# A Hybrid Hopfield Network-Genetic Algorithm Approach for the Terminal Assignment Problem

Sancho Salcedo-Sanz, *Member, IEEE* and Xin Yao, *Fellow, IEEE*

***Abstract*—This paper presents a hybrid Hopfield network-genetic algorithm (GA) approach to tackle the terminal assignment (TA) problem. TA involves determining minimum cost links to form a communications network, by connecting a given set of terminals to a given collection of concentrators. Some previous approaches provide very good results if the cost associated with assigning a single terminal to a given concentrator is known. However, there are situations in which the cost of a single assignment is not known in advance, and only the cost associated with feasible solutions can be calculated. In these situations, previous algorithms for TA based on greedy heuristics are no longer valid, or fail to get feasible solutions. Our approach involves a Hopfield neural network (HNN) which manages the problem's constraints, whereas a GA searches for high quality solutions with the minimum possible cost. We show that our algorithm is able to achieve feasible solutions to the TA in instances where the cost of a single assignment in not known in advance, improving the results obtained by previous approaches. We also show the applicability of our approach to other problems related to the TA.**

***Index Terms*—Genetic algorithms (GA), heuristics, Hopfield neural networks, terminal assignment (TA) problem.**

## I. Introduction

THE USE of telecommunication networks has increased significantly in the last decade, mainly due to the dramatic growth in the use of the Internet [1]. The reliability and quality of modern telecommunication service networks are critical in designing optimized networks, which meet the performance parameters. A large variety of combinatorial optimization problems have arisen not only in the design, but also in the management of communication networks [1], [2]. These new problems require, in many cases, the application of emergent optimization techniques for tackling them. One of these problems is the terminal assignment (TA) problem in a telecommunication network, on which this paper is focused.

TA is a NP-complete combinatorial optimization problem which arises in the design and management of telecommunications networks [3]–[5]. The objective of the TA involves determining minimum cost links to form a network by connecting a given set of terminals to a given collection of concentrators. The terminals have a known requirement of capacity to be assigned to a given concentrator, and this requirement varies from one terminal to another. Each concentrator has associated a given maximum capacity which limits the number of terminals it can handle. The capacity of all concentrators is also known.

TA involves identifying for each terminal the concentrator to which it should be assigned, subject to two constraints. First, each terminal must be assigned to one and only one of the concentrators, and second, the aggregate capacity requirement of the terminals connected to a given concentrator must not exceed the capacity of that concentrator.

Some interesting approaches for the TA can be found in the literature. Abuali *et al.* [3] proposed a greedy algorithm and a hybrid greedy-genetic algorithm (GA) for solving the TA. Khuri and Chui [4] proposed a GA with a penalty function as an alternative method for solving the TA. They showed its performance by means of the comparison with the greedy algorithm proposed in [3].

All the previous work on the TA provides powerful approaches when the cost of assigning a single terminal to a given concentrator is known before running the algorithms.[1] However, there are situations in which the cost of assigning a terminal to a concentrator cannot be known before having a feasible solution for the TA. For instance, consider a TA problem in which one of the aims of the network design is to have a balanced assignment of terminals to concentrators. In this case, the cost function depends on the entire solution provided to the communication network, and the cost of a single assignment cannot be calculated. Thus, some of the previous algorithms to the TA, such as the greedy algorithm in [3], are no longer applicable.

In this paper, we focus on TA instances where only the cost function of entire feasible solutions can be calculated, and where the cost of a single assignment cannot be known in advance. In this framework, heuristic approaches like GAs are an appealing option for achieving good solutions to the TA. The aim of this paper is twofold: first we present a novel hybrid Hopfield network (HNN) GA in which the problem's constraints are managed by the HNN and the quality of the solution obtained is improved by the GA; and second we apply this approach to the TA obtaining very good results and improving some previous heuristic algorithms in the literature.

Our algorithm combines the good performance of the HNN in solving problem's constraints with the ability of the GA for searching optimal solutions in combinatorial optimization problems. The use of the HNN reduces the search space of the GA to the space of feasible solutions. Note that this approach is applicable to other combinatorial optimization problems related to the TA. There are several of these problems in which our hybrid

Manuscript received January 26, 2004; revised June 2, 2004. The work of S. Salcedo-Sanz was supported by a postdoctoral fellowship from the Ministerio de Educación Cultura y Deporte of Spain, under Fellowship Number EX2003-0463. This paper was recommended by Associate Editor E. Santos.

The authors are with the School of Computer Science of the University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (e-mail: sancho@tsc.uc3m.es; sss@cs.bham.ac.uk).

Digital Object Identifier 10.1109/TSMCB.2004.836471

[1]For example, when the cost function is the Euclidean distance between a terminal and its associated concentrator.

approach can be applied, in the following sections we describe some of them.

The performance of our hybrid algorithm has been evaluated in several test TA problems and compared with the GA proposed in [4] with very good results in all test instances considered. We have also applied this hybrid heuristic to a problem directly related to TA, the task assignment problem, in order to show that our approach can be generalized to other problems.

The rest of the paper is structured in the following way. In the next section, we provide two equivalent formal definitions for the TA extracted from [3] and [4], respectively. We also give an illustration of the TA in a small problem and we show that the TA is directly related to other combinatorial optimization problems that have been previously solved in the literature. Section III provides a detailed overview of the most important existing approaches to the TA, whereas in Section IV we describe the scope of our paper and the proposed HNN-GA approach. Section V shows the TA test instances tackled and the results obtained by our algorithm, comparing the results with other GA's previously proposed in the literature. In this section we also include some results of our HNN-GA applied to the task assignment problem in an heterogeneous computer network. Finally, Section VI concludes the paper by giving some remarks and possible future work.

## II. Problem Definition

Given a set of

$$\begin{aligned} \text{Terminals}: &\quad l_1, l_2, \ldots, l_N \\ \text{Weights}: &\quad w_1, w_2, \ldots, w_N \\ \text{Concentrators}: &\quad r_1, r_2, \ldots, r_M \\ \text{Capacities}: &\quad p_1, p_2, \ldots, p_M \end{aligned}$$

where $w_i$ is the weight, or capacity requirement of terminal $l_i$. The weights and capacities are positive integers and $w_i < \min\{p_1, p_2, \ldots, p_M\}$ for $i = 1, 2, \ldots, N$. The N terminals and M concentrators are placed on the Euclidean grid, i.e., $l_i$ has coordinates $(l_{i1}, l_{i2})$ and $r_j$ is located at $(r_{j1}, r_{j2})$. The following are two formal equivalent definitions for the TA, which can be found in [3] and [4], respectively

*Problem Definition I:* Let $\mathbf{X}$ be a binary matrix such that for every element on it $x_{ij} = 1$ if terminal $i$ has been assigned to concentrator $j$, and $x_{ij} = 0$ otherwise.

Find $\mathbf{X}$ which minimizes

$$Z(\mathbf{X}) = \sum_{j=1}^{M} \sum_{i=1}^{N} \text{cost}_{ij} x_{ij} \tag{1}$$

subject to

$$\sum_{j=1}^{M} x_{ij} = 1, \qquad i = 1, 2, \ldots, N \tag{2}$$

$$\sum_{i=1}^{N} w_i x_{ij} \leq p_j, \qquad j = 1, 2, \ldots, M \tag{3}$$

where $Z$ is the cost of all of the links in the network and $cost_{ij}$ is the cost of assigning terminal $i$ to concentrator $j$. Note that the first constraint, from (2), ensures that each terminal must be associated with one and only one concentrator, and the second constraint from (3) implies that the capacity constraint on each concentrator cannot be violated.

*Problem Definition II:* Let $\tilde{\mathbf{x}} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N\}$ be a vector such that $\tilde{x}_i = j$ means that terminal $i$ has been assigned to concentrator $j$, $1 \leq \tilde{x}_i \leq M$ and $\tilde{x}_i$ is an integer.

Find $\tilde{\mathbf{x}}$ which minimize

$$Z(\tilde{\mathbf{x}}) = \sum_{i=1}^{N} \text{cost}_{ij}, \quad j = 1, 2, \ldots, M \tag{4}$$

subject to

$$\sum_{i \in R_j} w_i < p_j, \quad j = 1, 2, \ldots, M \tag{5}$$

where $\text{cost}_{ij}$ is the cost for associating terminal $i$ to concentrator $j$, and $R_j = \{i \,|\, \tilde{x}_i = j\}$, i.e., $R_j$ represents the terminals that are assigned to concentrator $j$.

### A. Analysis of Problem Definitions

We have presented two equivalent definitions for the TA, used in the literature. Problem Definition I codifies the TA into a binary matrix. These types of encodings are interesting for solving the problem by a classical GA [6] using traditional binary operators. On the other hand, this encoding method involves large binary strings for codifying the TA, because the solution matrix $\mathbf{X}$ is of size $N \times M$.

Problem Definition II codifies the problem in a more compact way, since the vector $\tilde{\mathbf{x}}$ has length N. On the other hand, this encoding involves integer numbers instead of binary ones, which may complicate the operators in a GA.

The transformation from one definition (representation) to another is straightforward:

If we have a matrix $\mathbf{X}$, the corresponding vector $\tilde{\mathbf{x}}$ can be calculated as

$$\tilde{x}_i = j, \quad \text{if } x_{ij} = 1. \tag{6}$$

If we have a solution for the TA encoded as a vector $\tilde{x}$, the corresponding binary matrix $\mathbf{X}$ can be calculated as

$$x_{ij} = \begin{cases} 1, & \text{if } \tilde{x}_i = j \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

### B. Example

Consider the TA defined by the collection of $N = 10$ terminals and $M = 3$ concentrators shown in Tables I and II. Following **Problem Definition I**, each assignment in this example will be represented by a $10 \times 3$ binary matrix $\mathbf{X}$, whereas following **Problem Definition II**, each assignment will be represented by a vector of ten integer numbers between 1 and 3. If we consider as cost function $cost_{ij}$ the Euclidean distance from terminal $i$ to concentrator $j$, the optimal assignment is shown in Fig. 1.

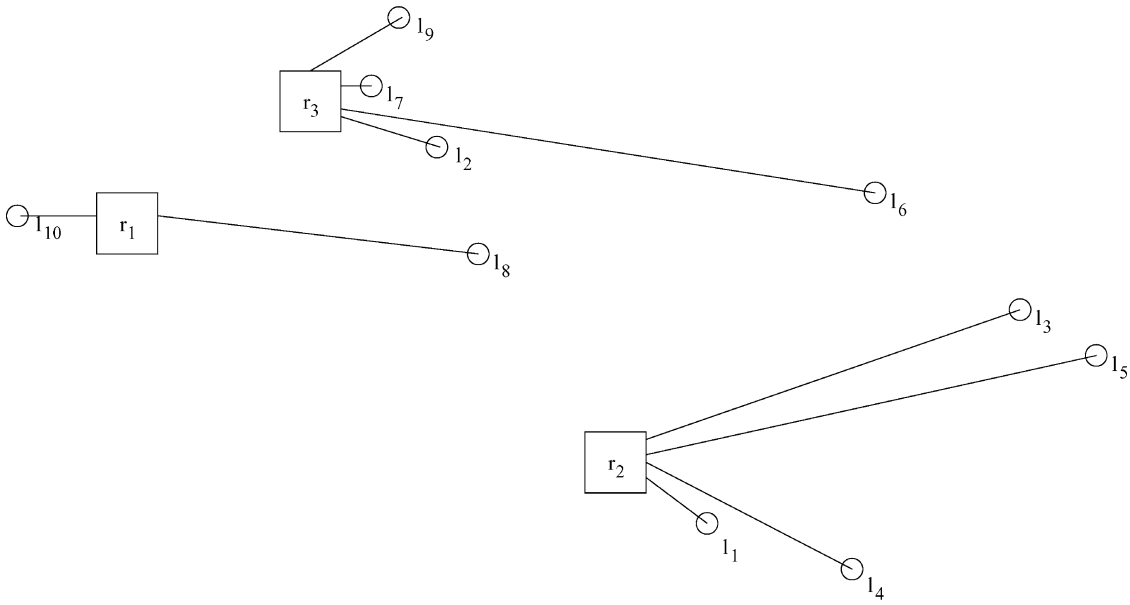



Fig. 1. Optimal TA for the example in Section II-C.

TABLE I
TERMINAL CAPACITY REQUIREMENTS (WEIGHTS) AND TERMINAL COORDINATES FOR THE PROBLEM IN SECTION II-C

| Terminal # | Weight | Coordinates |
|---|---|---|
| 1 | 5 | (54,28) |
| 2 | 4 | (28,75) |
| 3 | 4 | (84,44) |
| 4 | 2 | (67,17) |
| 5 | 3 | (90,41) |
| 6 | 1 | (68,67) |
| 7 | 3 | (24,79) |
| 8 | 4 | (38,59) |
| 9 | 5 | (27,86) |
| 10 | 4 | (07,76) |

This optimal assignment can be represented as a binary matrix.

$$\mathbf{X_o} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

or, as an integer vector:

$$\tilde{\mathbf{x}}_\mathbf{o} = \{2, 3, 2, 2, 2, 3, 3, 1, 3, 1\}.$$

TABLE II
CONCENTRATORS CAPACITY AND COORDINATES FOR THE PROBLEM IN SECTION II-C

| Concentrator # | Capacity | Coordinates |
|---|---|---|
| 1 | 12 | (19,76) |
| 2 | 14 | (50,30) |
| 3 | 13 | (23,79) |

### C. Related Problems

The TA is not only interesting because its applications to the design of fixed telecommunications networks, but also because it is directly related to a large amount of problems in very different contexts. First, note that the TA is very similar to a classical combinatorial optimization problem, known as the *bin packing* (BP) problem [7], [8]. BP is defined by means of a set of bins, $B = \{b_1, \ldots, b_N\}$ (concentrators in the TA), with associated capacity $c$ and a set of objects $O = \{o_1, \ldots, o_M\}$ (terminals in the TA) with associated weights $w_i$. A feasible solution would be represented by a vector $\mathbf{x} = \{x_1, \ldots, x_N\}$, where $x_i = j$ means that the $i^{th}$ object has been assigned to bin $j$. The goal of the bin packing problem is assigning the objects to the bins in such a way that

$$\sum_{i \in B_j} w_i \leq c$$

where $B_j = \{i | \ x_i = j\}$ represents the objects which are in bin $j$.

The objective function of the BP is usually the number of different bins required for obtaining a feasible solution to the problem. Note that this definition of the BP is similar to the definition II of TA given in this paper, and only the objective function is quite different from TA.

Another combinatorial optimization problem very close to the TA is the so called cell to switches assignment problem (CTAP). This problem consists of assigning the cells of a mobile telecommunications network to a set of switches, which route calls to another base station or telephone switched network. The optimal assignment of cells to switches allows a better management of the *handoff* between cells. The CTAP can be defined in a straight forward manner from the TA, changing terminals by cells, concentrators by switches and using an objective function which depends on the number of possible handoff from a given cell and its distance to the switch it is assigned. See [9] and [10] for further details on the CTAP.

The last problem related to the TA we consider in this small revision is the task assignment problem (TSAP) in computer networks with resources constraints. Let us consider a distributed computer network formed by a set of processors, and a set of tasks of a distributed application which must be executed on them. Each task requires some resources in order to be executed (memory for example) and each processor has a maximum of resources available for executing tasks. Thus, the TSAP can be obtained from the TA changing terminals by tasks and concentrators by processors, and including an objective which takes into account the total time for finishing the whole program (all the tasks) and the communication costs between tasks. More details about the TSAP can be found in [11]–[13].

## III. Previous Approaches

There are some previous approaches to the TA in the literature which used heuristic algorithms. One of the most important papers on TA was the approach by Abuali *et al.* [3]. In this article the authors proposed a greedy algorithm for solving the TA. This greedy approach used the notation in **Problem Definition I**, and started from a random permutation of terminals $\pi(l_N)$. They then considered the cost function $\mathrm{cost}_{\mathrm{ij}}$ as the Euclidean distance between terminal $i$ and concentrator $j$. The terminals were assigned to concentrators following the order in $\pi(l_N)$ in such a way that a terminal is allocated to the closest concentrator if there was enough capacity to satisfy the requirement of the particular terminal. If the concentrator could not handle the terminal, the algorithm searched for the next closest concentrator and performed the same evaluation. This process was repeated until an available concentrator was found and the algorithm continued to assign the remaining terminals, if there were any. In the case that no concentrator could accommodate the required capacity of a given terminal, the search was considered failed, and the solution provided by the greedy algorithm was not feasible.

*Pseudo-code of the Greedy algorithm in [3].*
Choose a permutation $\pi(l_N)$, at random.
**for**(each terminal $\pi(l_i)$)
Determine $\mathrm{cost}_{\mathrm{ij}}$, the distance from $\pi(l_i)$ to the closest feasible concentrator $r_j$.
Assign $\pi(l_i)$ to $r_j$.
**end(for)**

Abuali *et al.* in [3] also proposed a GA which looked for the permutation $\pi(l_N)$ which would provide the lowest cost $\mathrm{cost}_{\mathrm{ij}}$. The results provided in [3] showed that the GA hybridized with the greedy algorithm was a powerful approach to the TA.

Another relevant work on TA is due to Khuri and Chiu [4]. In this paper, the authors proposed a GA for the TA, also using the encoding shown in **Problem Definition I**. This GA used a penalty function for managing the TA constraints, and could include some initial solutions given by the greedy algorithm presented above in order to speed up the convergence of the algorithm to a good feasible solution.

## IV. Our Hybrid HNN-GA Approach

Existing algorithms in [3] and [4], provide nearly optimal solutions for the TA in the case that the cost $\mathrm{cost}_{\mathrm{ij}}$ of assigning a given terminal $i$ to a concentrator $j$, is known in advance; for example in the case that cost function $\mathrm{cost}_{\mathrm{ij}}$ is the Euclidean distance between terminal $i$ and concentrator $j$, as was considered in [3], [4]. However, there are situations in which $\mathrm{cost}_{\mathrm{ij}}$ is not known in advance, but only after a feasible solution is provided to the communication network. Consider, for example, the case in which $\mathrm{cost}_{\mathrm{ij}}$ is a function of the number of terminals assigned to a given concentrator.[2] In this situation, the greedy algorithm is no longer a valid option, since it needs to have the cost function values in order to generate a feasible solution for the TA. Note that this fact implies that some kind of blind search procedures are needed in order to obtain a solution to the problem. The GA proposed by Khuri and Chiu would be able to provide solutions in this case. However, it cannot be seeded with feasible solutions from the greedy algorithm, and its performance would be worse than shown in [4].

In this paper, we propose new heuristic algorithms for TA instances in which $\mathrm{cost}_{ij} = f(x_{ij})$, though our approach can also be applied to any other types of cost functions. $\mathrm{cost}_{\mathrm{ij}}$ needs not be known in advance. Specifically, we propose a hybrid neural-GA in which a Hopfield neural network manages the problem's constraints and a GA looks for good solutions in terms of $\mathrm{cost}_{\mathrm{ij}}$. In the following subsections we present the Hopfield neural network and two GAs with different encoding schemas for hybridization.

### A. Hopfield Neural Network

The Hopfield network [14] we use as a local search algorithm for solving the TA constraints belongs to a class of binary Hopfield networks [15] where the neurons can only take values 1 or 0. The dynamics of this network depends on a matrix $C$ which defines the minimum distance in rows between two 1s in the network, and on the initial state of the neurons. See [15] for further details. The structure of the HNN can be described as a graph, where the set of vertices are the neurons, and the set of edges define the connections between the neurons. We map a neuron to every element in the solution matrix $\mathbf{X}$. In order to simplify the notation, we shall also use matrix $\mathbf{X}$ to denote the neurons in the Hopfield network. The HNN dynamics can then be described in the following way: After a random initialization of

[2]This could be useful, for instance, if we are interested in assignments with a balanced number of terminals handled in concentrators.

every neuron with binary values, the HNN operates in a serial mode. This means that only one neuron is updated at a time, while the rest remain unchanged. Denoting by $x_{ij}(t)$ the state of a neuron at time $t$, the updating rule is described by

$$x_{ij}(t) = \text{isgnta}\left(\sum_{\substack{\text{p=1}\\ \text{p}\neq\text{i}}}^{\text{N}} \sum_{\substack{\text{q=max(1,c}_{\text{i,p+1}}\text{)}\\ \text{q}\neq\text{j}}}^{\text{min(M,j+c}_{\text{i,p}}\text{)}} \text{x}_{\text{pq}}\right) \quad \forall \text{ i, j.} \tag{8}$$

where the *isgnta* operator is defined by

$$\text{isgnta(a)} = \begin{cases} 0, & \text{if } a > 0 \text{ or } \sum_{i=1}^{N} w_i x_{ij} > p_j \quad \forall j \\ 1, & \text{otherwise} \end{cases}.$$

Note that the updating rule only takes into account neurons $x_{pq}$ with value 1 within a distance of $c_{ip}$. The matrix $C$ is an $N \times N$ matrix which encodes the problem's constraint given by (2). Its elements are defined as follows

$$c_{ij} = \begin{cases} M, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Note that this matrix forces one and only one 1 per row, whereas there may be several 1s in the same column.

In the updating rule defined above, the neurons $x_{ij}$ are updated in their natural order, i.e., $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, M$. We introduce a modification of this rule by performing the updating of the neurons in a random ordering of the rows (variable $i$). This way the variability of the feasible solutions found will increase. Let $\pi(i)$ be a random permutation of $i = 1, 2, \ldots, N$. The new updating rule of the HNN is

$$x_{\pi(i)j}(t) = \text{isgnta}\left(\sum_{\substack{\text{p=1}\\ \text{p}\neq\pi\text{(i)}}}^{\text{N}} \sum_{\substack{\text{q=max(1,c}_{\pi\text{(i),p+1}}\text{)}\\ \text{q}\neq\text{j}}}^{\text{min(M,j+c}_{\pi\text{(i),p}}\text{)}} \text{x}_{\text{pq}}\right) \forall \text{ i, j.} \tag{10}$$

The resulting updating rule runs over the rows of $\mathbf{X}$ in the order given by the permutation $\pi(i)$, but the columns are updated in natural order $j = 1, 2, \ldots, M$.

A *cycle* is defined as the set of $N \times M$ successive neuron updates in a given order. In a cycle, every neuron is updated once following the given order $\pi(i)$, which is fixed during the execution of the algorithm. After every cycle, the convergence of the HNN is checked. The HNN is considered converged if none of the neurons have changed their state during the cycle. The final state of the HNN dynamics is a potential solution for the TA, which fulfils the problem's constraints given by (2) and (3). Note, however, that the solution found may be unfeasible if not all the terminals are assigned.

### B. Genetic Algorithm I

The first genetic algorithm (GA I) we propose to be hybridized with the Hopfield network codifies a population of $\chi$ potential solutions for the TA as binary strings of length $N \times M$. Each string represents a different matrix $\mathbf{X}$, i. e. this GA uses the notation given in **Problem definition I**. The population is then evolved through successive generations by means of the application of the genetic operators selection, crossover and mutation [6]. The selection operator is the roulette wheel selection, in which individuals are randomly sampled with probabilities inversely proportional to their objective function values (in order to minimize the objective function). In this case, values of fitness (i.e., objective) are given by the problem's function cost $\text{cost}_{\text{ij}}$. An elitist strategy, which always passes the highest fitness string to the next generation, is applied in order to preserve the best solution encountered thus far in the evolution. We consider two-point crossover with probability $P_c$ and random flip mutation with probability $P_m$. Finally, since crossover and mutation operators may cause the new string to be unfeasible, this offspring string is set as the initial state of the Hopfield network, and the result of the neural algorithm substitutes it in the new population. In the case that even the solution found by the Hopfield network is unfeasible (some terminals not assigned), a penalty term is added to the cost function. The penalty function we use is similar to the one defined in [4] for managing the unfeasible solutions.

In pseudo-code, this algorithm for the TA can be written as follows:

```
Genetic algorithm I
Initialize GA population at random
while(max. number of generations has not
been reached) do
 for(every individual X)
  Run the HNN to obtain a feasible X.
  Calculate the fitness value of the indi-
vidual cost_ij.
  if X is not feasible, apply a penalty
to cost_ij.
  Substitute the GA individual by the new
X obtained through the HNN.
 endfor
  selection
  crossover
  mutation
end(while)
```

### C. Genetic Algorithm II

The second genetic algorithm (GA II) we investigate in this work encodes potential solutions for the TA as shown in **Problem Definition II**: every individual is a string of N integers between 1 and $M$, $\tilde{\mathbf{x}}$. The selection and crossover operators are applied in the same way as that in the **Genetic Algorithm I**. The mutation operator consists of substituting each position in the string with a different integer value, with a very small probability $P_m$. In order to come up with feasible solutions, every individual in the population is passed to the Hopfield network: First, the string of integers $\tilde{\mathbf{x}}$ is passed to a binary matrix $X$ using (7); once a feasible solution has been obtained by means of the Hopfield network, the resulting matrix $X$ matrix is passed back to a vector of integers $\tilde{\mathbf{x}}$ using (6). In the case that the solution found by the Hopfield network is unfeasible

(some terminals were not assigned) the corresponding position in the string is filled at random, and a penalty term is added to the cost function. In pseudo-code, this algorithm for the TA can be written as

```
Genetic algorithm II
Initialize GA population at random
while(max. number of generations has not
been reached) do
 for(every individual x̃)
  x̃ → X
  Run the HNN to obtain a feasible X.
  Calculate the fitness value of the indi-
vidual cost_ij.
  if X is not feasible, apply a penalty
to cost_ij.
  X → x̃
  Substitute the individual in the GA by
x̃.
 end(for)
  selection
  crossover
  mutation
end while
```

## V. Computational Experiments and Results

In order to test the performance of our approach, we tackle a set of TA instances of different sizes. Table III shows the main characteristics of these problems. There are 15 test instances, of different difficulties. In general, the difficulty increases with the problem size. The coordinates of terminals and concentrators have been randomly obtained in a $100 \times 100$ grid, whereas the weights associated with each terminal were randomly generated between 1–6. The capacities of each concentrator vary from one problem to another, being in a range between 10 and 15. In all instances considered, the cost function $\mathrm{cost_{ij}}$ has been defined in the following way:

$$\mathrm{cost_{ij}} = 0.9 \cdot \mathrm{bal_{ij}} + 0.1 \cdot \sum_{\mathrm{i=1}}^{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{M}} \mathrm{x_{ij} dist_{ij}} \qquad (11)$$

where $\mathrm{dist_{ij}}$ is the matrix of the Euclidean distances between a terminal $i$ and a concentrator $j$, and $\mathrm{bal_{ij}}$ is defined, as shown in (12) at the bottom of the page.

Note that this cost function encourages solutions with a balanced assignment of terminals into concentrators, and there is also a term in (11) which imposes that the addition of Euclidean distances between terminals and concentrator must be as small as possible. Note also that this cost function $\mathrm{cost_{ij}}$ cannot be known before a feasible solution is provided to the communication network.

TABLE III
Main Features of the TA Problems Tackled

| Problem # | Number of terminals | Number of concentrators | Total terminals weight | Total concentrators capacity |
|---|---|---|---|---|
| 1 | 10 | 3 | 35 | 39 |
| 2 | 10 | 3 | 39 | 42 |
| 3 | 10 | 3 | 34 | 37 |
| 4 | 20 | 6 | 77 | 83 |
| 5 | 20 | 6 | 61 | 68 |
| 6 | 20 | 6 | 72 | 79 |
| 7 | 30 | 10 | 117 | 127 |
| 8 | 30 | 10 | 98 | 120 |
| 9 | 30 | 10 | 94 | 120 |
| 10 | 50 | 17 | 182 | 204 |
| 11 | 50 | 17 | 174 | 193 |
| 12 | 50 | 17 | 173 | 204 |
| 13 | 100 | 30 | 292 | 360 |
| 14 | 100 | 30 | 334 | 360 |
| 15 | 100 | 30 | 342 | 360 |

We compare the results obtained by the hybrid HNN-GA, using the two encodings proposed in this paper, [3] with the performance of the GA proposed in [4]. The parameters of the GAs used are fixed: population size $\chi = 50$, roulette wheel selection, two-point crossover operator with a probability $P_c = 0.6$, probability of mutation $P_m = 0.01$, and maximum number of generations fixed to 1000. We run every algorithm 30 times, keeping the best, mean and standard deviation results obtained.

Table IV shows the results obtained on the problems considered, whereas Table V shows the results of a $t$-test performed over the data obtained by the three compared algorithms. It is apparent that our approaches Hybrid I and Hybrid II, perform equally or better than GA with penalty function in all cases. The GA with penalty function [4] was not able to obtain feasible solutions for problems from #10 to #15. Fig. 2 shows the best solutions found by our approach for problems #7, #8, and #9. Note that there is a good balance in the number of terminals assigned to concentrators, as expected.

### A. Discussion

The results provided in Tables IV and V show that the hybrid HNN-GA scheme has a very good performance for the TA. It is possible to extract further information through a detailed

[3]Hereafter, we call Hybrid I to the hybrid algorithm with binary encoding (GA I) and Hybrid II to the hybrid algorithm with integer encoding (GA II), see Sections IV-B and IV-C

$$\mathrm{bal_{ij}} = \begin{cases} 10, & \text{if } \sum_{i=1}^{N} x_{ij} = round\left(\frac{N}{M}\right) + 1 \ \ \forall j, \\ 20 \cdot abs\left(\mathrm{round}\left(\frac{\mathrm{N}}{\mathrm{M}}\right) + 1 - \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{x_{ij}}\right), & \forall j, \quad \text{otherwise.} \end{cases} \qquad (12)$$

TABLE IV
COMPARISON OF THE RESULTS OBTAINED BY THE DIFFERENT ALGORITHMS CONSIDERED (BEST/AVG./STD DEV.). SYMBOL – MEANS THAT THE GA WITH PENALTY FUNCTION WAS NOT ABLE TO ACHIEVE FEASIBLE SOLUTIONS IN THOSE PROBLEMS (#10-#15)

| Problem # | GA | Hybrid I | Hybrid II |
|---|---|---|---|
| 1 | 65.6/66.6/1.2 | 65.6/65.6/0.0 | 65.6/65.6/0.0 |
| 2 | 76.3/78.5/2.1 | 76.3/76.3/0.0 | 76.3/76.3/0.0 |
| 3 | 85.6/88.2/2.6 | 85.6/85.6/0.0 | 85.6/85.6/0.0 |
| 4 | 135.5/162.3/12.8 | 135.5/137.0/1.9 | 135.5/137.2/2.3 |
| 5 | 139.2/153.0/7.7 | 139.2/141.4/1.9 | 139.2/142.9/2.7 |
| 6 | 143.6/153.7/4.6 | 137.6/139.0/1.6 | 137.6/139.2/1.2 |
| 7 | 278.8/293.0/7.8 | 257.3/264.9/3.5 | 256.9/268.7/3.8 |
| 8 | 270.7/289.8/11.0 | 261.9/277.3/7.0 | 268.6/280.9/5.5 |
| 9 | 269.9/286.4/11.0 | 269.2/280.1/7.25 | 263.0/284.1/7.4 |
| 10 | - | 440.4/460.8/12.0 | 434.1/465.6/12.4 |
| 11 | - | 453.9/476.1/11.5 | 458.3/478.8/11.5 |
| 12 | - | 483.7/508.7/12.0 | 479.9/509.5/12.9 |
| 13 | - | 811.4/852.9/18.8 | 809.6/851.7/20.0 |
| 14 | - | 740.5/808.7/28.4 | 748.5/804.0/24.7 |
| 15 | - | 798.7/848.1/21.1 | 812.1/856.3/19.1 |

TABLE V
$t$ VALUES OBTAINED BY A TWO-TAILED t-TEST FOR PROBLEMS #1 TO #15. † STANDS FOR VALUES OF $t$ WITH 29 DEGREES OF FREEDOM WHICH ARE SIGNIFICANT AT $\alpha = 0.05$. SYMBOL – MEANS THAT THE GA WITH PENALTY FUNCTION WAS NOT ABLE TO ACHIEVE FEASIBLE SOLUTIONS IN THOSE PROBLEMS (#10-#15)

| Problem # | Hybrid I-GA | Hybrid II-GA | Hybrid I-Hybrid II |
|---|---|---|---|
| | $t$-test | $t$-test | $t$-test |
| 1 | $-3.23^{\dagger}$ | $-3.23^{\dagger}$ | 0.0 |
| 2 | $-4.50^{\dagger}$ | $-4.50^{\dagger}$ | 0.0 |
| 3 | $-5.31^{\dagger}$ | $-5.31^{\dagger}$ | 0.0 |
| 4 | $-10.74^{\dagger}$ | $-10.59^{\dagger}$ | $-0.37$ |
| 5 | $-8.03^{\dagger}$ | $-7.29^{\dagger}$ | $-2.27^{\dagger}$ |
| 6 | $-16.70^{\dagger}$ | $-16.62^{\dagger}$ | $-0.62$ |
| 7 | $-18.18^{\dagger}$ | $-15.27^{\dagger}$ | $-4.08^{\dagger}$ |
| 8 | $-5.39^{\dagger}$ | $-3.94^{\dagger}$ | $-2.32^{\dagger}$ |
| 9 | $-2.36^{\dagger}$ | $-1.09$ | $-1.66$ |
| 10 | - | - | $-1.64$ |
| 11 | - | - | $-0.91$ |
| 12 | - | - | $-0.28$ |
| 13 | - | - | 0.26 |
| 14 | - | - | 0.21 |
| 15 | - | - | $-1.41$ |

analysis of the algorithms considered. First, note that there exist small differences of performance between the Hybrid I and Hybrid II approaches. Table V shows that differences in performance between both hybrid algorithms are only statistically significant in three problems, where Hybrid I performs better than the Hybrid II algorithm. It is important to note that both hybrid algorithms are able to achieve feasible solutions for all the TA instances considered. Fig. 3 shows the percentage of convergence of the HNN to feasible solutions. These data are calculated by launching 1000 HNN's for each instance with random initialization. The HNN convergence rate is over 85% in all instances, and over 95% for Problems #2 and #14. This high convergence rate of the HNN makes the hybrid algorithm easier to find feasible solutions for all the problems considered.

The increasing computational cost is the main drawback when using a hybrid algorithm for a combinatorial optimization problem such as the TA. Thus, the design of the hybrid local and global algorithms should take into account the computational cost as a primary factor. We have used a binary Hopfield network, known to be much faster than classical Hopfield networks (see [15], [16]). However, it is expected that the computational cost of the HNN-GA algorithm will be higher than a GA with penalty function. Table VI shows computation times for all the instances considered.[4] The GA with penalty function converges faster than our hybrid algorithm in problems #1-#9, however, the computation time of our approach is reasonable. In the rest of the problems it was not able to achieve any feasible solution.

### *B. Experiments on a Related Problem: The Task Assignment Problem*

As has been shown before in Section II-D, there are a large amount of problems directly related to TA. In this section we test the performance of our hybrid approach against two different GAs in some tests instances of the Task Assignment Problem (TSAP) in heterogeneous computer networks. Briefly, the TSAP can be defined in the following way:

Consider an heterogeneous distributed system formed by a set of processors/machines $P = \{P_1, \ldots, P_M\}$, of different speed but with the same architecture, in such a way that a set of different tasks $T = \{T_1, \ldots, T_N\}$ of a distributed application can be executed on them. Let $W = \{w_1, \ldots, w_N\}$ the amount of resources required (memory constraints, processor load, etc) by the task $T_i$, and $R = \{r_1, \ldots, r_M\}$ the maximum resources available for a given processor $P_i$, then the TSAP can be formulated as a TA with $P$ being the concentrators, $T$ being the terminals, $W$ being the weights of terminals and $R$ the capacity of concentrators. Usually, the number of processors in TSAP instances are smaller than the number of concentrators in TA instances.

The cost function for the TSAP would be different from the one discussed for the TA in this paper. It should take into account the minimization of the total execution time and the minimization of the communication time between tasks in different processors. We can consider the following model:

- All the tasks may be executed in all the processors. Two tasks executing in different processors incur in a communications cost. The communication cost between tasks executing in the same processor (intraprocessor communication) is negligible.
- A matrix of communication costs $\mathbf{K}$ is defined, where each element $k_{ijpq}$ represents the communication cost between

[4]The computation times of algorithms strongly depends on the simulation platform used for running the experiments. In our case it was a *Dual Xeon*/2.8 GHz. Computation times in Table VI are shown as round figures, only for given an idea of differences in computational cost between algorithms.

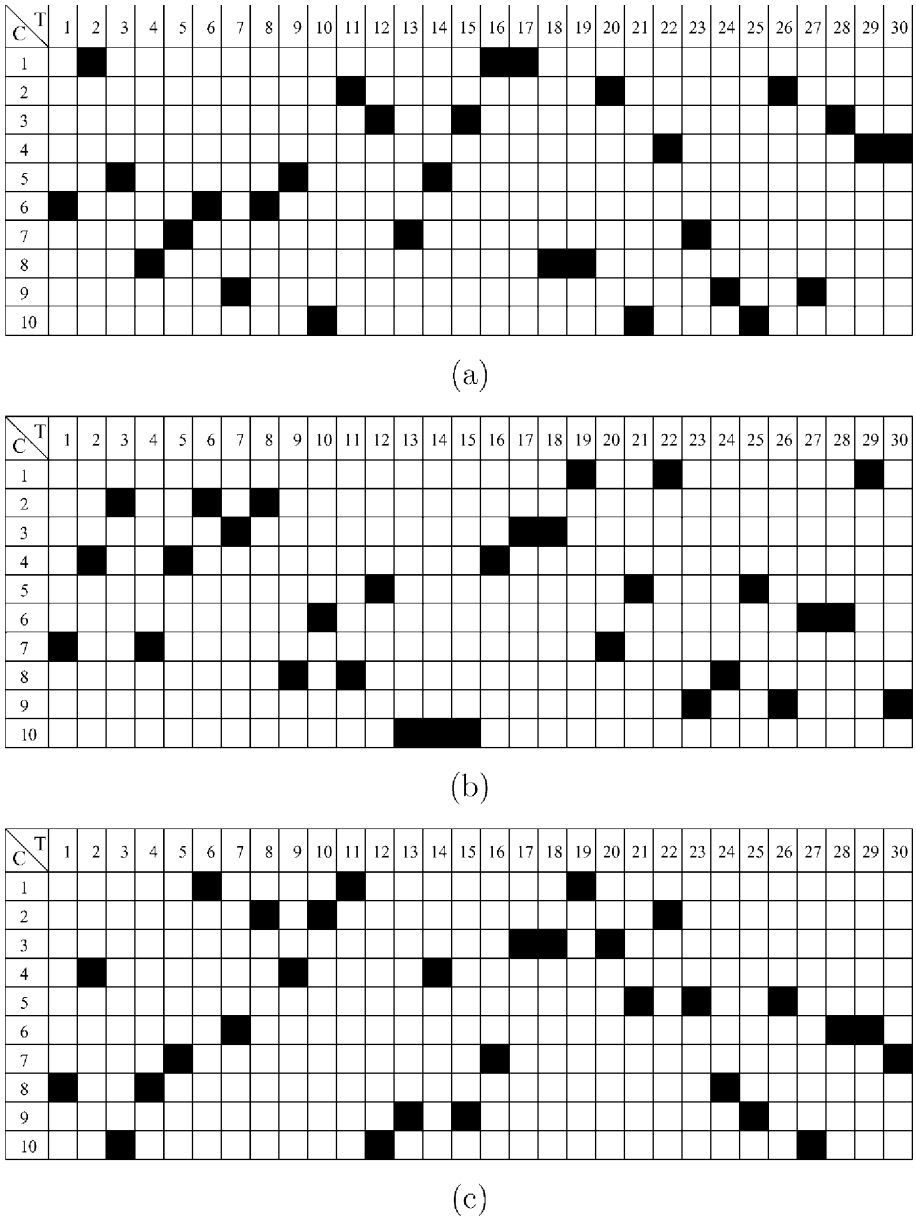



Fig. 2. Best assignments obtained by the Hopfield network-GA approach in problems #7 (a), #8 (b), and #9 (c).

a task $i$ executing in processor $j$ and a task $p$ executing in a different processor $q$.

- Let $v_j$ the relative speed of a given processor $j$ to the slowest processor in the system, being the speed of the slowest processor 1. Let $t_i$ be the time of execution of a given task $i$ in the slowest processor of the system. We define $t_j^e = (\sum_{i:x_{ij}=1} t_i)/v_j$ as the total amount of time needed by the processor $j$ in order to finish the tasks assigned to it.

The cost function $f(\mathbf{X})$ for the TSAP can be defined then as

$$f(\mathbf{X}) = \alpha_1 \sum_{j=1}^{M} t_j^e + \alpha_2 \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{\substack{q=1 \\ q \neq j}}^{M} k_{ijpq} x_{ij} x_{pq} \quad (13)$$

where $\alpha_1$ and $\alpha_2$ stand for two parameters which control the importance of each term in the cost function, and $\alpha_1 + \alpha_2 = 1$. Note that this objective function includes the minimization of the total execution time ($\alpha_1$) of the program and the communication costs between tasks ($\alpha_2$), as was required before. Note also that using this objective function only the cost associated with entire feasible solutions can be calculated.

In order to test the performance of our hybrid approach in this problem, we tackle a set of test TSAP instances. We consider five different instances, named *T*1-*T*5, involving 50 task and a different number of processors, from 3 to 7. The values of the rest of the parameters in the instances were generated as follows: the time a given task takes for finishing in the slowest processor ($t_i, i = 1 \ldots, N$) has been randomly generated with values between 1 and 10 from an uniform probability distribution. The

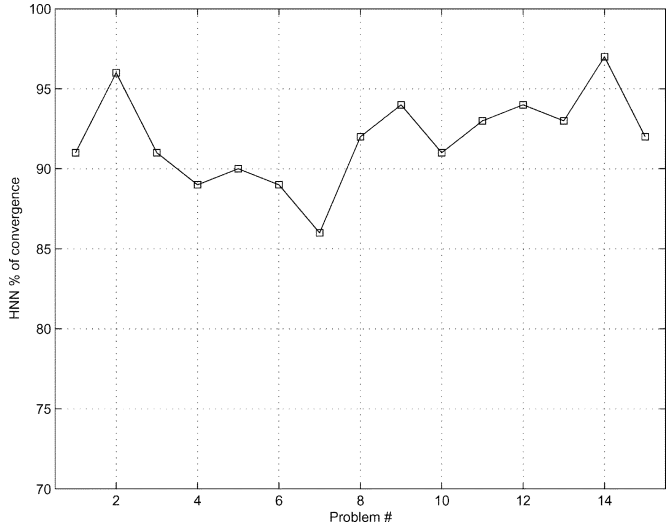



Fig. 3. HNN percentage of convergence in the TA test instances considered.

speed of the processors in the system ($v_j$, $j = 1,\ldots,M$) have also been randomly generated, with values between 1 and 5, ensuring that at least one processor has the value 1 (slowest processor). The amount of resources needed by each task ($w_i$) is randomly generated from an uniform distribution between 1 and 6. The total resources available for each processor ($r_j$, $j = 1 \ldots M$) have been generated in such a way that all the processors have the same resources available, and the total processors resources to be between 1% and 5% of the total task needed resources. Finally, each element of the matrix $\mathbf{K}$ of communication costs between tasks have been randomly generated from an uniform distribution with values between 1 and 10.

In the experiments performed, we will compare the results obtained by our Hybrid I algorithm, against the results obtained by the GA with penalty function, and a GA with a local search heuristic. In this latter GA, in addition to the term of penalty, we implement a local search heuristic for managing the infeasible assignments. We call this GA as $GA_{\text{local}}$. The local heuristic works in the following way:

*local search heuristic*
Calculate the load of the processors.
$\gamma = 1$;
**while**(max. number of loops $\gamma_{\max}$ has not been reached) **do**
 **for**(each overloaded processor $j$)
  **if** $\tilde{x}_i = j$, **then**
   $\tilde{x}_i = k$, $k \in P$ randomly chosen.
  Recalculate the load of processors.
 **end(for)**
$\gamma = \gamma + 1$;
**end(while)**

In all the simulations we have set the parameter $\gamma = 20$. The rest of the GA parameters are the same that were used in the TA instances. Table VII shows the results obtained by the considered algorithms in the five TSAP instances. The Hybrid I algorithm obtains better results than the GA and $\text{GA}_{\text{local}}$ approaches. The $t$-test applied to the data obtained by the three compared algorithms (Table VIII) shows that the differences between Hybrid I, GA and $\text{GA}_{\text{local}}$ are statistically significant. Note the large differences between the Hybrid I approach and the GA with penalty function, mainly in instances *T*4 and *T*5. It is easy to see that the local search included in the GA improves the quality of the solutions obtained, however, the Hybrid I approach proposed in this paper performs better.

TABLE VI
COMPUTATIONAL TIME (IN SECONDS) OF THE COMPARED ALGORITHMS. SYMBOL – MEANS THAT THE GA WITH PENALTY FUNCTION WAS NOT ABLE TO ACHIEVE FEASIBLE SOLUTIONS IN THOSE PROBLEMS (#10-#15)

| Problem # | GA | Hybrid I | Hybrid II |
|---|---|---|---|
| 1 | 2 | 5 | 5 |
| 2 | 2 | 5 | 5 |
| 3 | 2 | 5 | 5 |
| 4 | 5 | 20 | 20 |
| 5 | 5 | 20 | 20 |
| 6 | 5 | 20 | 20 |
| 7 | 15 | 45 | 45 |
| 8 | 15 | 45 | 45 |
| 9 | 15 | 45 | 45 |
| 10 | - | 120 | 120 |
| 11 | - | 120 | 120 |
| 12 | - | 120 | 120 |
| 13 | - | 360 | 360 |
| 14 | - | 360 | 360 |
| 15 | - | 360 | 360 |

Table IX shows the computational time of the three algorithms. The GA with penalty function is the fastest algorithm, as expected. The computation time of the Hybrid I approach proposed in this paper is larger than the other GAs compared, however, the Hybrid I algorithm is able to obtain much better solutions than the GAs in less than one minute, which is a reasonable computation time.

## VI. CONCLUSIONS

In this paper we have presented a hybrid Hopfield network-GA scheme for the TA problem. Our approach is focused on those TA in which the cost function is not known in advance. In these types of TA, previous approaches based on greedy heuristics are no longer valid, and "blind" algorithms (such a GA) are necessary for achieving high quality solutions to the problem.

Our algorithm hybridizes a binary Hopfield neural network with a GA. The Hopfield network solves the problem's constraints, giving feasible solutions to the problem, and a GA performs a global search for high quality feasible solutions. Two GAs with different encoding methods have been proposed to be hybridized with the Hopfield network. We have tested our algorithm on a set of TA instances of different difficulties, obtaining very good results that outperform previous approaches to the

TABLE VII
COMPARISON OF THE RESULTS OBTAINED BY THE HYBRID I APPROACH, GA WITH PENALTY FUNCTION AND GA WITH LOCAL HEURISTIC IN THE SET OF TASK ASSIGNMENT TEST PROBLEMS CONSIDERED (BEST/AVG./STD DEV.). T AND P STAND FOR THE NUMBER OF TASKS AND PROCESSORS, RESPECTIVELY

| Problem | T | P | Hybrid I | GA | $GA_{local}$ |
|---|---|---|---|---|---|
| $T1$ | 50 | 3 | 6166.1/6232.0/43.9 | 6208.1/6282.2/49.3 | 6228.1/6228.1/ 49.5 |
| $T2$ | 50 | 4 | 6114.2/6220.4/48.25 | 6273.6/6545.9/97.6 | 6303.7/6347.5/32.1 |
| $T3$ | 50 | 5 | 6161.6/6222.7/35.4 | 6347.6/6429.8/63.1 | 6232.5/6293.0/36.9 |
| $T4$ | 50 | 6 | 6055.8/6164.0/55.0 | 6422.5/6609.5/94.1 | 6124.6/6225.5/53.6 |
| $T5$ | 50 | 7 | 6178.2/6283.9/54.34 | 6547.6/6756.6/104.1 | 6297.1/6414.0/47.8 |

TABLE VIII
$t$ VALUES OBTAINED BY A TWO-TAILED t-TEST FOR THE TASK ASSIGNMENT PROBLEMS CONSIDERED. † STANDS FOR VALUES OF $t$ WITH 29 DEGREES OF FREEDOM WHICH ARE SIGNIFICANT AT $\alpha = 0.05$

| Problem | Hybrid I-GA | Hybrid I-$GA_{local}$ |
|---|---|---|
| | $t$-test | $t$-test |
| $T1$ | $-2.74^{\dagger}$ | $-4.03^{\dagger}$ |
| $T2$ | $-11.59^{\dagger}$ | $-6.63^{\dagger}$ |
| $T3$ | $-10.93^{\dagger}$ | $-4.78^{\dagger}$ |
| $T4$ | $-22.75^{\dagger}$ | $-4.64^{\dagger}$ |
| $T5$ | $-21.61^{\dagger}$ | $-8.43^{\dagger}$ |

TABLE IX
COMPUTATIONAL TIME (IN SECONDS) OF THE COMPARED ALGORITHMS FOR THE TASK ASSIGNMENT PROBLEMS CONSIDERED

| Problem | Hybrid I | GA | $GA_{local}$ |
|---|---|---|---|
| $T1$ | 18 | 5 | 10 |
| $T2$ | 25 | 10 | 15 |
| $T3$ | 35 | 15 | 20 |
| $T4$ | 45 | 20 | 27 |
| $T5$ | 55 | 25 | 35 |

problem. We have also shown the application of our approach in other combinatorial optimization problems related to TA.

Future work of this study includes further improvements of the efficiency of hybrid HNN-GA schemes and the investigation of other global search algorithms, such as simulated annealing [17] and tabu search.

## ACKNOWLEDGMENT

The authors wish to thank Prof. L. Hall and the anonymous referees for their interesting hints and suggestions which have contributed to improve the quality of this paper.

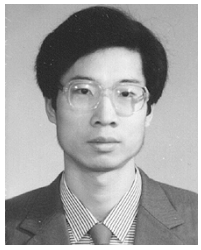
## REFERENCES

[1] C. H. Chu, G. Premkumar, and H. Chou, "Digital data networks design using genetic algorithms," *Eur. J. Oper. Res.*, vol. 127, pp. 140–158, 2000.
[2] A. Kershembaum, *Telecommunications Network Design Algorithms*. New York: McGraw-Hill, 1993.
[3] F. N. Abuali, D. A. Schoenefeld, and R. L. Wainwright, "Terminal assignment in a communications network using genetic algorithms," in *Proc. 22nd Annu. ACM Computer Science Conf.*, 1994, pp. 74–81.
[4] S. Khuri and T. Chiu, "Heuristic algorithms for the terminal assignment problem," in *Proc. 1997 ACM Symp. Applied Computing*, 1997, pp. 247–251.
[5] I. Brudaru, "Grouping-based hybrid genetic algorithm for the terminal assignment problem," in *Proc. 7th Int.. Research/Expert Conf. Trends Development of Machinery Associated Techniques*, Barcelona, Spain, 2003.
[6] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[7] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *J. Heuristics*, vol. 2, pp. 5–30, 1996.
[8] S. Khuri, M. Schülz, and J. Hitkötter, "Evolutionary heuristics for the bin packing problem," in *Proc. Int. Conf. Artificial Neural Networks Genetic Algorithms*, Ales, France, 1995, pp. 18–21.
[9] A. Merchant and B. Sengupta, "Assignment of cells to switches in PCS networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 521–521, Aug. 1995.
[10] S. Menon and R. Gupta, "Assigning cells to switches in cellular networks by incorporating a pricing mechanism into simulated annealing," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, pp. 558–565, Feb. 2004.
[11] W. Zhu, T. Y. Liang, and C. K. Shieh, "A Hopfield neural network based task mapping method," *Computer Commun.*, vol. 22, pp. 1068–1079, 1999.
[12] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, vol. 6, pp. 42–51, July/Sept. 1998.
[13] R. Szymanek and K. Kuchcinski, "Task assignment and scheduling under memory constraints," in *Proc. 26th Euromicro Conf.*, Mastrich, The Netherlands, 2000.
[14] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152.
[15] Y. Shrivastava, S. Dasgupta, and S. M. Reddy, "Guaranteed convergence in a class of Hopfield networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 951–961, Nov. 1992.
[16] S. Salcedo-Sanz, R. Santiago-Mozos, and C. Bousoño-Calzón, "A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, pp. 1108–1116, Apr. 2004.
[17] X. Yao, "A new simulated annealing algorithm," *Int. J. Comput. Math.*, vol. 56, pp. 161–168, 1995.
[18] R. M. Krzanowki and J. Raper, "Hybrid genetic algorithm for transmitter location in wireless networks," *Comput., Environ. Urban Systems*, vol. 23, pp. 359–382, 1999.
[19] T. Kurokawa and S. Kozuka, "A proposal of neural network for the optimum frequency assignment problem," *Trans. IEICE*, vol. J76-B-II, no. 10, pp. 811–819, 1993.
[20] C. Bousoño-Calzón and A. R. Figueiras-Vidal, "Emerging techniques for dynamic frequency assignment: Merging genetic algorithms and neural networks," in *Proc. Information Systems Technology Symp.*, Aalborg, Denmark, 1998, pp. 12.1–12.5.
[21] S. Salcedo-Sanz, C. Bousoño-Calzón, and A. R. Figueiras-Vidal, "A mixed neural-genetic algorithm for the broadcast scheduling problem," *IEEE Trans. Wireless Commun.*, vol. 2, pp. 277–283, Mar. 2003.
[22] Q. Wang, X. Sun, B. L. Golden, and J. Jia, "Using artificial neural networks to solve the orienteering problem," *Ann. Oper. Res.*, vol. 61, pp. 111–120, 1995.
[23] S. A. Cook, J. K. Pachl, and I. S. Pressman, "The optimal location of replicas in a network using a read-one-write-all distributed policy," *Distrib. Comput.*, vol. 15, no. 1, Apr. 2002.
[24] G. R. Mateus, H. P. L. Luna, and A. B. Sirihal, "Heuristics for distribution networks design in telecommunication," *J. Heuristics*, vol. 6, no. 1, 2000.

[25] E. Alba, C. Cotta, F. Chicano, and A. J. Nebro, "Parallel evolutionary algorithms in telecommunications: Two case studies," in *Proc. 8th Argentinian Computer Science Congr.*, Buenos Aires, Argentina, 2002.
[26] F. N. Abuali, R. L. Wainwright, and D. A. Schoenefeld, "The design of a multipoint line topology for a communication network using genetic algorithms," in *Proc. 7th Oklahoma Conf. Artificial Intelligence*, 1993, pp. 101–110.
[27] C. C. Charddaire, P. Kapsalis, J. W. Mann, V. J. Rayward-Smith, and G. D. Smith, "Applications of genetic algorithms in telecommunications," in *Proc. 2nd Int. Workshop Applications Neural Networks Telecommunications*, 1995, pp. 290–299.
[28] K. Smith, M. Palaniswami, and M. Krishnamoorthy, "Neural techniques for combinatorial optimization with applications," *IEEE Trans. Neural Networks*, vol. 9, pp. 1301–1318, Nov. 1998.
[29] Y. Watanabe, N. Mizuguchi, and Y. Fujii, "Solving optimization problems by using a Hopfield neural network and genetic algorithm combination," *Syst. Comput. Jpn*, vol. 29, no. 10, pp. 68–73, 1998.
[30] J. Balicki, A. Stateczny, and B. Zak, "Genetic algorithms and Hopfield neural networks for solving combinatorial problems," *Appl. Math Comp. Sci.*, vol. 7, no. 3, pp. 568–592, 1997.

**Sancho Salcedo-Sanz** (S'00–M'03) was born in Madrid, Spain, in 1974. He received the B.S degree in physics from the Universidad Complutense, Madrid, Spain, in 1998, and the Ph.D. degree in telecommunications engineering from the Universidad Carlos III, Madrid, Spain, in 2002.

He is currently a postdoctoral Research Fellow in the School of Computer Science, The University of Birmingham, U.K. He has co-authored more than 20 international journal and conference papers in the field of genetic algorithms and hybrid algorithms. His current interests deal with optimization problems in telecommunications, genetic algorithms, hybrid algorithms, and neural networks.



**Xin Yao** (M'91–SM'96–F'03) received the B.Sc. degree in computer science from the University of Science and Technology of China (USTC), Hefei, China, the M.Sc. degree in computer science from the North China Institute of Computing Technologies (NCI), Beijing, and the Ph.D. degree in computer science from the USTC, Hefei, China, in 1982, 1985, and 1990, respectively.

He is currently a Professor of computer science and the Director of the Center of Excellence for Research in Computational Intelligence and Applications (CERCIA), the University of Birmingham, U.K., a Distinguished Visiting Professor of USTC, and a Visiting Professor at several other universities in P. R. China and the Chinese Academy of Sciences. He was a lecturer, senior lecturer and associate professor at the University College, the University of New South Wales, the Australian Defence Force Academy (ADFA), in Canberra, Australia, between 1992 and 1999. He held post-doctoral fellowships from the Australian National University (ANU), Canberra, and the Commonwealth Scientific and Industrial Research Organization (CSIRO), Melbourne, Australia, between 1990 and 1992.

He is the Editor-in-Chief of IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and an associate editor or an editorial board member of several other international journals. He was the past chair (2001–2003) of IEEE NNS Technical Committee on Evolutionary Computation and the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award. He has given more than 30 invited keynote and plenary speeches at various international conferences and chaired/co-chaired more than 25 international conferences in evolutionary computation and computational intelligence, including CEC'99, PPSN~VI'00, CEC'02, and PPSN'04. His major research interests include evolutionary computation, neural network ensembles, global optimization, computational time complexity and data mining.