# Dynamic load balancing for client server assignment in distributed system using genetical gorithm

**Arepalli Peda Gopi*, V. Lakshman Narayana, N. Ashok Kumar**

*Vignan's Nirula Institute of Technology and Science for Women,*
*Peda Palakaluru, Guntur,Andhra Pradesh, Guntur, India*

*gopiarepalli2@gmail.com*

ABSTRACT. *Anetwork of computers consists a set of interconnected computers using an appropriate technique. In distributed systems every client and server is unique and has different processing capability. Each server is independent where resource allocation is an important feature for the system to appear as single network. So the performance of system depends on allocation of work among the servers effectively. It is the combination of variousfactors likelatency, throughput, consistency, reliability and performance. The concept of dynamic load balancing can be introduced to efficiently manage the factors to be fulfilled in a distributed network. Every clients in the network benefit from dynamic load balancing. In turn all tasks benefit from load balancing. The load balancing comprises of both physical and logical features. The time, cost, performance must be optimized through load balancing. The paper describes a model for load balancing in the system to manage the performance through internetindistributedsystems.Thisproposedalgorithmcanbe applied to n-processor dynamic systems. This will prove effective to reduce the serverload.*

RÉSUMÉ. *Un réseau d'ordinateurs est constitué d'un ensemble d'ordinateurs interconnectés utilisant une technique appropriée. Dans les systèmes distribués, chaque client et serveur est unique et dispose d'une capacité de traitement différente. Chaque serveur est indépendant lorsque l'allocation des ressources est une caractéristique importante pour que le système apparaisse en tant que réseau unique. Ainsi, la performance du système dépend de la répartition efficace du travail entre les serveurs. C'est la combinaison de divers facteurs tels que la latence, le débit, la cohérence, la fiabilité et les performances. Le concept d'équilibrage dynamique de la charge peut être introduit pour gérer efficacement les facteurs à remplir dans un réseau distribué. Chaque client du réseau bénéficie d'un équilibrage de charge dynamique. À leur tour, toutes les tâches bénéficient d'un équilibrage de charge. L'équilibrage de charge comprend à la fois des caractéristiques physiques et logiques. Le temps, le coût et les performances doivent être optimisés grâce à l'équilibrage de la charge. Cet article décrit un modèle d'équilibrage de charge dans le système pour gérer les performances via Internet dans des systèmes distribués. Cet algorithme proposé peut être appliqué aux systèmes dynamiques à n processeurs. Cela prouvera efficace pour réduire la charge du serveur.*

KEYWORDS*: distributed systems, dynamic load balancing, client-server assignment, networking, network traffic, server load, genetic algorithm.*

MOTS-CLÉS*: systèmes distribués, équilibrage dynamique de la charge, affectation client-serveur, réseau, trafic réseau, charge du serveur, algorithme génétique.*

DOI:10.3166/ISI.23.6.87-98 © 2018 Lavoisier

## 1. Introduction

The distributed systems have hardware, software and data distributed along the network. The performance, cost, throughput are major factors considered in a distributed system. These factors need to be improved constantly for the network to produce better results. The distributed system already have a major advantage over the parallel systems, hence they prove to be a better alternative to the parallel systems. Scheduling and load balancing are the most important challenges faced by the distributed systems when the demand on the network increases.

The paper mainly focuses on the dynamic load distribution technique for client server assignment using the genetic algorithm. The paper describes load balancing techniques for distributed systems over the internet where the client server allocation of clients in a network is fluctuating. We study a method for load balancing that fulfills the parameters of latency, throughput, consistency, reliability and performance.

The goal of load balancing is for each server to perform an equal and evenly distributed share of the total work load. In many applications the work distribution can be allocated previously and the load balancing structure can be built in a specific application. This is static load balancing. However in real time scenario, we consider the dynamic load on server over the internet in distributed system. Hence various factors come into consideration. These factors require a reliable run time processing. The following are some major issues that are analyzed during load balancing:

(1). In distributed network the server have finite capacity and bandwidth. Their processing units maybe different hence load balancing needs to decide the migration of clients request.

(2). The server capacity may vary.

(3). Each client request may consists of several smaller request and each of them can have different execution times.

(4). The load on each server and the network can vary from time to time based on the workload brought about by the clients.

## 2. Related work

The load balancing strategies reported in the literature survey are organized under a structure to show how different approaches are used to address the same load balancing issues. The algorithm attributes are also considered. More attention is given to a distributed system that has a rapidly changing environment. For this environment adaptive scheduling is one of the ways to maintain a consistency level of performance. The fundamentals of adaptive scheduling are described and a methodology for the design of adaptive load balancing algorithms is outlined. Load balancing is a major problem that is studied extensively from a long period of time. Load balancing strategies are grouped into two major categories namely – those for applications where new tasks are created and scheduled during execution (i.e. task scheduling) and another for iterative applications with persistent load patterns (i.e. periodic load balancing). Lots of work has been done to study the various scalable load balancing strategies of task scheduling, where applications can be expressed through the use of task pools A load balancing approach with centralized Load Balancer and two back-end servers. The load sharing algorithm is based on master-slave technique where the client can be termed as the slave and the server is the master. The performance of the CSA algorithm will be used as a base for evaluation of distributed algorithms . The CSA algorithm is based on the simulated annealing framework which is shown highly effective in solving many large scale combinatorial optimization problems..Using the DSA algorithm, the servers keep track of their local information such as the total load, the inter server communication loads, and exchange these information with all other servers. Once a server receives all information of other servers, it will compute the global parameters and be able to anticipate the change in the global objective function value for moving a user from one server to another. Client Server assignment problem is an instance of clustering problem. Clients and their communication pattern can be denoted as a graph, with vertices representing the clients and the edges between two vertices representing the communication between the respective clients. Communication frequency between two clients can be represented by the weight of the edge between the corresponding vertices(Jiang *et al*., 2013; Jiang and Huang, 2012).

## 3. Various load bala ncing techniques

### *3.1. Load sharing algorithm*

The load sharing algorithm is based on master-slave technique where the client can be termed as the slave and the server is the master. The master is responsible to divide the requests based on a criteria. The criteria depends on fixed, variable or adaptive work load. In fixed, the approach is fixed like static technique. Same server is assigned to every server irrespective of any different factors. In variable approach, the request maybe variable and is variable from start to end of processing. For adaptive approach, the system load is taken into consideration (Wiki and Partitioner, 2013).

### *3.2. Centralized simulated annealing algorithm*

The performance of the CSA algorithm will be used as a base for evaluation of distributed algorithms. The CSA algorithm is based on the simulated annealing framework which is shown highly effective in solving many large scale combinatorial optimization problems (Kirkpatrick and Vecchi, 1983). The CSA algorithm is a type of stochastic greedy search in which the probability of searching in the next conjuration is based on the objective value at the current and the next conjuration. Central to the design of the CSA algorithm is the gradually decreasing temperature parameter *Tb*, which allows for the algorithm to explore many conjuration that are possible before settling down to the approximately optimal conjuration. Also, the probabilistic search allows the CSA algorithm to overcome the local minimums (Nishida and Nguyen, 2011; Sabin *et al*., 2004).

### *3.3. Distributed simulated algorithm(DSA)*

In many practical systems, the number of users can be thousands or even millions. Thus, we propose a Distributed Simulated Annealing (DSA) algorithm as follows. Assume that each server has local information on its users as described It. Using the DSA algorithm, the servers keep track of their local information such as the total load, the inter server communication loads, and exchange these information with all other servers. Once a server receives all information of other servers, it will compute the global parameters and be able to anticipate the change in the global objective function value for moving a user from one server to another. Just as in CSA algorithm, DSA algorithm moves one user from one server to another probabilistically according to the simulated annealing method. To perform the simulated annealing process in a distributed manner, at each iteration, a server is selected uniformly at random. Next (Ga and Johnson, 1979; Kernighan and Lin, 1970), the chosen server selects its users and reassigns them to another server probabilistically based on local computation as in (3) (6). The only difference between DSA from CSA is that for DSA, only users (*V*(*i*)) belonging to the chosen server *i*can be moved while for CSA, any user can be moved. Consequently, the CSA algorithm has the potential to converge to a good assignment faster than the DSA algorithm since at each iteration, it has the potential to take a better move due to more options. On the other hand, the DSA algorithm does not need a centralized controller to keep track of all the global information (Johnso *et al*., 1989).

### *3.4. Clustering algorithms*

Client Server assignment problem is an instance of clustering problem. Clients and their communication pattern can be denoted as a graph, with vertices representing the clients and the edges between two vertices representing the communication between the respective clients (Bui and Moon, 1996).

Communication frequency between two clients can be represented by the weight of the edge between the corresponding vertices. Clustering algorithm forms fixed 0 number of clusters of clients based on a given objective. Objective of the clustering algorithm in this paper is to minimize the amount of inter group communication and balance the sum of weights of all edges in the group(Martin *et al.*, 1991).

### *3.5. Round robin algorithm*

The assignment of task in Round Robin algorithm is sequential and even among all nodes. All of the active tasks are assigned to computing nodes based on Round Robin order, hence the computing nodes are chosen in series and will be back to the first computing node if the last computing node has been reached (Jiang and Jiang, 2009). Each node maintains its load index locally independent of allocations from remote node. Inter- process communication is not required. Useful for jobs of equal processing time and nodes of same capabilities. Not useful when tasks have unequal processing time. Not useful when nodes have different capacities (Newman, 2006).

### *3.6. Randomized algorithm*

Randomized Algorithm(RA)uses random numbers in selecting computing nodes for processing, without having any information about the current or previous load on the node. The computing nodes are selected randomly following random numbers generated based on a statistic distribution (Jiang and Jiang, 2009; Yorozu *et al.*, 1987).

It works well for particular special purpose applications.

No inter process communication is required.

It is not considered elegant solution.

Maximum response time among all algorithms.

## 4. Geneticalgorithm

### 4.1. Methodology

In the client given below, the client request input to the system. The request is given to the load balancer. The load balancer accepts the request. This request is processed by the load balancer and it allocates it to the server (Saab, 2000).

Taking the factors into consideration the load balancing can be generalized into four basic steps:

Monitoring server load and state

Exchanging load information between servers in the network.

Calculating the new work distribution

Transfer of workload.

Transfer of work load.

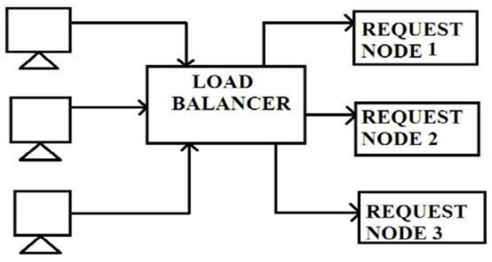

*Figure 1. Load balancer architecture*

## 5. Proposed systemarchitecture

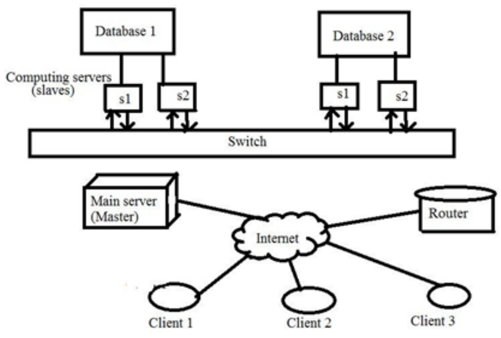

*Figure 2. Proposed system architecture*

The proposed system is basically made up of three main components, namely, load balancer, database computation, generic algorithm unit. The client and server communicate through these components. The client sends request to the server through the load balancer. The Load balancer handles this request and allocates it to the server that is available. The free server takes the request. If the server 1 is busy, the request is passed on to the second server and so on. The load balancer uses generic algorithm in our generic system. The load balancer is connected to the database computation unit. This unit is responsible for the load balancing using generic algorithm. The load balancing algorithm is responsible for balance the total system load by transferring the workload from heavily loaded nodes to lightly loaded nodes. This ensure good overall performance with respect to some specific metric of system performance. This allocation is done dynamically. Hence, the drawback of clustering algorithm is overcome in this approach (Deriche *et al.*, 1990; Ldberg, 1989).

### *5.1. Proposed system architecture*

A Genetic Algorithm (GA) is a search algorithm based on the principles of evolution and natural genetics. GAs combine the exploitation of past results with the exploration of new areas of the search space. By using survival of the fittest techniques combined with a structured yet randomized information exchange, a GA can mimic some of the innovative flair of a human search. A generation is a collection of artificial creatures (strings). In every new generation, a set of strings is created using information from the previous ones. Occasionally, a new part is tried for good measure. GAs are randomized, but they are not simple random walks. By contrast, GAs work from a database of points simultaneously (a population of strings), climbing many peaks in parallel. The probability of finding a false peak is reduced compared to methods that go point to point. The mechanics of a simple GA are surprisingly simple, involving nothing more complex than copying strings and swapping partial strings. Simplicity of operation and power of effect are two main attractions of the GA approach. The effectiveness of the GA depends upon an appropriate mix of exploration and exploitation. Three operators to achieve this are: selection, crossover, and mutation (Magrini *et al*., 2018).

Taking the factors into consideration the load balancing can be generalized into four basic steps:

Monitoring server load and state

Exchanging load information between servers in the network.

Calculating the new work distribution

Transfer of workload.

Genetic algorithms, as powerful and broadly applicable stochastic search and optimization techniques, are the most widely known types of evolutionary computation methods today. In general, a genetic algorithm has five basic components as follows:

An encoding method that is a genetic representation (genotype) of solutions to the program (Sharma and Singh, 2008).

A way to create an initial population of individuals (chromosomes).

An evaluation function, rating solutions in terms of their fitness, and a selection mechanism.

The genetic operators (crossover and mutation) that alter the genetic composition of offspring during reproduction (Mitchell, 2004).

Values for the parameters of genetic algorithm.

#### *5.1.1. Ge notype*

In the GA-Based algorithms each chromosome corresponds to a solution to the problem. The genetic representation of individuals is called Genotype.

### *5.1.2. Initial population*

A genetic algorithm starts with a set of individuals called initial population. Most GA-Based algorithms generate initial population randomly.

### *5.1.3. Selection*

The selection process used here is based on spinning the roulette wheel, which each chromosome in the population has a slot sized in proportion to its fitness. Each time we  require an offspring, a simple spin of the weighted roulette wheel gives a parent chromosome.

### *5.1.4. Crossover*

Crossover is generally used to exchange portions between strings. Crossover is not always affected, the invocation of the crossover depends on the probability of the crossover Pc. Two crossover operators are given. The GA uses one of them, which is decided randomly (Lu and Lau, 1994).

## ***5.2. Algorithm***

```
Initialize all the clients and servers
Check the load on each server
Evaluation for request message
Generic operation
Repeatfromstep2untiltaskqueueisempty
End
```

## ***5.3. Mathematical model***

This mathematical model describes the superset S which consists of 3 subsets namely: I {set of inputs to state the inputs provided or acquired by the system}, O {set of outputs to state the corresponding response given by the various modules or the system as a whole} F {a set of functions or software modules used for acquisition, processing and generating suitable responses and/or corresponding actions of the proposed system}

The Input set I = {inputs or data acquired by the client request to server}

The Output set O = {response or outputs generated after processing the acquired data from the input set I of the load balancer}

Therefore, O = {It provides the optimal solution to the client }

The Function set F = {It provide best path to client request and process faster.}

Thus, the overall system follows the above mentioned subsets for its working and this is combines in one single superset that is set S={I,F,O}. Using these I/O and functionality models, the dynamic load balancing is to be designed to achieve its objectives.

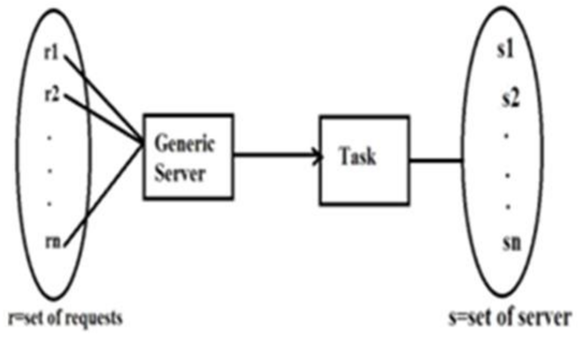

*Figure 3. Mathematical representation*

### *5.4. Performanceanalysis*

#### *5.4.1. Total communication cost.*

To make correct processor selection for selecting the processor in a distributed environment, first the analysis of load measure is required from each processor. The load measure will determine the load on each server. This is then given to the network to determine each server. The value to determine the processor load is analyzed and given to each processor. This calculated value is considered for further computations. The value should be computed swiftly and adapt for each change. The calculation of this value must happen frequently to clear off old data. The policy should be generalized to fit a variety of operating system environments (Wang and Morris, 1985).

#### *5.4.2. Load balancing technique*

Scheduling and load balancing are the most important challenges faced by the distributed systems when the demand on the network increases. The current distributed systems are linked together with medium to handle delay and load exchange. The computational power of any distributed system lies within the elements by working effectively so the large amount of load is allocated effectively and fairly among every server (Gilbert and Palmucci, 1991).

However there are uncertainties associated with the dynamic amount of traffic, conjunction and other unpredictable factors. The goal of load balancing is for each server to perform an equal and evenly distributed share of the total work load. In many applications the work distribution can be allocated previously and the load balancing structure can be built in a specific application.

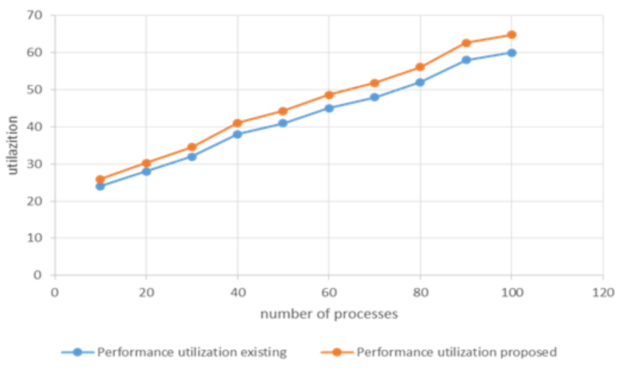

*Figure 4. Process utilization*

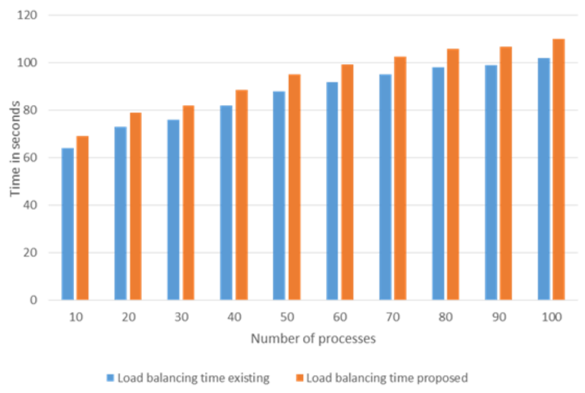

*Figure 5. Balancing time*

This is static load balancing. The load on each server and the network can vary from time to time based on the work load brought about by the clients. This paper focuses on the dynamic load distribution technique. The paper describes load balancing techniques for distributed systems over the internet where the client server allocation of tasks is fluctuating.

We study a method for load balancing that fulfills the parameters of latency, throughput, consistency, reliability and performance

## 6. Conclusion

The main aim in distributed system is to e xecute the process at minimum t ime limit. This is the most important factor that can be considered in cost calculation. This means that dynamicload balancing technique must achieve higher success as compared to the previously used load balancing techniques. Some of the major advantages achieved are that it reduces the clients waiting time and thus minimizes response time. The approach maximizes utilizat ion of server resources and maximizes server throughput. It improves reliability, and stability of the network. Long starvation is avoided for small requests. In load balancing overall system performance is enhanced by improving the performance of each server.

## Reference

Bui T. N., Moon B. R. (1996). Geneticalgorithmandgraph partitioning. *IEEE Trans. Comput*, Vol. 45, No. 7, pp. 841-855. https://doi.org/10.1109/12.508322

Deriche M., Huang M. K., Tsai Q. T. (1990). Dynamic load-balancing in distributed heterogeneous systems under stationary and bursty traffics. *Proc. 32ndMidwestSymp. Circuits and Systems*, Vol. 1, pp. 669-672. https://doi.org/10.1109/MWSCAS.1989.101943

Garey M. R., Johnson D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. *San Francisco, CA, USA: Freeman*, Vol. 23.

Goldberg D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Reading, Mass: Addison-Wesley*. https://doi.org/10.1111/j.1365-2486.2009.02080.x

Jiang Y., Huang Z. (2012). Therich get richer: Preferential attachment in the task allocation of cooperative networked multiagentsystems with resource caching. *IEEE Trans. Syst, Man, Cybern. A, Syst, Humans*, Vol. 42, No. 5, pp. 1040-1052. http://doi.org/10.1109/TSMCA.2012.2186439

Jiang Y., Jiang J. (2009). Conte xtualresource negotiation- based task allocation and load balancing in complex software systems. *IEEE Trans. Parallel Distrib*, Vol. 20, No. 5, pp. 641-653. http://dx.doi.org/10.1109/TPDS.2008.133

Jiang Y., Zhou Y., Wang W. (2013). Task allocation for undependable multiagent systems in social networks. *IEEE Trans. Parallel Distrib. Syst*, Vol. 24, No. 8, pp. 1671-1681, http://doi.org/10.1109/TPDS.2012.249.

Johnson D. S., Aragon C. R., Geoch L. A. M, Schevon C. (1989). Optimization bysimulated annealing: An experimentalevaluation, Part I, graphpartitioning. *Oper. Res*, Vol. 37, No. 6, pp. 865-892. http://dx.doi.org/10.1287/opre.37.6.865

Johnson D. S., Aragon C. R., Geoch L. A. M., Schevon C. (1989). Optimization bysimulated annealing: An experimentalevaluation. PartI, graphpartitioning. *Oper. Res*, Vol. 37, No. 6, pp. 865-892. https://doi.org/10.1287/opre.37.6.865

Kernighan B. W., Lin S. (1970). Anef_cient heuristic procedure for partitioning graphs. *Bell Syst. Tech*. Vol. 49, No. 2, pp. 291-307. http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x

Kirkpatrick S., Vecchi M. P. (1983). Optimization by simmulated annealing. *Science*, Vol. 220, No. 4598, pp. 671-680. http://dx.doi.org/10.1126/science.220.4598.671

Lu C., Lau S. M. (1994). A performance study on load balancing algorithms with process migration. *In Proceedings, IEEE TENCON 1994*, pp. 357-364.

Magrini A., Lazzari S., Marenco L., Guazzi G. (2018). Cost optimal analysis of energy refurbishment actions depending on the local climate and its variations. *Mathematical Modelling of Engineering Problems*, Vol. 5, No. 3, pp. 268-274. https://doi.org/10.18280/mmep.050321

Martin O., Otto S. W., Felten E. W. (1991). Large-step Markov chains for the traveling sales man problem. *Complex Syst*, Vol. 5, No. 3, pp. 299-326. https://digitalcommons.ohsu.edu/cgi/viewcontent.cgi?referer=https://scholar.google.com.hk/&httpsredir=1&article=1015&context=csetech

Mitchell M. (2004). An introduction to genetic algorithms: Easterm Economy. *Edition*.

Newman M. E. J. (2006). Modularity and community structure in networks. *Proc. Nat. Acad. Sci. USA*, Vol. 103, No. 23, pp. 8577-8582. https://doi.org/10.1073/pnas.0601602103

Nishida H., Nguyen T. (2011). Optimal client-server assignment for internet distributed systems. *in Proc. 20th ICCCN*, Vol. 24, No. 3. pp. 1-6. http://dx.doi.org/10.1109/ICCCN.2011.6006007

Saab Y. (2000). A new effective and ef_cient multi-level partitioning algorithm. *in Proc. Conf. Design, Autom. Test Eur*, pp. 112-116. https://doi.org/10.1145/343647.343715

Sabin G., Sahasrabudhe V., Sadayappan P. (2004). On fairness in distributed job scheduling across multiple sites. *in Proc. IEEEInt. Conf. Cluster Comput, Sep*, pp. 35-44. http://dx.doi.org/10.1109/CLUSTR.2004.1392599

Sharma S., Singh S., Sharma M. (2008). Performance analysis of load balancing algorithms. world academy of science. *Engineering and Technology,* Vol. 2, No. 2. pp. 367-370. http://waset.org/publications/5537

Syswerda G., Palmucci J. (1991). The application of genetic algorithms to resource scheduling. P*roc. Fourth International Conference on Genetic Algorithms*, pp. 502-508.

Wang Y. T., Morris R. J. T. (1985). Load sharing in distributed systems. *IEEE Trans. Comput*, Vol. 34, No. 3, pp. 204-217. https://doi.org/10.1109/TC.1985.1676564

Yorozu T., Hirano M., Oka K., Tagawa Y. (1987). Electron spectroscopy studies on magneto-optical media and plastic substrate interface. *IEEE Transl. J. Magn. Japan*, Vol. 2, No. 8, pp. 740-741. http://doi.org/10.1109/TJMJ.1987.4549593