MULTI ROBOT BOX PUSHING USING VIKOR INDUCED DIFFERENTIAL EVOLUTION ALGORITHM

A Thesis
submitted in partial fulfillment of the requirement for the Degree of
Master of Technology in Intelligent Automation and Robotics,
Offered by
the Department of Electronics and Tele-Communication Engineering
Jadavpur University
May 2024

By
Kaustav Das

Registration No: 160229 of 2021-2022
Examination Roll No: M6IAR24010

Under the Guidance of
Dr. Pratyusha Rakshit

Department of Electronics & Telecommunication Engineering
Jadavpur University, Kolkata-700032. India

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE

This is to certify that the dissertation entitled **"Multi Robot Box Pushing Using Vikor Induced Differential Evolution Algorithm"** has been carried out by **KAUSTAV DAS** (University Registration No.:**160229** of 2023-2024) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of **Master of Technology in Intelligent Automation and Robotics**. The research results presented in the thesis have not been included in any other paper submitted for the award of any degree to any other University or Institute.

Dr. Pratyusha Rakshit
Assistant Professor, Electronics and
Telecommunication Engineering Department
Jadavpur University

Dr.Manotosh Biswas
Head of department
Electronics & Telecommunication

Prof. Dipak Laha
Dean,Faculty Council of Engineering and Technology
Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL★

The forgoing thesis is hereby approved as a creditable study of an engineering subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.

Committee on final examination
for the evaluation of the thesis

Signature of the Examiner

Signature of the Supervisor

★**Only in the case the thesis is approved**

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMIC THESIS

I hereby declare that the thesis entitled **"Multi Robot Box Pusing Using Vikor Induced Differential Evolution Algorithm" c**ontains literature Survey and original research work by the undersigned candidate,as Part of his Degree of Master of Technology in Intelligent Automation & Robotics.

All information was obtained and presented in accordance with academic regulations and ethical conduct

I also declare that, in accordance with these rules and practices, I have fully cited and referenced all materials and results not derived from this study.

Name:-Kaustav Das

Examination Roll No: M6IAR24010

Tiltle of Thesis:- Multi Robot Box Pusing Using Vikor Induced Differential Evolution Algorithm

----

Signature of Candidate

ACKNOWLEDGEMENT

This thesis entitled "**Multi Robot Box Pushing Using Vikor Induced Differential Evolution Algorithm**" is the result of the work whereby many people accompanied and supported me:my guide, friends, and lab managers. I'm very happy to be able to express my gratitude to everyone now.

First of all, I would like to express my sincere gratitude to my supervisor **Dr. Pratyusha Rakshit,** Assistant of the Department of Electronics and Telecommunication Engineering, at Jadavpur University for her valuable guidance, insightful suggestions, and support while conducting this thesis work as well as in the writing of this thesis. I have been very fortunate to have a guide like her. Her positivity, confidence, and ideas helped me to complete my thesis work successfully and she guided me as a guardian.I am also very grateful to our Head, **Prof.Amit Konar** for his continuous help and support and for letting me use all the available resources for my thesis work.

I want to express my gratitude to my parents and family also, as, without their sacrifices, I can't do anything. And their invaluable love, encouragement, and support make me, whatever I am today.

Kaustav Das
JADAVPUR UNIVERSITY
Kolkata-700032

CONTENTS

PAGE NO.

List of Abbreviations---------- 2

ABSTRACT---------- 3

Chapter 1 Introduction ---------- 4

1.1Optimization---------- 5

1.1.1Apllication of Optimization---------- 5

1.2 Bottleneck of Traditional Optimization---------- 6

1.3 Evolutionary Algorithms---------- 7

1.4 Classifications of EAs---------- 7

1.4.1 Single Objective Optimization---------- 8

1.4.1.1 Differential Evolution---------- 9

1.4.1.2 Particle Swarm Optimization---------- 11

1.4.1.3 Ant Colony Optimization---------- 14

1.4.1.4 COMPARISONS of EAs ---------- 18

1.4.2 Multi Objective Optimization---------- 19

1.4.2.A)NSGA II---------- 21

1.5 Non-Pareto Based Multi Objective Optimization-- 22

1.5.1 Lexicographic ordering---------- 22

1.5.2 Criterion selection- VEGA---------- 22

1.6 Difference Between Single and MOO---------- 23

1.7 Objective of Thesis---------- 23

1.8 Organization of Thesis---------- 24

1.9 REFERENCES---------- 24

Chapter 2 VIKOR Optimization---------- 26

2.1 VIKOR for Multi Criteria Decision---------- 27

2.2 History---------- 27

2.3 Brief Introduction of VIKOR Variants---------- 27

2.4 VIKOR Algorithm---------- 29

2.5 Code for VIKOR using Python Optimization-------- 32

2.6 Advantages and Disadvantages of VIKOR---------- 33

2.7 REFERENCES---------- 34

Chapter 3 Differential Evolutionary Algorithm---------- 35
3.1 Differential Evolutionary Algorithm (DE)---------- **36**
3.2 Computation Background---------- **37**
3.3 Why Differential Evolution?---------- **37**
3.4 Differential Evolution Algorithm---------- **38**
3.5 Limitations of Traditional Differential Evolution----- **42**
3.6 APPLICATIONS---------- **43**
3.7 REFERENCES---------- **44**

Chapter 4 Multi Robot Box Pushing Using VIKOR Induced Differential Evolution Algorithm---------- 46
4.1 MOOs for Multi Robot Box Pushing---------- **47**
4.2 Problem Formulation---------- **48**
4.3 VIKOR Induced Differential Evolution---------- **53**
4.4 Benefits of combining VIKOR with DE---------- **53**
4.5 Intregration of VIKOR induced DE for MOO--------- **54**
4.6 MULTI-ROBOT BOX-PUSHING USING VIKOR INDUCED DE---------- **55**
4.7 Results and Simulations---------- **57**
4.8 Choosing VIKOR Weight Factors---------- **58**
4.9 CONCLUSION---------- **61**
4.10 FUTURE SCOPE---------- **61**
4.11 REFERENCES---------- **62**

List of Figures

Figure No. Figure name	Page No.
Fig:1.1 Flow Diagram of Differential Evolution	9
Fig:1.2 Crossover in DE	10
Fig:1.3 Rastrigin function of 2 variables	12
Fig:1.4 Sphere Function for 2 variable	13
Fig 1.5: Data Flow to store particle of Swarm	13
Fig:1.6 Ant Colony Optimization Process	15
Fig:1.7 Flow chart of MOOs	20
Fig:1.8 Mapping solution space and objective function space.	20
Fig:1.9 Distance calculation of Crowding	21
Fig:1.10 NSGA II Crowding distance sorting	21
Fig: 3.1 Flowchart of DE	36
Fig: 3.2 Illustrating a simple DE mutation scheme in 2-D parameters	39
Fig: 3.3 Each possibly trial vectors formed due to uniform or binomial crossover between the mutant vectors and the target vectors	40
Fig: 4.1 MOO conflicts and selection	47
Fig: 4.2 Three steps in local movement of the box towards its goal	49
Fig: 4.3 Computation of distance by robots from obstacles and arena boundary	52
Fig: 4.4 Multi-robot box-pushing problem workspaces	58

List of Tables

Table No. Table name	Page No.
TABLE 1: VARIATION OF OBJECTIVES BASED ON WEIGHT FACTORS IN WORKSPACE 1	59
TABLE 2: VARIATION OF OBJECTIVES BASED ON WEIGHT FACTORS IN WORKSPACE 2	59
TABLE 3: COMPARATIVE PERFORMANCE ANALYSIS IN WORKSPACE 1	60
TABLE 4: COMPARATIVE PERFORMANCE ANALYSIS IN WORKSPACE 2	60

List of Abbreviations

Abbreviations	Name
EA	*Evolutionary Algorithm*
DE	*Differential Evolution*
ACO	*Ant Colony Optimization*
PSO	*Particle Swarm Optimization*
GA	*Genetic Algorithm*
MOO	*Multi Objective Optimization*
MMO	*Multi Modal Optimization*
MOOEAs	*multi-objective optimization evolutionary algorithms*
MCDM	*Multi criteria decision making*

ABSTRACT

Multi-Robot Box-Pushing Using VIKOR Induced Differential Evolution

Coordination of multiple agents is an active research area in robotics and automation. The multi-robot box pushing problem studied in this work is a common multi-agent coordination scenario. In the classic two-robot box-pushing problem, two identical robots push a box over a confined workspace with one or more static obstacles so that the task is completed as quickly as possible. A local trajectory must be planned by moving from a predefined starting position to a fixed target position. If possible, it will require time and minimal energy input. In this work, the problem is a multi-objective one with the goal of minimizing a total of four objectives, including (i) total time, (ii) total energy, and (iii) Euclidean distance from the current box position to that box position. It is formulated in an optimization framework target, and (iv) the penalty imposed if the box collides with an obstacle.

CHAPTER-*1*

Introduction of Thesis

Introduction

The chapter gives a brief overview of the importance of optimization.
It deals with the study of methologies of optimization.
It also explores the capabilities of the Evolutionary Algorithms for solving problems.

It consists of classification of Evolutionary Algorithms and the objective of the thesis work with the organization.

1.1 Optimization

Optimization is a fundamental process that lies at the heart of problem-solving in numerous domains, spanning engineering, finance, logistics, healthcare, telecommunications, and many others. It involves finding the best possible solution from a set of feasible options, aiming to maximize efficiency, minimize costs, optimize performance, or achieve desired objectives. The importance of optimization cannot be overstated, as it enables decision-makers to make informed choices and allocate resources effectively.

Optimization as a concept, has a rich history rooted in various disciplines and fields of human endeavour. At its core, optimization seeks to find the best possible solution within a set of constraints, whether it's maximizing efficiency, minimizing costs, or achieving optimal performance. This fundamental idea has played a pivotal role in shaping the advancement of science, engineering, economics, and even the arts. The origins of optimization can be traced back to ancient civilizations, where early human societies began applying rudimentary optimization principles to solve practical problems.

The seeds of optimization were sown in the fields of engineering and agriculture. Ancient architects and builders intuitively optimized the design and construction of structures to ensure stability and longevity. The Egyptian pyramids, Roman aqueducts, and ancient Chinese bridges stand as testament to early optimization efforts, even though the term itself had not yet been coined. With the rise of artificial intelligence and machine learning, optimization techniques found new applications in training neural networks, resource allocation, and even game theory. These techniques enabled computers to autonomously optimize their performance based on data and predefined objectives, this seemingly straightforward activity becomes challenging

1.1.1 Application of Optimization

Optimization has a wide range of practical applications across various domains,By employing optimization techniques, practitioners and researchers can tackle complex problems, explore large search spaces, and find optimal or near-optimal solutions.

The continual advancement of optimization methods, coupled with the increasing availability of computational power, has fuelled innovation and led to significant improvements across industries, enabling organizations to achieve higher

efficiency, improved performance, and better resource allocation. Here are some examples of how optimization is applied in different fields:

1.Engineering and Design:
Employed in designing aircraft, automobiles, buildings, and bridges to optimize their structural integrity, minimize material usage, and enhance energy efficiency.

2.Finance and Investment:
Optimization techniques used to construct portfolios that maximize returns while considering risk tolerance and investment constraints
optimizing the allocation of assets across various investment options

3.Operations Research :
Used in optimizing production processes, resource allocation, scheduling, and workforce management.

4.Energy Systems:
Used in determining optimal energy generation schedules and optimizing power grid operations.Used in managing energy demand to minimize costs and maximize system efficiency

1.2 Bottleneck of Traditional Optimization

Limitation of traditional optimization techniques is that they may become too complex and too large to solve efficiently and effectively as the problem size and dimensionality increase.

For example, solving a linear programming problem with millions of variables and constraints may require a lot of computational power and memory, and solving a nonlinear or integer programming problem .

Optimization solvers, like any tool, have their limitations. Some common limitations include:

Complexity: As the complexity of the optimization problem increases, the computational resources required to solve it also increase. This can lead to longer solution times or even infeasibility for very complex problems.

Non-convexity: Many real-world optimization problems are non-convex, meaning they have multiple local optima. Traditional solvers may struggle to find the global optimum in such cases.

Discrete variables: Some optimization solvers are designed for continuous variables and may struggle with problems involving discrete variables, such as integer programming.

Sensitivity to initial conditions: Some solvers may be sensitive to the initial conditions or starting point, leading to different solutions for similar but not identical problems.
Scalability: Solvers may not scale well to very large problem sizes, leading to practical limitations on the size of problems that can be effectively solved.

1.3 Evolutionary Algorithms (EAs)

Evolutionary algorithms (EAs) are a class of computational techniques inspired by the principles of biological evolution. These algorithms are used to solve complex optimization, search, and machine learning problems by simulating the process of natural selection and genetic variation. EAs work by iteratively evolving a population of potential solutions to a problem, gradually refining and improving them over successive generations.

Evolutionary algorithms have gained attention as effective optimization techniques inspired by natural evolution processes. These algorithms operate by maintaining a population of candidate solutions and iteratively evolving the population to search for optimal solutions.

The history of evolutionary algorithms is a journey through the development of computational techniques inspired by the process of biological evolution. These algorithms have emerged as powerful tools for solving complex optimization and search problems across various disciplines. The evolution of these algorithms mirrors the progression of computational paradigms and the quest to harness the principles of natural selection for computational problem-solving.

1.4 Classifications of EAs

There are several types of evolutionary algorithms (EAs) that have been developed to address different optimization problems and cater to specific requirements.

Here are some of the commonly used types of EAs-

1.4.1 Single-Objective-Optimization

Single-objective optimization technique finds the best ever solution for a problem with only one objective function to optimize. In this type of optimization, the goal is to minimize or maximize a single objective function, which can typically be represented as a mathematical equation. The optimization process seeks to find the input values or parameters that result in the optimal value for the objective function.

Various optimization algorithms can be used for single-objective optimization, such as:

Gradient-Based Methods: These methods rely on calculating the gradient of the objective function to iteratively improve the solution.

Evolutionary Algorithms: Algorithms like Genetic Algorithms, Particle Swarm Optimization, and Differential Evolution can be used to explore the solution space and find the optimal solution.

Linear and Nonlinear Programming: Techniques such as Linear Programming (LP) and Nonlinear Programming (NLP) can be employed for optimization with linear or nonlinear constraints.

Simulated Annealing: This probabilistic technique mimics the annealing process in metallurgy to find the global optimum.

Gradient-Free Optimization: Methods like Simulated Annealing, Random Search, and Bayesian Optimization do not rely on calculating gradients and are suitable for complex, non-smooth functions.

Single-objective optimization is commonly used in various fields such as engineering, finance, machine learning, and operations research to solve problems like parameter tuning, function optimization, resource allocation, and more. The goal is to find the best possible solution that optimizes the single objective function according to the specified criteria.

1.4.1.1) Differential-Evolution Technique(DE)

DE is an evolutionary algorithm that was developed by Rainer Storn and Kenneth Price in the mid-1990s. It has gained widespread recognition as an effective and efficient optimization method due to its simplicity, versatility, and strong performance on a variety of problem types.

In DE, a population of candidate solutions, often referred to as individuals or vectors, is maintained throughout the optimization process. Individuals are recognized by vectors, where,
The length of the vectors = No. of variables.

The population undergoes **mutation, crossover, and selection** operations to evolve and improve the solutions over successive generations.

Evolutionarily computating, different day dissolution (DE) is a method that optimizing a problem by trying for improving a candidate solution with regarding to a given measure of qualities. Such methods are common known as metaheuristically assumptions about optimized problems are not a thing, and they can search very large spaces of solutions. However, DE like metaheuristics don't guarantee finding an optimal solution.

DE used for dimensional really-valued functions, but gradient not used which means DE not require optimization problems being differentiably, unlike classic optimizing methods.

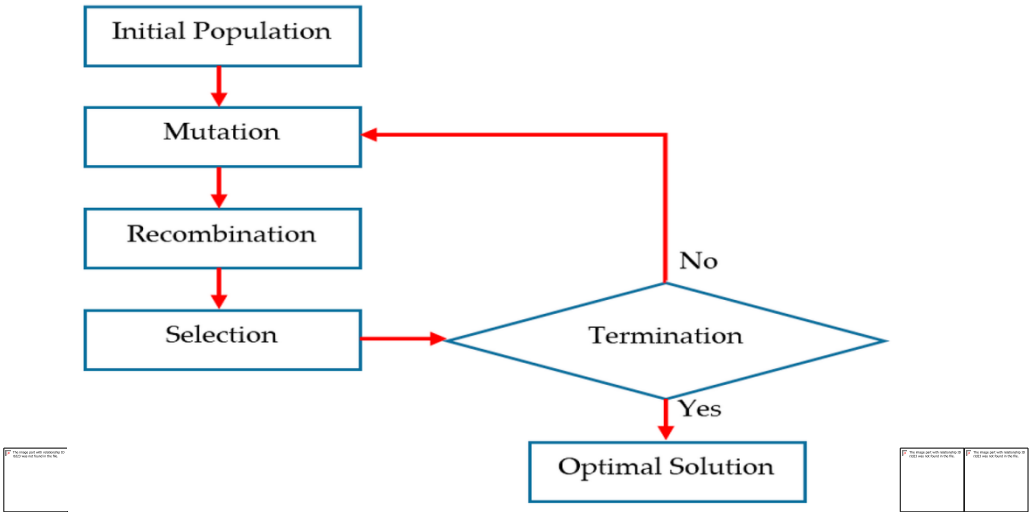


Fig:1.1 Flow Diagram of Differential Evolution

A basic varieties of DA algorithm working by generate a population of candidates answers. Those agents is being moving around in search-space by application easy mathematical equations to connect with positions of current agents from population. If new position of an agent is being enhancement it accepting and becoming part of population, else new position just discarded. Process repeating and expectation is to hopefully, but non-definitely, good outcome will ultimately be found.

Mutation- Mutation operation in DE plays a crucial role in exploring the search space. It generates new individuals by perturbing the existing population. DE uses a differential mutation strategy, where each candidate solution is perturbed by the weighted difference between two other randomly chosen individuals. The mutation is controlled by a parameter called the mutation factor (F), which determines the magnitude of the perturbation. By using the difference vectors, DE is able to explore different directions in the search space, which helps in escaping local optima and promoting diversity

Crossover-After mutation, DE performs a crossover operation to combine the mutated individual with a target individual to create a trial solution. The crossover determines how the components of the trial solution are inherited from the mutated individual and the target individual. The most commonly used crossover operator in DE is the binomial crossover, where each component of the trial solution is either inherited from the mutated individual or the target individual based on a randomly generated crossover probability. The binomial crossover ensures a good balance between exploration and exploitation by allowing both the mutated and target individuals to contribute to the trial solution.

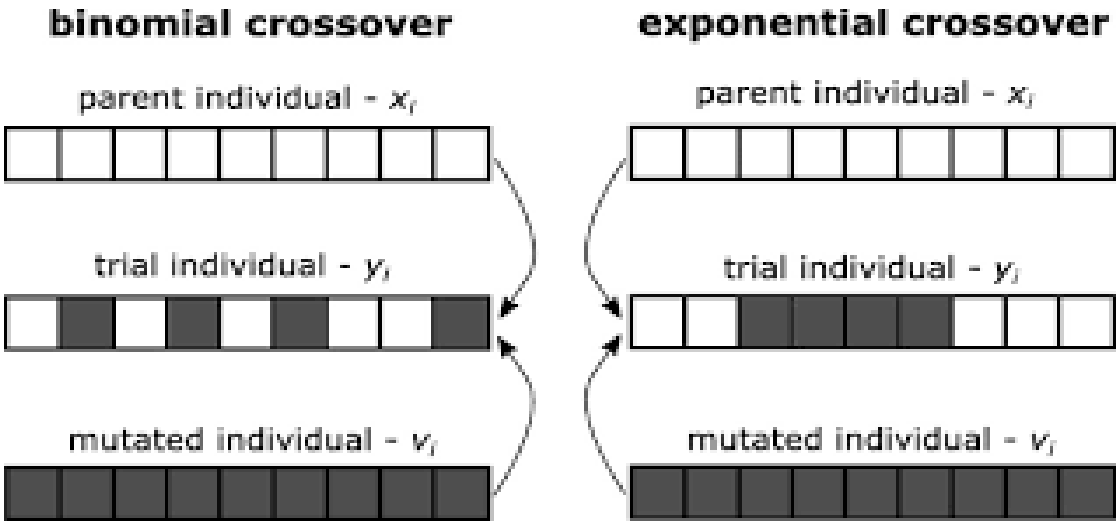


Fig:1.2 Crossover in DE

Selection-The trial solution generated through mutation and crossover is compared with the target individual in terms of fitness. If the trial solution is better, it replaces the target individual in the population. This selection process, known as the "greedy" selection, ensures that only individuals with improved fitness are retained in the population, leading to the convergence towards better solutions over generations

Termination-The DE algorithm continues to iterate through the mutation, crossover, and selection stages till a ending criterai has been met. Satisfactory level of fitness, or when the improvement in fitness becomes negligible or too much iterations are responsible for termination.

1.4.1.2)Particle Swarm Optimization

Particle Swarm Optimization falls under the Artificial Intelligence category. The hypothesis of computationally mimicking human behavior is known as artificial intelligence. It entails creating computer systems that are capable of doing activities that call for human intellect. For example, in the past, only humans were capable of understanding spoken language. However, these days, voice recognition is a standard feature on all digital devices. The development of artificial intelligence has made this feasible. Among other things, decision-making, language translation, and visual perception are examples of human intelligence. That is made feasible by a number of different strategies. The term "approaches of artificial intelligence" refers to several methods of integrating artificial intelligence into computers.

Implementation

We implement particle swarm optimization (PSO) for two fitness functions: 1) a Rastrigin function and 2) a spherical func$n$. This technique iterates for maximum iterations which are pre defined and tries to find out minimum from the fitness func$n$

Rastrigin function

equation:

$$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$$
$$\text{minimum at } f(0, \cdots, 0) = 0$$

This, an extremely difficult method to optimize. Because of its complicated behavior, optimization algorithms are frequently trapped at local minima. The complicated behavior of this function is introduced by the large number of cosine oscillations on the plane.

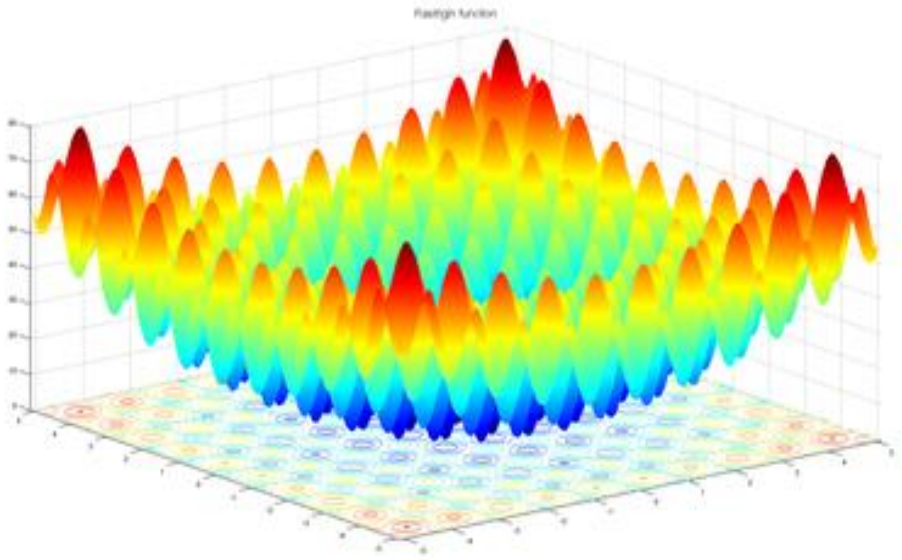


Fig:1.3 Rastrigin function of 2 variables

Sphere Function

A common function for assessing an optimization algorithm's performance is the sphere function.

equation:

$$f(x_1 \cdots x_n) = \sum_{i=1}^{n} x_i^2$$

$$\text{minimum at } f(0, \cdots, 0) = 0$$

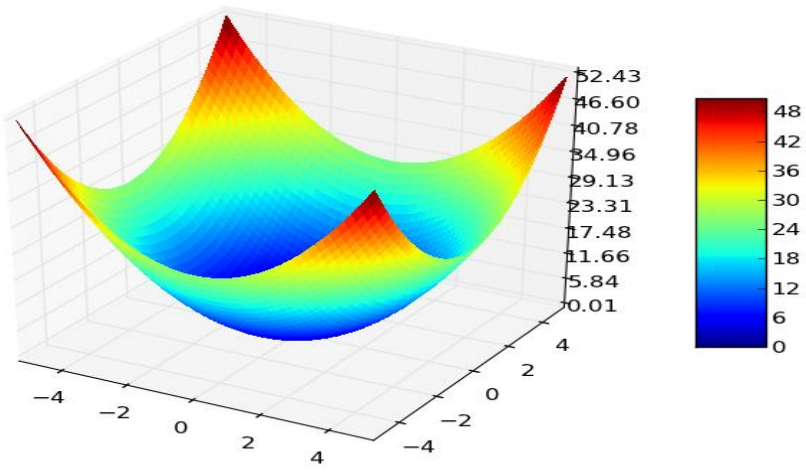


Fig:1.4 Sphere Function for 2 variable

ALGORITHM

A) In PSO every particles have association position, velocity, higher fit value.

B) Every particle tracks of the particle_best_Fit_valuing particle_best_Fit_positions.

C) A record of global-best_Fit_positions and global-greatest_Fit_values are happen bounding.

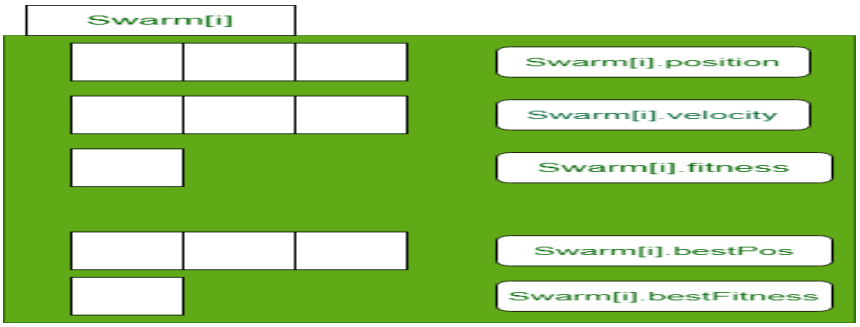


Fig 1.5: Data Flow to store i_{th} particle of Swarm

1.4.1.3) Ant-Colony-Optimization (AC*O*)

Ant Colony Optimization (ACO) algorithm was developed by taking inspiration from the foraging behavior of ants.ACO was the development by Marco Dorigo in the early 1990s and has since become a popular optimization technique for solving combinatorial optimization problems. ACO is particularly effective in solving problems that involve finding optimal paths or tours in graphs or networks. It has been applied successfully to a many situations, Vehicle Routing Problem (VRP), and graph coloring problems, among others.

The fundamental idea behind ACO is the simulation of the behavior of ant colonies when searching for food. In real-life ant colonies, ants deposit pheromone trails on the ground, which serve as indirect communication among the ants. These trails guide other ants towards food sources, creating a collective intelligence that helps the colony find the smallest route between the food sources and the home.

Stages in the Ant Colony Optimization algorithm are as follows:

1. **Initialize:** A population of virtual ants is created. Each ant is placed at astarting point, such as a nest or a depot, and assigned a random initial path or tour.

2. **Construction of Solutions:** Each ant iteratively builds a solution by probabilisticallyselecting the next component to visit based on a combination of pheromone trailinformation and heuristic values. The pheromone trail represents the amount of pheromone deposited on each edge or component of the problem, while the heuristicvalue captures the desirability of visiting a particular component based on domain specific knowledge. The selection process is typically guided by a parameter calledthe "pheromone trail intensity," which controls the balance between exploration and exploitation.

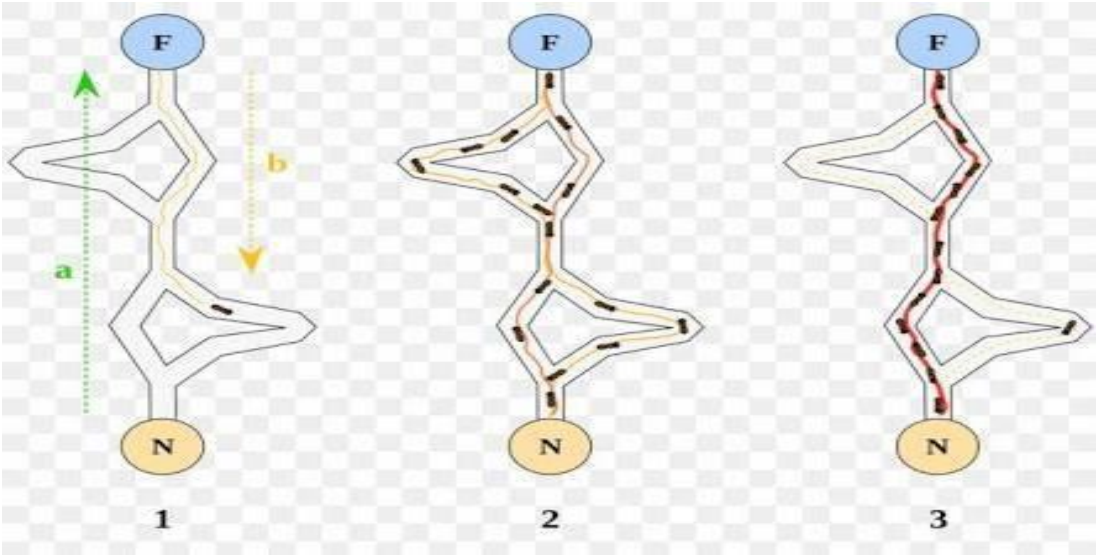


Fig:1.6 Ant Colony Optimization Process

3. Local Updating: After each ant moves from one component to another, it updates the pheromone trail intensity on the visited component, depositing or evaporating pheromone. This local updating process is essential for ants to reinforce good paths and encourage exploration of alternative paths.

4.Global Updating: Once all ants have constructed their solutions, a global updating procedure takes place. During this phase, the pheromone trails on all components are updated based on the quality of the solutions found. Stronger solutions leave a higher amount of pheromone,reinforcing the attractive paths in the search space

5.Termination: The technique iterates through the workspace, local updation, and global updation steps until a termination criterion is met.Closing(termit) criteria can include reaching a maximum number of iterations, a predefined computational budget, or achieving a satisfactory solution quality.

ACO has several advantages that contribute to its popularity:

Distributed Intelligence: ACO leverages the collective intelligence of the ant colony, allowing it to effectively explore and exploit the search space. The pheromone trail serves as a form of indirect communication that guides the ants towards promising areas.

Adaptability: ACO is adaptable to dynamic or changing problem instances since ants continuously update the pheromone trails based on the discovered solutions. This adaptability allows ACO to quickly respond to changes in the problem landscape.

Solution Diversity: ACO inherently promotes solution diversity due to the probabilistic nature of the construction process. This feature helps to explore different regions of the search space and avoids premature convergence to suboptimal solutions. Despite its advantages, ACO also has some challenges. It may suffer from slow convergence rates, especially in large-scale problems, and can be sensitive to parameter settings. Careful parameter tuning and problem-specific adaptations are often required to achieve good performance.

Parameter Tuning: ACO requires careful parameter tuning to achieve good performance. Parameters such as the pheromone evaporation rate, the influence of pheromone trails, and the balance between exploration and exploitation need to be properly set for different problem instances.

Convergence Speed: ACO may have slower convergence rates compared to other optimization techniques, especially in large-scale problems. Various strategies, such as the use of local search algorithms or hybrid approaches, can be employed to improve convergence speed.
Sensitivity to Problem Characteristics: ACO's performance can be sensitive to the characteristics of the problem being solved. Problem-specific adaptations or modifications of the basic ACO algorithm may be necessary to achieve better results.

By using pheromone trails and probabilistic decision-making, ACO effectively explores the search space, balances exploration and exploitation, and converges towards near-optimal solutions. Its adaptability, solution diversity, and successful applications make ACO a valuable optimization technique in various domains.

Advantages

The Ant Colony Optimization (ACO) algorithm offers several advantages that make it a powerful tool for solving optimization problems. Here are some key advantages of ACO:

1. Ability to find near-optimal solutions

ACO excels at finding near-optimal solutions, even in complex problem spaces. By leveraging the collective intelligence of artificial ants and the reinforcement of pheromone trails, ACO can efficiently explore and exploit promising regions of the solution space.

2. Adaptability to dynamic environments

ACO is adaptable to dynamic environments where the problem or its constraints change over time. The algorithm can dynamically adjust the pheromone trail

intensities, allowing it to quickly respond to changes and find updated optimal or near-optimal solutions.

3. Scalability

ACO is highly scalable, making it suitable for solving large-scale optimization problems. The parallel nature of ant exploration enables ACO to handle a high number of problem variables and constraints efficiently.

4. Robustness

ACO exhibits robustness in handling noisy or imperfect problem information. The pheromone trails and heuristics guide the ants' decision-making, reducing the impact of uncertainties and imperfect problem knowledge.

5. Exploration of solution space

Ant colony algorithms balance exploration and exploitation, allowing for effective exploration of the solution space. This property enables the algorithm to escape local optima and discover diverse solutions, providing a broader perspective of the problem.

The advantages of ACO make it a valuable optimization technique, offering the potential for finding high-quality solutions in various domains, including route optimization, scheduling, and resource allocation.

Disadvantages

While Ant Colony Optimization (ACO) is a powerful optimization algorithm, it also has some limitations that should be considered. Here are a few key limitations of ACO:

1. Convergence to suboptimal solutions

ACO may struggle to converge to the global optimum in complex problem spaces with multiple local optima. The algorithm's reliance on pheromone trails and local information exchange can lead to premature convergence to suboptimal solutions.

2. Sensitivity to parameter settings

ACO's performance is sensitive to the appropriate tuning of its parameters. The choice of parameters, such as the pheromone evaporation rate and exploration-exploitation trade-off, can significantly impact the algorithm's convergence speed and solution quality.

3. Computational complexity

ACO can be computationally expensive, especially when dealing with large problem instances. As the number of ants and iterations increases, the algorithm's execution time and memory requirements grow, making it challenging to handle real-time or time-critical applications.

4. Limited handling of dynamic environments

While ACO demonstrates adaptability to dynamic environments, sudden and drastic changes may pose challenges. The algorithm may require additional mechanisms or modifications to handle rapidly changing problem scenarios effectively.
Understanding these limitations helps in utilizing ACO effectively and considering alternative optimization algorithms when they align better with specific problem characteristics or constraints.

CONCLUSION:

To sum up, Ant Colony Optimization (ACO) is a potent metaheuristic algorithm that draws inspiration from ants' foraging habits. An effective technique for resolving challenging optimization issues is ant colony optimization, which is especially useful in the field of route optimization. It is used in many different industries, such as telecommunications, logistics, and transportation.

Initializing parameters, building solutions, updating pheromone levels, applying evaporation, and iteratively improving solutions until a termination criterion is satisfied are all steps in the ACO implementation process. ACO continues to significantly contribute to the solution of real-world problems and to the improvement of optimization by utilizing the collective intelligence of artificial ants.

1.4.1.4 COMPARISONS of EAs:

Criteria	Differential Evolution	Particle Swarm Optimization	Ant Colony Optimization
Inspiration	DE is inspired by the process of evolution and natural selection.	PSO is developed by behavior of birds, flocking or fish schooling in society.	ACO is inspired from the foraging behavior of real ants.
Search Strategy	DE uses mutation, crossover, and selection operations on a population of candidate solutions.	PSO searches for solutions by moving particles in the search space towards the best-known positions found by the swarm.	ACO uses artificial ants to build solutions by following pheromone trails and

			making probabilistic decisions.
Population vs Swarm vs Colony	DE operates on a population of candidate solutions where each solution is a potential individual.	PSO operates on a swarm of particles, with each particle representing a potential solution.	ACO operates with a colony of artificial ants where each ant contributes to building a solution.
Memory and Communication	DE typically does not have explicit communication between individuals in the population.	PSO relies on communication between particles to update their positions based on the best positions found by the swarm.	ACO uses pheromone trails to communicate information about good solutions and guide the search process.
Parameter Tuning	DE requires setting parameters like mutation rate and crossover probability.	PSO requires setting parameters such as inertia weight, cognitive, and social parameters.	ACO requires parameter tuning for parameters like the pheromone evaporation rate and heuristic information.

1.4.2

Multi-Objective Optimization

Multi-objective is an optimization technique where multi-criteria decisionis taken care of and is concerned with mathematical optimization problems in which multiple goals are optimized. MOOs have applications in many fields of science, including engineering, where optimal decisions need to be made when there are conflicts between two or more potentially contradictory goals. For example, maximizing vehicle performance while minimizing fuel consumption and pollutant emissions. In such cases, it is necessary to perform a multiobjective optimization study that provides multiple solutions that represent tradeoffs between objective functions.

Multi-objective optimization (MOO) occurs in everyday life, including mathematics, engineering, and the automotive industry. Many accounting methods are also applied to different types of MOO problems, global basis technique, sum-weighted technique, and ε-constraint method. MOO payment methods use complex solutions; arcane formulas. In this article, we propose a MOO accounting method that does not require complex formulas to make simple

the problem.Techniques include the pareto method and the scalarized method. On the other hand, solutions that use scalarization methods are performance indicator components that form scalar functions. Integrated fitness features.

Multi-objective optimization, or MOO, is the process of determining the best values for a number of objectives. The motivation for using MOO is that optimization does not require complex equations, resulting in a simpler problem. Decision-making problems in MOO allow for compromises on several contradictory issues. MOO was introduced by Vilfredo Pareto. MOO has a vector of objective functions.

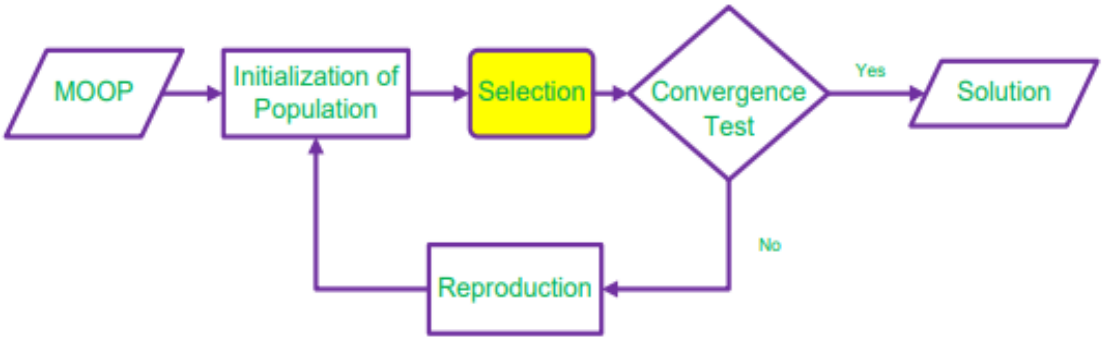


Fig:1.7 Flow chart of MOOs

The choice variable space of the solution vector and the objective function vector are multidimensional spaces in the MOO. There is a point on the objective function space for each x solution in the decision variable space. Figure shows the mapping between the objective function vector and the solution vector.

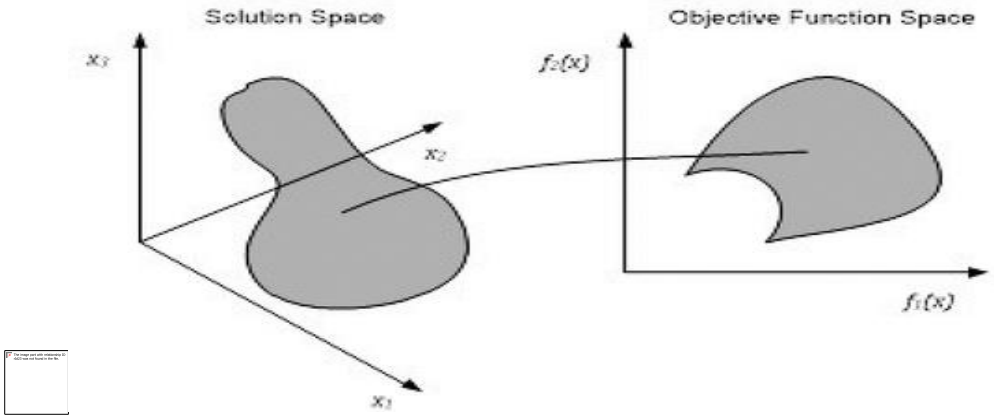


Fig:1.8 Mapping solution space and objective function space.

1.4.2.A)Non Dominated Sorting Genetic Algorithm II

NSGA-II is one of the strongest optimization technique. It is encouraged by natural selection of Darwin's Law.

If there is no dominant solution, solution x is considered non-dominant and selected as one of the Pareto fronts by NSGA-II. Considering a set of objectives fi, i ∈ 1, …, n to be maximized, if ∀ i, fi(x') ≤ fi(x) and ∃ j | Become. fj(x')

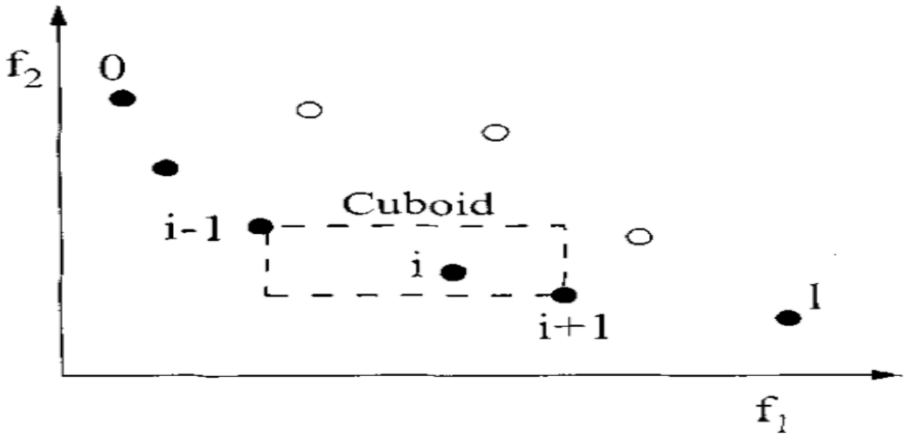


Fig:1.9 Distance calculation of Crowding

The solutions of the same non-dominated front are indicated by points with filled circles.

Crowd distance is the Manhattan distance of the area of interest. However, it is preferable to assign an infinite congestion distance because the extreme points are retained in each generation.

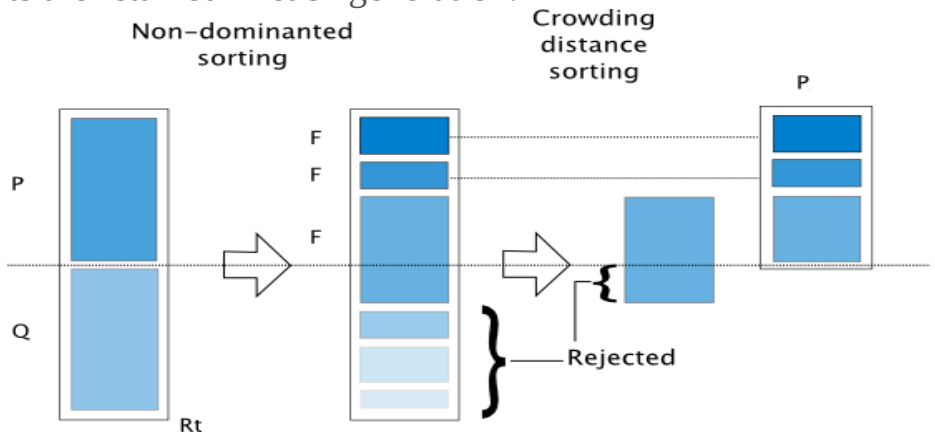


Fig:1.10 NSGA II Crowding distance sorting

To increase selection pressure, NSGA-II also uses combinatorial selection for binary tournaments. Individuals are compared first by rank and then by mass distance.

1.5 Non-Pareto Based Multi Objective Optimization

Non-Pareto based approches :

Lexicographic ordering

Criterion selection (VEGA)

1.5.1 Lexicographic ordering

Lexicographical optimization is a type of multi-objective optimization. Generally, multi-objective optimization deals with optimization problems that involve optimizing two or more objective functions simultaneously. Different goals are often ranked in order of importance to the decision maker (goal is most important, goal is her second most important, etc.). Lexicographical optimization assumes that the decision maker prefers a very small increase in her to a very large increase in her. Similarly, decision makers prefer even small increases, even very large increases, and so on. Stated differently, the decision-maker employs lexicographical preferences and arranges potential solutions based on the lexicographical arrangement of the values of the goal function.

For example, imagine a company that prioritizes safety. We want to ensure the maximum safety of our employees and customers. We want to maximize profits subject to the greatest possible safety.

Suppose, with k objectives and n constraints over a decision space x and is denoted as.

Minimize

$$f = (f_1, f_2, \ldots, f_k)$$

1.5.2 Criterion selection:- Vega

Due to the need to address a large number of objectives (M), Schaffer came up with the idea of randomly splitting the genetic algorithm population into M equal subpopulations at each iteration. The algorithm weights solutions that suit each individual objective functions.

1.6 Difference between Single and Multi-Objective Optimization

From our perspective, it is important to compare the solutions obtained by these multi-objective techniques with those obtained by algorithms particularly designed to optimize single and independent objectives, when each objective

value is taken into account independently. We conducted a comparison study between single-objective and multi-objective evolutionary algorithms.

According to the computational studies carried out, MOEA proves to be more effective than single-objective approaches in some cases, although it requires addressing multiple objectives simultaneously.

Further evaluation could be carried out in the future using a more representative and independent set of problems and cases. Therefore, the key factors influencing the improvement of the MOEA approach for single-objective environments can be further investigated. In MOEA design, each target can have ancillary target effects on other targets,

It is important to keep in mind that we aim for meaningful diversity. High diversity may be counterproductive, since it does not necessarily mean that exploration and exploitation are balanced.

1.7 Objective of Thesis

The objective of thesis is to propose an algorithm According to computational studies conducted, MOEA has proven to be more effective than single-target approaches in some cases, despite the need to address multiple targets simultaneously. In particular, the multi-target approach showed better performance in large cases and highly correlated targets. In such special cases, applying a multiobjective solver directly to a multiobjective problem is a better choice than converting the multiobjective problem to a single objective problem and solving it uses SO algorithm.The more negatively correlated the goals are, the greater the discrepancy. Otherwise, in contexts with a consistent objective function.Therefore, the key factors influencing the improvement of the MOEA approach for single-objective environments can be further investigated. Because the MOEA design allows each target to have additional targeting effects on other targets, this feature potentially provides more degrees of freedom to maintain individual diversity within the population. there is. Such functionality does not exist in single-target approaches. It's important to remember that we strive for meaningful diversity. **“Compromise solutions''**, denoted as Pareto optima. However, one major demerit of traditional MOEAs is the time consuming nature of Pareto front implementation. The work in this thesis succeeds to alleviate this drawback posed by the traditional MOEA in solving real-world problems by hybridization of the single-objective differential optimization (DE) with the VIKOR multi-criteria decision making to serve multiple objectives. The proposed algorithm, referred to as VIKOR-DE henceforth, unlike other

algorithms generates a single solution even in the contradicting multi-objective scenario. In contrast to traditional evolutionary algorithms, the proposed approach allows fast convergence of solutions on the Pareto Front, thus making it ideal for solving real-world multi-objective optimization problems, opening a new direction to successful multi-agent coordination.

1.8 Organization of Thesis

In this thesis,

Chapter 1 includes the introduction to the Optimization and Evolutionary Algorithm used for optimization problem. Here we also discussed the classification of Evolutionary Algorithm and the differences between them with discussing the pros and cons of each algorithm.

Atlast it includes the objective and the outline.

Chapter 2 Provides the literature review about strategies developed optimization problem using VIKOR method and its types.

Chapter 3 The chapter gives overview of the importance of DE method.It deals with the study of methologies of optimization using DE.It also explores the capabilities of the Algorithms for solving problems and optimizesthe path planning.

Chapter 4 The chapter gives overview of the importance of DE and VIKOR methods.It deals with the study of methologies of optimization using VIKOR-DE.It also explores the capabilities of the Algorithms for solving problems and optimizesthe path planning.It also briefs about different literary works done towards the advancement in the field of different evolutionary algorithms.It consists of the algorithms and pseudo-code for the above methodologies.

1.9 REFERENCES

[1]. Swagatam Das, Ponnuthurai Nagaratnam Suganthan “Differential Evolution: A Survey of the State-of-the-Art” IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 15, NO. 1, FEBRUARY 2011

[2]. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan”A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II” IEEE

TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 6, NO. 2, APRIL 2002

[3]. Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms, 1(1), 69-93.

[4]. Branke, J. (2006). Evolutionary optimization in dynamic environments. Kluwer Academic Publishers.

[5]. Storn, R., Price, K.V.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces (1995) Technical Report TR-95-012, ICSI (March 1995)

[6]. Storn, R., Price, K.V.: Minimizing the real functions of the ICEC 1996 contest by differential evolution. In: Proceedings of the 1996 IEEE international conference on evolutionary computation, Nagoya, Japan, pp. 842–844. IEEE Press, New York (1996)

[7]. Bunday, B.D. and Garside G.R. (1987), *Optimisation Methods in Pascal*, Edward Arnold Publishers.

[8]. I.F. Sbalzarini *et al.* Multiobjective optimization using evolutionary algorithms

[8]. R.C. Purshouse *et al.*On the evolutionary optimization of many conflicting objectives IEEE Trans. Evol. Comput.(2007)

[9]. C. Gil *et al.*A hybrid method for solving multi-objective global optimization problemsJ. Global Optim.(2007)

[10]. R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," Appl. Soft Comput., vol. 11, no. 2, pp. 1679–1696, 2011.

CHAPTER-*2*

VIKOR Optimization

Introduction

The chapter gives overview of the importance of VIKOR method.
It deals with the study of methologies of optimization using VIKOR.
It also explores the capabilities of the Algorithms for solving problems and optimizes the path planning.

2.1 VIKOR for Multi Criteria Decision

The MCDM technique is very well-known procedure extensively useful for to finding the best solution amongst numerous experiments having several features. The VIKOR optimization technique was established for multi optimization of difficult systems. VIKOR technique emphases on ranking and select from all alternatives. VIKOR is useful tool in MCDM. Compromise solution provides a lowest of the individual regret of the opponent and a highest group utility of the majority so decision maker might accept the compromise solution. The distance from the ideal solution representing by an aggregating function in VIKOR technique. VIKOR ranking index is an aggregation of all criteria. Experiment having highest ranked VIKOR index is close to the best solution.

2.2 History

Po-Lung Yu and Milan Zeleny introduced the concept of the VIKOR compromise solution to MCDM in 1973.

The fundamental concepts of VIKOR were formulated by S. Opricovic in his doctoral in the year 1979 and released the application in 1980.Term VIKOR firstly intoduced in 1990. In 2004 study helped the VIKOR technique gain international prominence.

Here is how the MCDM problem is expressed: From the set of J feasible alternatives,$(A_1, A_2, \ldots A_j)$ evaluate each one in accordance with the set of n criterion functions to determine the optimal (compromise) solution in the multicriteria sense. The performance (decision) matrix's elements f_{ij}, where is the alternative's i-th criterion function's value, are the input data.

2.3 Brief Introduction of VIKOR Variants

In addition to the VIKOR approach, several variations have also been developed based on the nature of the choice problem and the decision maker's requirements. These variations include the complete VIKOR, fuzzy VIKOR, regret theory-based VIKOR, modified VIKOR, and interval VIKOR methods.

A) **Comprehensive VIKOR:**

Jahan et al. (2011) presented the complete VIKOR method, which should be applied in circumstances where it is desirable to approach the goal values for certain of the criteria. Thus, it resolves the primary flaw in the original VIKOR technique.

One of the most important steps in the design process is choosing the materials.

B) Fuzzy VIKOR:

The original technique is extended by fuzzy VIKOR to handle fuzzy environments, where weights and criteria can both be fuzzy sets.Imprecise numerical quantities are represented by triangular fuzzy numbers.A flexible framework for making decisions in uncertain situations is offered by fuzzy VIKOR.

C) Regret Theory VIKOR:

The original VIKOR approach and regret theory are combined in the regret theory-based VIKOR method to incorporate regret and celebration into the decision-making process. According to regret theory, people's decisions are occasionally impacted by their feelings. Emotions, which are modifications of mental and psychological states, were once considered to be a major factor in the decision-making process. The basic idea behind regret theory is that the decision maker's expected emotions are added to the traditional utility function.

The value that a person can obtain from a consequence if they have experienced it without having made a choice is known as the choiceless utility, and it is determined irrespective of choices. The VIKOR technique and regret theory vary in that the former defines regret as the choiceless utility.

D) VIKOR in Modifification:

The modified VIKOR method finds applications in various fields, including supply chain management, healthcare, and environmental assessment.

It allows decision-makers to make informed choices by considering multiple criteria simultaneously.

E) Interval VIKOR:

Since interval numbers are the most straightforward way to express uncertainty in a decision matrix, they are better suited to handle decision-making issues in

imprecise and uncertain environments. The minimal quantity of data on the attribute values is needed to calculate the interval numbers. A property in a decision matrix that has an interval specified for it can have any value within the period. The interval numbers don't tell you which of the numerous values in the interval is most likely to occur, or how likely it is that the value will be in the interval. Therefore, it is more suitable to think of the qualities as interval numbers when it is difficult or impossible to determine their exact values.

2.4 VIKOR Algorithm

The decision maker identify a near ideal solution to a multi-criteria problem by evaluating all candidates based on M decision criteria.

Let $\{\overrightarrow{Y_1}, \overrightarrow{Y_2}, \ldots, \overrightarrow{Y_{MP}}\}$ be MP options in the W-dimensional search space, which stand for potential solutions to the M-criteria multi-criteria issue. The following is a summary of the VIKOR approach for identifying the near optimal solution from $\{\overrightarrow{Y_1}, \overrightarrow{Y_2}, \ldots, \overrightarrow{Y_{MP}}\}$:

STEP (I):

DM of MP×M is built by M objective functions of different MP alternatives

Objective / Alternative	**1**	**2**	**---**	**3**
$\overrightarrow{Y_1}$	$t_1(\overrightarrow{Y_1})$	$t_2(\overrightarrow{Y_1})$	**---**	$t_M(\overrightarrow{Y_1})$
$\overrightarrow{Y_2}$	$t_1(\overrightarrow{Y_2})$	$t_2(\overrightarrow{Y_2})$	**---**	$t_M(\overrightarrow{Y_2})$
- -	- -	- -	- -	- -
$\overrightarrow{Y_{MP}}$	$t_1(\overrightarrow{Y_{MP}})$	$t_2(\overrightarrow{Y_{MP}})$	**---**	$t_M(\overrightarrow{Y_{MP}})$

STEP (II):

The objective function values are then normalized by setting

$$t_{j(}\overrightarrow{Y_i)}=t_{j(}\overrightarrow{Y_i)}/\sqrt{\sum_{k}^{MP} t_j^2(\overrightarrow{Y_k})} \quad \textbf{(1)}$$

for d =1,2, …, MP and j = 1, 2, …, M.

STEP (III):

We will identify the positive and negative ideal solutions in this step. The optimal set of goal functions for an MCDM problem requiring the minimization of M objectives is found below:

$$t^* = \{t_1^*, t_2^*, \ldots, t_M^*\} \quad \textbf{(2.a)}$$

$$t_j^* = \min(t_j(\overrightarrow{Y_i}) \quad \textbf{(2.b)}$$

the negative ideal set of objective functions is denoted similarly by-

$$t^- = \{t_1^-, t_2^-, \ldots, t_M^-\} \quad \textbf{(3.a)}$$

$$t_j^- = \max(t_j(\overrightarrow{Y_j}) \quad \textbf{(3.b)}$$

STEP (IV):

By capturing discrepancy from the ideal function values of the goal,calculating Chebyshev(C) and Manhattan (N) distances of MP alternatives

$$\mathrm{N}(\overrightarrow{Y_d}) = \sum_{j}^{N} w_j \frac{t_j(\overrightarrow{Y_d}) - t_j^*}{t_j^- - t_j^*} \quad \textbf{(4)}$$

$$C(\overrightarrow{Y_d}) = \max \frac{t_j(\overrightarrow{Y_d}) - t_j^*}{t_j^- - t_j^*} \quad \textbf{(5)}$$

for d = {1, 2, …, MP}. The weight w_j indicates the importance of the decision maker's imposed objective role t_j.

STEP 5:

In this stage we calculate the VIKOR index-

$$L(\overrightarrow{Y_d}) = \theta \frac{N(\overrightarrow{Y_d}) - N^*}{N^- - N^*} + (1-\theta) \frac{C(\overrightarrow{Y_d}) - C^*}{C^- - C^*} \quad \textbf{(6)}$$

for d = {1, 2, …, MP}

$$N^* = \min N(\overrightarrow{Y_d}) \quad \text{and} \quad N^- = \max N(\overrightarrow{Y_d}) \quad \textbf{(7)}$$

$$C^* = \min C(\overrightarrow{Y_d}) \quad \text{and} \quad C^- = \max C(\overrightarrow{Y_d}) \quad \textbf{(8)}$$

Θ{0, 1} is a factor for weighting that is typically set to 0.50.

STEP 6:
Lastly, the options are arranged according to their corresponding VIKOR indices, which are sorted ascending. The closest to optimum solution is represented by the option at the top of the sorted list.

2.5 Code for VIKOR using Python Optimization

In [1]:

```
def vikor(alternatives, criteria, weights)
  # Step 1: Determine the best t* and the worst t- values of all criterion functions.
  best_t = []
  worst_t = []
  for criterion in criteria:
    best_t.append(max(criterion))
    worst_t.append(min(criterion))
  # Step 2: Compute the values Nj and Cj.
  N = []
  C = []
  for alternative in alternatives:
    n = 0
    c = 0
    for d in range(len(criteria)):
      n += weights[d] * (best_t[d] - alternative[d]) / (best_t[d] - worst_t[d])
      c = max(r, weights[i] * (best_t[i] - alternative[i]) / (best_t[d] - worst_t[d]))
    N.append(n)
    C.append(c)
  # Step 3: Compute the values Qj.
  Q = []
  for d in range(len(alternatives)):
    Q.append(v * (N[d] - min(N)) / (max(N) - min(N)) + (1 - v) * (C[d] - min(C)) / (max(C) - min(C)))
  # Step 4: Select the best alternative.
  best_alternative = alternatives[Q.index(max(Q))]
  return best_alternative
```

2.6 Advantages and Disadvantages of VIKOR method

Advantages:

1. **Handling insufficient or conflicting Data:** VIKOR can manage situations where data is insufficient or conflicting,accomodating both quantitative and qualitive aspects.

2. **Compromise Solution:** It provides a compromise solution that balances conflicting criteria, making it suitable for decision problems with noncommensurable criteria.

3. **Application Flexibility:** VIKOR can be applied across various domains, including engineering, economics, and environmental management.

Disadvantages:

1.**Imprecise Handling:** The original VIKOR method assumes that comprising is acceptable for conflicting resolution.However when dealing with imprecise or uncertain information,the interval VIKOR method may perform unsatisfactorily

2.**Bipolar Information Challenges:** The VIKOR strategy cannot handle MCDM problems with bipolar information(bipolar reasoning).

2.7 REFERENCES

[1]. Po Lung Yu (1973) "A Class of Solutions for Group Decision Problems", Management Science, 19(8), 936–946.

[2]. Milan Zelrny (1973) "Compromise Programming", in Cochrane J.L. and M.Zeleny (Eds.), Multiple Criteria Decision Making, University of South Carolina Press, Columbia.

[3]. Lucien Duckstein and Serafim Opricovic (1980) "Multiobjective Optimization in River Basin Development", Water Resources Research, 16(1), 14–20.

[4]. Serafim Opricović., (1990) "Programski paket VIKOR za visekriterijumsko kompromisno rangiranje", SYM-OP-IS

[5]. Serafim Opricovic and Gwo-Hshiung Tzeng (2004) "The Compromise solution by MCDM methods: A comparative analysis of VIKOR and TOPSIS", European Journal of Operational Research, 156(2), 445–455.

[6]. Serafim Opricovic and Gwo-Hshiung Tzeng (2007) "Extended VIKOR Method in Comparison with Outranking Methods", European Journal of Operational Research, Vol. 178, No 2, pp. 514–529.

[7]. Heydari, Majeed; Kazem Sayadi, Mohammad; Shahanaghi, Kamran (2010). "Extended VIKOR as a new method for solving Multiple Objective Large-Scale Nonlinear Programming problems" (PDF). *Rairo - Operations Research*. 44 (2): 139–152. doi:10.1051/ro/2010011.

CHAPTER-*3*

Differential Evolutionary Algorithm

Introduction

The chapter gives overview of the importance of DE method.
It deals with the study of methodologies of optimization using DE.
It also explores the capabilities of the Algorithms for solving problems and Optimizes the path planning.

It also concludes the importances and applications of the algorithm.

3.1 Differential Evolutionary Algorithm (DE)

Storn and Price introduced this algorithm.

DE is **"a new heuristic approach for minimizing possibly nonlinear and non-differentiable continuous space functions"** according to Storn and Price.
Any strategy to problem-solving that employs a methodical technique to generate solutions—which might not be ideal—but are adequate in situations when traditional approaches are unable to locate the exact solution or are too slow is referred to as a heuristic.The optimization problem does not have to be differentiable, in contrast to the Gradient Descent approach.

Differential Evolution (DE) is a numerical optimization technique used to find solutions to optimization problems. It belongs to the class of evolutionary algorithms and is particularly well-suited for solving optimization problems in continuous spaces where the objective function may be complex, noisy, or non-linear.The basic idea behind Differential Evolution is to mimic the process of natural evolution by iteratively improving a population of candidate solutions (individuals) to find the optimal solution. DE maintains a population of potential solutions, each represented as a vector in the search space. At each iteration, new candidate solutions are generated by perturbing and combining existing individuals based on a set of predefined rules.

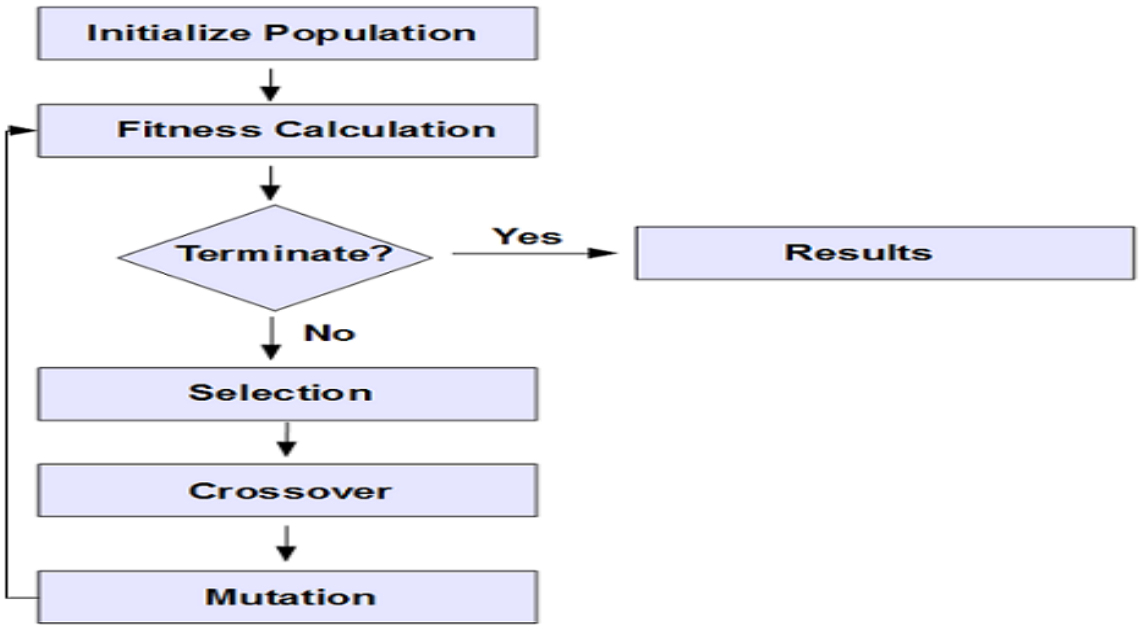


Fig: 3.1 Flowchart of DE

3.2 Computation Background

Within the larger field of evolutionary computation, Differential Evolution (DE) is a nature-inspired search method that includes a range of optimization algorithms that are inspired by natural selection and genetics.

In genetic algorithms, individuals are treated as potential solutions by the population, which then evaluates their fitness by subjecting them to genetic operations (such as crossover and mutation).The fundamental ideas behind genetic algorithms provide the framework for understanding DE's population-based optimization strategy.

3.3 Why Differential Evolution?

Differential Evolution (DE) is useful and relevant for several reasons, particularly when it comes to solving optimization problems. Those reasons can be easily listed down as table :–

Sl. No.	Factors	Description
1.	Complex Objective Functions	These functions can be difficult to optimize using traditional gradient-based methods. Differential Evolution's ability to explore the search space through mutation and crossover makes it well-suited.
2.	No Derivative Information	DE does not require any gradient information. It makes DE applicable to problems where Derivatives unavailable or expensive to compute.
3.	Global and Local Optimization	It has a good balance between exploitation and exploration thus effectively handle both global and local optimization tasks.
4.	Easy Implementation	DE's algorithmic simplicity and ease of implementation make it accessible to a wide range of users
5.	Parameter Tuning	DE can be used to tune the parameters of

		complex models or algorithms. Often used to optimize,hyper-parameters in machine learning algorithms,
6.	Constraint Handling	Can be extended to handle optimization problems with constraints by incorporating constraint handling mechanisms.
7.	Parallelization	DE is amenable to parallel computing, which can accelerate the optimization process.

3.4 Differential Evolution Algorithm

This section overviews the main steps of differential evolution (DE) , a well-known meta-heuristic algorithm.

STEP:1

Initialize

Individuals are assigned a fitness value, based on their corresponding objective function values and possibly constraint values. Originally DE has no rule for constraint handling, which was later the focus of several articles.The population then iterates through successive generations until some stopping criteria are met. In each of these iterations, new trial vectors are produced by operations known as mutation and crossover. The trial vectors are then compared to their corresponding parents of the same index and the best of each pair is passed to the next generation. Stopping criteria are usually based on improvements in the objective function and the number of generations.

DE commences from a population $\boldsymbol{\Omega}(g)$ $\{X_1(g), X_2(g),..., X_S(g)\}$of randomly initialized W-dimensional T target vectors, within the search space boundary $[X^l, X^h]$ where
$X^l=\{x^l, x^l,..., x^l\}$ at generation $g= 0$.

The i-th target vector's j-th component is initialized by setting-

$$x_{i,j}(0) = x_j^l + rand \times (x_j^h - x_j^l) \quad \textbf{(9)}$$

for the variables i = 1, 2,..,W. and j = 1, 2,…,T In this case, rand stands for a value chosen at random from the uniform distribution in (0, 1). In [0, 1], the crossover ratio CR is initialized. For i = 1, 2,..., T, the fitness value $f(x_i(0))$ is calculated.

STEP:2
Mutation

This step is concerned with generation of a donor vector $V_i(g)$ for each target vector $X_i(g)$ by the expression (2). a is generated by randomly picking up two different members $X_{r1}(g)$ and $X_{r2}(g)$ from $\boldsymbol{\Omega}(g)$ for i = 1, 2, …, S. The policy is given by

$$\vec{V}_i(g) = \vec{X}_i(g) + F_1(\vec{X^{best}}(g) - \vec{X}_i(g)) + F_2(\vec{X}_{r1}(g) - \vec{X}_{r2}(g))$$
(10)

Here $F1$ and $F2$ are scale factors in [0, 2]. $\vec{X^{best}}(g)$ symbolizes the population best at

generation g. $\vec{X}_{r1}(g)$ and $\vec{X}_{r2}(g)$ are randomly selected from the population $\boldsymbol{\Omega}(g)$ such that $\vec{X}_{r1}(g) \neq \vec{X}_{r2}(g) \neq \vec{X}_i(g)$. The mutation operation described by (22) is called DE/current-to-best/1. This step is repeated for i = 1, 2, …, S

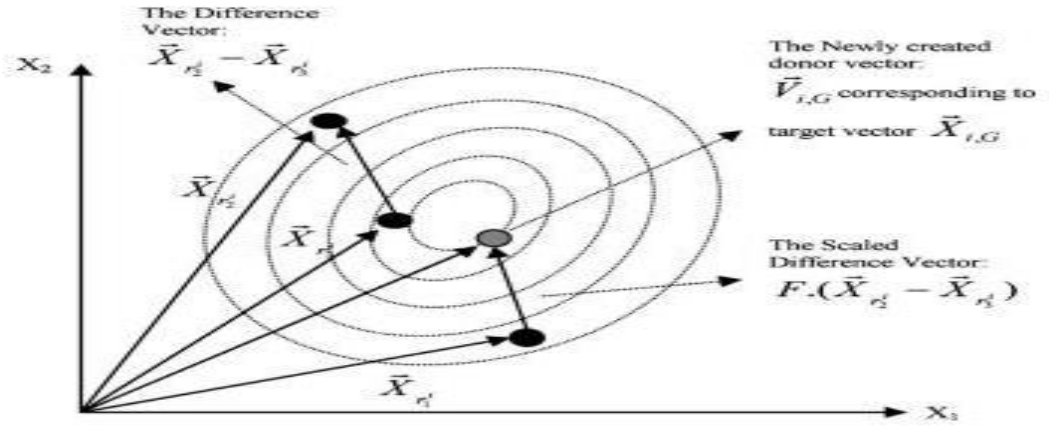


Fig: 3.2 Illustrating a simple DE mutation scheme in 2-D parameters

STEP 3:
Crossover

The binomial crossover policy is employed here for generating a trial vector $\vec{Z}_i(g)$ corresponding to each pair of $\vec{X}_i(g)$ and $\vec{V}_i(g)$ for i = 1, 2, , S. This is realized by

$$Z_{i,j}(g) = \begin{cases} v_{i,j}(g) & \text{,if rand} \le CR \text{ or } j = j_r \\ x_{i,j}(g) & \text{, otherwise} \end{cases} \quad \textbf{(11)}$$

for j = 1, 2, …, W and i = 1, 2, …, T. Here j_r symbolizes an integer chosen at random from {1, 2, …, W}

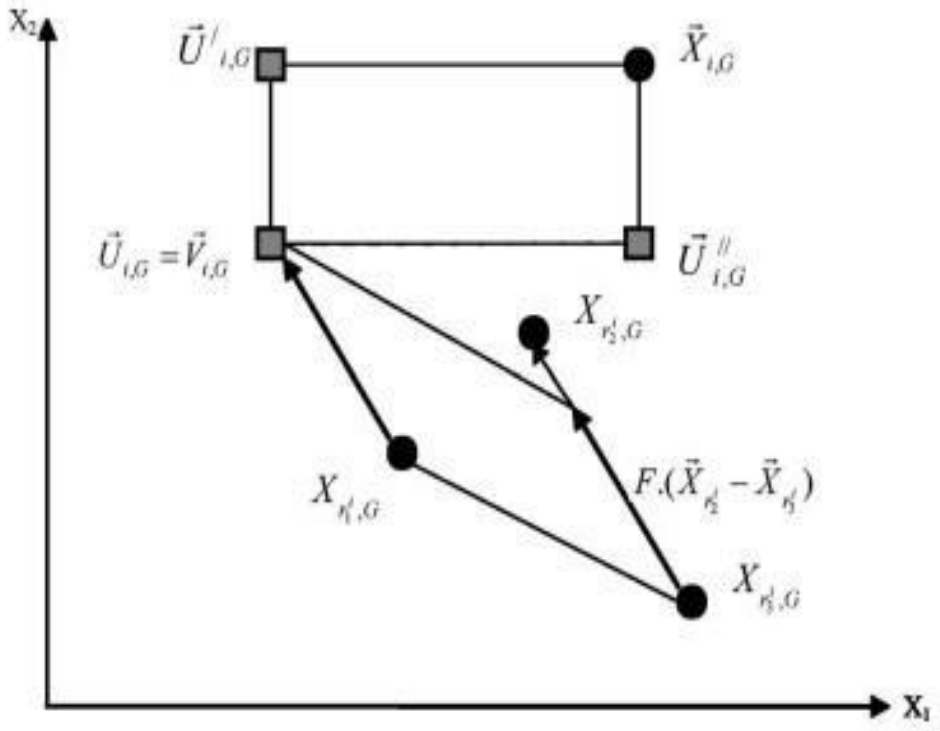


Fig: 3.3 Each possibly trial vectors formed due to uniform or binomial crossover between the mutant vectors and the target vectors

STEP 4:

Selection

For the minimization problem, the greedy selection strategy is described by

$$\vec{X}_i(g+1) = \begin{cases} \vec{Z}_i(g), & \text{if } f(\vec{Z}_i(g)) \leq f(\vec{X}_i(g)) \\ \vec{X}_i(g), & \text{otherwise} \end{cases} \quad \textbf{(12)}$$

STEP 5:

Convergence

Steps 2 through 4 are repeated after every evolution cycle until the terminating condition is met. The requirement is to limit the number of generations, uphold error restrictions, or do both at the earliest opportunity.

3.5 Limitations of Traditional Differential Evolution

Traditional Differential Evolution (DE) has several limitations that can impact its performance in certain optimization scenarios. These limitations are:

A. Premature Convergence: Traditional DE can converge to a suboptimal solution prematurely, especially in complex optimization problems.This premature convergence occurs when the algorithm quickly converges to a local optimum and fails to explore other regions of the search space that may contain better solutions.This limitation restricts the algorithm's ability to find globally optimal solutions and can hinder its performance in challenging optimization scenarios.

B. Sensitivity to Parameter Settings: Inadequate parameter settings can result in poor convergence, slow convergence speed, or even algorithm failure. Finding the optimal parameter settings for a given problem can be challenging and time-consuming.This sensitivity to parameter settings makes traditional DE less robust and requires careful parameter tuning for optimal performance.

C. Lack of Spatial Information: Traditional DE primarily relies on fitness
evaluations and does not consider the spatial information of individuals in the search space.This limitation can lead to a loss of population diversity and hinder the algorithm's ability to explore the entire search space effectively.Ignoring the spatial information can be problematic in problems with complex or irregular fitness landscapes where the location of solutions plays a crucial role in finding optimal solutions.

D. Limited Adaptation to Problem Characteristics: Traditional DE typically uses fixed control parameter settings throughout the optimization process. However, different optimization problems may have distinct characteristics and require different exploration and exploitation strategies.Without the ability to adapt to problem-specific requirements, traditional DE may struggle to perform optimally across various problem domains. The lack of adaptation can result in inefficient search and suboptimal convergence.

E. Inefficiency in High-Dimensional Spaces: Traditional DE can face challenges in high-dimensional optimization problems where the search space is large.As the dimensionality increases, the search space becomes more expansive, making it difficult for DE to explore

effectively.Traditional DE may encounter increased computational complexity, longer convergence times, and difficulties in maintaining population diversity in high-dimensional spaces.

3.6 APPLICATIONS

Differential Evolution (DE) is a robust and efficient method for addressing optimization problems, which makes it useful in a wide range of applications.

I. Engineering Field

Engineering optimization projects make considerable use of the Differential Evolution Algorithm. For example, DE can effectively optimize parameters like wing profiles to improve performance metrics like lift and drag in the design of aerodynamic shapes for airplanes.

The effectiveness of engineering procedures is increased by this proactive approach, which guarantees that the final designs fulfill or surpass predetermined requirements.

II. Neural Network

DE is essential for training neural networks in the field of machine learning. Through the optimization of the network's weights and biases, DE guarantees that the model effectively captures patterns in the data.

In order to enhance the neural network's capacity to recognize and categorize objects, for instance, DE actively modifies the neural network's parameters in image recognition, hence improving the model's overall performance.

III. Game Strategy

DE is a tool used by game creators to maximize strategies in competitive games. For example, DE actively refines decision-making algorithms in the design of non-player character (NPC) behaviors to provide more dynamic and tougher opponents.

Players are guaranteed an exciting and dynamic gaming experience with this application.

IV. Financial Optimization

DE is frequently used by the financial industry to optimize investment portfolios.DE assists investors in building portfolios that balance risk management and return maximization by modifying the asset allocation in accordance with past performance and risk preferences. To reach certain financial objectives, DE, for example, actively chooses the best combination of stocks, bonds, and other assets via portfolio optimization.

3.7 REFERENCES

[1]. Rocca, P.; Oliveri, G.; Massa, A. (2011). "Differential Evolution as Applied to Electromagnetics". *IEEE Antennas and Propagation Magazine*.53 (1):38–49. doi:10.1109/MAP.2011.5773566. S2CID 27555808.

[2] Storn, Rainer; Price, Kenneth (1995). "Differential evolution—a simple and efficient scheme for global optimization over continuous spaces" (PDF). *International Computer Science Institute*. TR (95). Berkeley: International Computer Science Institute: TR-95-012. Retrieved 3 April 2024.

[3]. Storn, R.; Price, K. (1997). "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces". *Journal ofGlobalOptimization*. 11 (4):341–359. doi:10.1023/A:1008202821328. S2CID 5297867.

[4]. Storn, R. (1996). "On the usage of differential evolution for function optimization". *Biennial Conference of the North American Fuzzy InformationProcessingSociety(NAFIPS)*.pp. 519–523. doi:10.1109/NAFIPS.1996.534789. S2CID 16576915.

[5]. Price, K.; Storn, R.M.; Lampinen, J.A. (2005). *Differential Evolution:A Practical Approach to Global Optimization*.Springer. ISBN 978-3-540-20950-8.

[6]. Feoktistov, V. (2006). *Differential Evolution: In Search of Solutions*. Springer. ISBN 978-0-387-36895-5.

[7]. G. C. Onwubolu and B V Babu, "New Optimization Techniques in Engineering". Retrieved 17 September 2016.

[8]. Chakraborty, U.K., ed. (2008), *Advances in Differential Evolution*, Springer, ISBN 978-3-540-68827-3

[9]. S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art", IEEE Trans. on Evolutionary Computation, Vol. 15, No. 1, pp. 4-31, Feb. 2011, DOI: 10.1109/TEVC.2010.2059031.

[10]. S. Das, S. S. Mullick, P. N. Suganthan, "Recent Advances in Differential Evolution - An Updated Survey," Swarm and Evolutionary Computation, doi:10.1016/j.swevo.2016.01.004, 2016.

[11]. Liu, J.; Lampinen, J. (2002). "On setting the control parameter of the differential evolution method". *Proceedings of the 8th International Conference on Soft Computing (MENDEL)*. Brno, Czech Republic. pp. 11–18.

CHAPTER-*4*

Multi Robot Box Pushing Using VIKOR Induced Differential Evolution Algorithm

Introduction

The chapter gives overview of the importance of DE and VIKOR methods.
It deals with the study of methodologies of optimization using VIKOR-DE.
It also explores the capabilities of the Algorithms for solving problems and
Optimizes the path planning.

It also briefs about different literary works done towards the advancement in the
field of different evolutionary algorithms.

It consists of the algorithms and pseudo-code for the above methodologies.

4.1 Multi Objective Optimization for Multi Robot Box Pushing

Multi agent Coordination is a vital research field of robotics and automation world. In multi robot box pushing problem we are using two classical robot to plan the local trajectory in shifting box from start state to goal state.In between the start state and goal state there are multiple obstacles inside the workspace.

In Multi-objective optimization we aim to minimize four objectives,

a) Total traversal time

b) Total energy

c) The penalty occurred if collision happens because of obstacles

d) Euclidean distance of boxes from current position to goal

There is a conflict between two objectives;to minimize the time of traversal the force should be maximized.Whereas to obtain minimum energy consumption minimum forces should be applied on the robots,and for this time is increased proportionally to reach the goal state.
Single optimization's perfomance fails here due to conflicting nature in between the objectives;that's why MOO has been introduced to solve real world problems efficiently.
Normal optimization technique is failed to give satisfactory result due to the large work space.

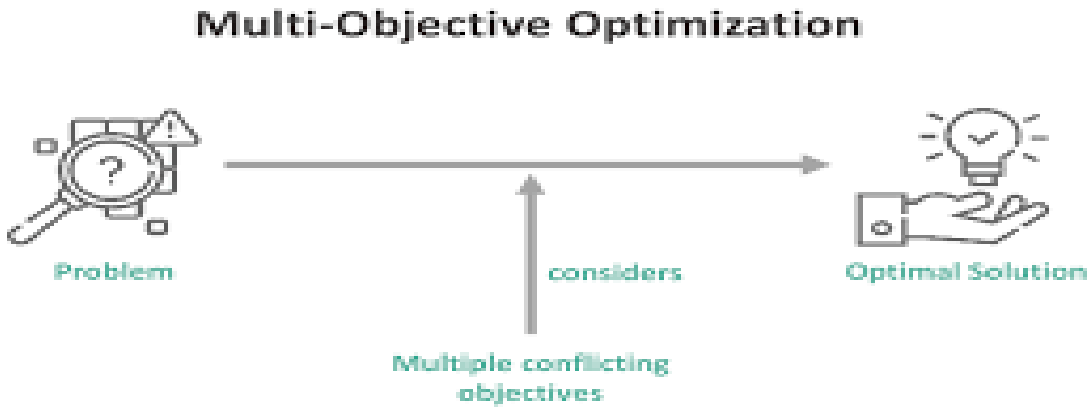


Fig: 4.1 MOO conflicts and selection

Using MOEA, we were able to achieve the optimal goal of traditional MOEA to solve real-world problems by hybridizing DE and VIKOR making to achieve MO.Thealgorithm named as VIKOR-Differential Evolution.

4.2 Problem Formulation

Multi robot problem consists of two identical homogeneous robots Q1 and Q2, controlled torque and force are jointly applied to rotate or move the box from a starting position to a target position. in a workspace by avoiding collisions of obstacles in the traversal path.The box-pushing problem consists –

I)Time	t_1
II)Energy	t_2
III)Route travelled by bots	t_3
IV)Collision Penalty	t_4

For each movement, **(iii)** steps are performed by bots.

The steps consists

(i) rotation by an angle **α** through the center of gravity of the box encompasses steps such as moving in specifically diagonal directions rather than conventional linear movements.

(ii) translating around length by d_t,

(iii) translation perpendicular to its length by d_p.

α_c**(f)** is the current alignment of the box with respect to the horizontal axis, (m_c(f), n_c(f)) be the current position of its center of the mass and (m_d(f), n_d(f)) is the current positions of corners which is d∈**{T,U,V,W}**. Here f represents the time index.Upnext position represented as ((m_c(f +1),n_c(f+1)), and 4 corners (m_d(f+1), n_d (f+1)),

later proceedings 3 steps are followed by

$\alpha_c(f+1) = \alpha_c(f) + \alpha$

$$m_c(f+1) = m_c(f) - d_t \cos(\alpha_c(f+1)) + d_p \sin(\alpha_c(f+1))$$
$$n_c(f+1) = n_c(f) - d_t \sin(\alpha_c(f+1)) - d_p \cos(\alpha_c(f+1))$$

$$m_d(f+1) = m_c(f)(1-\cos\alpha) + m_d(f)\cos(\alpha) - (n_d(f) - n_c(f))\sin(\alpha) - d_t \cos(\alpha_c(f+1)) + d_p \sin(\alpha_c(f+1)) \quad \textbf{(13)}$$

$$n_d(f+1) = n_c(f)(1-\cos\alpha) + n_d(f)\cos(\alpha) + (m_d(f) - m_c(f))\sin(\alpha) - d_t \sin(\alpha_c(f+1)) - d_p \cos(\alpha_c(f+1))$$

With d∈ {T,U,V,W}.

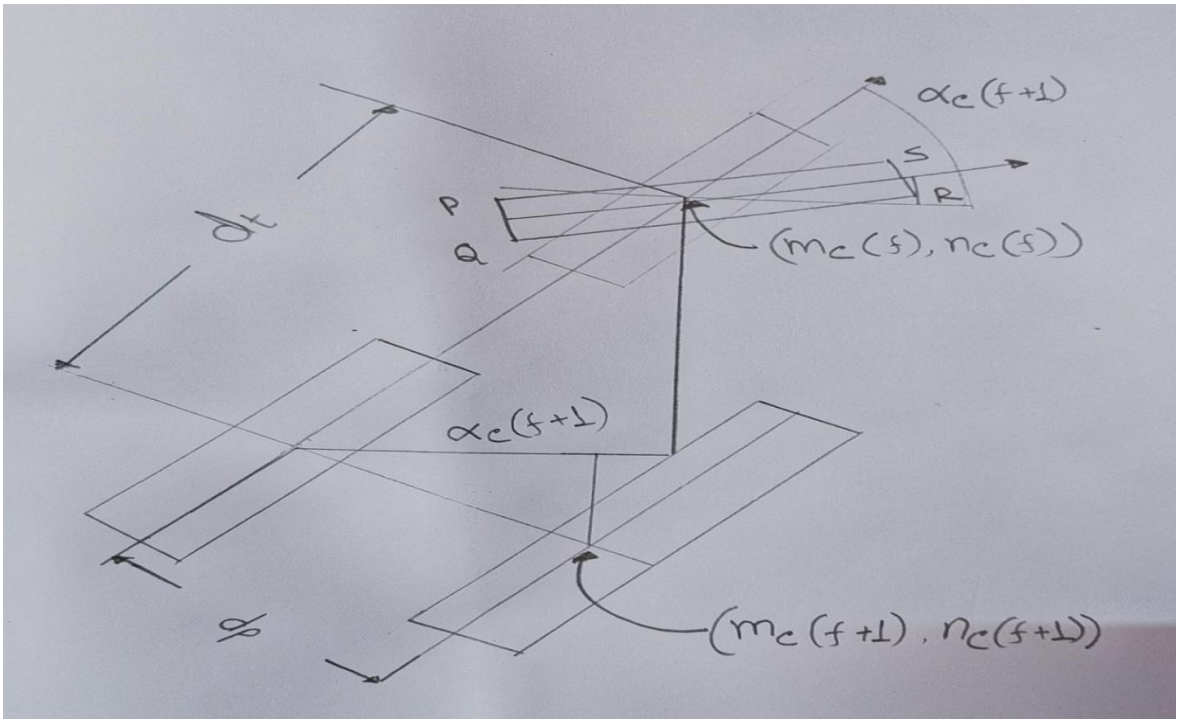


Fig: 4.2 Three steps in local movement of the box towards its goal

Forces

Denotion:

Torque	Γ
Rotational translation force	T_w
Perpendicular translation force	T_l

The time for 3 steps are calculated as below:-

$$F_1 = \sqrt{\frac{2\alpha J}{\Gamma}} \quad \textbf{(14)}$$

$$F_2 = \sqrt{2M\frac{d_t}{T_w}} \quad \textbf{(15)}$$

$$F_3 = \sqrt{\frac{2d_p M}{T_l}} \quad \textbf{(16)}$$

J denotes the moment of inertia of the box whereas V is the width, E is the mass, R is the length of each boxes.

Energies

The following are the energy components that the robots used during the three steps that involved rotating and translating the box:

$$G1 = \alpha\Gamma \quad \textbf{(17)}$$

$$G_2 = T_w d_t \quad \textbf{(18)}$$

$$G_3 = T_l d_p \quad \textbf{(19)}$$

First and Second Objective

The robots needed the following total time and energy to finish this one local movement in three steps:

$$t_1 = \textbf{F1+F2+F3} \quad \textbf{(20)}$$

$$t_2 = \textbf{G1+G2+G3} \quad \textbf{(21)}$$

Third Objective Equation

Total time t_1 and total energy t_2 are contradictory in nature as because of their conflicting nature because of normal optimization. In MOO because of this conflict third objective function t_3 has been introduced to guarantee that the robots' time and energy-optimal planning should direct the package toward its intended position (m_g, n_g) .

It is satisfied by minimizing -

$$t_3 = \sqrt{(m_g - m_{c(f+1)})^2 + (n_g - n_{c(f+1)})^2} \quad \textbf{(22)}$$

Fourth Objective Equation

The fourth objective is to avoid collisions with obstacles by planning workmap.
It is obtained by minimizing the penalty function-

$$t_4 = 2^{-dmin} \quad \textbf{(23)}$$

d_{min} indicates the shortest path between the box's corners and the closest barrier or arena edge, which is shown by

$$d_{min} = \text{minimum}(d_{t1}, d_{t2}, d_{t3}, d_{t4}, d_{p1}, d_{p2}) \quad \textbf{(24)}$$

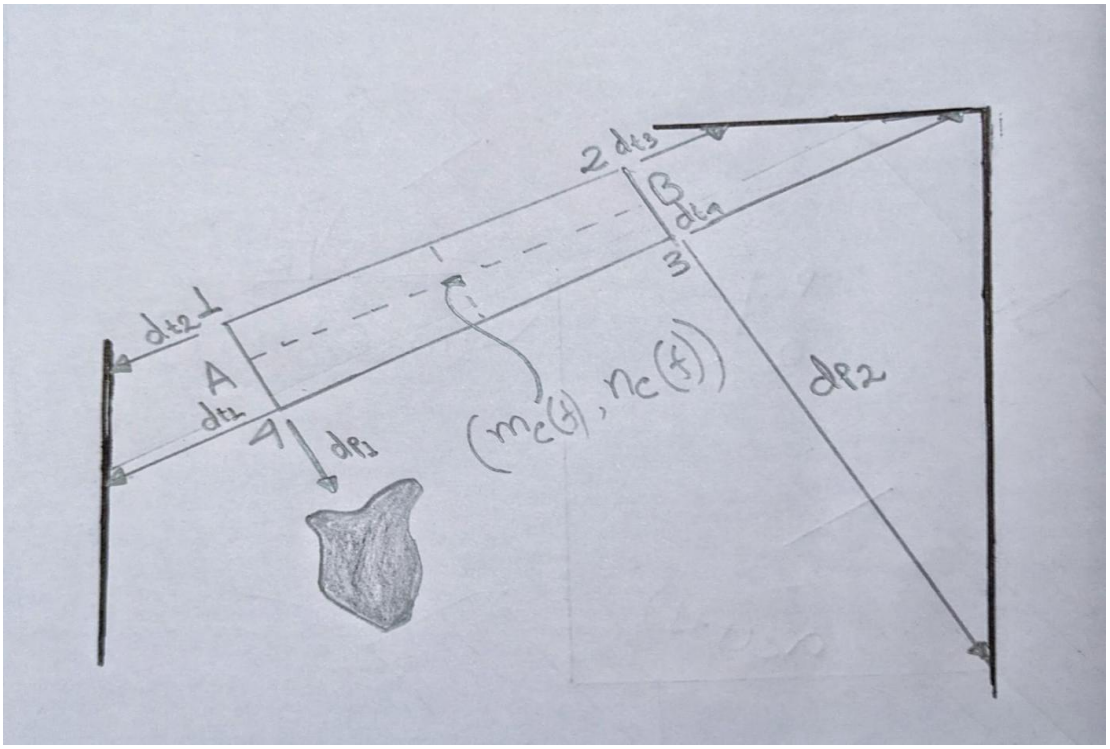


Fig: 4.3 Computation of distance by robots from obstacles and arena boundary

4.3 VIKOR Induced Differential Evolution

A rapid decision-making algorithm is needed for the multi-robot box-pushing problem, a real-world complex MOO problem, instead than a diverse Pareto front. Here, the MOO issue is resolved with the aid of the DE optimizer. In this case, the VIKOR technique makes this possible. Put differently, VIKOR helps to modify the traditional single objective DE in the direction of MOO.

The system is able to identify an optimal or nearly optimal solution that balances all of these criteria, resulting in an efficient and successful multi-robot box-pushing operation. This is achieved by employing DE to explore different coordinating techniques and using VIKOR to assess and rank these strategies.

The integration of VIKOR with Differential Evolution provides a powerful approach to solving multi-criteria optimization problems like multi-robot box pushing. It leverages the strengths of both methods to ensure effective coordination, path planning, and force distribution among the robots, resulting in a balanced and optimized solution.

4.4 Benefits of combining VIKOR with DE

a) **Enhanced decision making:** VIKOR helps in making balanced decisions considering multiple conflicting criteria.

b) **Improved Optimization:** DE provides a robust optimization framework to explore and exploit the search space effectively.

c) **Effective Coordination:** The combination ensures that the multi-robot system is optimized not just for a single criterion but a balanced compromise among several important factors.

4.5 Intregration of VIKOR induced DE for Multi Objective Optimization

Steps Involved

1. Initialize:

Population	An initial population of candidate is generated. Solutions where each candidate represents a possible way of coordinating the robots for the box pushing task.
Evaluate	Each candidate solution is calculate based on different criteria

2. Criteria Identification:-

Objective Criteria: Define the criteria for evaluating the effectiveness of a solution. Common criteria might include:

a) Total traversal time

b) Total energy

c) The penalty occurred if collision happens because of obstacles

d) Euclidean distance of boxes from current position to goal

3. DE Operators:-

Mutation	For each candidate solution, create a mutationed vector by adding the weight difference between two randomly selected population vectors to a 3rd vector.
Crossover	Combine the mutationed vector with the target vector to create a trial vector.
Selection	Evaluate trialed vector using the defined criteria.When the trial vector performs better it replaces the target vector in the population.

4. Compromise Ranking using VIKOR :-

VIKOR calculation-For each generation, use the VIKOR method to rank the candidate solutions. This involves:

Normalization: Normalize the values of each criterion across all solutions.

Utility and regret measures: Calculate the utility (benefit) and regret (shortfall) measures for each solution.

Compromise Ranking: Determine the VIKOR index for each solution, which is a compromise measure balancing the utility and regret.

5. Iterative Optimization:-
Continue the DE operations (mutation, crossover, selection) and VIKOR ranking over multiple generations until convergence criteria are met (e.g., a set number of generations or minimal improvement).

6. Solution Selection:
Thefinal solution is chosen based on the VIKOR ranking from the last generation, ensuring a compromise between the different criteria.

4.6 MULTI-ROBOT BOX-PUSHING USING VIKOR INDUCED DE

The box-pushing algorithm uses the suggested VIKOR-DE algorithm to find$[\alpha, d_t, d_p, \lceil, T_w, T_l]$ by jointly minimizing (20) to (24). It starts at the current position $(m_c(f)n_c(f))$ of the box's center of mass at the f_{th} instant. Then, for a single local movement of the box, the optimal values of the decision variable determined by VIKOR-DE are utilized to update $(m_c(f+1), n_c(f+1))$. Until $(m_c(f),n_c(f))$ approaches the objective position sufficiently, the operation is repeated (m_g, n_g). The following is the pseudo-code for resolving the multi-robot box-pushing issue.

The following is the pseudo-code for resolving the multi-robot box-pushing issue.

The multi-robot-box-pushing procedure

Incoming: Starting location of the box's center of mass $(m_c(f)n_c(f))$ and corners $(m_d(f),n_d(f))$ for d falling inside {T, U, V, W} at f = 0, goal Position (m_g, n_g), **width V,mass E, length R** small +ve constant ɣ .

Outgoing:

Translation Forces	T_w and T_l
Angle of Rotation	α
Torque due to rotation	Γ
Translation Lengths	d_t and d_p

START

I. Set $f \Longleftarrow$ 1.

II.Set J $\Longleftarrow$E×($R^2 + V^2$)/12.

III. **Repeat**

1. Call **DE-VIKOR** $((m_c(f)n_c(f)),(m_d(f), n_d(f))$ for i∈ {**T,U,V,W**}, J, E; $\alpha, d_t, d_p, \lceil, T_w, T_l)$.

2.Calculate $(m_c(f+1), n_c(f+1))$ and $(m_d(f+1), n_d(f+1))$ for i∈ {**T,U,V,W**} using (13).

3. Move the box from $(m_c(f+1), n_c(f+1))$ to $(m_c(f+1), n_c(f+1))$.

4. Set $f \Longleftarrow$ f+1

Until

$$\sqrt{(m_g - m_c(f))^2 + (n_g - n_c(f))^2} \leq \gamma$$

End

Process VIKOR

Decision matrix (DM) as **input.**
Output: Almost perfect substitute.

BEGIN

a) Use (1), normalize the objective function value entries of the Decision Matrix.

b) Apply (II) and (3) to find the positive and negative ideal sets of values for the objective function, respectively.

b) Apply (4) to (8) to the VIKOR index evaluation of the DM alternatives.

d) Arrange the options according to their individual VIKOR indices in ascending order.

e) Return the candidate solution, or alternative, to the top of the sorted list.

CLOSE

4.7 Results and Simulations

The experiment is run on an Intel Core i5-7th generation CPU with a maximum clock speed of 3.10GHz using simulation software. In order for the robots to transport the box from the starting position to the ultimate objective coordinate, the experiment creates a world space.A script that may be used with GNU Octave 9.1.0 is created to solve the simultaneous time and energy optimization challenge.

In order to compare the results suggested VIKOR-DE and NSGA-II and SPEA-II, two differently designed workspaces, as shown in Fig. 4.4 (a) and (b), have been taken into consideration. About the center of mass of the box, the mass and moment of inertia are fixed at 50 kg and 800 kg, respectively.

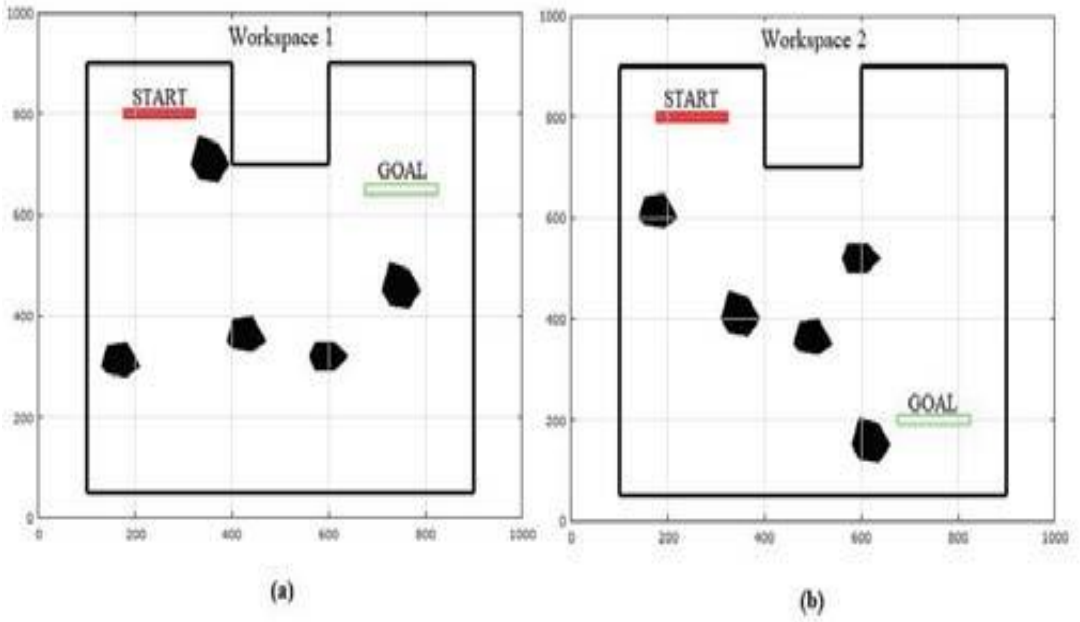


Fig: 4.4 Multi-robot box-pushing problem workspaces

4.8 Choosing VIKOR Weight Factors

The optimization efficacy of VIKOR-induced DE is highly dependent on the appropriate tuning of weight factors z_j for j = 1, 2,..., M, as can be shown from (16). It makes sense that different robot worldmaps have different values for these weight considerations.The objective function values for every conceivable combination of wjs for j = 1, 2,..., M are recorded in order to determine the ideal values of the four weight factors for each workspace (Fig. 4.4). The values of the relevant objectives t_1 to t_4 for various conceivable combinations of z_1 to z_4 are shown in Tables 1 and 2.

A few space economy results are tabulated.The two best sets of weight factors for workspaces 1 and 2 are $\{z_1, z_2, z_3, z_4\}$ are {1, 1, 700, 1} and {1, 1, 500, 1} respectively, according to the data shown in Tables 1 and 2.

Weight Factors				Function Objective Values			
z_1	z_2	z_3	z_4	t_1	t_2	t_3	t_4
1	1	100	1	318.5	31.1	730.4	2.87
1	10	100	0.1	293.7	44.7	696.5	2.89
1	1	200	1	226.6	31.8	714.7	1.76
1	1	500	1	266.8	29.8	685.6	1.78
2	5	500	2	353.4	33.9	687.9	2.88
1	**1**	**700**	**1**	**235.8**	**25.9**	**676.6**	**1.78**
1	1	800	1	225.0	41.7	615.1	1.81
1	1	1000	1	231.5	25.3	616.8	1.84
5	2	1000	2	234.8	27.9	634.4	1.81
1	1	1200	1	241.3	32.4	604.8	1.85

TABLE 1: VARIATION OF OBJECTIVES BASED ON WEIGHT FACTORS IN WORKSPACE 1

Weighting Factors				Function Objective Values			
z_1	z_2	z_3	z_4	t_1	t_2	t_3	t_4
1	1	100	1	424.1	47.76	1332.3	3.75
1	10	100	0.1	447.4	63.12	1484.5	3.71
5	10	100	1	689.9	58.77	2341.7	7.30
1	1	200	1	502.2	64.79	1485.7	4.49
1	1	300	1	760.8	80.20	1572.7	6.50
1	**1**	**500**	**1**	**419.4**	**48.62**	**1183.9**	**3.69**
1	1	700	1	430.08	55.11	1139.4	3.71
5	2	700	2	505.35	50.945	1686.0	5.59
1	1	1000	1	481.2	48.12	1267.9	3.64
1	1	1200	1	495.3	40.64	1117.9	3.73

TABLE 2: VARIATION OF OBJECTIVES BASED ON WEIGHT FACTORS IN WORKSPACE 2

Algorithm	No. of Steps	Total Time (Seconds)	Total Energy (KJ)
VIKOR-DE	**2 (0.828)**	**234.88 (88.6)**	29.927 (5.387)
NSGA-II	6 (2.362)	1130.183 (456.55)	**23.361 (2.937)**
SPEA-II	8 (1.140)	677.58 (88.26)	55.102 (9.913)

TABLE 3: COMPARATIVE PERFORMANCE ANALYSIS IN WORKSPACE 1

Algorithm	No. of Steps	Total Time (Seconds)	Total Energy (KJ)
VIKOR-DE	4 **(0.836)**	**458.36 (62.630)**	48.15 (4.959)
NSGA-II	5 (0.836)	1124.31 (356.269)	**15.86 (3.185)**
SPEA-II	6 (2.168)	888.244 (160.806)	67.18 (7.766)

TABLE 4: COMPARATIVE PERFORMANCE ANALYSIS IN WORKSPACE 2

4.9 CONCLUSION

In this thesis we discussed the box pushing using twin similar identical robots using VIKOR-DE algorithm technique.There are 4 objectives for the problem for the robots' travelling path.
To minimize the objectives we had faced conflicts between forces and time.

After that using MOO strategy and hybridizing VIKOR algorithm with DE we were abled to netralise the conflict.

We have compared the results of perfomances with NSGA-II with VIKOR-DE.We have seen that NSGA-II efficient on the basis of energy.On other hand we have seen that VIKOR-DE optimized both the objective values

4.10 FUTURE SCOPE

Advanced optimization algorithms:More sophisticated optimization algorithms that integrate VIKOR with other techniques such as GA and particle swarm optimization to more efficiently handle larger problem instances and complex environments. Develop.

Dynamic Environment Adjustment: Incorporates real-time data and feedback from the environment to dynamically adjust criteria weights and update decision-making processes as conditions change. This could include the use of sensors and machine learning algorithms for adaptive decision-making.

Multi-goal optimization: Extend VIKOR-DE to manage large numbers of goals or criteria in multi-robot box-pushing scenarios and more effectively consider trade-offs between conflicting goals.

Collaborative decision making: Explore methods for collaborative decision making between multiple robots. Each robot contributes to the

decision-making process based on local observations and goals, resulting in a more efficient and coordinated box-pushing strategy.

Integration with simulation and virtual environments: Integrate Vikor-DE based optimization with simulation and virtual environments to test different box push strategies risk-free and cost-effectively before deploying them to real-world scenarios. and verify.

Overall, VIKOR-DE's future application in multi-robot box pushing has the potential to provide efficient and effective decision-making capabilities, resulting in optimized performance, improved resource utilization, multiple Improved coordination between robots is achieved.

4.11 REFERENCES

[1] Pratyusha Rakshit, Amit Konar,Swagatam Das,Lakhmi C. Jain, and Atulya K. Nagar, "Uncertainty Management in Differential Evolution Induced
Multi-objective. Optimization in Presence of Measurement Noise", IEEE Transactions on Systems, Man, and Cybernetics: Systems (Volume: 44, Issue: 7, July 2014).

[2] Jayasree Chakraborty, Amit Konar, Atulya Nagar, and Swagatam Das, "Rotation and Translation Selective Pareto Optimal Solution to the BoxPushing Problem by Mobile Robots Using NSGA-II", *IEEE Congress on Evolutionary Computation, 2009.*

[3] James Kennedy, and R. Eberhart, "Particle swarm optimization", IEEE International Conference on Neural Networks, 1995.

[4] Serafim Opricovic, Gwo-Hshiung Tzeng, "Extended VIKOR method in comparison with outranking methods", European Journal of Operational
Research (Volume: 178, Issue: 2, April 2007).

[5] C. A. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with Particle Swarm Optimization," IEEE Transactions on Evolutionary Computation (Volume: 8, Issue: 3, June 2004).

[6] K. Deb, A. P. S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA II,” IEEE Transactions on Evolutionary Computation (Volume: 6, Issue: 2, April 2002).

[7] Carlos. M. Fonseca, and Peter. J. Flaming, “An overview of evolutionary algorithms in multiobjective optimization”, Evolutionary Computation Journal (Volume: 3, Issue: 1, Spring 1995).

[8] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the Strength Pareto Evolutionary Algorithm”, TIK-Report 103, May 2001.

[9] T. Langle, and H. Worn, “Human-robot cooperation using multi-agent systems,” Journal of Intelligent and Robotic system, (Volume: 32, Issue: 2, October 2001.

[10] B. Innocenti, B. Lopez, and J. Salvi, “A multi-agent architecture with cooperative fuzzy control for a mobile robot”, Robotics and Autonomous Systems (Volume: 55, Issue: 12, December 2007).

[11] Serafim Opricovic "Fuzzy VIKOR with an application to water resources planning", Expert Systems with Applications (Volume: 38, Issue: 10, September 2011).

[12] E.Zitzler, “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications” dissertation submitted to the Swiss Federal Institute of Technology Zurich for the degree of Doctor of Technical Sciences Diss (ETH No. 13398, 1999).