An Advanced Computer Algorithm for Implementation of TRIZ for Automated Decision-Making

*** Ashish Mangalampalli**
Scholar
Computer Science and Engineering Department
ICFAI Institute of Science and Technology, ICFAI University
4-102, Jeedimetla, Medchal Road
Hyderabad – 500 055, INDIA.
Email: amangalampalli2@yahoo.com

Surya K Pala
Consultant, Wipro Ltd
Bangalore, INDIA.

Srirama Moorthy Mangalampalli
Senior Manager, S S Consultants
Hyderabad, INDIA.

* Corresponding Author

Abstract

The Theory of Inventive Problem Solving (TRIZ) has been simplified and made more user-friendly using the Algorithm for TRIZ (ARIZ), Systematic Inventive Thinking (SIT), and Unified Structured Inventive Thinking (USIT). And like most other applications, TRIZ has been automated using various software applications developed by different companies. But this automation is not enough as no concrete computer algorithm exists which can be used to write highly sophisticated software to solve problems using TRIZ and its derivatives like SIT. Also ARIZ and SIT cannot be directly used to write software as they are computer-oriented. In this paper, we propose an advanced computer algorithm and a corresponding Information System framework which can be used to write software that can solve problems in any domain using the TRIZ methodology. The algorithm facilitates the programmer to develop various levels of software, from very basic to the most advanced – according to his requirements. This is achieved using advanced computer techniques like Neural Networks, Fuzzy Logic, Unsupervised Learning, and Supervised Learning.

Introduction

The Algorithm for TRIZ (ARIZ) [Altshuller, 1997, Domb and Tate, 1997, Mazur, 1996, Nakagawa, 2000, and Altshuller, 1992] focuses on how a person can apply it to a particular inventive problem in order to find a suitable solution. But ARIZ has been designed with a human user in mind. The level of details and the step-by-step explanation it provides is suitable for direct use by a human analyst or inventor. The same is the case with Systematic Inventive Thinking (SIT) [Horowitz and Maimon, 1997] and Unified Structured Inventive Thinking (USIT) [Sickafus, 1997 and Nakagawa, 2000]. Neither ARIZ, SIT nor USIT, in the present form, can be directly used to write code for a computer program to assist humans solve inventive problems. Actually, ARIZ, SIT, and USIT do not incorporate the usage of any software techniques or tools and are thus are not computer-oriented.

Moreover, software implementations of TRIZ or other methods for innovation have not yet attained this level of sophistication, still leading the user through the intermediate steps between their specific problem and a specific solution [Mann and Hey]. Thus, in this paper we have combined the essence of ARIZ and SIT and developed a step-by-step computer algorithm [Terninko, Zusman, and Zlotin, 1997] which can used to develop a TRIZ computer program in any popular computer language, like Java and Microsoft Visual C++. The algorithm has been developed keeping in mind the computer as the direct user or direct agent to solve an inventive problem. It also incorporates the latest computer technologies like Neural Networks, Fuzzy Logic, Supervised Learning and Unsupervised Learning. The paper also proposes an Information System framework compatible with the computer algorithm

The Computer Algorithm for TRIZ

The actual algorithm is divided into two parts which are explained below (Refer Fig 1). The algorithm provides the user the option to use Artificial Intelligence (AI) techniques, like using heuristics through Neural Networks and Fuzzy Logic. The data in the Contradiction Matrix as well as the data input by the user is stored by the program in the most fundamental and atomic form possible (emulates a neuron). This ensures that a Neural Network structure of the program is maintained. This structure lends robustness and stability to the program and makes the program execution faster and error-free. Neither does the Contradiction Matrix be static nor should it be the standard one that is generally used. It can be dynamic and customized according to the domain where TRIZ is applied. For each contradiction in the Matrix, there can be multiple rules which can be used to eliminate the contradiction, but in different ways. The rules involve the application of one or more Inventive Principles, either simultaneously or individually. A sample rule to resolve a contradiction is shown below:

```
def_rule (
        (slot inventive_principle_1    weight_1)
        (slot inventive_principle_2    weight_2)
        (slot inventive_principle_3    weight_3)
                ……..
        (slot inventive_principle_n    weight_n)
)
```

This algorithm lets the program apply various rules for each individual contradiction and generate different solutions with different parameters. The user can utilize the solution which best fits his situation.

Also the weights attached to each individual datum (neuron) help in implementing Supervised Learning and Unsupervised Learning, according to one's needs. Moreover, Fuzzy-Logic-based decisions can be made by using the weights. Thus, the proper assignment of weights to the data in the Contradiction Matrix is paramount in the decision-making process.

The algorithm uses two domains, namely specific domain and general domain [Mann and Hey]. The user enters data which is domain-specific, e.g. automobile or aircraft manufacturing. The program has to convert the user-input data from the specific domain to the general domain [Mann and Hey] so that TRIZ can be used to solve the problem. Once the Contradiction Matrix and the Inventive Principles have been used to resolve the contradictions in the general domain, the solution should be converted back from the general domain to the specific domain and displayed to the user [Mann and Hey]. Furthermore, the program need not necessarily use the Contradiction Matrix (39x39) and the Inventive Principles (40). In some cases, like the Services Industry, the programmer can design the software with slightly modified Contradiction Matrix and Inventive Principles. The algorithm has been designed to support even modified Contradiction Matrices and Inventive Principles.

Part I (Refer Fig 2)

1) Take input from the user in the required format. This step is as important as the actual computation process. Input of data in a proper format would make the actual computation process, to follow, much faster and easier to code and implement.

2) If the problem is not in its most elementary form, we try to break it using various techniques. The parameters specified by the user help us in doing so.

3) We apply S-Field Analysis and try to the break the problem into different discrete mini-problems in the operating zone and operating time.

   Sample format of each mini-problem:

```
def_prob (

        def_improving_feature (
                (slot time)
                (slot space)
                (slot parameter)
                (slot value)
        )

        def_worsening_feature (
                (slot time)
                (slot space)
                (slot parameter)
                (slot value)
        )
)
```

4) Once the problem is broken into its most elementary forms or mini-problems, we go to Part II. For each mini-problem, we apply the pseudo-code in Part II and get the results.

5) Once we get the results of all the mini-problems, we combine them to get a single comprehensive solution or solutions to the original problem. The combining process should take into account all the combinations of the results of all the mini-problems. The final comprehensive solution or solutions would be combined using the weights of the all the mini-problems involved and thus would also have a weight in between 0 and 1. A weight of 1 suggests that the solution is a perfect one for the problem and a weight of less than 1 suggests that the solution is a partial one.

Part II (Refer Fig 3)

1) Take input about the mini-problem from Part I.

2) If the solution (domain-specific) for the mini-problem is already present in the neural network system, then use the solution, else use the system to get a result.

3) If the problem is in its most elementary form (mini-problem), convert its parameters from the specific domain to the general domain.

4) Apply the Contradiction Matrix and the Inventive Principles to the mini-problem. When using the Contradiction Matrix and the Inventive Principles, we should use popular TRIZ techniques, like Smart Little People and Closed World Method, depending upon the our requirements and prevalent circumstances to generate proper results.

   Note: We attach weights to all the Inventive Principles and if required, make more than one combination of the Inventive Principles to generate multiple solutions for the same mini-problem.

   i) Using the Neural Network System, we compute the result with a scale of 0 to 1 (probability of the feasibility and correctness of the solution)

   ii) If the weight is 1, the result is a perfect solution for the mini-problem.

   iii) If the weight is less than 1, the result is a partial solution for the mini-problem.

5) Save the result for future use and for Unsupervised Learning.

Unsupervised Learning

Once a solution is selected from the pool of solutions offered by the system and is implemented, the user can provide feedback to the Neural Network Software on various aspects of the solution provided. This will help the system (software) to analyze itself and restructure its knowledgebase, mainly modifying rules and adjusting weights involved in them.

Supervised Learning

The user can provide training sets to the system. These training sets would contain new domain-specific parameters and their conversion to general parameters. These can be absorbed by the system and used for future decision-making.

Conclusions

This algorithm can be used to automate the decision-making to solve an inventive problem using TRIZ. The heart of the TRIZ technique is the re-formulation a given problem in the form of a contradiction and consequently the resolution of this contradiction [Altshuller, 1997, Domb and Tate, 1997, and Mazur, 1996]. The algorithm has been designed keeping this in mind and resolves problems TRIZ does. Moreover, the algorithm goes a step further and gives the user the option to use any of the additional features, viz. Neural Networks, Fuzzy Logic, Unsupervised Learning, and Supervised Learning. Please note that these additional features and the use of weights in the rules are optional. The programmer can code his software without using any of these features – by using the basic algorithm and avoiding assigning weights to data in the Contradiction Matrix.
But the basic algorithm will not provide the advanced features mentioned above.

References

1) Altshuller, G S. 40 Principles: TRIZ Keys to Technical Innovation, Translated by Lev Shulyak. Technical Innovation Center, Worcester, MA. 1998.

2) Domb, E and Tate, K. 40 Inventive Principles with Examples. TRIZ Journal (www.triz-journal.com). July 1997.

3) Mazur, G. Theory of Inventive Problem Solving (TRIZ). Published on the Web. February 26, 1996, Update.

4) Mann, D and Hey, J. TRIZ as a global and local knowledge framework. www.creax.com.

5) Nakagawa, T. Approaches to Application of TRIZ in Japan. TRIZCON2000: The Second Annual AI TRIZ Conference, Sheraton Tara, Nashua, NH, USA. 2000, pp. 21-35.

6) Altshuller, G S. The History of ARIZ Development. Journal of TRIZ. 1992. Volume 3, No 1.

7) Sickafus, Ed N. Unified Structured Inventive Thinking: How to Invent. NTELLECK, Grosse Ile, MI. 1997. 488p.

8) Nakagawa, T. USIT: Creative Problem Solving Procedure in Simplified TRIZ. Design Engineering (JSDE). 2000. Volume 35.

9) Horowitz, R and Maimon, O. Creative Design Methodology and the SIT Method. ASME Design Engineering Technical Conference, Sacramento, California. Sept. 14-17, 1997.

10) Terninko, J, Zusman, A, and Zlotin, B. Step-by-Step TRIZ. Nottingham, NH. 1997.

11) Zhang, J, Chai, K, and Tan K. 40 Inventive Principles with Applications in Service Operations Management. The TRIZ Journal.

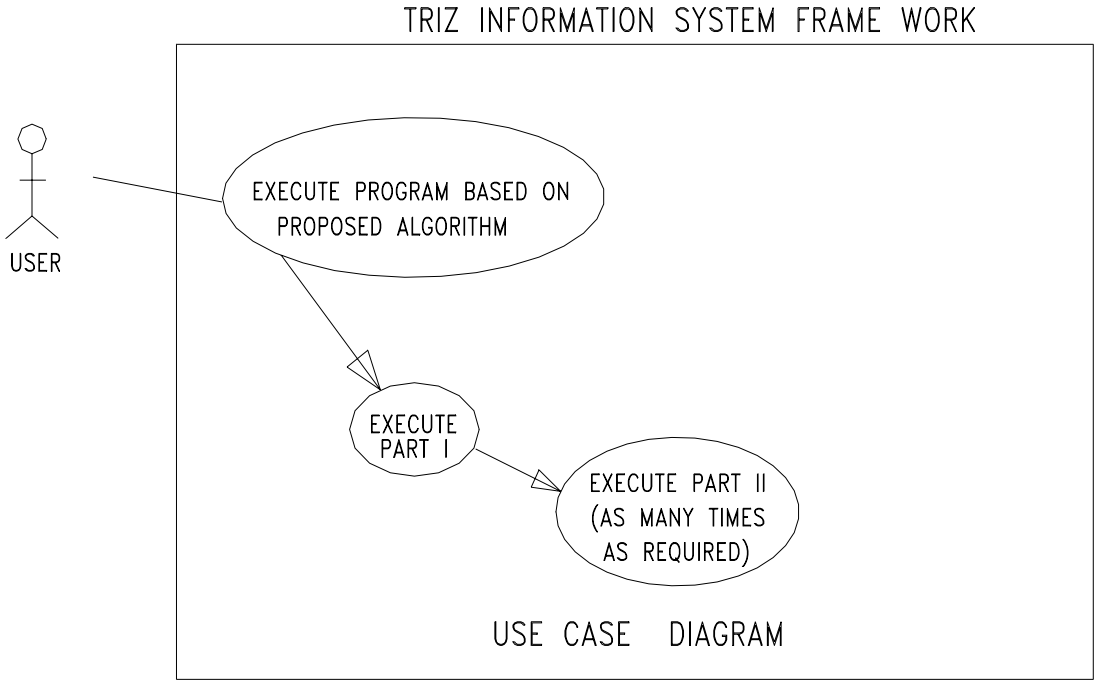

Fig 1
UML Use Case Diagram of the Proposed Computer Algorithm for TRIZ

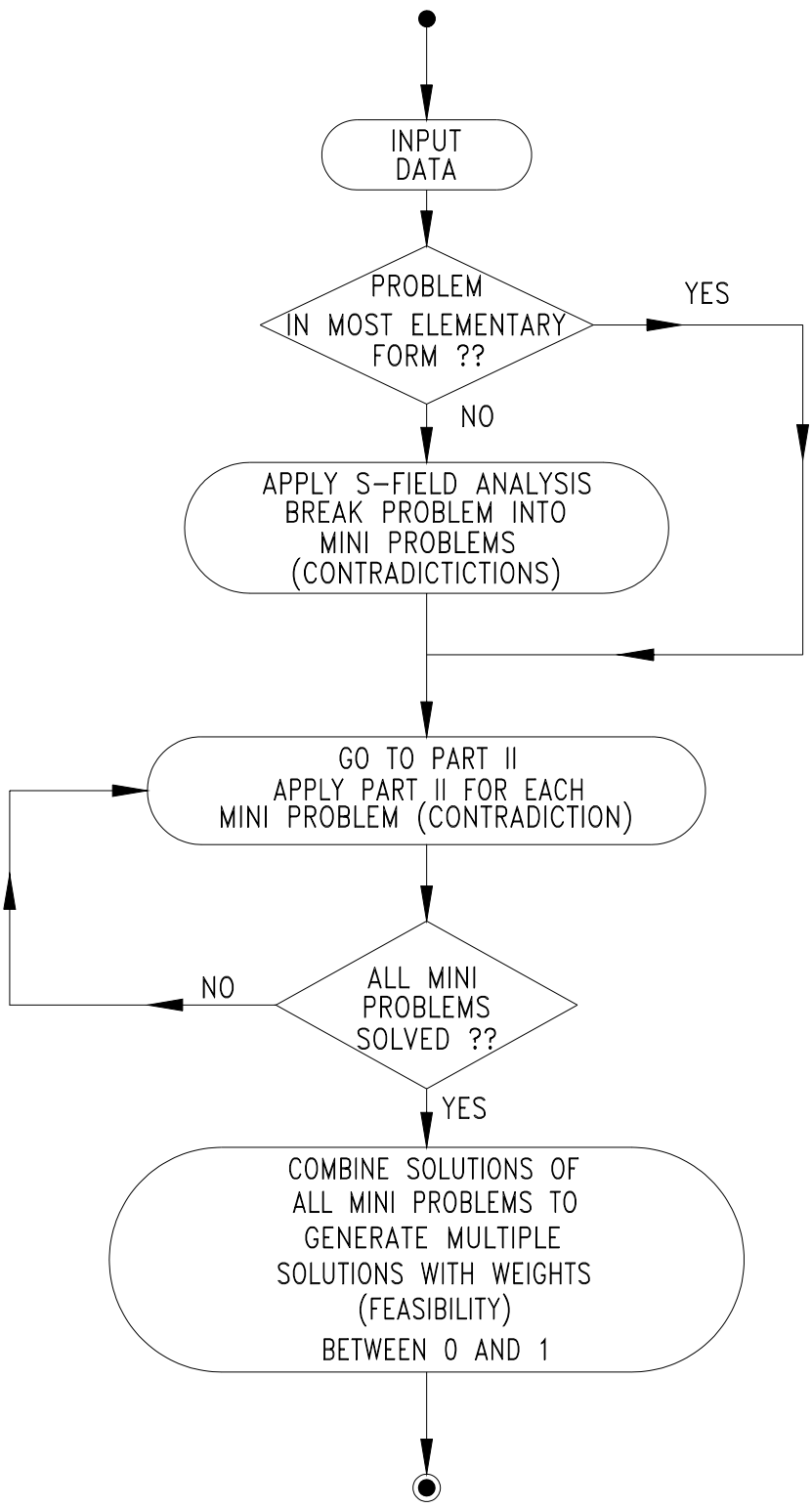

Fig 2
UML Activity Diagram of Part I of the Proposed Computer Algorithm for TRIZ

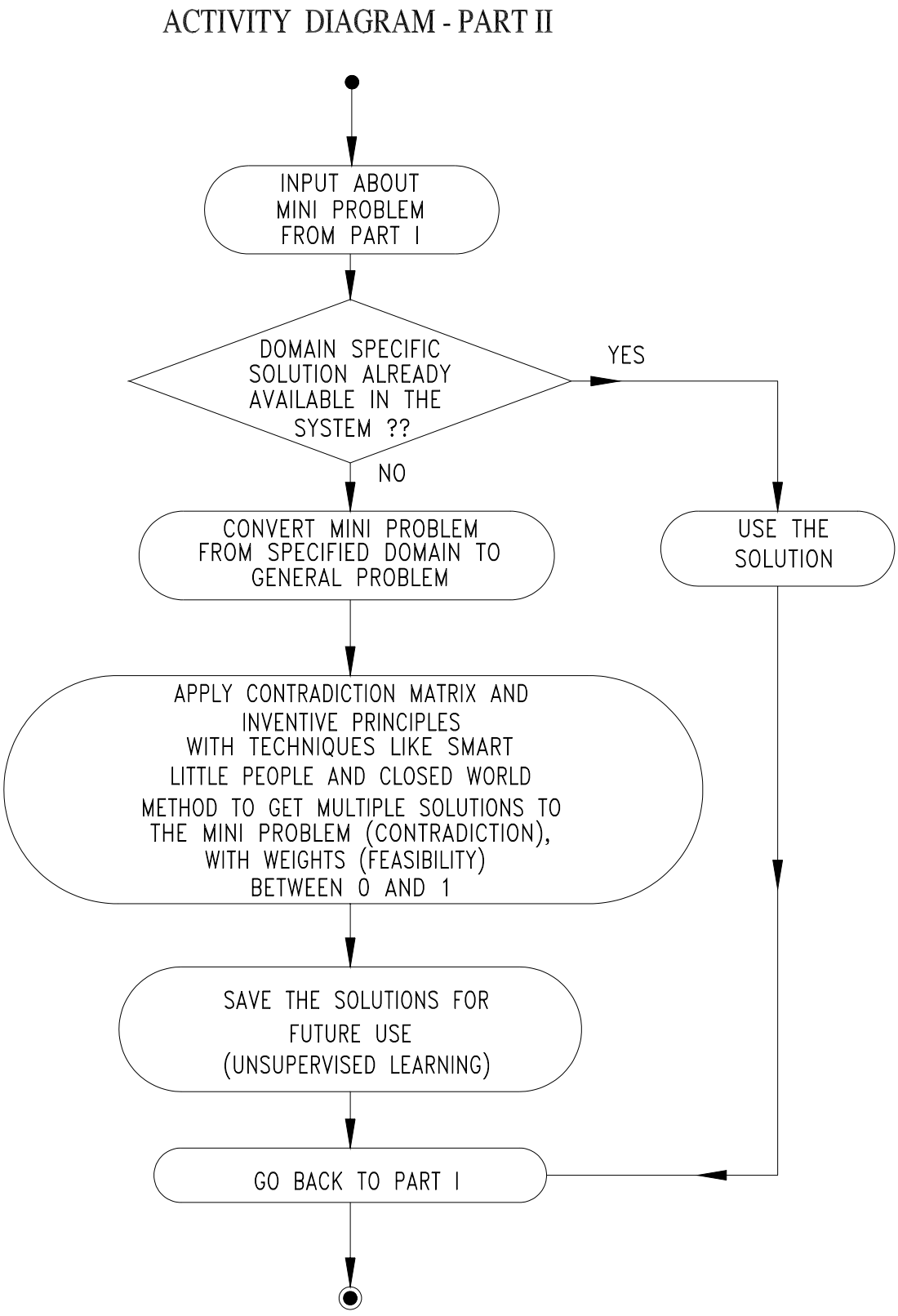

Fig 3
UML Activity Diagram of Part II of the Proposed Computer Algorithm for TRIZ