Table 7 the average of the computational times (in CPU-seconds) on the best (ITER=10.000, PLITI=0.3 and LNR=20) and worst cases (ITER=50.000, PLITI=0.7 and LNR=10).

TABLE 7

5 Conclusions

In this paper we have presented a new approach to optimize combinatorial problems called Viral System due to its inspiration in the infection behaviour of viruses. The method was tested with the Steiner problem, which is NP-Complete. VS was applied to a library of medium-to-large-sized problems for which the optimal solution is

known and the results were compared with the best Tabu Search approach (and perhaps the best metaheuristic for the Steiner problem) and the best Genetic Algorithm approaches that we found in the scientific literature. VS clearly improves the results from the Genetic Algorithms (a bio-inspired evolutionary methodology close to our proposal) and for several cases VS obtains better results than the Tabu Search, even taking into account that the Tabu Search starts from a very good seed provided by the MPH heuristic.

Our future research is focused on generating an optimisation library associated to different classes of viruses (Figure 1 showed several cases). Here we have only addressed the phagocyte case, which corresponds to the less complex virus infection. We want to test different type of infections with different set of problems trying to identify the most adequate one to each type of problem.

6 Acknowledgements

The authors acknowledge two anonymous referees for their helpful comments on earlier versions of this manuscript that have significantly contributed to enrich the quality of the paper in its final form.

7 References

[1] Kirkpatrick, S. Optimization by simulated annealing: Quantitative studies. Journal of Statistical Physics 1982; 34: 975-986.

[2] Rumelhart, D.E., McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume I: Foundations*, MIT Press, 1986.

[3] Holland, J.H. Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[4] Holland, J.H., Holyoak, K.J., Nisbett, R.E., Thagard, P.R. Induction: Processes of Inference, Learning and Discovery, MIT Press, 1986.

[5] Goldberg, D.E. Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[6] Glover, F. Tabu Search – Part I. ORSA Journal on Computing 1989; 1 (3): 190-206.

[7] Glover, F. Tabu Search – Part II. ORSA Journal on Computing 1990; 2 (1): 4-32.

[8] Van Dyke Parunak, H. "Go to the ant": Engineering principles from natural multi-agent systems. Annals of Operations Research 1997; 75: 69-101.

[9] Jennings, N.R., Sycara, K., Wooldridge,M. A roadmap of agent research and development. Autonomous Agents and Multi-Agent Systems 1998; 1: 7-38.

[10] Luck, M., McBurney, P., Preist, C. Agent Technology: Enabling Next Generation Computing, AgentLink, EC V FP Information and Service Technologies Programme, 2003.

[11] Farmer, J.D., Packard, N. and Perelson, A. The immune system, adaptation and machine learning. Physica D 1986; Vol. 22: 187--204.

[12] Kubota, N., Fukuda, T., Shimojima, K. Virus-evolutionary genetic algorithm for a self-organizing manufacturing system. Computers and Industrial Engineering 1996; 30 (4): 1015-1026.

[13] Saito, S. A Genetic Algorithm by Use of Virus Evolutionary Theory for Combinatorial Problems. Optimization and Optimal Control, Vol.1, World Scientific Publishing Co. 2003; 10: 251−268.

[14] Garey, M.R. and Johnson, D.S. Computers and Intractability. W.H. Freeman, New York, 1979.

[15] Talukdar, S., Gove, A., De Souza, P. Asynchronous teams: cooperation schemes for autonomous agents. Journal of Heuristics 1998; 4: 295-321.

[16] Takahashi, H., Matsuyama, A. An approximate solution for the Steiner problem in graphs. Mathematica Japonica 1980; 24: 573-577.

[17] Beasley, J.E. A Heuristic for Euclidean and Rectilinear Steiner Problems. European Journal of Operational Research 1992; 58: 284-292.

[18] Wong, R.T. A Dual Ascent Approach for the Steiner Tree Problems on a Directed Graph. Mathematical Programming 1984; 28: 271-287.

[19] Chopra, S. Gorres, E.R., Rao, M.R. Solving the Steiner Tree Problem on a Graph Using Branch and Cut. ORSA Journal on Computing 1992; 4 (3): 320-335.

[20] Grötschel, M., Martin, A., Weismantel, R. Packing Steiner Trees: A Cutting Plane Algorithm and Computational Results. Technical report SC92-09, Konrad-Zuse-Zentrum für Informationstechnik, Berlín, 1992.

[21] Gendreau, M., Larochelle, J.-F., Sansò, B. A tabu search heuristic for the Steiner tree problem. Networks 1999; 34 (2):162-172.

[22] Esbensen, H. Computing near-optimal solutions to the Steiner problem in a graph using genetic algorithm. Networks 1995; 26, 173-185.

[23] Voss, S. and Gutenschwager, K. A chunking based genetic algorithm for the Steiner tree problem in graphs. In Pardalos, P.M., Du, D., (Eds.) Network Design: Connectivity and Facilities Location, DIMAC series in Discrete Mathematics and Theoretical Computer Science 40, AMS, Providence, 1999. p. 335-355.

[24] Koch, T. and Martin, A. Solving Steiner tree problems in graphs to optimality. Networks 1998; 32 (3) 207-232.

[25] Winter, P. Steiner Problem in Networks. Networks 1987; 17: 129-167.

[26] Cortés, P. , Larrañeta, J., Onieva, L., García , J.M. Multi-Injection Model to Solve the Steiner Problem. In Sforza, A., (Ed.) Simulation and Optimisation in Operations Management, Edizioni Scientifiche Italiane, Napoli, 1999.

[27] Montgomery, D.C. Design and Analysis of Experiments, John Wiley & Sons, Inc., 1991.

FIGURES

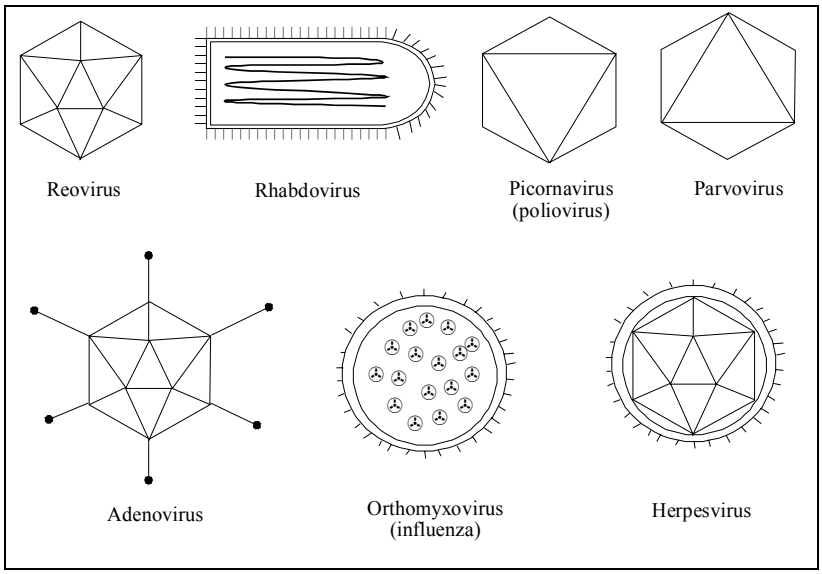

Figure 1. Some types of known viruses

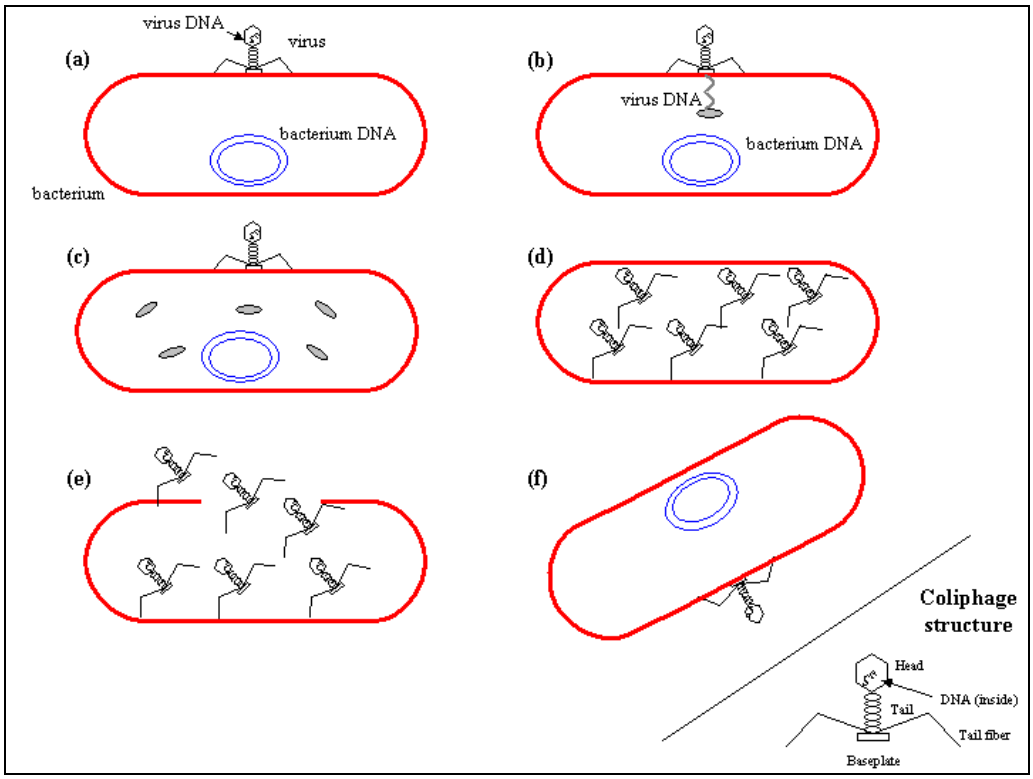

Figure 2. Lytic replication

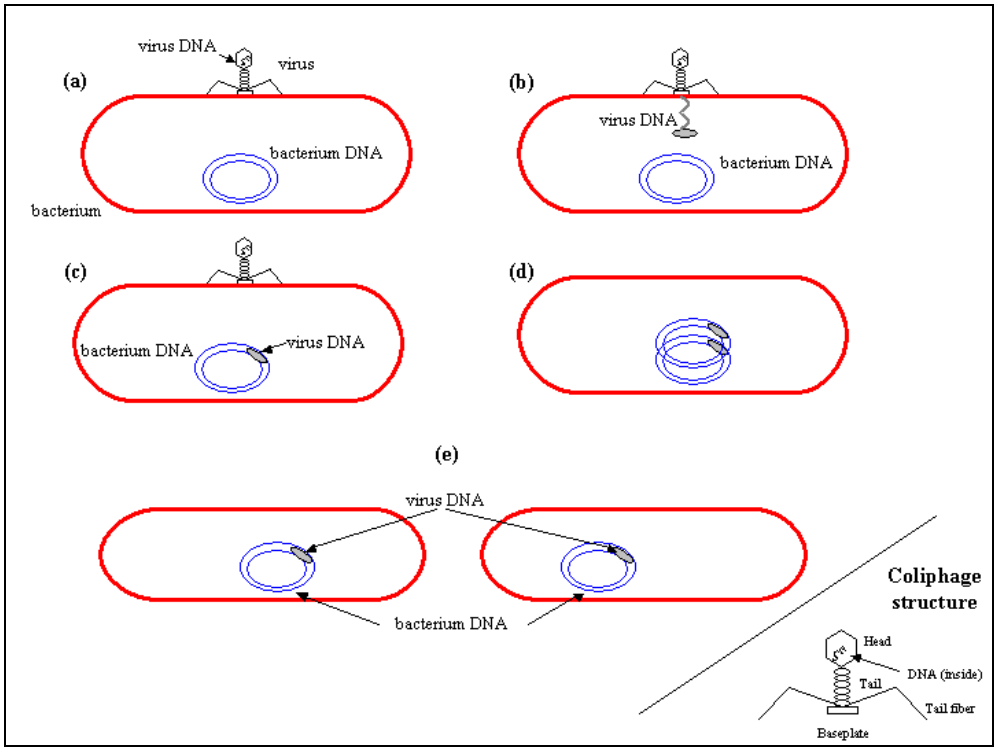

Figure 3. Lysogenic replication

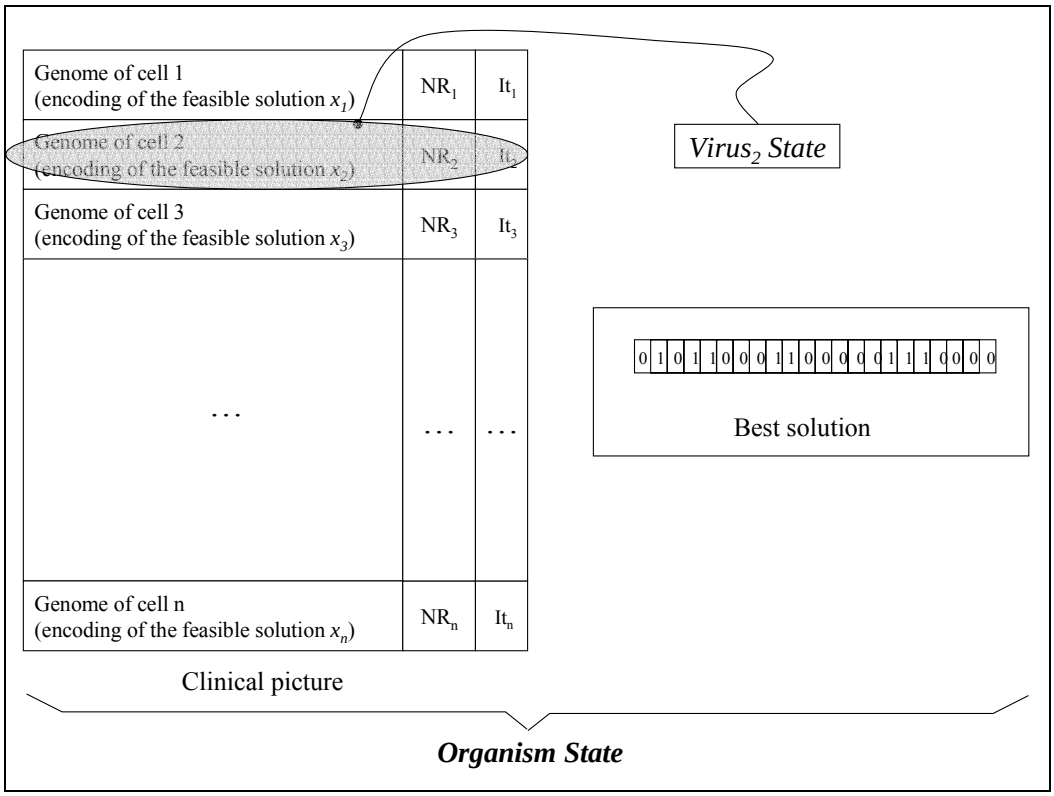

Figure 4. Organism and Virus State

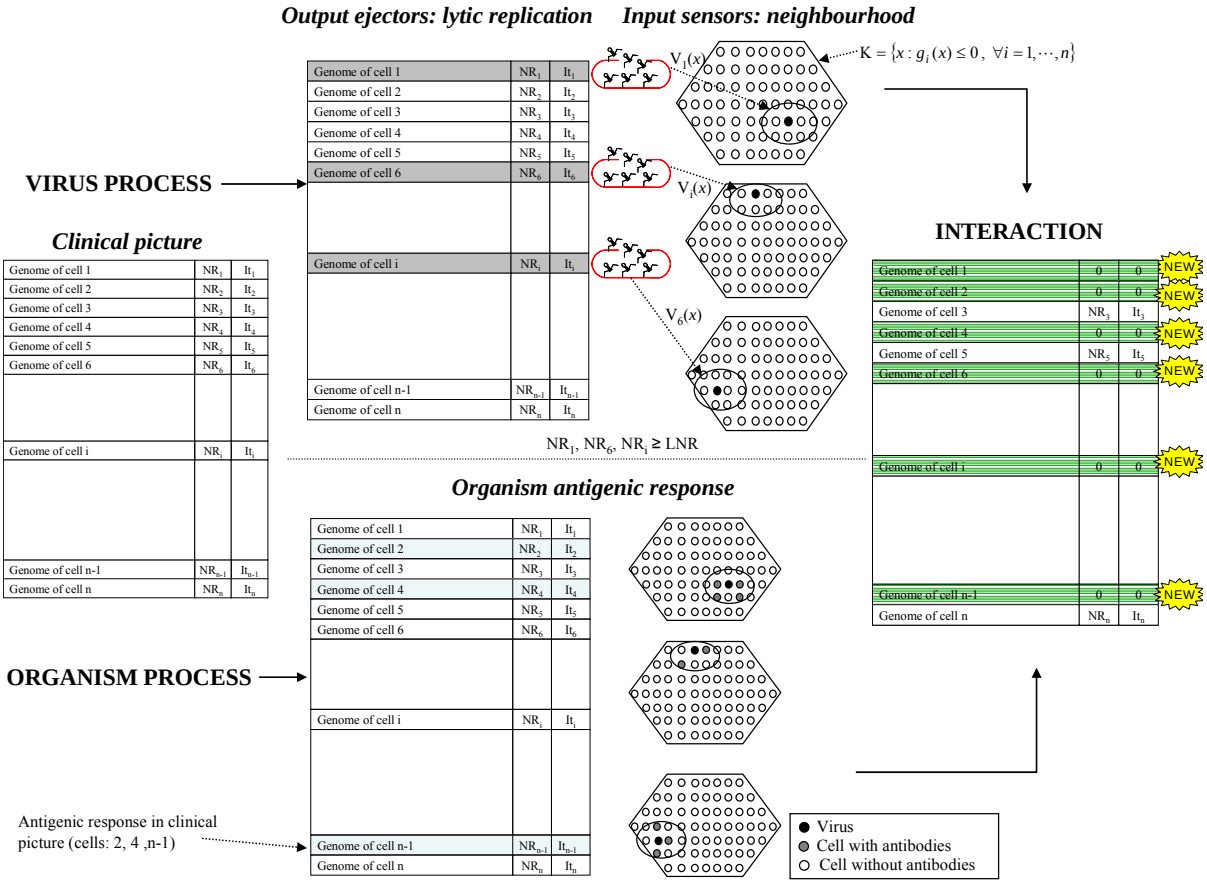

Figure 5. Algorithm evolution for lytic replication case in massive infection

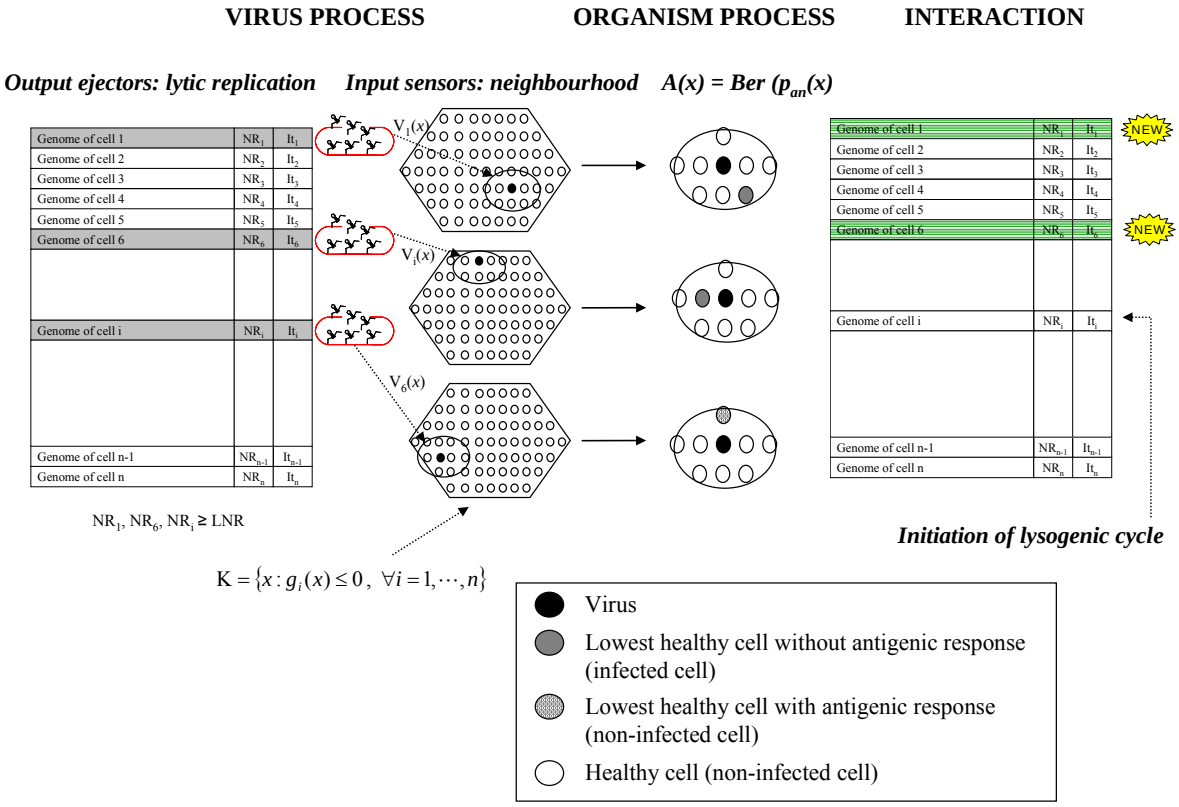

Figure 6. Algorithm evolution for lytic replication case in selective infection

TABLES

Table 1. States transition law for the clinical picture

State	Transition Law						
	0	1	2	…	J	…	LNR
0	$p_0 = \mathrm{P}(\mathrm{Z}=0)$	$p_1 = \mathrm{P}(\mathrm{Z}=1)$	$p_2 = \mathrm{P}(\mathrm{Z}=2)$	…	$p_j = \mathrm{P}(\mathrm{Z}=\mathrm{j})$	…	$p_{\mathrm{LNR}} = \mathrm{P}(\mathrm{Z}=\mathrm{LNR})$
1	0	$p_0 = \mathrm{P}(\mathrm{Z}=0)$	$p_1 = \mathrm{P}(\mathrm{Z}=1)$	…	$p_{j-1} = \mathrm{P}(\mathrm{Z}=\mathrm{j}\text{-}1)$	…	$p_{\mathrm{LNR}} + p_{\mathrm{LNR}-1}$
2	0	0	$p_0 = \mathrm{P}(\mathrm{Z}=0)$	…	$p_{j-2} = (\mathrm{Z}=\mathrm{j}\text{-}2)$	…	$p_{\mathrm{LNR}} + p_{\mathrm{LNR}-1} + p_{\mathrm{LNR}-}$
…	…	…	…	…	…	…	…
j	0	0	0	…	$p_0 = \mathrm{P}(\mathrm{Z}=0)$	…	$\sum_{k=0}^{j} p_{\mathrm{LNR}-k}$
…	…	…	…	…	…	…	…
LNR	1	0	0	…	0	…	0

Table 2. Least squares regression analysis for Stein-C set of problems

	Stein-C: group 1 Adjusted R-squared = 0.92115 Standard error = 0.90721662E-03			**Stein-C: group 2** Adjusted R-squared = 0.93107 Standard error = 0.74477109E-03			**Stein-C: group 3** Adjusted R-squared = 0.98426 Standard error = 0.18575403E-03		
Parameter	Coefficient	t-ratio	P[|T|>t]	Coefficient	t-ratio	P[|T|>t]	Coefficient	t-ratio	P[|T|>t]
ITER	*-.1627359097E-01*	*-17.938*	*.0000*	*-.1273674287E-01*	*-17.102*	*.0000*	*-.6201052638E-02*	*-33.383*	*.0000*
POB	*.6196468609E-02*	*6.830*	*.0000*	*.5878508007E-02*	*7.893*	*.0000*	*.2075200906E-02*	*11.172*	*.0000*
PLITI	*-.9859702694E-02*	*-10.868*	*.0000*	*-.1150348920E-01*	*-15.446*	*.0000*	*-.7409459177E-02*	*-39.889*	*.0000*
LNR	-.3670145379E-03	-.405	.6879	-.2338535169E-03	-.314	.7551	*.2033251897E-03*	*1.095*	*.2799*
LIT	*.6165346813E-02*	*6.796*	*.0000*	*.1751795999E-02*	*2.352*	*.0234*	*.4951710153E-03*	*2.666*	*.0109*
PZ	-.1663867492E-03	-.183	.8554	-.2691466718E-04	-.036	.9713	-.5265526171E-04	-.283	.7782
ITER_POB	*-.7003059497E-02*	*-7.719*	*.0000*	*-.5167419501E-02*	*-6.938*	*.0000*	*-.1364551560E-02*	*-7.346*	*.0000*
ITER_PLI	*.8558958037E-02*	*9.434*	*.0000*	*.9824820706E-02*	*13.192*	*.0000*	*.5792975690E-02*	*31.186*	*.0000*
ITER_LNR	.7465536683E-04	.082	.9348	.2547066287E-03	.342	.7341	.6346707768E-04	.342	.7343
ITER_LIT	*-.6091337370E-02*	*-6.714*	*.0000*	*-.1309931377E-02*	*-1.759*	*.0859*	*-.2162974302E-03*	*-1.164*	*.2508*
ITER_PZ	.1520681087E-03	.168	.8677	.4128354307E-04	.055	.9561	-.7542575442E-04	-.406	.6868
POB_PLIT	*-.3592776392E-02*	*-3.960*	*.0003*	*-.4961598593E-02*	*-6.662*	*.0000*	*-.1731482463E-02*	*-9.321*	*.0000*
POB_LNR	-.7503267980E-03	-.827	.4129	-.5292303695E-04	-.071	.9437	.9390983973E-05	.051	.9599
POB_LIT	*.2544590416E-02*	*2.805*	*.0076*	.3891250218E-03	.522	.6041	*.2019797493E-03*	*1.087*	*.2831*
POB_PZ	-.6631948513E-03	-.731	.4688	.1516634356E-03	.204	.8396	-.6409390469E-04	-.345	.7318
PLITI_LN	.2567051887E-03	.283	.7786	.2020066744E-03	.271	.7875	-.1561495836E-03	-.841	.4053
PLITI_LI	*-.1712201851E-02*	*-1.887*	*.0660*	*-.1237158692E-02*	*-1.661*	*.1041*	*-.3410437225E-03*	*-1.836*	*.0734*
PLITI_PZ	-.6425612504E-03	-.708	.4827	.1878275183E-03	.252	.8021	.1433866844E-03	.772	.4445
LNR_LIT	-.8883046612E-03	-.979	.3331	-.2012397331E-03	-.270	.7883	.8122341766E-04	.437	.6642
LNR_PZ	-.6213004108E-03	-.685	.4972	.3794939438E-04	.051	.9596	.1411620695E-03	.760	.4515
LIT_PZ	.2776950605E-04	.031	.9757	.9490056945E-04	.127	.8992	-.1452575337E-03	-.782	.4386

Table 3. Adequate selection of parameters for each group of problems in the Steiner collection

	Group 1		Group 2		Group 3	
Parameters	1st set	2nd set	1st set	2nd set	1st set	2nd set
ITER	50,000 (+)	10,000 (-)	50,000 (+)	50,000 (+)	10,000 (-)	50,000 (+)
POB	100 (+)	50 (-)	100 (+)	50 (-)	50 (-)	50 (-)
PLITI	0.7 (+)	0.7 (+)	0.7 (+)	0.7 (+)	0.7 (+)	0.7 (+)
LNR	15 (~)	15 (~)	15 (~)	15 (~)	10 (-)	10 (-)
LIT	10 (-)	10 (-)	20 (+)	10 (-)	10 (-)	10 (-)
Pz	0.5 (~)	0.5 (~)	0.5 (~)	0.5 (~)	0.5 (~)	0.5 (~)

Table 4. Comparison on the solution quality for the reduced C graphs of the Steiner collection

Problem	Optimum	GA-E	GA-V	MPH	P-Tabu	F-Tabu	VS
C1	106	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C2	220	1,67%	0,83%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C3	1565	0,13%	0,13%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C4	1935	0,11%	0,04%	0,09%	**0,00%**	**0,00%**	**0,00%**
C5	3250	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C6	67	0,73%	1,09%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C7	103	1,76%	2,75%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C8	1072	0,63%	0,51%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C9	1448	1,05%	1,30%	0,99%	0,14%	0,14%	**0,00%**
C10	2110	0,26%	0,27%	0,09%	**0,00%**	**0,00%**	**0,00%**
C11	29	1,88%	1,88%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C12	42	1,30%	0,43%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C13	500	1,01%	1,32%	0,78%	**0,00%**	**0,00%**	**0,00%**
C14	667	0,87%	0,68%	1,24%	0,31%	0,31%	**0,00%**
C15	1116	0,25%	0,22%	0,18%	**0,00%**	**0,00%**	**0,00%**
C16	13	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C17	23	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
C18	223	0,71%	0,71%	5,31%	0,88%	**0,00%**	**0,00%**
C19	310	0,41%	0,82%	4,79%	0,68%	**0,00%**	**0,00%**
C20	537	**0,00%**	**0,00%**	0,37%	**0,00%**	**0,00%**	**0,00%**
Best approach		5	5	11	16	18	20

Table 5. Comparison on the solution quality for the reduced D graphs of the Steiner collection

Problem	Optimum	GA-E	GA-V	MPH	P-Tabu	F-Tabu	VS
D1	106	0,57%	0,88%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D2	220	**0,00%**	0,73%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D3	1565	0,92%	1,25%	0,77%	0,26%	0,06%	**0,00%**
D4	1935	0,52%	0,63%	0,16%	**0,00%**	**0,00%**	**0,00%**
D5	3250	0,12%	0,19%	0,06%	**0,00%**	**0,00%**	**0,00%**
D6	67	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D7	103	1,94%	3,62%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D8	1072	1,55%	2,28%	1,59%	0,47%	**0,37%**	0,47%
D9	1448	0,50%	1,15%	0,83%	0,41%	**0,21%**	0,69%
D10	2110	0,13%	0,44%	0,38%	**0,00%**	**0,00%**	**0,00%**
D11	29	2,07%	1,84%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D12	42	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D13	500	0,56%	1,48%	2,00%	**0,00%**	**0,00%**	**0,00%**
D14	667	0,30%	0,75%	0,75%	**0,15%**	**0,15%**	**0,15%**
D15	1116	0,16%	0,39%	0,36%	**0,00%**	**0,00%**	**0,00%**
D16	13	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D17	23	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
D18	223	1,26%	1,43%	6,28%	1,35%	**0,90%**	**0,90%**

D19	310	1,03%	1,20%	5,81%	0,65%	**0,32%**	0,65%
D20	537	0,15%	0,16%	0,19%	**0,00%**	**0,00%**	0,37%
Best approach		5	4	8	15	19	16

Table 6. Comparison on the solution quality for the reduced E graphs of the Steiner collection

Problem	Optimum	GA-E	MPH	P-Tabu	F-Tabu	VS
E1	111	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E2	214	0,93%	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E3	4013	**0,00%**	1,07%	0,42%	0,32%	0,24%
E4	5101	0,02%	0,18%	**0,00%**	**0,00%**	**0,00%**
E5	8128	**0,00%**	0,02%	**0,00%**	**0,00%**	**0,00%**
E6	73	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E7	145	**0,00%**	2,07%	2,07%	**0,00%**	**0,00%**
E8	2640	**0,23%**	1,63%	0,49%	0,42%	1,14%
E9	3604	0,19%	1,17%	0,42%	**0,14%**	0,47%
E10	5600	**0,00%**	0,21%	0,04%	0,04%	0,14%
E11	34	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E12	67	1,49%	1,49%	1,49%	1,49%	**0,00%**
E13	1280	0,70%	1,88%	0,78%	**0,63%**	1,33%
E14	1732	**0,23%**	1,04%	0,29%	**0,23%**	0,64%
E15	2784	**0,00%**	0,22%	0,11%	0,11%	**0,00%**
E16	15	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E17	25	**0,00%**	**0,00%**	**0,00%**	**0,00%**	**0,00%**
E18	564	3,37%	7,62%	2,66%	**1,60%**	2,66%
E19	758	1,26%	4,35%	1,19%	1,19%	**1,18%**
E20	1342	**0,00%**	0,67%	**0,00%**	**0,00%**	0,15%
Best approach		12	6	9	14	12

Table 7. Computational times (in seconds)

Set of Problems	Group	Best Case	Worst Case
Stein-C	1	1	2
Stein-C	2	71	205
Stein-C	3	54	112
Stein-D	1	4	13
Stein-D	2	1,471	4,329
Stein-D	3	166	750
Stein-E	1	6	33
Stein-E	2	4,692	19,892
Stein-E	3	480	3,292