A New Task Scheduling Algorithm using Firefly and Simulated Annealing Algorithms in Cloud Computing

Fakhrosadat Fanian
Department of Computer Engineering,Kerman Branch,
Islamic Azad University,
Kerman,Iran

Vahid Khatibi Bardsiri
Department of Computer Engineering, Bardsir Branch,
Islamic Azad University,
Kerman,Iran

Mohammad Shokouhifar
Department of Electrical Engineering,
Shahid Beheshti University G.C.
Tehran,Iran

Abstract—**Task scheduling is a challenging and important issue, which considering increases in data sizes and large volumes of data, has turned into an NP-hard problem. This has attracted the attention of many researchers throughout the world since cloud environments are in fact homogenous systems for maintaining and processing practical applications needed by users. Thus, task scheduling has become extremely important in order to provide better services to users. In this regard, the present study aims at providing a new task-scheduling algorithm using both firefly and simulated annealing algorithms. This algorithm takes advantage of the merits of both firefly and simulated annealing algorithms. Moreover, efforts have been made in regards to changing the primary population or primary solutions for the firefly algorithm. The presented algorithm uses a better primary solution. Local search was another aspect considered for the new algorithm. The presented algorithm was compared and evaluated against common algorithms. As indicated by the results, compared to other algorithms, the presented method performs effectively better in reducing to make span using different number of tasks and virtual machines.**

Index Terms—**Firefly; make span; simulated annealing; task scheduling; cloud**

I. Introduction

Cloud computing has recently been introduced as a new technology for users. From a historical perspective, the first computers used were those of the first generation, mainly the mainframes. As time went by, these computers became smaller with higher processing power until personal computers were developed and distributed amongst all users. Next, the technology of networks providing higher processing power emerged by connecting a few small personal computers. However, processing requirements increased exponentially and the need for bigger computing systems became crucially essential.

Thus, smaller networks were privately joined to form bigger networks across the internet. By then, millions of users had access to the internet mostly never using their computers processing power to its full capacity and preferring to give away the idle processing time of their computers to be used for computational tasks. Therefore, many small computational resources were connected; however, it was not possible to completely use these sources within the created network, since these computers were not purposefully created to handle commercial applications. This led to the establishment of a new approach. An approach in which the details were hidden from the user and users did not need to allocate or control infrastructural cloud technologies they were using [1].

In layman's terms, cloud computing was a new user-driven model based on users demands with easy access to flexible and configurable computational sources such as networks, servers, storage areas, practical applications, and services, such that this access is rapidly made with the minimum need for resource management or intervention by the service provider. In general, cloud-computing users are not proprietors of the cloud infrastructure, but rather rent these services from third parties in order to avoid large costs [2]. These users utilize the existing resources in the form of services and only pay for whichever sources they are using [3]. Like any other public service, the costs are based on the amount of service the user requires [4]. Hence, considering that hundreds of people make use of virtual machines, manual allocation of computational sources for different tasks is very troublesome in cloud technology [5]. This highlights the need for an efficient algorithm for task scheduling in cloud environments. This scheduler must be consistent with environmental changes and change in task types [6]. At any moment, millions of users are demanding cloud resources. Scheduling this number of tasks is a serious challenge in cloud processing environments, especially since allocation of optimized resources or task scheduling in clouds must be done in accordance with optimized number and need of systems within the cloud environment so as to maintain the clouds integrity. On the other hand, this scheduling must be done in a way minimizing energy consumption within the cloud. Ergo, this study tries to present an efficient algorithm for

Identify applicable funding agency here. If none, delete this.

task scheduling in clouds using the combination of both firefly and simulated annealing optimization algorithms. This study is organized as follows: Section II reviews related and previous works. Section III discusses and presents a new method. Section IV contains the results of the presented algorithm, and finally Section V gives a conclusion of the entire study

II. Review of Literature

Cloud computing is currently made up of various aspects, making it a challenging subject. Thus, many researchers have made efforts to investigate the various aspects of cloud computing [7] and have tried to make virtualization and automation technologies focus on improving services in clouds. In this regard, task scheduling and reducing energy consumption in clouds is a very challenging issue for these environments. Kusic [8] investigated the issue of energy management in virtual heterogeneous environments and used Kalman filters as a method complying with system demands and as a means for prediction and actual implementation. Kalman filters are used for estimating future demands in order to predict system status and allocate resources accordingly. On the other hand, some researchers focused on the effects of scheduling virtual machines on I/O virtual performance and emphasized on monitoring optimization for better I/O performance. For instance, Ongaro et al. [9] studied the effect of virtual machine observer on performance and presented an idea for arranging processors in an executional queue based on remaining and current value. They ultimately presented an optimization algorithm for scheduling even I/O distribution. However, this scheduling procedure did not take into account the workload and the reallocation of virtual machines. In [10], Kim presented a task-aware scheduler with an emphasis on developing I/O performance.This scheduler did not consider the heterogeneous workload and variety of weights only focusing on I/O performance. Liao [11] presented a scheduler for scheduling real time applications for supporting respond time, and instead of placing the processor at the end of the executive queue, this method compute the state in which the virtual processor is inserted based on its delay. Goiri [12] presented a task dynamic scheduling policy for allocating informed sources at cloud data centers. The presented scheduler worked to stabilize workload by connecting large tasks of individual devices with necessary hardware, in order to maintain service quality. In other words, these methods reduced energy consumption at data centers turning off servers. Wood [13] presented a virtual machine-driven scheduling policy based on using resources including processor, memory, and subnet components. However, instead of optimizing and scheduling operational energy, his study mainly focused on developing an algorithm for avoiding local traps. Dorigo et al. presented the ACO algorithm [14]. The ACO was a random search algorithm, which used positive feedback and followed actual ant colony behavior. In [15] this algorithm was used to allocate optimized sources for tasks in a dynamic cloud environment in order to minimize make span. Liu et al. [16] worked on a scheduling algorithm based on genetic and ant colony algorithms. They tried to make use of the advantages of both algorithms. This algorithm uses the global search in genetic algorithm in order to reach the optimized solution faster. It also utilizes initial values for pheromones in the ACO algorithm. Guo et al. [17] used a formulated particle swarm optimization (PSO) model for minimizing process costs. They also tried to use crossover and mutation functions of the genetic algorithm along with the PSO model. Lakro et al. [18] investigated various variables and their optimization in cloud computing environments. They tried to present a multi-variable optimization algorithm for scheduling and improving performance of data centers. Jia et al. [19] investigated scheduling of various tasks of different sizes on a set of parallel batch machine and presented a meta-heuristic algorithm based on max-min and ant system for minimizing make span.

III. Methodology and Suggested Algorithm

Cloud computing is one of the newest technologies today, which allows users to send their requests to clouds and pay a certain amount of fees based on the service provided. On the other hand, cloud environments are in fact homogenous systems suitably storing large applications and data for services. Considering this, scheduling of these data and large applications in these systems is of great importance. The present study tries to present a new algorithm based on firefly and simulated annealing algorithms called FA-SA in order to schedule tasks in clouds. The details of the suggested combination are expressed below. The general framework for this study is shown in Fig. 1.

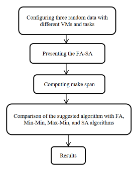

Configuring three random data with different VMs and tasks
Presenting the FA-SA
Computing make span
Comparison of the suggested algorithm with FA, Min-Min, Max-Min, and SA algorithms
Results

Fig. 1. General framework for the study

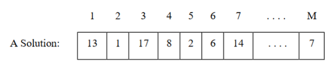
A. Problem Statement

The allocation of tasks to virtual machines in cloud computing systems is a problem, in which m number of tasks, V= t1, t2, . . . , tm are to be allocated to certain virtual machines. In this study, the total number of tasks are randomly selected from 10 to 100 tasks and categorized into three different data sets with different number of virtual machines. The tasks are made randomly. Also P= v1, v2, . . . , vn are the n virtual machines used. All systems are the same, meaning tasks are performed in a homogeneous environment.

B. Possible Solutions

This study uses a combination of firefly algorithm (FA) and simulated annealing (SA). The feasible solution in this study is a string of m characters, where m is the total number of tasks. According to (1), if task i is allocated to a virtual machine, j, the ith place in the relative string, has a value of “j”. 20 virtual

$$Solution_i = j \qquad \text{if task } i \text{ is allocated to machine } j \qquad (1)$$



	1	2	3	4	5	6	7	....	M
A Solution:	13	1	17	8	2	6	14	....	7

Fig. 2. A feasible solution: for example, the first task is given to machine I3 and the second task is given to machine 1

machines are considered for all m tasks. A feasible solution for the problem is shown in Fig. 2.

C. Objective function in the Suggested Algorithms (FA, SA and FA-SA)

As previously mentioned, an objective function is needed for all algorithms in order to schedule tasks and minimize amount of make span. Task scheduling is an optimization problem in which tasks are to be allocated to sources at certain times. In other words: n tasks, j1, j2, . . . , jn, with different sizes are to be allocated to m identical scheduler machines such that make span is minimized. Make span is defined as the total amount of time required to perform all tasks (after all tasks have been done). Recently, this problem has been introduced as a dynamic scheduling problem, in which for every task, the dynamic algorithm must use the existing information to make a decision before the next task comes. This is one of the most famous dynamic problems and the first for which a competitive analysis was presented by Graham in 1966 [20].

D. Overall Stages of Allocating Tasks to Virtual Machines using the Suggested FA-SA Algorithm

Evolutionary algorithms are generally based on population and make use of a very suitable global search strategy. The firefly algorithm [21] was used in this study. This algorithm is a meta-heuristic algorithm inspired by the behavior and motion of fireflies in nature. This algorithm is similar to other population-based algorithms and computes the optimized solution (or near to optimized) in an iterative manner. The algorithm starts by performing a search procedure in a randomly developed population. Each member of the population (location of each firefly in the search space) is a possible solution for the problem, which is shown in Fig. 2 according to (1). Each iteration in the FA algorithm has two main stages: Stage 1, evaluating the suitability of the solutions and Stage 2, updating the population (establishing a new population). These two stages are continuously performed in iteration until the termination criteria of the algorithm is satisfied. The termination condition in this study is the completion of all tasks. The FA algorithm is a population-based algorithm with the ability to perform a very suitable global search since it has a very high convergence rate and each firefly tries to find the best state individually; thus, it avoids local optimums and searches for the global optimum [22]. On the other hand, the SA algorithm has a very convenient local search procedure. It is for this reason that both of these algorithms were combined in this study to form the FA-SA algorithm in order to benefits from the advantages of both of these algorithms for performing a better scheduling of tasks in clouds. n the presented method, the FA algorithm initiates first in order to perform a global search in the search space. After the FA algorithm, the SA algorithm is executed to perform a local search near the previous solution provided by the FA algorithm. In other words, the initial population for the SA algorithm is not selected randomly, rather it gets the value provided by the FA algorithm which is in fact the optimum value provided by the FA algorithm. The general flowchart for the suggested method is shown in Fig. 3. The stages of the suggested algorithm will be explained in more detail in the following section.

1) Producing a random initial population for the FA algorithm): As previously mentioned, the first stage for all evolutionary algorithms is producing initial solutions, which are mostly done randomly. The initial solutions for the FA algorithm in this study are produced considering the following regulations:: 1) Perform the following stages for m iterations (where m is the total number of tasks): find the virtual machine(s) with the least termination time (since the data are random multiple machines may have the same value). 2) Perform the following stages for m iterations (where m is the total number of tasks): find the virtual machine(s) with the least termination time (since the data are random multiple machines may have the same value). 3) If a virtual machine is found, select the virtual machine, otherwise randomly select a virtual machine with the least termination time (since data are random multiple machines may produce the same value). 4) Search the initial data set (containing tasks and virtual machines) and find the virtual machine selected in stage 2 and choose the task with the least time from the unallocated tasks for that machine. 5) If a task exits with the least amount of time, select that task; otherwise, randomly select a task. 6) All tasks are assigned? 7) no, go to stage 1, otherwise terminate. In other words, each task is allocated to a virtual machine according to the regulations mentioned above. It is worth noting that since the data sets of this study are random in nature and according to the regulations, random selection is performed two times, the initial population or rather the initial solutions are different for each iteration, though due to the nature of the regulations, these initial solutions are near optimum.

2) Competency assessment for produced solutions: The solutions produced by the FA algorithm are evaluated in each iteration after the population has been updated. This evaluation works on the basis of the objective function. In order to evaluate each member of the population (each firefly), allocated tasks for each machine are considered first. Next, execution time on each machine is computed and finally termination time for all tasks are computed.:

3) Updating population in the FA Algorithm: The firefly algorithm was presented by Yang [23] and is inspired by the motion and behavior of fireflies in nature. Fireflies produce short and rhythmic lights. These rhythmic lights, light radiation rate, and distance are what make two fireflies attract each other. Light intensity at a distance of r from the light source has a relationship with the reverse squared amount *of distance. In the firefly algorithm, light can be considered as the objective function to be optimized. In short, the firefly algorithm is based on the following three principles::* 1) All fireflies are unisexual and each firefly attracts the other firefly despite their sexuality. 2) Attraction of fireflies is proportional to their radiance such that the firefly with less light intensity is attracted to the one with higher light intensity, and if there is no firefly with a higher light intensity in the locality, fireflies move randomly. 3- The light intensity of fireflies is determined as the objective function [24], [25]. In FA algorithm, the location of each firefly in m-dimension space determines a solution for the optimization problem, where m is the number of optimization variables (total number of virtual machines). Considering that fireflies' location is defined in a continuous space, this study considers the location of each firefly within the (0,n] range, where n is the total number of machines. Therefore, each dimension value for each firefly is a value from 0 to n. In each iteration of the evaluation stage, each dimension for each firefly is rounded up to the nearest natural number that is bigger than the current number. Therefore, evaluation of fireflies takes place in a discrete space. However, fireflies' motion and attraction are done continuously. After determining the time for the solution of each firefly using relative objective function, radiance of each firefly i is computed using (2) (since radiance in this algorithm denotes higher competency, every firefly with a lower objective function has a higher), where and denote error rate (objective function) and radiance for the ith firefly, respectively. Each iteration selects fireflies with the highest radiance. Then, each of the remaining fireflies moves towards the nearest radiant firefly. The distance between firefly i and firefly j is computed by (3) : where xi and xj are the locations of the ith and jth fireflies, respectively. d is the number of optimization variables, which in this case is equal to the total number of tasks. Movement of firefly i towards firefly j is formulated as (4). The second expression in this statement shows the attraction if firefly i towards firefly j and the third expression shows a random movement in the attraction procedure. and are two static variables that configure the effect of the two expressions when firefly i moves. determines the way fireflies move and is usually selected between 0 and infinity. ( ) d) Addition of local search to the FA algorithm: Three types of local searches were added to the firefly algorithm in this study, where each type is used with a probability of 1/3 for each iteration (generation). These searches include exchange mutation, inverted exchange mutation, and a suggested local search called hybrid max-min to exchange (HHME). These procedures are explained in more detail in the following section. Exchange mutation: In this procedure, two machines are randomly selected and their tasks are exchanged [26]. Fig. 4 shows the search procedure used in the proposed algorithm. Inverted exchange mutation: two machines are randomly selected and their tasks are inverted [26]. Fig. 5. shows the search procedure used in the proposed algorithm, before inverted exchange mutation (current solution) and after the inverted exchange mutation (new solution). 7410268653Before mutation: 7462018653:After the mutation

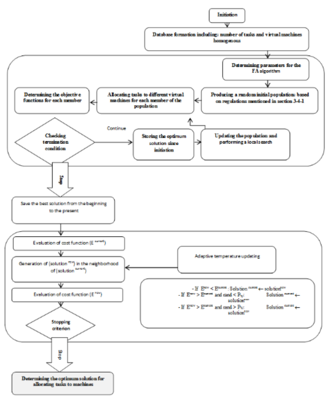

Initiation
Database formation including: number of tasks and virtual machines homogenous
Determining parameters for the FA algorithm
Producing a random initial population: based on regulations mentioned in section 3-4-1
Allocating tasks to different virtual machines for each member of the population
Determining the objective functions for each member
Checking termination condition
Continue
Storing the optimum solution since initiation
Updating the population and performing a local search
Stop
Save the best solution from the beginning to the present
Evaluation of cost function (E current)
Generation of (solution new) in the neighborhood of (solution current)
Evaluation of cost function (E new)
Adaptive temperature updating
Stopping criterion
Stop
Determining the optimum solution for allocating tasks to machines

Fig. 3. General flowchart for the FA-SA algorithm

Fig. 5. search procedure used in the suggested algorithm, before inverted exchange mutation (current solution) and after inverted exchange mutation (new solution). Hybrid max-min to exchange (HMME): In this procedure, virtual machines with the highest and lowest value of termination time are selected and; 1) a minimum task from the machine with the highest termination time is transferred to the machine with the lowest termination time or 2) a task is randomly selected from the machine with the highest termination time and transferred to the machine with the lowest termination time or 3) a maximum task from the machine with the highest termination time is transferred to the machine with the lowest termination time. e) Local search using SA algorithm: the simulated annealing, presented in 1983 [27] algorithm, is an optimization algorithm that uses local search. The gradual annealing technique is used by metallurgists in order to reach a state where the solid material is sorted properly with minimized energy. In this technique, the substance is placed at high temperature then cooled down gradually. During this algorithm, each state s in the search space is similar to a state of a physical system and the E(s) function which must be minimized is similar to the internal energy of the system in that specific state. The purpose of this procedure is to transfer the system from its initial random state to a state where the system has the lowest amount of energy. For an optimization problem, the algorithm starts with a random initial solution and gradually moves towards neighboring solutions in an iterative manner. In each iteration, if the neighbor solution (solutionnew) is better than the current solution (solutioncurrent), the algorithm selects the former solution as the new current solution. Otherwise, the algorithm selects the new solution with a probability of of , where is the difference between the objective function value of the current solution and that of the neighboring solution and T is the temperature variable. This algorithm iterates for each temperature, and gradually decreases the temperature. The temperature is initially high so that the possibility of choosing worse solutions is high. However, with the gradual decrease in temperature, the possibility of choosing worse solutions decreases and better solutions are selected. Therefore, the algorithm converges to a proper solution. As seen in Fig.

type data	Number of task	Number of VM	Min$_{Task}$	Max$_{Task}$
Data 1	100	8	10	100
Data 2	200	20	10	100
Data 3	500	20	10	100

TABLE I
SPECIFICATIONS OF RANDOMLY SELECTED DATASETS

6, in this study, random changes in one dimension of the solution have been selected for local search. The possibility for performing this procedure is defined as Pm, where the value of each dimension in the current solution changes with the probability of Pm

The suggested FA-SA algorithm was simulated in MATLAB and was compared with three data sets and the following algorithms: min-min, max-min, firefly, and simulated annealing. Comparison results are provided below. A. Datasets Three different datasets were randomly selected in this study considering that the minimum and maximum numbers of tasks were 10 and 100, respectively. The number of tasks and machines were chosen randomly such that the first dataset, name data1, contained 100 tasks and 8 homogeneous virtual machines randomly selected according to the mentioned criteria. Detailed specification of these three datasets are provided in Table 1

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

E. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

F. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [7].

G. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

H. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is “Heading 5”. Use “figure caption” for your Figure captions, and “table head” for your table title. Run-in heads, such as “Abstract”, will require you to apply a style (in this case, italic) in addition to the style provided by the drop down menu to differentiate the head from the text.

Text heads organize the topics on a relational, hierarchical basis. For example, the paper title is the primary text head because all subsequent material relates and elaborates on this one topic. If there are two or more sub-topics, the next level head (uppercase Roman numerals) should be used and, conversely, if there are not at least two sub-topics, then no subheads should be introduced.

I. Figures and Tables

a) Positioning Figures and Tables: Place figures and tables at the top and bottom of columns. Avoid placing them in the middle of columns. Large figures and tables may span across both columns. Figure captions should be below the figures; table heads should appear above the tables. Insert figures and tables after they are cited in the text. Use the abbreviation “Fig. 4”, even at the beginning of a sentence.

TABLE II
TABLE TYPE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 4. Photo of the owner

Figure Labels: Use 8 point Times New Roman for Figure labels. Use words rather than symbols or abbreviations when writing Figure axis labels to avoid confusing the reader. As an example, write the quantity “Magnetization”, or “Magnetization, M”, not just “M”. If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write “Magnetization (A/m)” or “Magnetization {A[m(1)]}”, not just “A/m”. Do not label axes with a ratio of quantities and units. For example, write “Temperature (K)”, not “Temperature/K”.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks …”. Instead, try “R. B. G. thanks…”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use “Ref. [3]” or “reference [3]” except at the beginning of a sentence: “Reference [3] was the first …”

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors’ names; do not use “et al.”. Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.