Using Genetic Programming to Improve the Performance of Wireless LAN Access Point Configuration

Yuta Yasuda and Yuji Sato

Graduate School of Computer and Information Science
Hosei University
i05t0010@k.hosei.ac.jp

Abstract. Radio communication speed has improved by leaps and bounds with developments in communication technology. In a wireless LAN, clients connect to a network to communicate with access points. Various approaches have benn used over the years to maximize the efficiency of access point configurations, such as the simplex method, Tabu search, and the genetic algorithm. This paper describes how genetic programming can be applied to the problem in a more deterministic function to improve the convergence speed. Linear genetic programming using arrays to shorten the running time and "bonsai" manipulation, hierarchichal pruning of genes based on evaluation of terminal nodes, are proposed. An experiment was carried out to compare the performance with that of standard genetic programming. Consequently, it was confirmed that linear genetic programming reduced the running time and that the proposed "bonsai" manipulation improves not only the convergence speed, but also the evaluation results.

1 Introduction

Radio communication speed has improved by leaps and bounds with developments in communication technology. As a result, wireless LANs have come to be used as alternatives for wired networks. In these LANs, communications proceed not via cables but electromagnetic waves, infrared radiation, and other means. Wireless LANs have the advantages that clients can connect to the Internet while moving and that the time and cost of wiring are eliminated. Because of these properties, wireless LAN has been used in new fields where it is difficult for wired networks to be applied. Therefore, it is forecast that opportunities to use wireless LANs will continue to increase, since the utility value of communication networks as part of the social infrastructure will be further enhanced. However, the cost of access points (APs) and wiring to connect APs cannot be ignored. Therefore, efficient design of wireless LANs is important if they are established fully.

Various methods of AP configuration have been tried, including the simplex method [2], Tabu search [3], simulated annealing [4], and the genetic algorithm

[5]. The genetic algorithm has been criticized as inappropriate for this problem, as its performance depends on its many running parameters and there is no convergence guarantee [6]. However, there are many tradeoffs among multiple objectives of system configuration, and evolutionary algorithms have proved to be among the most effective approaches for identifying a set of Pareto solutions simultaneously [7]. Consequently, evolutionary algorithms are considered to be more effective than others, and the genetic algorithm has been applied to AP configuration in various types of wireless LANs. In particular, Tang et al. proposed an innovative variable-length genetic algorithm named the hierarchical genetic algorithm (HGA) to solve the configuration problem with a variable number of APs [5]. In this method, the number (K) of necessary APs is first estimated, but if K is too big or too small, optimal solution search encounters an obstacle. However, the research has shown that evolutionary algorithms are very effective for optimizing the AP configuration in wireless LANs and for operating complex conditions. In that research, AP configuration of wireless LANs was treated as a variable-length search problem. Therefore, Jianjun et al. applied Genetic Programming (GP) [1] to this problem. Since genetic programming searches a space starting from small solutions, the difficulty of estimating K in HGA is largely avoided. As a result, it was found that GP is suitable for representing variable lengths and for evolving open-ended solutions [8]. The remaines of this paper reports the result of applying linear GP to reduce the running time and pruning of genes ("bonsai" manipulation) to improve the convergence speed, with the aim of improving GP performance, and describes our experiments and evaluation.

2 GP Approach for AP Configuration

2.1 Construction of Wireless Networks

In wireless LANs, clients connect to a network to communicate with APs. An AP has a certain communication range, and when clients are in that range, they can communicate [9]. A wireless LAN is set up in such way that APs can cover all clients in their range and are them wired. In this research, an optimal design is defined as one that minimizes the cost of the setup. Therefore, it is necessary to find the minimum possible number of APs and the minimum length of wire. For wiring, the minimum spanning tree (MST) algorithm is used. We used Prim's algorithm, in which MST evolves into a tree composed of one point [10]. Wiring between APs is represented in generating MST by weighting the distances between APs. GP is applied to configuration of APs.

2.2 Design of an AP Configuration Using GP

In this research, we investigated the configuration of APs, and searched for an optimal configuration using GP. The necessary functions are to set and remove APs, and the arguments of functions are the coordinates of APs.

In a continuing series, an evaluation function is defined. Definition of an evaluation function is a difficult problem in efficient design of a wireless LAN

[8]. Consequently, only a limited number of clients and APs are evaluated in this research, since the MST algorithm is assumed to give a design for optimal wiring. The evaluation function is defined in such a way that the configuration receives a high evaluation if fewer APs cover more clients. The evaluation function uses two parameters: the number of clients that can communicate (nCovered) and the number of APs (numofAP). It is defined as follows:

$$fitness = nCovered * (nCovered - numofAP) \tag{1}$$

The difference between nCovered and numofAP is used in the evaluation function, because a high evaluation score is assigned to a configuration in which fewer APs cover more clients. A coefficient of the above difference, called nCovered, is used. This is a variable number whose value increases when the individual is excellent, because the experimental results were poor when the coefficient was set to a constant number. When the coefficient is a small number, not only is the convergence speed slow, but also multiple redundant APs are set. On the other hand, when the coefficient is a large number, a design in which a few specific APs are set receives an extremely high evaluation. Therefore, individual diversity is lost during generation shifts, and the population is certain to converge to a very poor local solution. Consequently, for the coefficient of the above difference, nCovered, not a constant number but a variable number is used.

3 Proposition for Improving Convergence Speed

In this section, we propose a method of program implementation (using linear GP) and a genetic manipulation ("bonsai" manipulation) to improve the convergence speed.

3.1 Speeding Up Running Time by Using Linear GP

In this paper, GP is implemented in C language because of the latter's versatility. Generally, tree structures are implemented by pointers in C language. In GP, a tree structure grows in generational shifts, and consequently large amount of memory is used if tree structures are implemented by pointers. To prevent this problem, application of linear GP has been suggested. One characteristic of linear GP is that tree structures are implemented by arrays [11]. In linear GP, tree structures implemented by pointers are first replaced by arrays in prefix order. Next, the arrays are analyzed by using StackCount (SC) to detect tree structures. Fig. 1 shows how SC is used in linear GP. SC is the difference between the numbers of pushes and pops on a stack. The figure shows the SC corresponding to each node and the sum of the SCs according to the progress of operation from left to right. The numbers in circles are the total values of the SCs in Fig. 1. In a structurally correct tree, the sum of the SCs is always less than '1' except at the end of trees, scanning from left to right. Similary, the total sum of the SCs must be '1' at the end of a structurally correct tree. The above properties of SCs are valid in scanning from any node selected at random. By applying

genetic operation, subtrees are detected by adapting the above properties of SCs. It seems that linear GP can reduce memory consumption and perform at high speed by eliminating the use of pointers.

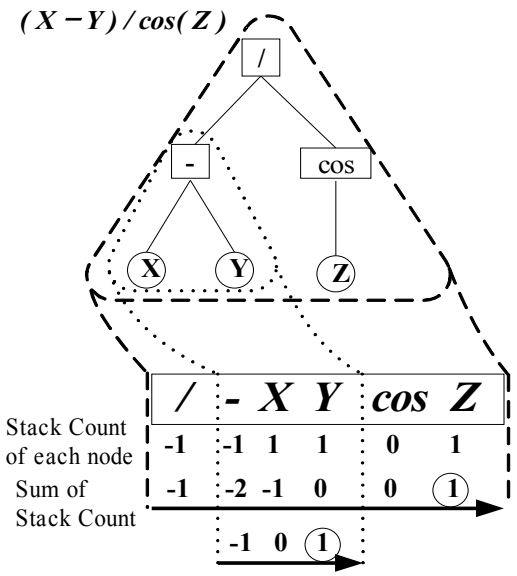

Fig. 1. Diagram of StackCount. Tree structures are replaced by arrays, using a prefix format. Next, the arrays are analyzed by StackCount to detect their tree structure. The total sum of StackCount must be '1' at the end of a structurally correct tree.

3.2 Proposal of "Bonsai" Manipulation

Improvement proposal applied pruning. In the field of artificial intelligence, a type of manipulation called pruning can be used to improve search efficiency [12]. Pruning improves search efficiency by avoiding tracing below specific nodes. In this field, a tree structure represents a transition of states, and pruning is used to search for states efficiently in terms of time and search depth .

GP adopts a similar approach for handling tree structures. However, the difference is that the pruning manipulation in artificial intelligence avoids tracing tree structures, whereas the manipulation in GP controls the growth of tree structures. Because of the evolution of tree structures as chromosomes, GP has the problem that unnecessary patterns emerge, such as meaningless subtrees and multiple subtrees with the same pattern in identical individuals. To solve or avoid these problems, GP uses manipulations that control to unnecessary growth of tree structures and remove unnecessary subtrees.

Minimum description length (MDL) and pattern matching have been previously proposed as manipulations for controlling the size of tree structures and removing subtrees, respectively.

For controlling tree growth in MDL, the expression below is employed as a fitness function [13]. The fitness function is used to describe the length of data. As a result, individuals represented in short form can be easily selected and spread among the population.

$$MDL = description\_length\_of\_data + complexity\_of\_model$$

Next, pattern matching is explained. Pattern matching is applied to subtrees below a node selected randomly from tree structure. Redundant patterns used in pattern matching are defined beforehand. When the subtree below the selected node includes redundant patterns, each part of the subtree that is the same as a redundant pattern is removed [14].

In this research, GP is applied to AP configuration in wireless LANs. The evaluation function is in inverse proportion to the number of terminal nodes. Therefore, it is considered that a concept like MDL has been previously actualized to add size of tree structure to evaluation value [15]. However, it is difficult to promote emerging redundant patterns sufficiently by using MDL above. Since redundant patterns are very numerous in AP configuration, it is difficult to define redundant patterns beforehand. Consequently, "bonsai" manipulation is proposed as a new pruning method, which prunes in ways such as bottom-up based on evaluation of terminal nodes.

"Bonsai" manipulation. The proposed "bonsai" manipulation is shown in Fig. 2 to 5. The arrangement of APs is converted into a tree structure such as that shown in Fig. 2. Here, square nodes denote function nodes and circular nodes denotes terminal nodes in the tree structure. In addition, function nodes are indexed alphabetically to identify each node, and terminal nodes are indexed in numerical order corresponding to the labels of the AP arrangement in Fig. 2. The following section describes the procedures of "bonsai" manipulation for removing unnecessary nodes. The following three rules are the criteria for unnecessary nodes in the AP configuration:

Rule 1 The node being investigated covers no clients.

Rule 2 The node being investigated and other nodes are set to the same coordinates.

Rule 3 The node being investigated covers the same clients as are covered by other nodes, and evaluation result for the node being investigated is less than that of other nodes.

The evaluation result for Rule 3 primarily compares the number of clients covered by each node. A node covering many clients receives a high evaluation. If the number of clients covered by a node is the same on multiple nodes, the

evaluation result compares the average distance from a node to each client covered by the node. If the average distance is short, the evaluation is high, since the quality of communication is linked to be better.

In accordance with the above rules, unnecessary nodes are specified. At first, APs covering no clients are removed form Fig. 3, in accordance with Rule 1. In the arrangement shown in Fig. 3, the AP labeled '3' covers no clients, so the node indexed '3' is removed from the tree structure. The arrangement after the node has been removed is shown in Fig. 4. Subsequently, unnecessary nodes are removed in accodance with Rules 2 and 3. In the arrangement shown in Fig. 4, the AP labeled '5' is set to the same coordinates as the AP labeled '4', thus satisfying Rule 2. In addition, the APs labeled '1' and '6' cover the same client. The AP labeled '1' covers two clients, while the AP labeled '6' covers only one client. Therefore, the AP labeled '6' satisfies Rule 3. As a result, the two nodes indexed '5' and '6'are removed from tree structure, as shown in Fig. 4.

In the tree structure of left-hand part of Fig. 5, the function node labeled 'D' has no child nodes. Therefore, the node is unnecessary. Furthermore, function nodes having one child node are also considered unnecessary for the properties investigated in this research. Therefore, the node labeled 'B' is also unnecessary. Consequently, the above two function nodes indexed 'B' and 'D' are removed from the tree structure, shown in Fig. 5. The only remaining function nodes are the nodes labeled 'A' and 'C' in the tree structure shown in Fig. 5. In this regard, if even the root node is removed, individual information is also lost. Therefore, the root node can generate a new child node when the root node has no child nodes. The eventual arrangement of APs is shown in the right-hand part of Fig. 5.

It might be thought that the proposed manipulation is unnecessary for adding restrictions such as only generating APs covering clients on early ontogeny. However, this manipulation can be used in generation shifts, and can therefore remove redundant patterns appearing as a result of crossover or mutation.

Furthermore, by judging as necessary whether terminal nodes are redundant, pruning can be applied without the need to define redundant patterns beforehand. Pruning can also be applied to patterns whose emergence is difficult to promote in concepts such as MDL. Therefore, we think that the convergence speed can be improved. The manipulation is named "bonsai" because it represents a tree structure in what we believe is a smart way, and also because leaf nodes and branches are pruned to increase the evaluation result.

4 Experiment

4.1 Evaluation Method

In a GP Experiment, the number of individuals was set at 1000 and the number of clients was set at 30, communication range of the APs was set at 30. The extent of the areas was assumed 400 400. The rates of crossover and mutation were set at 0.9 and 0.1 respectively. The computer used in this experiment had

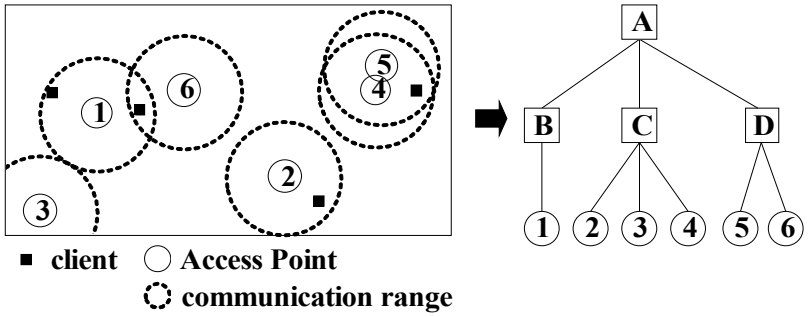

Fig. 2. Conversion of an access point configuration into a tree structure. In the tree structure, function nodes are labeled in alphabetical order, and terminal nodes are labeled in numerical order corresponding to the labels of access points in the arrangement on the left.

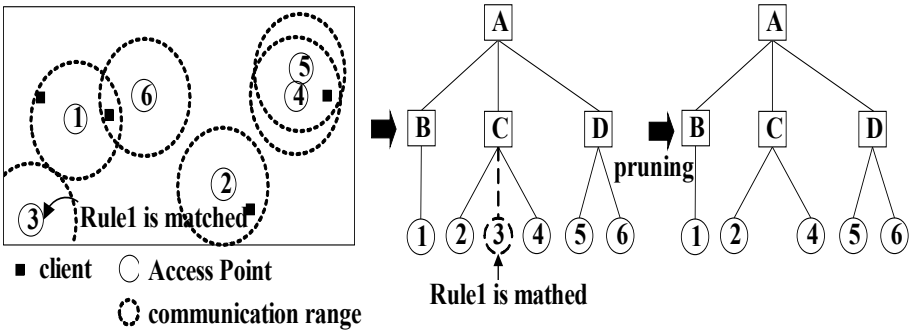

Fig. 3. Pruning of terminal node labeled e3e. In the arrangement, the access point labeled '3' satisfies Rule 1. Therefore, the node indexed '3' is removed from the tree structrure.

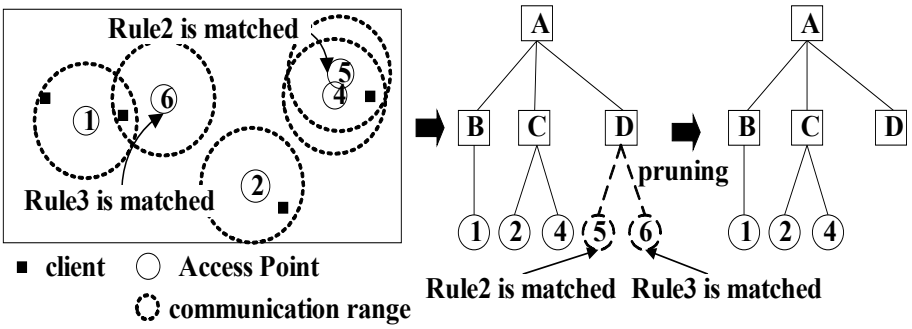

Fig. 4. Pruning of the terminal nodes labeled '5' and '6'. In the arrangement, the access point labeled '5' satisfies Rule 2, and the access point labeled '6' satisfies Rule 3. Therefore, the nodes indexed '5' and '6' are removed from the tree structure.

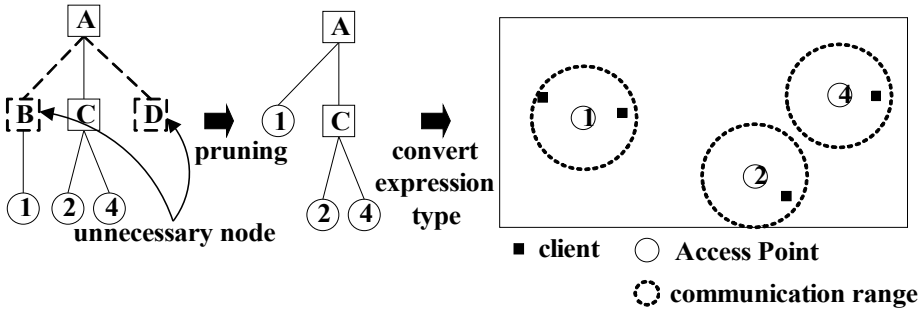

Fig. 5. Pruning of unnecessary function nodes labeled 'B' and 'D'. The node labled 'B' has one branch, and the node labeled 'D' has no branches. Therefore, the nodes are removed from tree structure. In addition, the final state of the arrangement is shown on the right.

a 2.6-GHz CPU and 1 GB of memory. First, we checked the effect of running time of linear GP and the "bonsai" manipulation exercise on the AP configuration. The running time was recorded for each generation. The effect of "bonsai" manipulation on the convergence speed was then examined. Changes in the number of convergence generations, with or without "bonsai" manipulation, and the timing of textrm"bonsai" manipulation were also recoreded.

4.2 Experiment Results

Comparison of Running Times. The running time in the AP configuration is investigated by comparing standard GP with linear GP and linear GP when "bonsai" manipulation is applied. Fig. 6 shows the running time of standard GP and linear GP. Fig. 7 presents an expanded view of Fig. 6 to show the running times of linear GP and linear GP when "bonsai" manipulation is applied. Each approach was executed 10000 times, and the running time was recorded up to 100 generations for each generation. When standard GP is compared with linear GP, the experimental results in Fig. 6 show that linear GP shortens the running time drastically. In addition, the running time of standard GP increases nonlinearly, while the time of linear GP increases linearly. The slope of the graph on Fig. 6 change from 1.3 at first 10 generations to 3.9 at last 10 generations in standard GP. While in linear GP, the slope change from 0.16 to 0.04. Next, when linear GP is compared with linear GP using "bonsai" manipulation, the latter increases time before 40 generations in Fig. 7, but reduces the time after 40 generations.

Comparison of Convergence Speeds. In the AP configuration, the change of convergence speed is examined in relation to the use or non-use of "bonsai" manipulation and the timing of this manipulation. Fig. 8 shows the results after 20000 executions and the number of termination generations counted for each 20 generations, respectively. The results are shown without "bonsai" manipulation

(standard GP in Fig. 8), for GP with "bonsai" manipulation applied only to initial individuals, and for GP with "bonsai" manipulation applied at every generation. Fig. 9 shows the transition of best fitness for standard GP and GP with "bonsai" manipulation applied. The best fitness in Fig. 9 is to get the average of 20000 executions. Table 1 shows the deviation of fitness and the average of convergence generations and calculation time, fitness value in the experiment of Fig. 8. Table 1 also shows rate of local solution that means the rate of the experiments spent to converge 400 generations and over. The experimental result in Fig. 8 shows that a higher peak in fewer generations by applying "bonsai" manipulation inrespective of the timing. Fig. 8 shows that the peak value in the graph is higher when "bonsai" manipulation is applied every generation. The transition of the fitness in Fig. 9 shows that fitness is achieved earlier and that the eventual fitness is higher with "bonsai" manipulation than without it. Furthermore, all the values in Table 1 for GP without "bonsai" are worse than those for GP with "bonsai". The best running time in Table 1 is for GP with "bonsai" manipulation applied only to initial individuals. Incidentally, the average running time of "bonsai" manipulation is 5 ms.

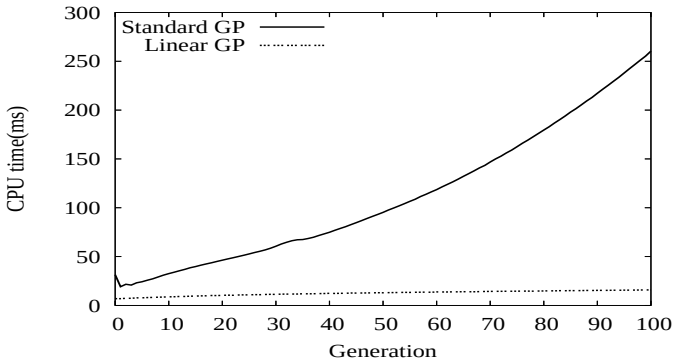

Fig. 6. Average running time of standard GP and linear GP. The experiment was executed 10000 times up to 100 generations.

5 Discussion

In a comparison of standard GP and linear GP, linear GP shortens the running time drastically, as shown in Fig. 6. We confirmed that the running time is improved by applying linear GP in which arrays are used instead of pointers.

In a comparison of the running time for linear GP and linear GP with "bonsai" manipulation in Fig. 7, the magnitude relation of the running time changes in the vicinity of 40 generations. In Fig. 8, GP with "bonsai" manipulation starts to converge in the vicinity of 40 generations, while GP without "bonsai" manipulations starts to converge after 60 generations. Furthermore, in

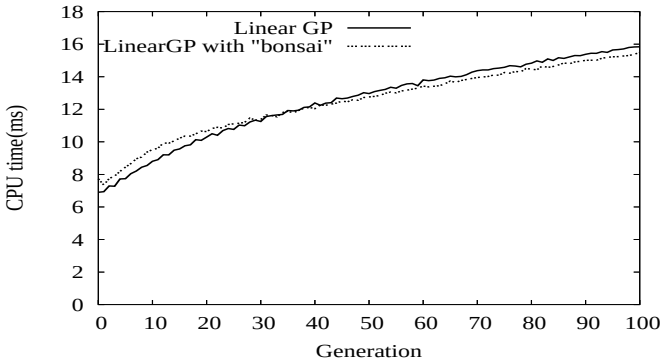

Fig. 7. Expanded view of Fig. 6 to compare running time of linear GP and linear GP using "bonsai" manipulation.

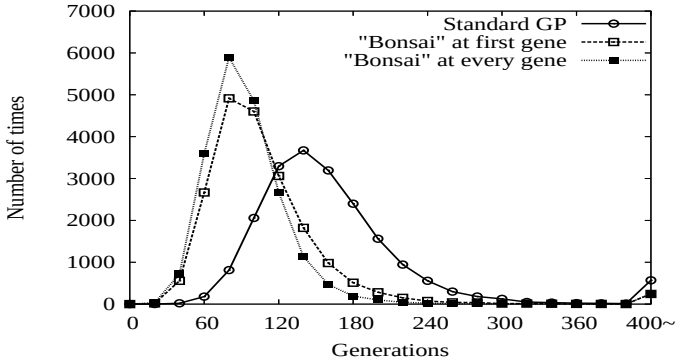

Fig. 8. The graph is to compare convergence speed. Termination generation is counted up each 20 generations on 20000 executions for standard GP and "bonsai" at first gene, "bonsai"every gene.

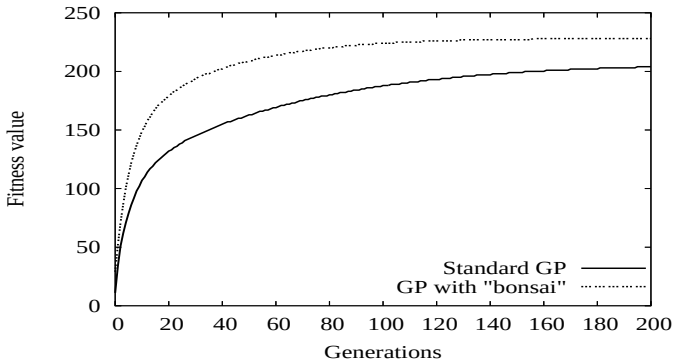

Fig. 9. The figure is plotted average of best fitness up to 200 generations per each generation. The experiment is executed 20000 times for standard GP and GP with "bonsai".

Table 1. Experimental result at comparison of convergence speed. Each value is 20000 executions average.

	Standard linear GP	"Bonsai" at first gene	"Bonsai" at every gene
Convergence generations	180	105	94
Rate of local solution(%)	2.8	1.2	1.2
Calculation time(ms)	2672	1396	1729
Fitness value	204	227	228
Fitness deviation	38.2	36.4	36.1

Fig. 9, the fitness of GP with "bonsai" manipulation is 90% of the eventual fitness after 40 generations, while the fitness of GP without "bonsai" manipulation is only 68% after 40 generations. This shows that fitness can be increased faster by applying "bonsai" manipulation. In AP configuration, high-fitness individuals come to make up a rather large propotion of the tree structure, covering many clients. Briefly, the speed of tree growth becomes faster when "bonsai" manipulation is applied. Linear GP uses arrays, and must therefore operate on arrays in genetic manipulation. The fact that the operating time of arrays depends on the length of arrays shows that the running time of linear GP is related to the average size of tree structures in a population. When "bonsai" manipulation is applied, it is likely that the speed of tree growth becomes faster in a early phase because the excellent part of the tree structure can be easily found, while a solution space can be searched without increasing the tree structure unnecessarily in the final phase.

As the regards the effect on convergence speed of "bonsai" manipulation, GP applying "bonsai" manipulation shapes higher top at less generation in Fig. 8. In Table.1, by "bonsai" manipulation, the rate which GP lapse local solution reduce by half, and running time spent on convergence also reduce. Furthermore, average fitness of optimal solution is improved about 10%. For these reasons, removing unnecessary node artificially after generating individual provides not only improvement of both convergence speed and convergence accuracy lapsed no local solutions, but also improvement of evaluation value. This indecates that the AP configuration more closely approximate the required condition. It is regarded that the cause is removing redundant part of tree structure without losing diversity of individual unnecessarily by "bonsai" manipulation. Simultaneously, removing redundant parts, there is also a chance that hidden excellent pattern of the structure is easy to be found and come down to next generation. As the result, it is regarded leading to find solution fitted arrangement better, searching various combination of excellent pattern.

6 Conclusions and Future Work

In this paper, GP was applied to AP configuration in wireless LANs. A method of improving the convergence speed was proposed its validity and was exemined. Specifically, we implemented system using linear GP and the proposed "bonsai" manipulation, which performs hierarchical pruning of genes based on the data of terminal nodes, and evaluated the running time and convergence speed. Consequently, we confirmed that linear GP reduces the running time. It is found that "bonsai" manipulation not only increases the convergence speed, but also improves evaluation result and avoids local solutions, while determination of redundant nodes depends only on the terminal node in the AP configuration. Therefore, the criterion for when to apply "bonsai" manipulation is extremely simple. Futher research is required to increase the method's generality so that it can be applied to a wider variety of problems.

References

1. John R. Koza: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT press(1992)
2. M.H.Wrigh1: Optimization Method for Base Station Placement in Wireless Applications. Proc. IEEE Vehicular Technologv Conference 98(1998)
3. C.Y.Ire and H.G. Kang: Cell planning with capacity expansion in mobile commuoications: a tabu search approach. Proc. IEEE VTCZOOO, 49 (2000)
4. S. Hurley: Planning effective cellular mobile radio networks. IEEE Tram. On Vehicular Technology (2002)
5. Kit-Sang Tang, Kim-Fung Man and S. Kwong: Winless Communication Network Design in IC Factory. IEEE Transactions on Industrial Electronics, Vol. 48, No. 2 (2001)
6. E. Koichi and W. Yoshimori: Automatic Cell Design for Wide Area Wireless LAN Systems. NEC research & deveopmet,44[4]2003. Special Issue on Devices and Systems far Mobile Communications(2003)
7. Carlos A. Coello, David A. Van Veldhuizen and Gary B. Lamont: Evolutionary Algorithm for Solving Multi-Objective Problem. Kluwer Academic Publishers (2002)
8. Jianjun Hu, E. Goodman: Wireless Access Point Configuration by Genetic Programming. Evolutionary Computation 2004, vol. 1 (2004)
9. Shiro Sakata: Ubiquitous Techniques Wireless LAN. Ohm press (2004)
10. R.Sedgewick: Algorithms in C Vol.3EGraph E Math E Topic. Kindai Kagaku press (1996)
11. Nao Tokui, Hitoshi Iba: Emprical and Statistical Analysis Genetic Programming with Linear Genome. In Proc. 1999 IEEE International Conference on Systems, Man and Cybernetics (SMC99) (1999)
12. Stuart Russell, Peter Norvig: Artificiall Intelligence. Kyoritsu Press (1997)
13. Hitoshi Iba: Genetic Programming. Tokyo Denki University Press (1996)
14. Ayahiko Niimi, Eiichiro Tazaki: Extended Genetic Programming using Reinforcement Learning Operation. Proc. 1999 IEEE International Conference on Systems, Man and Cybernetics (SMC99) (1999)
15. Kinnear, Jr.K.E.: Generality and Difficulty in Genetic Programming: Evolving a Sort: Proc. of 5th International Joint Conference on Genetic Algorithms (1993)