A CLONAL SELECTION ALGORITHM TO INVESTIGATE DIVERSE SOLUTIONS FOR THE RELIABLE SERVER ASSIGNMENT PROBLEM

Abdullah Konak and Sadan Kulturel-Konak
Penn State Berks, PA, Reading, U.S.A.

Keywords: Network reliability, Reliable server assignment, Clonal selection algorithm, Artificial immune systems.

Abstract: A reliable server assignment (RSA) problem in networks is defined as determining a deployment of identical servers on a network to maximize a measure of service availability. In this paper, a simulation optimization approach is introduced based on a Monte Carlo simulation and embedded into a Clonal Selection Algorithm (CSA) to find diverse solutions for the RSA problem, which is important in simulation optimization. The experimental results show that the simulation embedded-CSA is an effective heuristic method to discover diverse solutions to the problem.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

The reliable server assignment (RSA) problem on networks is described as assigning identical servers to network nodes to maximize a measure of service availability under a budget constraint. Given an undirected network $G=(V, E)$, where $V=\{1,\dots,n\}$ is the node set, $E=\{(i, j)\}$ is the edge set, the RSA problem is expressed as

$$\begin{aligned}&\max \quad z = R(\{s_1,\dots,s_n\},G)\\ &\sum_{i=1}^{n} c_i s_i \le C\\ &s_i \in \{0,1\}\end{aligned}$$

where s_i is the binary server assignment decision variable indicating whether a server is assigned to node i (s_i=1) or not (s_i=0), c_i is the cost of deploying a server at node i, and $R()$ is a measure of the availability of a service provided by identical servers.

The definition of service availability $R()$ depends on the nature of the service is provided by a network. For example, a computer network will not function properly if the servers providing the Domain Name System (DNS) service to the network are inaccessible. Therefore, DNS servers are replicated over a network to ensure their availability. In this paper, the critical service rate (CSR), which was first defined in (Kulturel-Konak and Konak, 2009), is used as the objective function. The CSR quantifies a network's ability to continue providing a service in the case of catastrophic edge or node failures. In a catastrophic failure mode, the edges and nodes are assumed to be in only two states, operative or failed. Let $r_{(i,j)}$ be the reliability of edge $(i, j)\in E$ and r_i be the reliability of node $i\in V$. For computational tractability, edge and node failures are assumed to be state-independent. When edge (i,j) fails, the communication between nodes i and j is interrupted if it cannot be rerouted through an alternative path. When a node fails, the network traffic cannot be routed through the node.

For a given server assignment $\mathbf{S}=\{s_1,\dots,s_n\}$, CSR($\mathbf{S}$) is defined as the probability that more than a predetermined fraction (α) of the operational nodes have access to at least one server in case of a component failure as follows:

$$\mathrm{CSR}(\mathbf{S}) = \Pr\{\frac{\sum_{i\in V}\delta_i(\mathbf{X}\mid\mathbf{S})}{\sum_{i\in V}v_i(\mathbf{X})} \ge \alpha\} \qquad (1)$$

where $\mathbf{X}$ denotes a binary state vector of the network such that at least one node is operational, $\delta_i(\mathbf{X}|\mathbf{S})$=1 if there exists at least one path between node i and a server node ($\delta_i(\mathbf{X}|\mathbf{S})$=0 otherwise), $v_i(\mathbf{X})$=1 if node i is operational in state $\mathbf{X}$ ($v_i(\mathbf{X})$=0 otherwise), and α is the critical service level.

The RSA problem as described in this paper was first defined in (Kulturel-Konak and Konak, 2009), and an Ant Colony Optimization (ACO) algorithm

156 Konak A. and Kulturel-Konak S..
A CLONAL SELECTION ALGORITHM TO INVESTIGATE DIVERSE SOLUTIONS FOR THE RELIABLE SERVER ASSIGNMENT PROBLEM.
DOI: 10.5220/0003644101560161
In *Proceedings of the International Conference on Evolutionary Computation Theory and Applications* (ECTA-2011), pages 156-161
ISBN: 978-989-8425-83-6
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

was developed to solve the problem. (Kulturel-Konak and Konak, 2010) also applied a simulation embedded-Particle Swarm Optimization (PSO) algorithm to the problem with very promising results. Recently, three nature-inspired heuristics, namely, ACO, PSO, and Clonal Selection Algorithm (CSA), were compared in (Konak and Kulturel-Konak, 2011) in terms of their performances and convergence properties. In this paper, a CSA is applied to the problem with an emphasis to discover not only good but also diverse solutions.

In general networks, exactly calculating CSR(**S**) is not practical in reasonable CPU times because its evaluation is NP-hard as it is the case in many network reliability analysis problems (Ball, 1980). Therefore, solutions searched by the CSA are evaluated by a Monte Carlo (MC) simulation. The CSA, the MC simulation, and a hashing method are integrated in a way to minimize the computational effort to efficiently evaluate candidate solutions and to reduce the effect of the noise in the objective function due to the MC simulation.

When simulation is used to evaluate alternative solutions in a heuristic algorithm, it is important for the algorithm to return multiple solutions because the evaluation of solutions includes error due to simulation. In the context of the RSA problem, decision makers wish to have a set of alternative server assignments with high levels of service availability. This observation is motivated by the fact that in most real-life reliability optimization problems, component reliabilities are not hard facts, but estimations based on historical observations. Therefore, providing decision makers with a set of alternative solutions allows them to determine the final design by weighing other factors in addition to system availability. The CSA in this paper returns a set of solutions instead of a single best solution. The CSA is compared with two previous meta-heuristics, namely the ACO and the PSO, with respect to its ability to discover good solutions with diverse structures. The computational results show that the CSA can discover more diverse set of solutions than the ACO and PSO while its performance is par with them.

2 A CSA ALGORITHM FOR THE RSA PROBLEM

Artificial Immune Systems (AIS) are relatively new bio-inspired computational algorithms. The early theoretical immunology work (Farmer et al., 1986); (Perelson, 1989); (Varela et al. 1988) has prepared the origins for AIS. There has been an increasing interest in AIS approaches to different problems, such as scheduling, image processing, bio informatics etc., after early applications on machine learning (Dasgupta, 1998); (Dasgupta and Nino, 2009); (Hart and Timmis, 2009).

One of the population based AIS algorithms, namely CSA, is applied to solve the RSA problem in this paper. The first CSA was developed based on the foundational clonal selection theory of Burnet (1959). The basic immunological components are: maintaining a specific memory set, selecting and cloning most stimulated antibodies, removing poorly stimulated or non-stimulated antibodies, hypermutating (i.e., affinity maturation) activated immune cells and generating and maintaining a diverse set of antibodies (Dasgupta, 1998). The principle of the theory is mainly as follows: The antigen (i.e., the foreign molecule that the immune system is defending against) selects the lymphocytes (i.e., B-cells or white blood cells that detect and stop antigens) with receptors which are capable of reacting with a part of the antigen. The rapid proliferation of the selected cell occurs during the selection to combat the invasion (i.e., clonal expansion and production of antibodies). While duplicating the cells, copying errors occur (i.e., somatic hypermutation) resulting in an improved affinity of the progeny cells receptors for the triggering antigen. CSA has been applied to many engineering and optimization problems. A basic CSA has been proposed by De Castro and von Zuben (2000) and later renamed as CLONALG by De Castro and von Zuben (2002), and they also provide the extensive analysis on the algorithmic computational complexity. In the general CLONALG first, antibodies (i.e., candidate solutions) are selected based on affinity either by matching against an antigen pattern or via evaluation of a pattern by an objective function (affinity means objective function value in optimization problems). Then, selected antibodies are cloned proportional to their affinity, and each clone is subject to a hypermutation inversely proportional to its affinity. Finally, the resulting clonal-set competes with the antibody population to survive for membership in the next generation, and low-affinity population members are replaced with randomly generated antibodies (Brownlee, 2007). The main steps of CLONALG are as follows:

```
Initialization
While (stopping criteria is not) {
Antigen Selection
```

```
    Exposure and Affinity Evaluation
    Clonal Selection
    Affinity Maturation (Hypermutation)
    Metadynamics (Replacement)
    }
    Termination
```

2.1 Solution Representation and Hyper Mutations

In the CSA in this paper, a solution j is represent by binary server assignment vector $\mathbf{S}_j=\{s_{j1},\ldots,s_{jn}\}$ such that s_{ji} indicates whether a server is assigned to node i ($s_{ji}=1$) or not ($s_{ji}=0$).

In the CSA, each solution is cloned and mutated according to its rank in the population such that a higher-ranked solution produces a higher number of clones than a lower-ranked solution, and it is mutated at a lesser degree. To achieve this, solutions in the population are sorted and ranked into three equal sized tiers as top 1/3, middle 1/3, and bottom 1/3 based on their objective functions. Each solution at the top, middle, and bottom tiers respectively produces $\mu/2$, $\mu/3$, and $\mu/4$ clones which are mutated by bitwise invert operators based on its tier. This ensures that solutions with better objective function values will produce a higher number of clones. On the other hand, each clone is mutated at a rate inversely proportional to its tier. Therefore, three different invert mutation operators are applied to the clones of the three tiers as follows. Top tier clones are perturbed by inverting one or two decision variables (Mutation1), middle tier clones by three decision variables (Mutation2), and bottom tier clones by four decision variables (Mutation3). In other words, new solutions created from top tier solutions are similar to their parents, and solutions created from bottom tier solutions are much more different than their parents. Top tier solutions are promising; therefore, a local search around these solutions is justified. On the other hand, bottom tier solutions are probably in non-promising regions of the search space. Therefore, moving away from those regions by large perturbations is desired. The mutation operators of the CSA are as follows:

```
Procedure Mutation1 ($\mathbf{S}_j$) {
    Randomly select two nodes $i1$ and $i2$ such that $s_{ji1}+s_{ji2} \le 1$
    If $s_{ji1}+s_{ji2}=1$, then flip the values of $s_{ji1}$ and $s_{ji2}$ to obtain new solution $\mathbf{S}$
    If $s_{ji1}+s_{ji2}=0$, then flip the value of $s_{ji1}$ to obtain new solution $\mathbf{S}$
    Return $\mathbf{S}$
}
Procedure Mutation2 ($\mathbf{S}_j$) {
    Randomly select three nodes $i1$, $i2$, and $i3$ such that $1\le s_{ji1}+s_{ji2}+s_{ji3}\le 2$
    Flip the values of $s_{ji1}$, $s_{ji2}$, and $s_{ji3}$ to obtain new solution $\mathbf{S}$
    Return $\mathbf{S}$
}
Procedure Mutation3 ($\mathbf{S}_j$) {
    Randomly select four nodes $i1$, $i2$, $i3$, and $i4$ such that $s_{ji1}+s_{ji2}+s_{ji3}+s_{ji4}=2$
    Flip the values of $s_{ji1}$, $s_{ji2}$, $s_{ji3}$, $s_{ji4}$ to obtain new solution $\mathbf{S}$
    Return $\mathbf{S}$
}
```

2.2 Solution Evaluation

As mentioned earlier, the objective function of the problem, CSR, is estimated using a crude MC simulation (Konak, 2009). In the crude MC simulation, a state vector $\mathbf{X}$ is sampled by individually sampling the state of each edge $(i, j) \in E$ and node $i \in N$ using Bernoulli random variables with means r_{ij} and r_i, respectively. To estimate CSR, K state vectors are sampled in this way, and the accessibility of servers by each operational node is checked for each state sampled. The ratio of the number of network states where critical service level α is achieved to the sample size (K) yields an unbiased estimator of CSR.

Evaluating candidate solutions using simulation within a heuristic algorithm has two important drawbacks. Firstly, simulation output includes a statistical error which might affect the performance of the algorithm. Secondly, simulation is computationally expensive, especially if a small margin of estimation error is required. In most population-based heuristics, as the search converges, same solutions will appear in the population with increasing frequencies. Therefore, computationally expensive simulation would be used to evaluate same solutions again and again in the case of the RSA problem. To address these problems, a hierarchical solution evaluation approach with hashing is used in the CSA.

All solutions investigated during the search are stored in a list, called solution list (SL). A hash table (HT) is used as a pointer to quickly access the solutions stored in SL. A second list, called collision list (CL) is used to store solutions with a hash collision. After a solution $\mathbf{S}$ is created, the hash value of the solution is calculated as follows:

$$d(\mathbf{S}) = \mathrm{mod}\left(\prod_{i=1}^{n} (e_i)^{s_i}, H\right) \qquad (2)$$

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

where H is the hash size and e_i is a prime number corresponding to decision variable s_i. Hash table HT is an integer array such that HT[d(**S**)]=0 if a solution with a hash value of d(**S**) has not been searched yet, or HT[d(**S**)]=t if the first solution with a hash value of d(**S**) is the t^{th} solution in solution list SL. In other words, SL[t] stores the t^{th} evaluated solution without any hash collision. After calculating d(**S**) for a solution **S**, there are three cases possible.

Case 1: If HT[d(**S**)]=0, then **S** has not been investigated before.

Case 2: If HT[d(**S**)]=t and **S**=SL[t], then **S** has been investigated before.

Case 3: If HT[d(**S**)]=t and **S**≠SL[t], then a hash collision occurs (i.e., two different solutions have the same hash value). In this case, **S** is compared with all solutions in collision list CL. If **S** is not in CL, then it has not been searched before and added to CL.

In the CSA, instead of a single best solution-found-so-far, the best b solutions found-so-far during the search are maintained in a sorted list called elitist list (EL) such that CSR(EL[1])> CSR(EL[2])>…> CSR(EL[b]). When a new solution is created, it is compared to previously evaluated solutions using the hashing technique used in (Kulturel-Konak and Konak, 2009) as briefly described above. If the solution has not been searched before, it is first is evaluated using a low number of simulation replications (K_1). After the first evaluation, the solution is identified as promising if the solution has a better estimated objective function value than the worst elitist solution in EL, and then it is rigorously evaluated using a higher number of replications (K_2). The statistical error due to simulation can be controlled by increasing the size of EL. After the search is terminated, all elitist solutions are evaluated one more time using a very high number of simulation replications (K_3). The solution evaluation procedure of the CSA is given below.

```
Procedure Evaluate_Solution (S) {
  If solution S has not been searched before {
    Evaluate S using $K_1$ simulation replications
    If (CSR(S)> the worst CSR in EL) {
      Evaluate S using $K_2$ simulation replications
      Update EL by including S if necessary
    }}}
```

2.3 The Overall CSA Algorithm

In this section, the overall procedure of the CSA algorithm is presented. Initially, μ solutions are randomly generated by a simple construction heuristics as follows:

```
Procedure Create_Random_Solution ($S_j$) {
  Set A=$V$, $C'=C$, $s_{ji}$ =0 for each node $i \in V$
  While (A≠{}) {
    Randomly and uniformly select node $i$ from A
    Set $s_{ji}$ =1 and $C'=C'-c_i$
    Set A=A\ {$i$} }
  Return solution $S_j$ }
```

The replacement strategy of the CSA is to replace the worst γ% of the population with randomly generated new solutions using Procedure *Create_Random_Solution*(), and it is applied once in every five iterations. Infeasible solutions are not evaluated and discarded because of the high computational cost of the MC simulation. The overall procedure of the CSA is as follows:

```
Procedure CSA {
  Randomly generate $\mu$ solutions
  While (stopping criterion is not satisfied) {
    Sort and rank the population
    If (replacement=TRUE) {
      Replace worst $\gamma$% of the population    with
        randomly generated solutions  }
    For each top tier solution $S_j$ do {
      Generate $\mu/2$ clones of $S_j$ using Mutation1($S_j$)
      Evaluate the generated clones
      Replace $S_j$ if a better solution is found }
    For each middle tier solution $S_j$ do {
      Generate $\mu/3$ clones of $S_j$ using Mutation2($S_j$)
      Evaluate the generated clones
      Replace $S_j$ if a better solution is found }
    For each bottom tier solution $S_j$ do {
      Generate $\mu/4$ clones of $S_j$ using Mutation3($S_j$)
      Evaluate the clones
      Replace $S_j$ if a better solution is found  }
    Simulate the elitist solutions using $K_3$
  replications and return them.
}
```

3 COMPUTATIONAL EXPERIMENTS

In (Konak and Kulturel-Konak, 2011), the CSA was compared with the previous PSO and ACO approaches with promising results, and it was reported that the performance of the CSA was par with the PSO in some cases and between the PSO and the ACO in most cases. Hence, there is no clear evidence to justify the CSA to solve the RSA problem. In (Dasgupta and Nino, 2009), CSA is recommended as an approach to multi-modal optimization problems where there are multiple optimal points in the solution space. To test this claim, we designed test problems with multiple optimal solutions and investigated whether the three heuristics can discover these multiple optimal solutions in a single run. Not that in simulation optimization approaches such as the CSA in this paper, it is particularly important to discover a set of diverse solutions because of the noise in solution evaluation.

All test problems have ring topologies with identical node reliabilities and costs (0.99 and 1, respectively), as well as identical edge reliabilities. The test problems were solved for C=5 and α=0.5. In other words, the test problems were defined as how to locate five servers on a ring network. Since all nodes have identical reliabilities and costs, and in addition all edges have the same reliability, there are many alternatives with the same optimal CSR. For example, for n=30 two solutions where five servers are located at nodes {6, 12, 18, 24, 30} and {7, 13, 19, 25, 1}, respectively, have the same CSR. To minimize the sampling error, simulation parameters were set as $K_1=10^4$, $K_2=2\text{x}10^4$, $K_3=10^6$, H=99001, and |EL|=20. Each problem was solved for ten random replications with the CSA parameters μ=50 and γ=20%. The search was terminated after 8000 solutions were searched (the stopping criterion). The worst 20% of the population was replaced with randomly generated solutions once in every five iterations. The ACO and PSO were run with the same algorithm parameters given in (Konak and Kulturel-Konak, 2011) with the stopping criteria of 8000 maximum solutions searched.

The Hamming distances between each pair of the elitists solutions found by the ACO, PSO, and CSA were calculated, and the average Hamming distance (AHD) in the final EL was used as the indicator for the diversity of the solutions found by each heuristic as follows:

$$\text{AHD} = \frac{\sum\limits_{j=1,\ldots,|\text{EL}|} \sum\limits_{k<j} \sum\limits_{i=1,\ldots,n} |s_{ji} - s_{ki}|}{n \times |\text{EL}| \times (|\text{EL}|-1)/2} \tag{3}$$

Table 1 presents the averaged CSR of the best and worst elitist solutions found in ten replications and the AHD among the elitist solutions. For problems n=30, 40, 50, 60, and 70, the three heuristics performed equally well with respect to the best and worst elitist solutions. However, the CSA provided the highest average Hamming distance, particularly compared to the PSO. For problems n=80 and 100, the ACO did not perform as well as the other heuristics. The ACO and the CSA have the same average Hamming distance, but the CSA performs better. In Table 1, the column titled Normalized Average Hamming Distance (NAHD) gives the AHD divided by the difference between the best and worst elitist solution. It is clear that the CSA was able to discover the most diverse sets of elitist solutions while it performed as well as the PSO based on the results in Table 1. For problems n=80 and n=100, both ACO and CSA have the same level of diversity in the elitist list, but the CSA outperformed the ACO in these problems in terms of the objective function. These results are consistent with the claim in the CSA literature that CSA is a suitable heuristic for multi-modal optimization problems.

Figure 1 illustrates three best solutions found by the ACO and the CSA for a multi-modal problem with n=30. As seen the figure, the solutions found by the CSA are very different in terms of server assignments. For example, the best two solutions, {20, 11, 50, 40, 29} and {19, 10, 1, 40, 30} have only one common nodes, which is node 40. On the other hands, the best two solutions found by the ACO, {19, 8, 48, 38, 28} and {19, 9, 48, 38, 28}, have four common nodes. This example demonstrates the ability of the CSA to investigate diverse solutions.

Table 1: Results of multi-modal problems.

	ACO				PSO				CSA			
(n, m)	Best	Worst	AHD	NAHD	Best	Worst	AHD	NAHD	Best	Worst	AHD	NAHD
(30,30)	0.602	0.592	0.204	21.77	0.602	0.593	0.186	19.40	0.603	0.595	0.265	34.97
(40,40)	0.440	0.431	0.165	18.75	0.440	0.433	0.134	19.28	0.441	0.433	0.202	28.72
(50,50)	0.662	0.652	0.151	15.20	0.663	0.656	0.112	17.21	0.663	0.656	0.168	26.18
(60,60)	0.925	0.921	0.135	34.78	0.926	0.923	0.098	33.83	0.926	0.923	0.143	50.09
(70,70)	0.907	0.902	0.122	22.65	0.908	0.904	0.079	23.65	0.908	0.904	0.124	34.28
(80,80)	0.876	0.870	0.109	17.26	0.877	0.872	0.075	17.02	0.877	0.872	0.108	26.73
(100,100)	0.819	0.811	0.088	10.56	0.821	0.815	0.059	11.14	0.821	0.816	0.088	19.17

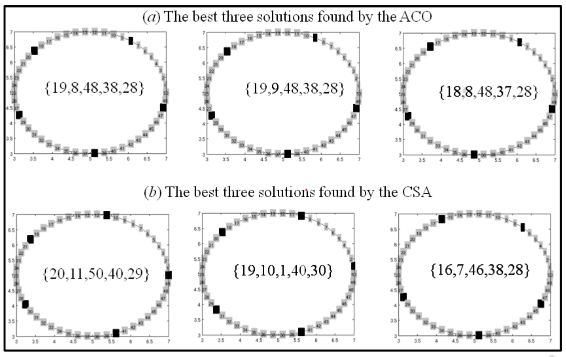

Figure 1: Best three solutions found by (*a*) the ACO and (*b*) the CSA for multi-model problems with *n*=30.

4 CONCLUSIONS

The reliable server assignment problem (RSA) in networks is studied in this paper. A simulation optimization approach is used based on a MC simulation and embedded into a CSA to solve the RSA problem. During the search, hashing method is used to rapidly detect whether a solution has been previously investigated or not to prevent costly evaluation process of same solutions again and again. The experimental study shows that the CSA algorithm is capable of searching very diverse solutions to the problem. This is an important concern in the RSA problem because very different solutions may have very similar system reliability, and decision makers can choose the best alternative by weighing in other factors.

REFERENCES

Ball, M., 1980. Complexity of Network Reliability Calculation. *Networks,* 10**,** 153-165.

Brownlee, J. (2007) Clonal Selection Algorithms. Victoria, Australia, Complex Intelligent Systems Laboratory (CIS), Swinburne University of Technology.

Burnet, F. M., 1959. The Clonal Selection Theory of Acquired Immunity, *Cambridge University Press*.

Dasgupta, D. (1998) An overview of Artificial Immune Systems and their applications. *Artificial immune systems and their applications.* New York, Springer-Verlag

Dasgupta, D. and Nino, L. P., 2009. Immunological Computation: Theory and Applications, *CRC Press*.

De Castro, L. N. and Von Zuben, F. J., 2000. The clonal selection algorithm with engineering applications. In: *Proceedings of GECCO'00*, Las Vegas, USA. 36-37,

De Castro, L. N. and Von Zuben, F. J., 2002. Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation,* 6 (3)**,** 239-251.

Farmer, J. D., Packard, N. H. and Perelson, A. S., 1986. The immune system, adaptation, and machine learning. In: *Physica D (Netherlands)*, Netherlands. 187-204.

Hart, E. and Timmis, J., 2009. Application areas of AIS: the past, the present and the future. *Applied Soft Computing Journal,* 8 (1)**,** 191-201.

Konak, A., 2009. Efficient event-driven simulation approaches to analysis of network reliability and performability. *International Journal of Modelling and Simulation,* 29 (Copyright 2010, The Institution of Engineering and Technology)**,** 156-168.

Konak, A. and Kulturel-Konak, S., 2011. Reliable Server Assignment in Networks Using Nature Inspired Metaheuristics. *IEEE Transactions on Reliability* 60 (2)**,** 381-393.

Kulturel-Konak, S. and Konak, A., 2009. Reliable network server assignment using an ant colony approach. In: *Joint ESREL (European Safety and Reliability) and SRA-Europe (Society for Risk Analysis Europe) Conference*, September 22, 2008 - September 25, 2008, Valencia, Spain. 2657-2663,

Kulturel-Konak, S. and Konak, A., 2010. Simulation optimization embedded particle swarm optimization for Reliable Server Assignment. In: *Winter Simulation Conference (WSC), Proceedings of the 2010*, 5-8 Dec. 2010 2897-2906,

Perelson, A. S., 1989. Immune network theory. *Immunological Reviews,* 110**,** 5-36.

Varela, F. J., Coutinho, A., Dupire, B. and Vaz, N., 1988. Cognitive networks: Immune, neural and otherwise. *Theoretical Immunology* 2**,**359-375.