Optimization and Comparison of Human Methods for Solving the Rubik’s Cube

Govind Sharma   
*Something*  
*University of Waterloo*Waterloo, Canada  
[g45sharma@uwaterloo.ca](mailto:g45sharma@uwaterloo.ca)

Govind Sharma   
*Something*  
*University of Waterloo*Waterloo, Canada  
[g45sharma@uwaterloo.ca](mailto:g45sharma@uwaterloo.ca)

*Abstract*—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. *\*CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract*. (*Abstract*)

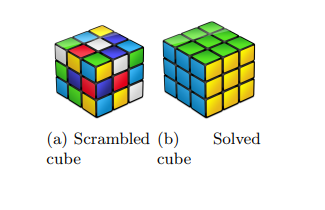
Keywords—component, formatting, style, styling, insert (key words)

# Introduction

The Rubik’s Cube (3D combination puzzle) was invented in 1974 by an Architecture professor Erno Rubik. Ever since its invention and conception as a toy, it has taken over the toy and puzzle industry by storm. It was a household object in the 1980s when everyone from children to adults were trying their hand at a Rubik’s Cube. Although its popularity declined at the turn of the century but recently it has picked up again as this toy has turned into a competitive game under the WCA (World Cubing Association) which organizes thousands of yearly events worldwide with events related to Rubik’s Cube puzzles of different shapes, sizes and stipulations.

The inspiration behind designing the cube was for professor Rubik to demonstrate a stable 3-dimensional design structure that can move in all directions. The first ever design was made using wood and rubber bands. But through many versions, he finally settled on a primitive version of the cube we see today. The 6 sides of the cube were colored for aesthetic purpose. He had no intentions of the structure being conceived as a puzzle. But after having done a few twists and turns on the cube, the professor soon realized that it was not at all straight forward to put the pieces back into their original position so as to have a solid color on each side of the cube. It was then that he realized that he has invented a revolutionary puzzle which both very easy to conceive and annoyingly difficult to solve. In fact, it is this innate simplicity of the puzzle that makes it so tempting as it firmly remains intractable for most people of try it. To say that Rubik’s Cube was a success would be an understatement of the highest order. With over 350 million copies of the cube being solved worldwide, the Rubik’s Cube takes the crown as the most successful toy/puzzle ever to sold in human history.

The classic version of the Rubik’s Cube also known as the Magic Cube, has the dimensions of 3x3x3. This simply means that the cube with 6 faces has 9 stickers of the same color on each side. The most common color scheme for the cube includes the colors yellow, green, white, blue, orange and red even though other color variations do exist. The most natural way of visualizing the cube is to see it in layers that can move together. The puzzle can be scrambled by rotating any of its layers on the fixed central pieces together. The rotations are 90 degree turns in any direction. A combination of these rotations of any arbitrary number of layers causes cube to be in a scrambled state.



Even though the number of ways to turn the cube is small but even with the limited possibilities of turning, the cube can have 43,252,003,274,489,856,000 combinations in total. Such a large number of possible legal states makes the cube an interesting challenge even for computers.

## Application Domain

This project delves into the application domain of Game Playing. As discussed, Rubik’s Cube is a competitive game nowadays and there are many methods suitable for humans to solve the cube as quickly as possible. These methods are broken down into various intuitively recognizable stages in order to make the whole solution tractable. The intended results from this project can help draw new insights into how each stage in a solution can be optimized and therefore possibly improve the current methods. The mechanics of the Rubik’s cube allow the pieces to move in conjunction to each other hence making it a classic example of a problem in Group Theory.

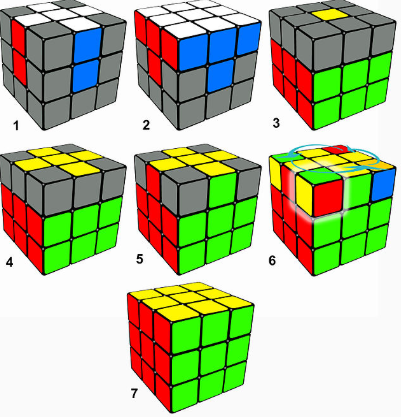
## Problem To Be Solved

The following problems are of interest to this project:

1. Can there be a system that can produce solutions to the Rubik’s Cube which can be interpreted or learned by human solvers?

There are already many solutions of the Rubik’s Cube that solve it completely and as efficiently as mathematically possible (God’s Number). Some of these methods use popular search techniques like IDA\* in conjunction with popular algorithms designed specifically for computer solutions of the Rubik’s Cube such as the Thistlethwaite’s algorithm (and its variations). But the solutions produced by these methods are beyond our comprehension. There is not much intuitive interpretation to the moves in these solutions. The major reason for this lack of interpretability is the fact that these computer algorithms by design try to put as many pieces as possible back into their intended positions on the cube in one go i.e. the rotations are intended to effect as many pieces as possible. Therefore, the purpose of each move becomes very obscure from a human perspective.

This issue could be solved if the instead of solving the entire cube using these efficient techniques, we solve the cube partially. Specifically, the solution can be divided into stages that correspond to popular human methods. Each stage of the solve can be attempted using intelligent search techniques to get a sequence of moves that solve only that stage of the solution. Knitting the sequence of moves generated for each stage together will produce a solution for the entire cube and by design this solution should be close to what a human can comprehend.

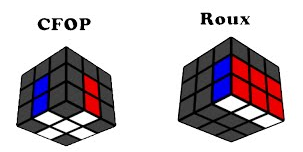


1. Which family of methods: Block Building or Layer by Layer produce better solutions using intelligent heuristic search techniques.

There are many popular methods of speed solving the Rubik’s Cube. Some of these methods are CFOP (Fridrich method), Roux method, ZZ method, Petrus method, etc. But most of these methods fall into two categories or families. These are layer by layer category of methods and the block building family of methods. In the layer by layer category, the methods involved focus on solving each layer of the cube at a time. The CFOP method, for example solved the first two layers together before moving on to the last layer. On the other hand, the block building methods do not limit themselves to consider only layers. They rather focus on building blocks of solid colors and of varying shapes and sizes across the cube as efficiently as possible and then join these blocks to get to completed layers of the cube. An example of a method in this category is the Roux method. It has been empirically established in the speed cubing community that the block building methods on average require a smaller number of moves to solve the cube as compared to the layer by layer methods. But the layer by layer methods are generally faster as they involve a much more rigid procedure which can be practiced into muscle memory of the solvers and produce better times. This happens because the block building methods leave more space for the solver to innovate and build blocks with freedom and since there is more thinking involved in these methods, the solves themselves tend to be longer. The layer by layer methods don’t require as much thinking as the block building methods and therefore, are faster generally.

It should be noted that even though the speed and efficiency tradeoff between the two families of methods is well established in the cubing community, the margins between these methods are very small and that is why both types of methods enjoy a heathy subscription. And since speed is the primary objective of solving the cube, the layer by layer methods are the most popular.

Due to the difference in approach of these methods, the stages involved are different. Therefore, it is interesting to see how the intended results from this project effect different types of methods. To be specific, comparisons will be made between the two most popular methods of each type i.e. CFOP for the layer by layer type and Roux for the block building type. Since, the solutions are produced by computers and time of the solves are irrelevant, we will focus on the move count for the solutions.



# Techniques to solve the problem

## Current Techniques

There have been many (successful) attempts of coming up with efficient solutions to the Rubik’s Cube. Plenty of research has been done in this regard and there many programs across the web that can solve the Rubik’s Cube in the most efficient way possible. Following sections offer a brief summary of some of techniques described across the literature pertaining this particular topic.

### Depth First Search

Depth First Search is a fundamental search/traversal technique popular in the domain of maze finding. Straight forward implementation and simplicity of the data structures involved in DFS lead it to be a very popular technique used in Artificial Intelligence. As mentioned, this technique is applied the most in graph search/traversal.

The solution to a Rubik’s Cube can also be thought of as a graph problem where the starting node of the graph is the cube in its given scrambled position. Each rotation of the cube leads to another node in the graph with the rotation being the edge between the two nodes. The goal is of course to find a path from the starting node to a solved state in this graph. Since it is not advisable to have the entire graph of all possible nodes and moves in memory, the solution can be obtained by growing the tree as various move sequences are applied from the initial scrambled stage. By the very nature of the Depth First Search, one branch of the graph is to be fully exhausted in search of the solution before moving onto a another one. By design, the only path stored in memory for this type of search will be the path between the starting node and the current node at the end of the branch (specified to a specific depth). The time complexity of this search is where is the branching factor i.e. the average number of moves (edges) that can be applied to a given state of the cube in the graph and is the specified depth for graph. The space complexity is where is the branching factor and is the maximum number of moves in a branch.

The obvious advantage of this technique is the efficient (linear) space complexity as only the nodes in one branch are needed in memory for this solution to work. Another benefit is that the solution can be found without exploring the whole tree/graph. But since the state space of the Rubik’s Cube is extremely large, it is probable that DFS will spend most of the its time searching in branches that do not actually contain the solution and since branches can be infinitely long (as there is no limit to the number of moves that can be applied to a Rubik’s Cube) this could take a very long time even if the branches are chopped off at a certain depth (knowing that if a solution were to be found, it would have happened before a certain move count i.e. God’s Number). Such pruning of the branches can be applied to reduce the time complexity but since this method only keeps track of a single branch there is no way of determining if the current branch has morphed into a state that previously seen and subsequently discarded in a previous branch.

### Breadth First Search

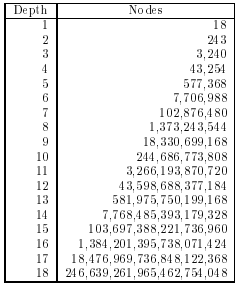
Breadth First Search is another fundamental graph search/traversal technique applied heavily in Artificial Intelligence and game playing domain. As was the case with DFS, BFS too has a straight forward implementation requiring simple data structures. Perceiving the solution to the Rubik’s Cube as a graph search problem is easy as described above. And therefore, BFS can be applied to find the solution.

Unlike in DFS, Breadth First Search looks at all nodes of the graph at a given level before moving in depth in the tree. This means that graph is grown in level by level instead of just growing a single branch. And since the state space for the Rubik’s Cube is very large, the nodes stored in memory for BFS could be very large and often not tractable even for super computers of moderate processing power and memory. The advantage of using Breadth First Search is that it will always find the most efficient solution by design i.e. a solution that uses the least number of moves between the scrambled state and solved state of the cube. This means that if the solution to scramble exist, BFS will find it given enough time and space. The time complexity of this method depends on the number of nodes in the state space and it also depends on the branching factor and the allowed depth of the graph . The time complexity is . But the space complexity of BFS is not linear (unlike in DFS) as all the nodes in the graph are needed to be in memory until a solution is found. This makes the space complexity to also be .

As mentioned, the inherent advantage of using BFS is that it will find the shortest path to the solution and that is complete i.e. if a solution exists, BFS will find it eventually. Since all the nodes in the graph are stored in memory this makes it easier to check for loops in the search space i.e. ignore states that are already present in the graph. But since space required is too much, BFS alone can not be used to solve the cube even on decently powered machines.

### *Iterative Deepening A\**

Exhaustive searches like BFS and DFS are not very effective specially on problems with large state spaces. In order to tackle the problem of large search spaces where exhaustive and blind searches are not practical, we need to apply search methods that make use of the domain knowledge and nudge the searches in the right direction so as to ensure that the solution state is found as soon as possible without having to waste processing time and memory on branches or parts of the graph that are away from the solution. Algorithms that use heuristics to nudge the search towards the solution have better performance over such state spaces. A\* and IDA\* are algorithms of this kind. The following table shows the number of nodes at different depth level of an exhaustive search in the Rubik’s Cube search space.



Even with A\* algorithm, the state space for the Rubik’s Cube is too large and the algorithm is bound to get stuck due to memory limitations. Therefore, it is prudent to apply a heuristic based approach but one which involves an exhaustive search. IDA\* algorithm (as the name suggests) performs an iterative search that grows in depth during subsequent iterations. This algorithm is based on depth search but the gain in effectiveness are made by the pruning of branches based on heuristic functions.

Deciding which heuristics to use is very important consideration. One possible heuristic is Manhattan distance applied to a flattened-out version of the 3-dimensional cube. This includes finding the minimum number of moves needed to place each piece of the puzzle in it right place (as in the solved state) with the correct orientation (position of the colors in the piece). To make this heuristic admissible, this value has to be divided by 8 because each move changes the position of 8 pieces in total (4 corners and 4 edges). A better variation of this heuristic is to calculate the Manhattan distance of corners and edges separately. The expected value of this heuristic is 5.5 (5 for 12 corners each 3 for the 8 edges).

Using these primitive heuristics increase the efficiency of the solution but only to a particular extent (specifically, search is fast till depth of 14 and beyond that the node count supersedes the efficiency gains). A better heuristic function is needed for IDA\* to be more effective in this setting. Heuristic are generally considered to be functions that are computed for each node as it is encountered in the search space. But it is often the case for large enough search spaces that the heuristics are precomputed and stored in look up tables. This ensures that calculation of heuristic for each node in the graph takes a constant and cheap time and thus improving the efficiency of the algorithm greatly. Consider a single corner piece on the Rubik’s Cube. The maximum number of moves required to solve a piece (put it in its solved position in the correct color orientation) for each possible initial position of the piece can be found by applying a simple breadth first search on just that one piece (leaving out the rest of the pieces in the cube). Performing such reduces searches enables the enumeration of the maximum move count of the solution for each corner of the cube. The same search can be performed on the edge pieces as well. Finally, the results of these searches can be cumulated in look up tables that can be made accessible to the IDA\* algorithm during its search for the solution of the entire cube. Such lookup tables for heuristics are referred to as pattern tables. Since, the maximum move count is considered separately for each piece, the heuristic calculation for each move on the Rubik’s Cube at a given state can only be done by considering the maximum of the move counts of the individual pieces. This will ensure that the overall heuristic is admissible.

Using the lookup tables as heuristics and applying the IDA\* algorithm on 10 scrambles (scrambled using 100 random moves each), the results as reported in the original text are promising as each scramble was solved in reasonable time with search space remaining stable in terms of the node count at each level.

### Thistlethwaite's Algorithm

Morwen B. Thistlethwaite is a professor of mathematics out of Britain. He invented the Thistlethwaite’s algorithm in 1981. The algorithm is notable because despite producing very shot move count solution, it has the potential of being morphed into a variation that can be replicated for human solvers. The original method although remains rather complex and is beyond human comprehension. This algorithm in its original form is meant for computer application. There are variations to this algorithm like the Human Thistlethwaite and Heise Method that human interpretable while being very close to the efficiency of the original. The major selling point of this algorithm is that it does not rely heavily on search algorithms to solve the cube. This is partly because at the time its conception, the computing power available was not apt enough to handle search procedures on the problem the size of the Rubik’s Cube. Rather it focuses on the mathematics of the Rubik’s Cube and applies graph theory in order to solve the cube. This algorithm is more important from theoretical standpoint (as better procedures are available) but for a long time, it remained the method that produced the lowest move count solutions to the Rubik’s Cube.

This method approaches the solution in a very different way than the popular layer by layer or block building approaches. It does not focus on putting together groups of pieces (layer or block) in place sequentially. Rather it looks to improve the position of all of the pieces of cube simultaneously. It works by restricting the possibilities for each piece at each step and finally having only one possible solved state. As discussed in the notation section, to any given state of the cube, we can apply 18 different moves. In other words, a combination of these 18 moves can solve any state of the cube. Thistlethwaite’s algorithm works on reducing the size of this set of moves needed to solve the cube at each stage. The first step is to arrive at a position of the cube that can be solved using just the <U, D, L, R, F2, B2> moves. This step is also termed as edge orientation because after completion of this step, all edges in the cube are oriented correctly. Next stage is reduction to <U, D, L2, R2, F2, B2>. This known as corner orientation. Next stage is reduction to <U2, D2, L2, R2, F2, B2>. This is followed by the final step which solves the cube i.e. no moves are required to solve the cube further.

The original algorithm solved the cube in at most 52 moves which is way more than the God’s Number. Improvements on this algorithm such as the Kociemba’s algorithm reduced this move count further.

### Kociemba’s algorithm

Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

# Prepare Your Paper Before Styling

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

## Abbreviations and Acronyms

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

## Units

* Use either SI (MKS) or CGS as primary units. (SI units are encouraged.) English units may be used as secondary units (in parentheses). An exception would be the use of English units as identifiers in trade, such as “3.5-inch disk drive”.
* Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity that you use in an equation.
* Do not mix complete spellings and abbreviations of units: “Wb/m2” or “webers per square meter”, not “webers/m2”. Spell out units when they appear in text: “. . . a few henries”, not “. . . a few H”.

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

* Use a zero before decimal points: “0.25”, not “.25”. Use “cm3”, not “cc”. (*bullet list*)

## Equations

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.

Number equations consecutively. Equation numbers, within parentheses, are to position flush right, as in (1), using a right tab stop. To make your equations more compact, you may use the solidus ( / ), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

*a**b* 

Note that the equation is centered using a center tab stop. 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 . . .”

## Some Common Mistakes

* The word “data” is plural, not singular.
* The subscript for the permeability of vacuum **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].

# Using the Template

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

## Authors and Affiliations

**The template 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).

### For papers with more than six authors: Add author names horizontally, moving to a third row if needed for more than 8 authors.

### For papers with less than six authors: To change the default, adjust the template as follows.

#### Selection: Highlight all author and affiliation lines.

#### Change number of columns: Select the Columns icon from the MS Word Standard toolbar and then select the correct number of columns from the selection palette.

#### Deletion: Delete the author and affiliation lines for the extra authors.

## 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. Styles named “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” are prescribed.

## Figures and Tables

#### 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. 1”, even at the beginning of a sentence.

1. Table Type Styles

| Table Head | Table Column Head | | |
| --- | --- | --- | --- |
| Table column subhead | Subhead | Subhead |
| copy | More table copya |  |  |

1. Sample of a Table footnote. (*Table footnote*)
2. Example of a figure caption. (*figure caption*)

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 *(Heading 5)*

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

The template will 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].

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. *(references)*
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.

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**