Evolution of Topology and Weights of Neural Networks Using Semi Genetic Operators

I. A.Yousif
Department of Computer, College of Education University of Al-Mustansiriyah

Abstract

Evolutionary computation is a class of global search techniques based on the learning process of a population of potential solutions to a given problem, that has been successfully applied to variety of problems. In this paper a new approach to design neural networks based on evolutionary computation is present. A linear chromosome representation of the network is used by genetic operators, which allow the evolution of the architecture and weights simultaneously without the need of local weights optimization. This paper describes the approach, the operators and reports results of the application of this technique to several binary classification problems.

Introduction

Artificial neural networks [ANNs] are a class of computational tools inspired by the biological neurons system (1). They consist of partially or fully interconnected simple processing units, called neurons.ANNs derive their power from their parallel distributed structure, and their ability to learn underlying relations from a given set of representatives (2). The connections between units have weights, which must be adjusted through a training process, to solve a particular problem. The design of a neural network is still largely performed using a lengthy process of trial and error definition of the topology, followed by the application of learning algorithm such as back propagation (3). The most ambitious combination attempts to evolve the architecture and weights simultaneously without a separate training process (2). Two common strategies to do this are the destructive and constructive algorithms. Evolutionary computation is a class of global search techniques based on the evolution process of each individual (4)(5)(6), representing a potential solution to a given problem. Typical evolutionary computation updates this population seeking for better regions of the search space using the operations of selection, recombination and mutation, inspired by biological evolution (7).

An approach based on genetic programming [GP]. has been largely limited by the lack of a good encoding mechanism, which is a first step in this direction, for example, in (8) ANNs have been represented as parse trees. This representation was proposed by Koza, which is recombined using a crossover operator that swaps sub-trees representing sub-nets. The graph like structure of a neural network is not ideally represented directly with parse trees either. An alternative method based on genetic programming, known as cellular encoding (9) is indirect approach that avoids the problem of representing the network directly as a parse tree. It represents the rules to construct the network instead. However, there are still constrains on the weights evolved, and to construct the network using the rule at each generation which can be a considerable overhead. A recent work based on a new form of [GP], known as parallel distributed genetic programming (PDGP) was proposed by R.Poli, in which programs are represented as graphs (instead of parse trees) has been applied to evolution of NNs with promising results (10). R. Poli and J. Pujol proposed new methods upon the preliminary work done with PDGP by introducing a dual representation and new operators. In PDGP, instead of usual parse tree representation a graph representation was used, where nodes are allocated in two-dimensional grid of a fixed size and shape, which forms the chromosome. This method does not include any operators specialized in handling the connection weights. By using this structure to represent ANNs we cannot apply the standard genetic operators (such as one-point crossover...etc)could not be applied because the structure is nonlinear. Although the results were encouraging but led us to believe that a proper representation with fixed size and shape of the chromosomes could make inefficient use of the available memory. In addition anothen disadvantage of this method when representing ANNs is that it does not take care with the design rules of neural network structure.

This paper describes a new form of Evolutionary computation which is suitable for evolution of artificial neural networks. It represents each individual with a linear chromosome of variable size and shape. This feature gives the power and ability of the new approach to passing most problems of representation in Genetic Programming.

The new approach proposes a new combination of crossover and mutation operators. Applying any one from both (standard or special genetic operators) allows the evolution of topology and weights of ANNs concurrently and very efficiently. The new method has been successfully applied to determination of architecture and weights of (three layered) feed forward networks.

Representation

In the new form, instead of the usual parse tree or the graph representation, an array of pointers is used to represent the chromosome. Each pointer represents a neuron and points to a linked list of nodes that describe its connections with the sending neurons in a previous layer.

Graph representation uses a linear genotype and a separate grid description to make it more natural. The new(linear or two-dimensional) representation avoids this problem where no need to descriptions because it represents NNs directly . The linear chromosome consists of a number of genes which represent the neurons of network. This chromosome is divided into three sub chromosomes. The genes of the first sub chromosome represent the input neurons, the genes of the second sub chromosome represent the hidden neurons, and the genes of the third sub chromosome represent the output neurons. This chromosome uses an array of pointers structure (see figure (1)).

In the standard GP the size of chromosome (parse tree) may grow excessively, but the new representation avoids this problem because the length of the chromosome is limited by the length of the array which equals the total sum of neurons in a particular neural network. Graph representation must represent the same number of neurons in all layers, While in this new form no need to do that. This characteristic can make the new method of representation more efficient in using the available memory. Like in standard GP there are three classes of neurons. The input class, the internal class and the output class. Also there are three types of nodes, in the linked lists pointed to by the neurons: function node, terminal node and body node . Each node consists of four fields. The function node fields are (see figure 2.a):

- The first field contains a value representing the number of nodes in the linked list.
- The second field contains a value representing the type of activation function.
- The third field contains a value representing the bias of the receiving neuron.
- The fourth field contains a pointer to next node.

The body node fields are (see figure 2.b):

- The first field contains a value representing the class of sending neuron.
- The second field contains the index of sending neuron.
- The third field contains a weight value of the link.
- The fourth field contains a pointer to the next node.

The terminal node fields are (see figure 2.c):

- The first field contains a value representing the the class of sending neuron.
- The second field contains the index of sending neuron.
- The third field contains the input value or a weight value of the link.
- The fourth field contains the nil pointer.

The input class must contain terminal nodes only, but the internal class and output class may consist of all types of nodes.

In the following are descriptions of the neurons in all parts of the chromosome, see figure (1): -

a. *The first sub chromosome* (Input neurons or input class*)*. The neurons in this part are representing the input layer, which contains input values. The node type in this part is a terminal node only (see figure 3).

b. *The second sub chromosome. (*Hidden neurons or internal class) The neurons in this part are representing hidden layer see figure (4). Each neuron in this sub chromosome points to a linked list of nodes that describe its connections with the sending neurons in the input layer. The nodes of this list consist of three types the first is a function node; the last is a terminal node and otherwise is a body node (internal node).

c. *The third sub chromosome.* (Output neurons or output class)The neurons in this part represent output layer see figure (5). Each neuron in this sub chromosome points to a linked list of nodes that describe its connections with the sending neurons in a previous layer. The nodes of this list consist of three types. The first is a function node; the last is a terminal node and otherwise is a body node (internal node).

The number of input neurons and output neurons depends on the problem (fixed number), and the number of hidden neurons is variable but subject to the condition that at least there is one neuron in the hidden layer connected with input and output layers.

Genetic operators

Genetic operators are performed on the last two sub chromosomes. Crossover operators

- *Both nodes are functions (Semi two points crossover).* This is the most important case. By combining the description of two functions, the topology and weights of the network can be changed see figure7.

The descriptions are combined by selecting two random crossover points, k1 in node a and K_2 following node b, and by exchanging the nodes between the two outs.

- *Both nodes are function (Semi one point crossover).* In this case the crossover operator applied by selecting randomly one point following both nodes (k1), and then replacing the right block of the node *a* with the right block in the node *b*. see figure 6, this case applied successfully.

Note

In all types of node level GP crossover operators, if the sub index of one or more of nodes is repeating in the offspring, the nodes which contain the iteration sub index are deleted, and one of them and is stay make its weight equal to the average of the iterations weights.

The genes of last two sub chromosomes of parents are scanned from left to right and exchanged with probability pc.

Mutation operator

- Changing mutation

The neurons of last two subcromosomes of a child are scanned from left to right and with small probability (Pm) we mutate the bias and weight in all nodes associated with the neuron. Each of the mutation values is generated randomly in the range $[-\alpha .. +\alpha]$.

If a new weight is close to zero, the node is removed. If all weights are close to zero the hidden neuron is removed.

- Removing mutation

The neurons of second sub chromosome of a child are scanned from left to right and with small probability Pm we remove the neuron (remove all its associated nodes and set the neuron to nil).

Note

If all hidden neurons are removed, the child is refused.

Results

To show the performance of the method proposed, it was applied to a test suit of benchmarks present in the literature: N-bit odd parity problems (for N=2(XOR),3,4 and 5), where the network output must be 1 if there is an odd number of 1's in the input pattern, and 0 otherwise.

In all experiments a population of 100 chromosomes was evolved for a maximum of 100 generations.

For each problem, we performed 50 runs with different random seeds. In the experiments, we used a generational genetic algorithm with: tournament selection (tournament size=2) and crossover and two types of mutation probabilities of 0.4 and 0.09 and 0.1 respectively.

The weights and biases were randomly initialized within the range [-1.0, +1.0].

The mutation operator used was the addition value to the weights during the evolution. The fitness function was the standard mean square error of the output of the network for all input patterns. The function set included a Sigmoid activation function.

The results obtained of using the XOR-problem (N-party problems) when all types of crossover are applied is shown in the table 1, the results table2, table3 and table4.

Column 2 represents the average number of generations at which reprsents the solution. Column 3 and 4 show the number of hidden neurons and the number of connections of such solution, respectively. Column 5 shows the copulated effort, and column 6 shows the percentages of runs in which solutions were found.

The representation of a typical solution for the XOR problem and the corresponding.

Further work

1.The results obtained so far are promising , but they were achieved without any optimization of relative probabilities of operators. Although some operators seem to perform better than others.

2.To keep diversity within the population, other mutation operators should be explored.

3.The extension to recurrent neural networks is a natural one, allowing the method to be applied to a wide range of tasks.

Conclusion

In this paper, a new approach to the automatic design of neural networks based on evolutionary computation has been presented. new operators were introduced , which exploited a liner representation , in which a linear encoding is used in conjunction with a array representation .

The representation of the neural network in a liberalized form allowed the development of efficient forms of crossover operations and the introduction of strategy to reduced the complexity of solutions, whereas the array of pointer description allowed to control the connectivity properties of the network.

The method was applied to evolve feed forward networks for a variety of binary classification problems showing promising results.

Refrences

1.Michalewicz Zbigniew. (1994). Genetic Algorithms [plus] Data Structures = Evolution Programs. Springer Verlag, Berlin.

2. Hammel, U. and Schwefel, H.(1997). IEEE Transactions on Evolutionary Computation, 1(1):3–17, Apr.

3. Moriarty, D. and Miikkulainen, R.(1993). Evolving complex Othello strategies using marker-based genetic encoding of neural networks. Technical Report AI93–206, Department of Computer Science, The University of Texas.

4.Kitano, H.(1994). Physical D, 75:225-238.

5.Fujita, S. and Nishimura, H. (1994). An evolutionary approach to associative memory in recurrent neural.

6. Angeline, P. J; Saunders, G.M and Pollack, J.B.(1994). IEEE Transactions on Neural Networks, 5(1):54–65.
7.Schwefel, H.(1996). In Proceedings of The third IEEE Conference on Evolutionary Computation, pages 20–. IEEE Press
8. Zhang, B. and Mühlenbein, H.(1993). Genetic programming of minimal neural nets using Occam`'s razor . In S.Forrest ,editor ,Proceeding of the Fifth international conference on genetic algorithms(ICGA '93), pages 342-349.Morgan Kaufmann .
9. Gruau F.(1994). Neural network synthesis using cellular encoding and the genetic algorithm. PhD thesis, Laboratoire de L'informatique du Parall_elisme, Ecole Normale Sup_eriere de Lyon, Lyon, France.
10. Fogel D.(1995). IEEE Press, Piscataway, NJ, USA.

Table :(1-1) The results of (XOR) problem for new method

4.1 The best solutions for (N-parity) problems:-

4.1.1. XOR problem :-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
Semi one-crossover	45	3 / 2.5/ 2	9/ 7.5 / 6	350	%100
Semi two-crossover	39	3 / 2.5/ 2	9/ 7.5 / 6	350	%100

Table: (1-2) The results of 3-parity problem for new method

4.1.2. 3-parity problem:-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
Semi one-crossover	60	5 / 4.0 / 3	17 / 16 / 12	350	%90
Semi two-crossover	69	4 / 3.5 / 3	17 / 16 / 12	310	%95

Table: (1-3) The results of 4-parity problem for new method

4.1.3.4-partty problem

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max		Solutions found
Semi one-crossover	100	8 / 6.5 / 5	48 / 42.0 / 30	360	%12
Semi two-crossover	100	9 / 7.0 / 5	54 / 42.0 / 30	300	%15

Table:(1-4)The results of XOR-parity problem for (Poli&Pujol)

4.2.The best solutions for (N-parity) problems for tow searcher (Poli&Pujol) [Pol.98]:-

4.2.1.XOR problem:-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
End node-end node	66	3 / 1.2 / 1	11 / 5.5 / 5	40,200	100%
End node-not end node	67	10 / 2.0/ 1	/ 5.5 / 5 11	39,800	100%
Not end node-end node	69	6 / 1.5/ 1	6.2 / 5 18/	40.000	100%
Semi one-crossover	48	8 / 1.8/ 1	11 / 5.5 / 5	40,200	100%

Table: (1-5) The results of 3-parity problem for (Poli&Pujol)
4.2.2.3-parity problem:-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
End node-end node	95	9 / 2.2/ 1	35 /10.7 / 7	72,000	98%
End node-not end node	103	10 / 2.0/ 1	31 /10.3 / 7	78.800	93%
Not end node-end node	105	6 / 2.0/ 1	24 /10.0 / 7	152,000	93%
Semi one-crossover	69	10 / 3.3/ 1	36 /13.6 / 7	141.000	52%

Table: (1-6) the results of 4-parity problem for (Poli&Pujol)
4.2.3.4-parity problem :-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
End node-end node	136	9 / 5.2/ 2	/ 14 42 /26.5	720,000	23%
End node-not end node	121	8 / 5.3/ 2	/ 14 37 /26.1	884,400	19%
Not end node-end node	112	5 / 5.0/ 5	/ 22 28 /25.0	5,472,000	2%
Semi one-crossover	130	6 / 6.0/ 6	/ 29 29 /29.0	12,025,800	2%

Table: (1-7) The results of 5-parity problem for (Poli&Pujol)

4.2.4.5-parity problem:-

The type of crossover	generations	Hidden neurons Min/avg./max	Connections Min/avg./max	effort	Solutions found
End node-end node	149	7/6.7/6	38/37.3/37	5,350,400	3%
End node-not end node	179	5/5.0/5	33/33.0/33	16,524.00	1%
Not end node-end node	188	7/7.0/7	38/38.0/38	18,500.00	0%
Semi one-crossover	190	9/7.5/6	43/40.0/37	20,500.00	0%

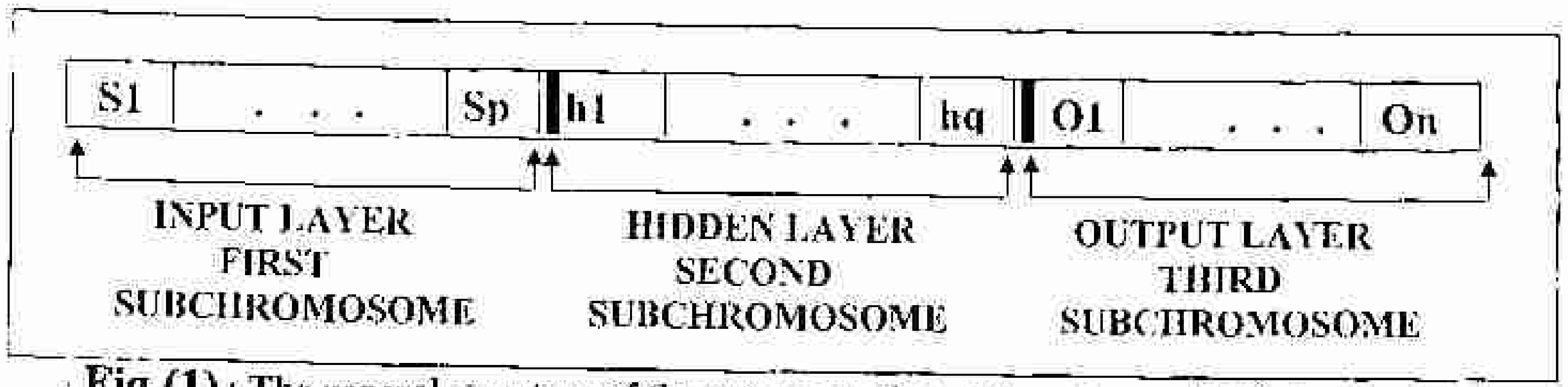
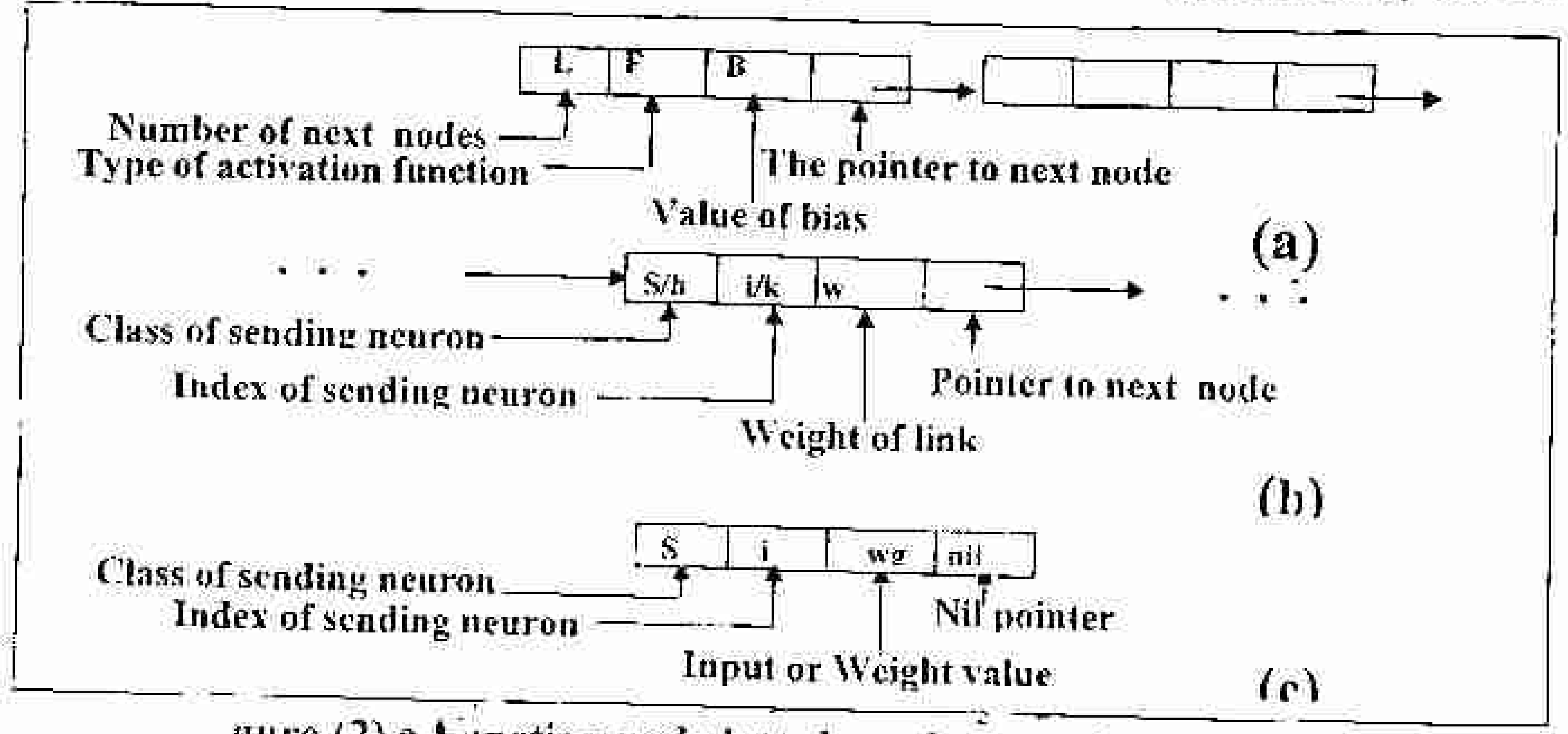


Fig.(1) The general structure of the representation of three layered NNs by our new form



_gure (2) a.function node b.body node c.terminal node.

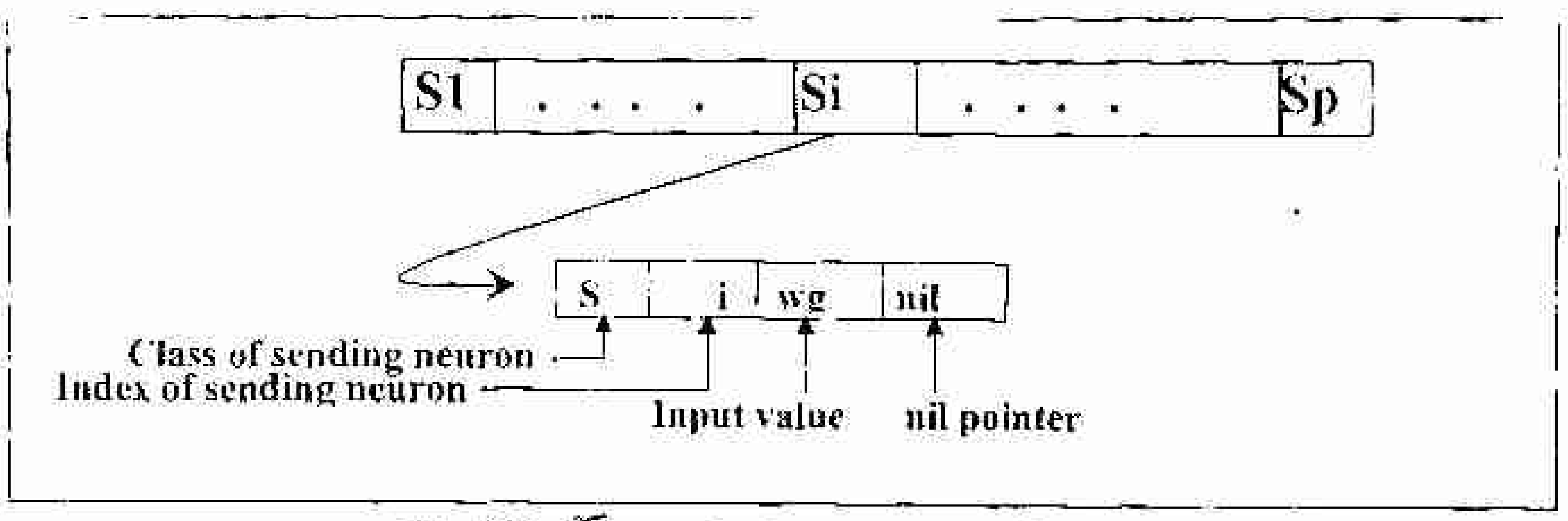


Fig.(3) first sub chromosome (input layer)

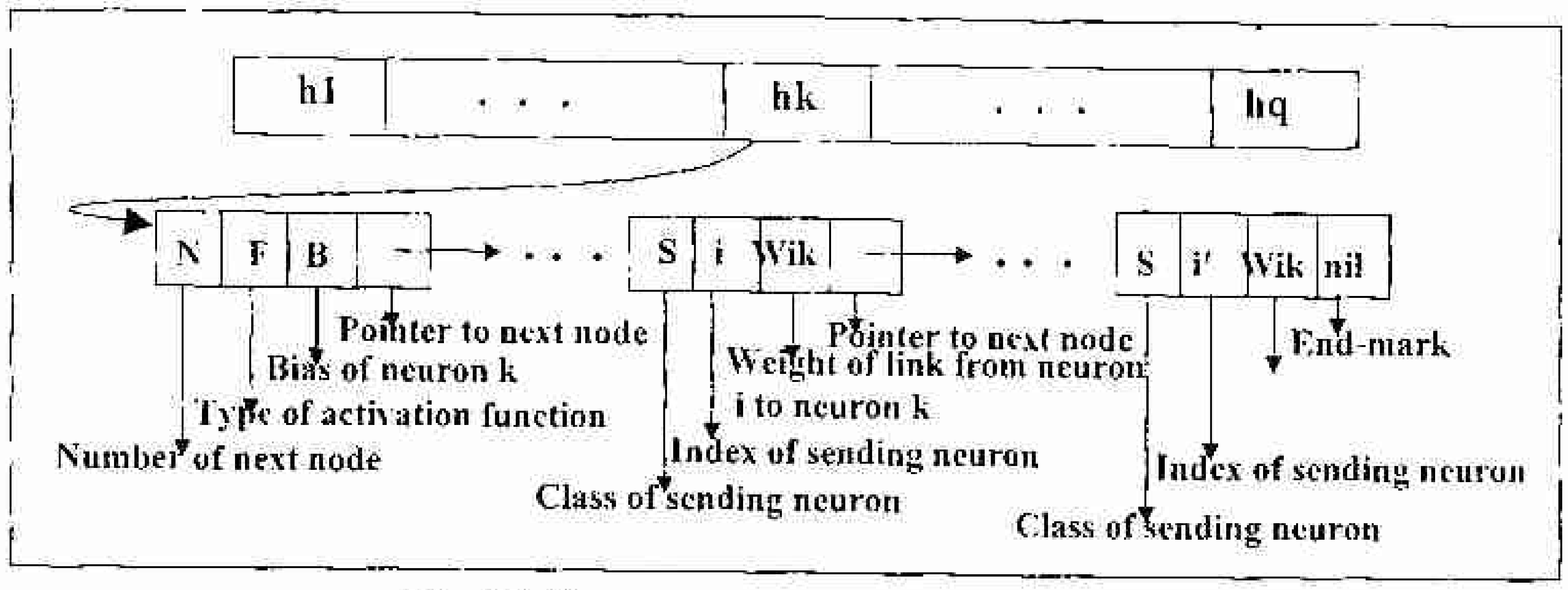


Fig.(4) Second sub chromosome (hidden layer)

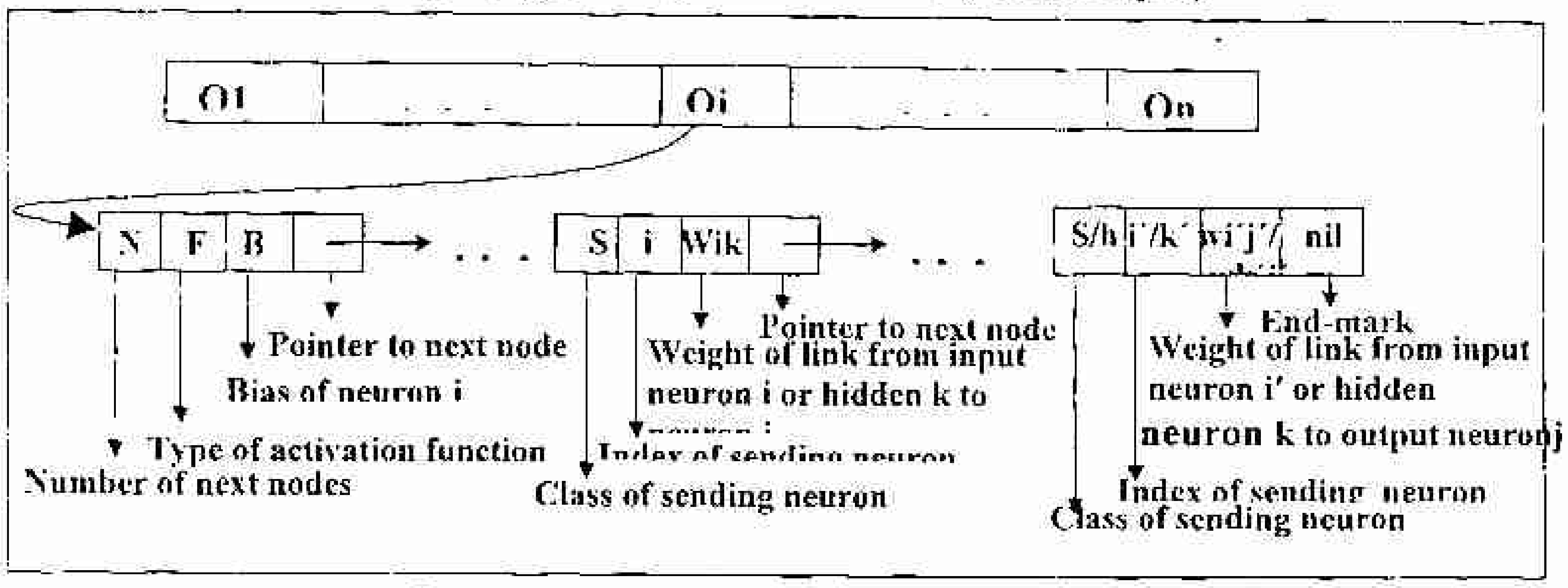


Fig.(5) Third sub chromosome (output layer)

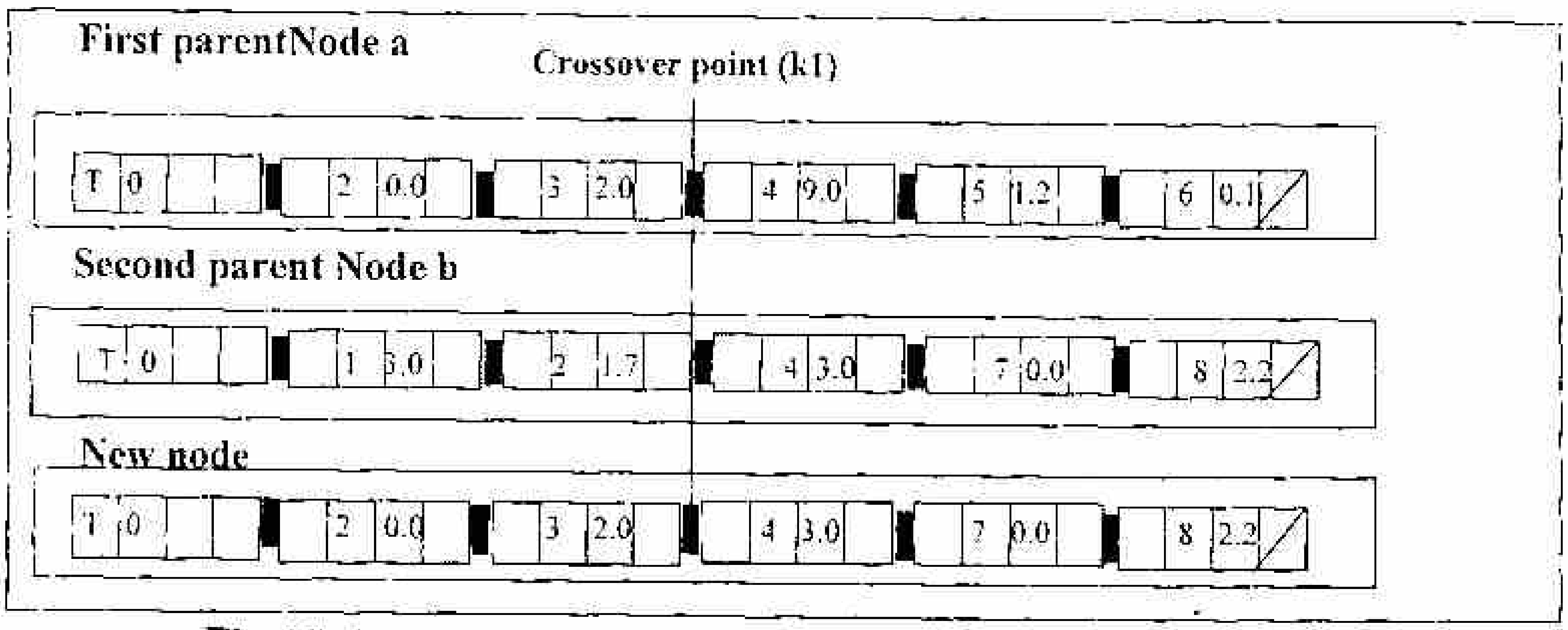


Fig.(6) Semi one-point crossover (both nodes are functions).

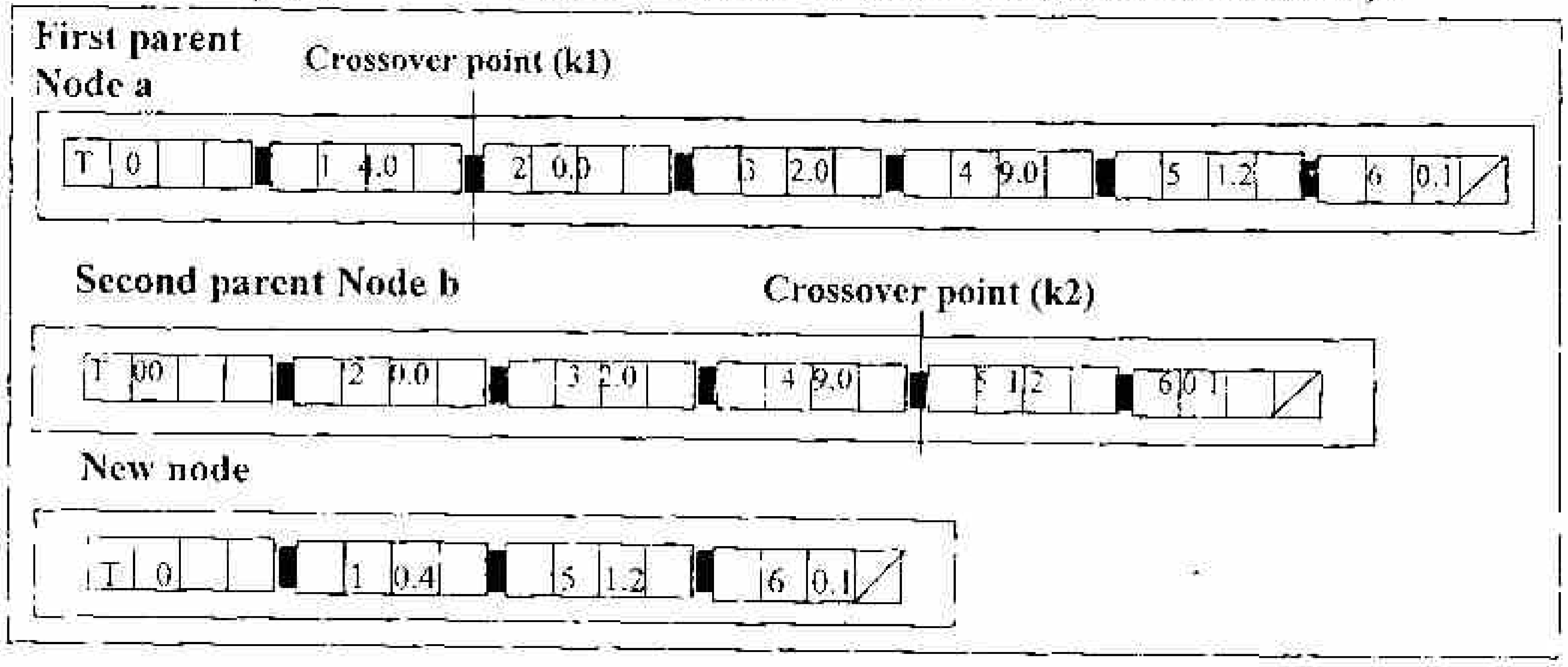


Fig.(7) Semi tow-point crossover (both nodes are functions).

تطوير تبولوجية واوزان الشبكات العصبية الاصطناعية باستخدام شبه العمليات الجينية

انتصار عبد يوسف
قسم علوم الحاسبات ،كلية التربية، الجامعة المستنصرية

الخلاصة

هي تقنيات أمثلية عالمية مستلهمة من التطور البيولوجي ، مـن خـلال تطبيـق العمليات التطويرية من اختيار(selection)، وتزاوج(crossover)، وطفرة(mutation)، على أفراد في مجتمع من حلول ممكنة لمسألة معطاة، هذه العمليات تطبق بشكل صـدفي لتنتقي الفرد الأفضل إنجازية. يقترح هذا البحث طريقة جديدة للتشفير في البرمجة الجينية ملائمة جدا لتطوير الشبكات العصبية الاصطناعية ، إذ يشفير كـل فـرد علـى شـكل كروموسوم خطي متغير الحجم والشكل، هذه الخصائص التي تميزت بها طريقة التـشفير الجديدة اعطت القوة والقدرة على تجاوز معظم المشاكل التي عانت منها مـسألة تمثيـل الكروموسوم في الخوارزميات التطويرية. طبقت الطريقة المقترحة بنجاح لتحديد معمارية واوزان شبكة ثلاثية الطبقة امامية الاتجاه لحل مجموعـة مـن مسائل التصنيف.