PATH PLANNING OF INTELLIGENT MOBILE ROBOT USING SIMPLIFIED SWARM OPTIMIZATION ALGORITHM

A THESIS SUBMITTED TO THE GRADUATE
SCHOOL OF APPLIED SCIENCES
OF
NEAR EAST UNIVERSITY

By
RAWAND BASSAM ISMAIL

In Partial Fulfillment of the Requirements for
The Degree of Master of Science
in
Software Engineering

NICOSIA, 2017

RAWAND BASSAM
ISMAIL

PATH PLANNING OF INTELLIGENT MOBILE ROBOT
USING SIMPLIFIED SWARM OPTIMIZATION ALGORITH

NEU
2017

PATH PLANNING OF INTELLIGENT MOBILE ROBOT USING SIMPLIFIED SWARM OPTIMIZATION ALGORITHM

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES
OF
NEAR EAST UNIVERSITY

By
RAWAND BASSAM ISMAIL

In Partial Fulfillment of the Requirements for
The Degree of Master of Science
in
Software Engineering

NICOSIA, 2017

Rawand Bassam Ismail: PATH PLANNING OF INTELLIGENT MOBILE ROBOT USING SIMPLIFIED SWARM OPTIMIZATION ALGORITHM

Approval of Director of Graduate School of
Applied Sciences

Prof. Dr. Nadire Çavuş

We certify this thesis is satisfactory for the award of the degree of Masters of Science in Software Engineering

Examining Committee in Charge:

Assist. Prof. Dr. Eser Gemikonakli	Computer Engineering Department, University Of Kyrenia
Assist. Prof. Dr. Kamil Dimililer	Electrical & Electronic Engineering Department, NEU
Assist. Prof. Dr. Yöney Kırsal Ever	Supervisor, Software Engineering Department, NEU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Rawand Ismail
Signature:
Date:

ACKNOWLEGEMENTS

I would like to thank almighty Allah for the grace that he has shown to me in this research. My special thanks go to my supervisor Assist. Prof. Dr. Yöney Kirsal Ever, for her encouragement and mentorship, assistance, and advice all through my research, having her as my supervisor has been wonderful. Eventually, I would like to thank my wife for giving me all the types of support to get this chance to carry on my education overseas at the Master level in Software Engineering Department at NEU.

Furthermore, there is a long list of friends that I would like to thank. I can’t mention them all but I would like to thank them from all of my heart for their valuable help and support since I was in my early study until now.

To my parents….

ABSTRACT

This study examined how path planning of intelligent mobile robot could be enhanced by utilizing simplified swarm optimization (SSO) in working environment with irregular obstacles. The conceptual framework of this study was driven from an inspiration of communal behavior of birds flocking and fish schooling. This conceptual framework was supported by swarm intelligence, which is one of the famous research areas in the field of computational swarm intelligence such as Particle swarm optimization (PSO) algorithm. Based on this, observations have been made which signified that mobile robots are significantly affected by path planning problems. However, solutions have been established recently on how to tackle the aforementioned problems. Nevertheless, such solutions have also been discovered to possess numerous weaknesses. Therefore, based on these weaknesses, this study tried to propose effective solutions which can yield a high quality and efficient mobile robots. The technique this research adopted was SSO. This technique was adapted in order to provide an effective solution to the weaknesses. After careful simulations, the algorithm result showed that *SSO* does not converge in the closed work interface in which there is no path between the initial and the target point. When particle is updated SSO using social effect term only. Furthermore, the result displayed that when the particles path falls within an obstacle space, it automatically relocated to an area of an obstacle free. This showed autonomy and energy efficiency within the particles.

Keywords: Swarm intelligence; path planning; particle swarm optimization; intelligent mobile robot; simplified swarm optimization

ÖZET

Bu çalışma, düzensiz engeller olan çalışma ortamında ve alanında, basitleştirilmiş sürüler optimizasyonunu (SSO) kullanarak akıllı mobil robotların yol planlamasının nasıl geliştirileceğini incelemiştir. Bu çalışmanın kavramsal çerçevesi, kuş sürüleri ve balık sürüleri için ortak bir davranışın esin kaynağı olmuştur. Kavramsal çerçeve; parçacık sürüsü optimizasyon (PSO) algoritması gibi, hesaplamalı(analitik) sürü zekası alanındaki ünlü araştırma alanlarından biri olan sürü uyumu ve aklı tarafından desteklenmiştir. Buna dayanarak, mobil robotların yol planlama problemlerinden önemli ölçüde etkilendiğini gösteren gözlemler yapılmıştır. Son zamanlarda yukarıda bahsedilen sorunlarla nasıl başa çıkılacağı üzerine çözümler belirlenmiş ve saptanmıştır. Bununla birlikte, bu gibi çözümlerin çok sayıda zayıflığa sahip olduğu keşfedilmiştir. Buradaki zayıflıklara dayanarak, çalışmada yüksek kalitede ve verimli mobil robotlar üretebilecek etkili çözüm önerileri sunulmaya çalışılmıştır.Bu araştırmanın benimsediği teknik basitleştirilmiş sürüler optimizasyonudur. Bu teknik zayıf noktalara etkili bir çözüm getirmek için uyarlanmıştır. Dikkatli bir şekilde simülasyonlar yapıldıktan sonra, algoritma sonucu, basitleştirilmiş sürüler optimizasyonun (SSO), başlangıç ve hedef nokta arasında bir yol bulunmadığını kapalı iş ara yüzünde birleşmediğini göstermiştir. Parçacık(partikül) güncellendiğinde basitleştirilmiş sürüler optimizasyonu (SSO) yalnızca sosyal etki dönemini kullanmıştır. Dahası, görüntülenen sonuç; parçacık yolu bir engel alanına düştüğünde otomatik olarak engelin bulunduğu bölgeye taşındığını göstermiştir. Sonuç olarak; parçacıklar(partiküller) içinde özerklik ve enerji verimliliği olduğu gösterilmiştir.

Anahtar Kelimeler: Sürü zekası; yol planlaması; parçacık sürüsü optimizasyonu; akıllı mobil robot; basitleştirilmiş sürüler optimizasyonu

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC:	Artificial Bee Colony.
ACO:	Ant Colony Optimization.
BFO:	Bacteria Foraging Optimization.
AIS:	Artificial Immune System.
ANN:	Artificial Neural Networks.
FIPS:	Fully Informed Particle Swarm optimization algorithm.
FL:	Fuzzy Logic.
GA:	Genetic Algorithm.
GCPSO:	Guaranteed Convergence Particle Swam Optimization.
NF:	Neuron-Fuzzy.
SI:	Swarm Intelligence.
PSO:	Particle Swam Optimization.
SSO:	Simplified Swarm Optimization.
MPSO:	Multi-start Particle Swarm Optimization.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The academic research fraternity has been characterized by vast studies that sought to address how mobile robots are being impacted by path planning problems. This commenced in the mid-1970s. Despite the proposition of numerous solutions, discoveries made contend that such solutions have inherent weaknesses and strengths. Thus, the decision to produce a sound solution in research hinges on efforts to attain high efficiency and quality (Yun, Ganapathy and Chong, 2010). Hence, satisfactory results from a path can be obtained when hindrances in local minimum positions, unrequired procedures and wastage of time are minimized by the robot. In addition, a satisfactory path is the one that eradicates all obstacles in the area (Han, 2007).

Mohanty and Parhi (2013) contends that there are numerous strategies that can employ in navigating an intelligent mobile robot. Their study also exhibited that (AIS) an acronym for Artificial Immune System, Particle Swam Optimization (*PSO*), Genetic Algorithm (*GA*), Neuron-Fuzzy (NF), Artificial Neural Networks (*ANN*), Ant Colony Optimization (*ACO*) and Fuzzy logic (*FL*) which constitute part of heuristic approaches are capable of offering highly probable and effective mobile robot navigation results (obstacle-avoidance and target reaching). Thus, heuristic approaches enable the mobile robots to navigate reliably among the obstructions without knocking them and thereby moving to their predefined destination point. Such approaches are indispensable in addressing the local smidgens delinquent. Scholars have immensely been looking for supplementary productive approaches in order to tackle this issue relating to accompanying area, prior studies on path planning and robot’s navigation utilizing molecule swarm are inspected. Raja and Pugazhenthi (2012) gave a review of research progress about on-line and offline environments and mobile robot path planning.

Ordinarily utilized exemplary and developmental methodologies of path planning of mobile robots have been deliberated, and demonstrated that the transformative algorithms are computationally proficient. Similarly, challenges required in building up a computationally effective path planning algorithm are tended to. According to Zhao and Zu (2009), they proposed a modified particle swarm optimization algorithm for the robot path planning in vigorous atmosphere; two parameters of distribution degree in relation to particles and dimension and distance in relation to particle were presented into the recommended algorithm in order to circumvent untimely conjunction. Furthermore, Ahmadzadeh and Ghanavati (2012) proposed an understanding way to deal with the triangulation of mobile robots in obscure situations, the triangulation problem develops to an optimization problem and it was later comprehended by a PSO algorithm. In view of the position of the objective, an assessment work for each element in PSO is ascertained. It is accepted that robots can distinguish just hindrances in a constrained range of its sensors. It is worthy to note that the hindrances can be portable and dynamic but the environment should be dynamic.

1.2 Background to Study

Path planning algorithms can be categorized into two distinct elements; these are the local (confined) path planning (on-line) and global (universal) path planning (off-line). Advance awareness of trajectory of moving obstacles and stationary obstacles is always available in global path planning of robots in environments such that the robot initially assesses the desired path and confines to it until it reaches its destination. But in local path planning, all information is not available in advance.

1.3 Research Objectives

This study examines the use of Simplified Swarm Optimization (SSO) in order to deal with the issue of mobile robot triangulation (navigation). This issue for ascertainment is the most productive mode which requires least time and shortest path in order to move from a starting locus to an objective locus in relation to environment with haphazard circumstances.

1.4 Organization of the Study

This study is organized into five chapters:

Chapter 1: Provides introductory insights about the study.

Chapter 2: Deals with related theoretical and empirical literature review.

Chapter 3: The methodological procedures that were employed in this study so to arrive at study findings and offer recommendations are addressed.

Chapter 4: Simulations and results of study findings is given.

Chapter 5: Concludes this study by looking at conclusions, recommendation and suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Particle Swarm Optimization

To understand or have a clear understanding of what particle swam optimization algorithm, resolution swarm differs and have different criteria with bird swarm optimization process, the birds' navigating from point A to point B is employed in the improvement of resolution swarm, while, ad-hoc communication is necessary in solving most utopian solution, the location of the target address is equivalent to the most utopian resolution during the whole sequence. Casually, the main address of the target address or in our case utopian resolution is determined by controlled collective sustenance which is the main idea of swarm optimization algorithm which is founded by collective progress to try and achieve a desired goal (Dos Santos et.al., 2011).

According to Trelea 2003, the Particle swarm optimization is a technique in which each node behavior in the network is aimed at achieving a collective goal of the swarm as a group. In PSO, a problem is solved by following or moving in a collective manner (Zhao and Zu, 2009). Every particle or node in the swarm is aware of its location and coordinates regarding to the problem statement and try to find the best or optimized option to the problem (Shiltagh and Jalal, 2013). This is termed or referred to as *pbest*. Similarly, another optimization value that is used in measuring particle swarm optimization is the best value, also another value is *lbest* which is obtained by choosing the best and closest node. And finally, the node or a particle that has the highest population of neighbors in its topological location is called global best and is termed gbest (Shiltagh and Jalal, 2013).

The definition of PSO consists of time interval, requires change in velocity and acceleration, every node is linked to the *pbest*, *lbest*. Movement is by nodes is autonomous, which is distinguished by the figure of nodes at risks of acceleration using the *pbest*, *lbest* locations. When the optimized values are found, the nodes retrieves the velocity, points using the given equation (see Equation 1.1).

$$Vi(k+1) = Vi(k) + c1 * rand(\ ) * (Pi(k) - Xi(k)) + c2 * rand(\ ) * (g(k) - Xi(k)) \tag{1.1}$$

$$Xi(k+1) = Xi(k) + Vi(k+1) \tag{2.1}$$

where

$Vi(k)$ is velocity of particle i at iteration k.

$Xi(k)$ is the position of particle i at iteration k.

$Vi(k+1)$ is new velocity of particle i at iteration k+1.

$Xi(k+1)$ is the new position of particle i at iteration k+1.

rand( ) is random variable with number between (0,1).

c1 local acceleration coefficient.

c2 global acceleration coefficient.

usually $c1 = c2 = 2$.

The main idea is designing an algorithm that can effectively address various optimization problems.

2.2 Literature Review

In this section, existing Particle Swarm Optimization are discussed in detail. One of the existing studies established by Russell Eberhart and James Kennedy (1995) revealed that the existence of concept of a particle swarm optimization with reference. However, analyzing the steps of its advancement from communal simulation to optimization standard, and subsequent presentation offers some of the limited procedures that can be adopted to execute the concept (Yun, Ganapathy and Chong, 2010).

The adoption of one model is presented in enhanced detail, accompanied by conclusions gathered from applications and tests superimposed on the model or criterion and these been established to operate successfully. Eberhart and Kennedy (1995) further proposed a distinct model of the particle swarm optimizer and they explored on how shifting in relation to paradigm transmute the expanse of restatements needed to reach a maximum error that can be handled, and each node is reused to find a global minimum in the swam (Han, 2007).

Based on this context, three accounts were examined, two samples types of *lbest* form, one node has two neighbors and the latter node having six neighbors as an assessment criteria. It seems that the primary GBEST form functions with regards to number of recapitulations to nodes becoming unified, as the *lbest* is created by creating two or more repellent nodes in confined minimal space (Shi, 2001). Observations have been made that the hunt for a PSO procedure without the initial segment is a procedure where the search space factually declines through the eras (Zhao and Zu, 2009). Then again, by including the initial segment, the particles tend to increase the space. That is, they can examine a new area. Along these lines, they have a more probable potency to undertake a worldwide search capacity by including the initial segment. Both the global and local searches are important in addressing similar complications.

A trade off exists between universal and confined discovery for diverse situations, a unified solution to the diverse stabilities among the confined discovery criteria of and universal discovery ability. In relation to this, Mohanty and Parhi (2013) established that Yuhui et al, 1998 proposed the inertia weight is represented by (w) into the Equation (1.1) as shown in Equation (1.2).

$$Vi(k+1)= \boldsymbol{w} * Vi(k) + c1* rand( ) * (Pi(k) - Xi(k)) + c2* rand( ) * (g(k) - Xi(k)) \qquad (1.2)$$

$$Xi(k+1) = Xi(k) + Vi(k+1) \qquad (2.2)$$

Researches has shown that a substantial latency weight encourages worldwide investigation (snew zones discovery), few among the nodes has a tendency of conducting within searches, i.e. calibrating the present search point of interest. In PSO, the area merged is termed as *local well* comprising a *local minimum*, initially it may look like a trustworthy as an escape route be a promise because it is kind of force incorporated into the process by means of the forces within; periodically, in any case, particles momentum declines reaching a low stage making the swarm fall into a situation known as *stagnation*, making it harder for the swarm to breach the algorithm difficult to escape. Casually, few nodes proceed in this state which triggers a case known as "*solution refinement"* or use taking after an underlying period of *exploration* has elapsed, empirical observations have been shown that sufficiently with time variant, velocities turn out to be small at their point of normal rate of reduction, even the closest arrangement might be wiped out of the segment of the space provided for the search, thus particles can for all intents and purposes be relied upon to achieve the goals in future recapitulations. In conventional Particle Swarm Optimization, in the absence of a universal algorithm to be established by a node molecule for quite a while, all nodes focalize broadcast the present universal best, perhaps taking out which is the closest neighborhood minimizer. Raja and Pugazhenthi (2012) contend that this has been addressed the Guaranteed Convergence PSO (GCPSO) which utilizes an alternate speed redesign the algorithm to be suit the best particle, though it has its own particular best, universal best both lie at a comparative opinion, theoretically in standard Particle Swarm Optimization employs the limits of the node that the best optimization, since it is unequivocally drawn toward that specific location, it possess an incapacitating energy, expanding velocities toward that point keeping it searching by any extend of the creative ability (Raja and Pugazhenthi, 2012). GCPSO is along these lines said to guarantee joining to a neighborhood minimizer. There is still an issue, in any case, in which nodes are bound to join a neighborhood which limits the number of occurrence before encountering a certifiable universal limiter. Regards to this, Van proposed in give in Bergh made multi-start PSO (MPSO) basically initiate a restart when stagnation is initiated (Zhao and Zu, 2009).

Restarting in MPSO implies starting another hunt with a substitute assembling of irregular numbers created so that even beginning positions are not the same as it its initial past ventures. When restarted, nodes forget their record of its initial past inquiry eventually each inquiry is self-ruling from those did before. After each node discovery, a universal best tested to compare its efficiency with the previous global best discoveries. A defined number of restarts is set and completed, the node with the best universal search is proposed of the test sample. Different leveled type of PSO metaheuristic was proposed by Stefan et al in 2005 (Ahmadzadeh and Ghanavati, 2012). With the new proposal termed as H-PSO, the nodes neighborhoods are being engineered in a newly dynamic chain of significance which is used to describe a zone formation. Dependent upon how their so-far best is discovered and takes course of action, the node climb or degrade the dynamic system. Thus, making remarkable nodes that roam in the dynamic system of greater effect on the swarms at large. Propose a variety of H-PSO, in which the structure is effectively balanced even with midst of the execution of the optimization. Another variety is how can it be dole out different approach to the each node with regards to its level in the ascending order.

Propositions thus sought to enhance the performance of H-PSO by examining a progressively advancing expanding level of the tree topology of a variation of H-PSO (AH-PSO). An alternative option that can be employed to increase in H-PSO is to utilize particular qualities of the first weight (w) of the nodes as per each stage in the packing order. This algorithm has proved that it can be used to accomplish a categorized goal for various test work with exemption of Rastrigin work progressively than each and every variety of Particle Swarm Optimization. Yang and Simon (2005) proposed a substitute procedure aimed at a prevalent course of action. In the proposed New PSO Technique introduced here, each node changes location in perspective of its own past most exceedingly most noticeably awful answer and its gathering's past most observably terrible to find perfect esteem. The procedure here is to avoid a particle's past most exceedingly terrible organization and its grouping's past most observably most exceedingly bad in light of similar formulae of the general PSO. The condition for position and speed proceeds as before, nevertheless the term used is more abysmal position instead of the node. Particle Swarm Optimization has proven to be handy in its quick discovery in a couple of mind boggling inquiry and enhancement interferences. Moreover, Particle Swarm Optimization usually consist of neighborhood optimization techniques.

To achieve a superior elucidation, Wang et al. (2007) suggested resistance which is based upon PSO. OPSO aims in fortifying the haphazard of Particle Swarm Optimization and avert early on multi-modal capacities. Konar (2000) recommended a procedure that exploits constraint relied learning on every node and relates ad-hoc Cauchy joining on the best node. A few test shows that nodes in the Particle Swarm Optimization will permits inside their aforementioned best node and universal best node by the entire node prior to joining them. One of the differences that the inquiring about the nodes that are close to the best node are included every hierarchy will draw out the area within confinement with best node. It is precarious for nodes in the swarm to broadcast for better positions. This is actualized by having Cauchy's transformation on the whole node which is regarded as the best node in the swarm. A PSO was proposed by Montes and Stutzle (2008) as informed particle swarm optimization algorithm (FIPS) as a sensitive to discrepancies in the population network (Atyabi, et.al., 2010). The velocity upgrade rule utilized as a part of the FIPS considers every one of the nearby nodes to redesign its speed and not just the one with the highest speed as observed with different results. Tests have shown that an arbitrary demonstration of the node collectiveness when a completely associated network is connected. Thus, frequent dispute may portray a frequently obtained low outcome from the algorithm in accordance to the situation. Also, literature has proven to be appropriate in little discovery areas. An extended H-PSO (AH-PSO) which possess a rapid changing extensions level of the tree topology was showcased which improves execution of H-PSO. Similarly, another augmentation of the H-PSO is the ability to utilize distinctive qualities of the initial number of nodes with reference to the number in the hierarchy. Test have shown that the said algorithm can attained a precedent objective for each test function (aside from the Rastrigin formula) faster than any other variation of Particle Swarm Optimization. Chunming et al. (2005) built up another ideal approach to optimize results.

In the proposed newly structured new PSO technique suggested herein, every molecule changes to its position as per its own past most exceedingly bad solution and its group’s past most exceedingly bad to locate the optimal value esteem. The solution here is to maintain a strategic distance from a particle's past most noticeably bad solution and its group's past most exceedingly worst in light of comparative optimization of the consistent Particle Swarm Optimization. Conditions for speed and location stay unchanged, however the optimization is utilized in worst position instead of the best one. PSO has showcased the rapid quick discovery in a convoluted enhancement. Notwithstanding, Particle Swarm Optimization may effectively vary in local optimization. To obtain superior results, Wang et al. (2007) proposed a PSO which is based on opposition criteria, thus naming it OPSO, OPSO aims in fastening the merging of Particle Swarm Optimization and avert early haphazard of its multi-modular abilities. Konar (2000) show cases techniques that utilizes learning that is funded by opposition for every nodes and implements unanimous Cauchy commute on the best node. A few tests shows how particles in Particle Swarm Optimization will bridge across the best node and global best particle found by all the particles before it merges. On the differences that exists between the discovering closest to the global best nodes are included in every tree, it would expand the discovery span of the best node. Carefully, its implemented using the entire node to move to the better locations. This can be refined by obtaining the Cauchy's bridge across the global best node in each tree. Montes de Oca and Stutzle (2008) developed a fully informed particle swarm optimization algorithm (FIPS) that was is more prone to variations in population topology (Atyabi, Phon-Amnuaisuk and Ho, 2010). All the neighbors of a particle expressed by the FIPS and in most variants, it is presumed to frequent its speed rather than the best as postulated by the velocity update rule. When subjected to a full topology, this rule will cause the particle swarm to begin to behave randomly. In line with such an observation, it can be utilized to offer explain about the algorithm's observed poor performance. Additionally, it is more applicable to small search regions. As a result, a particle swarm optimization (PSO) algorithm with numerous variants is advisable. Divergences among variants are attributed to the presence of an optimized formula in only one sample.

In consequence, Montes de Oca and Stutzle proposed that refinements be made in 2009 to the new PSO and this was achieved through research on numerous Particle Swarm Optimization samples based on component types point of view (Dutta, 2010). Consequently, the composite algorithm Frankenstein's PSO was developed and this new PSO algorithm constituted of various algorithmic components through an investigation on optimization speed and reliability. The algorithm components had inherent significant benefits in which several changes to the primary (PSO) algorithm are made. In various occasions, a distinction between two types are caused by the presence of an algorithmic element being only available in one variant. Ultimately, a new PSO was established and the acquired ideas and were based on empirical studies of various PSO samples analyzed (Dutta, 2010). The latter section, therefore offers a new optimized PSO algorithm which is developed from numerous algorithmic elements. This is what has been termed the composite algorithm Frankenstein's PSO and has attracted favor as it offered significant benefits in experimental investigations. The PSO developed by Frankenstein is made of three main algorithmic components which are as follows:

- Dynamics in time population tree that diminishes QOS over time.
- FIPS contrivance for renewing node's speed.
- A decreasing inertial weight.

The elements derived are generated from decreasing the initial weight sample, FIPS and AHPSO. The initial sample is added as a catalyst that enhancing the differences between velocity, quality characterized by the changes in different connectivity measures. Another element that was utilized because the examination exhibited in FIPS includes sore optimization that can supers the performance with regards to various connectivity parameters. Lastly, the declining weight constituent integrates in an augmented way to support the discoverable behavior of the optimization process. PSO in recent research has shown how it succumb to unproductivity when nodes have early convergence to any specific area in the point of discovery. A new regrouping Particle Swarm Optimization (RegPSO) which averts the implement issue by dynamically initiating swarm regrouping, although early haphazard is observed was thus proposed (Saska, Macas, Preucil, and Lhotska, 2006). This tool disengages nodes facilitates continuous increase towards dynamic positioning. Nodes are rearranged within a specified area on every width equivalent to the extent of uncertainty suggested with the highest variation nodes from the collection of best nodes. When discovered, any early occurrence, the numbers in which nodes are to be rearranged the best is determined when the minimum of:

- An initial amount of the discovery location on dimension.
- The product of the rearranging element with the highest distance along dimension j of any node from the overall best.

2.3 Advantages of the Basic Particle Swarm Optimization Algorithm:

- Particle Swarm Optimization is based upon intelligence as it backbone, it is implemented in scientific research point of interest to try and simplify problems that have scenarios that has to be with group of swarms.
- Particle Swarm Optimization has no evident cases of mutation in it inner algorithms, optimization can be done using vectors like speed, mobility, communication among others.
- Particle Swarm Optimization implements a case where codes can be translated into solutions.

2.4 Disadvantages of the Basic Particle Swarm Optimization Algorithm:

- PSO has setbacks which include optimism which arises from less correct guidance of the nodes speed and direction.
- PSO won't be able to suffice scenarios where the nodes are dismantled from swarm or collective optimization.
- PSO can't be employed when the systems is disfigured, lack of coordination, energy resources and conservation.

2.5 Review Summary

The conventional techniques frequently fail to solve and provide solutions for optimization problems that have many local and confined ideals. In order to tackle these problems, there is a desire to invent more effective optimization techniques. Examples of the most popular population based optimization techniques include;

- Particle swarm optimization (PSO),
- Genetic algorithm (GA),
- Bacteria foraging optimization (BFO),
- Artificial bee colony (ABC),
- Ant colony optimization (ACO) etc.

For the sake of this research PSO algorithm is going to be chosen for the optimization algorithms in order to examine its nodes mobility and velocity, path discovery and its autonomy. This is because the particle's size has been chosen on the base of experimentation, to study tries to implement SSO optimization, the recommended optimization technique has traces of connected workplace to locate an ideal path.

CHAPTER 3

METHODOLOGY AND PROPOSED APPROACH

3.1 Proposed Approach

Numerous immobile complications were enunciated by 2D, also known as two dimensional square guides which was enunciated by a network based model in relation to mobile robot environment. This is overlaid on an enduring case of framework core interests. Cross section based model makes the most of the partition and representation of tangle less requesting. To check the ampleness of the SSO, the reenactment has been associated with the working environment which is shown in Figure 3.1. As appeared in the figure, the systems network deterrent where mobile robot can change position freely. The path used has no unit for measurement which can symbolize individuality within the cells. The blue circle in the environment shows beginning and end of path. The grid size and the directions of the beginning and target point with number of obstructions are showed up in Table 3.1.

Table 3.1: A table showing start and target point, grid size, number of obstacles

Start point [unit]	**Target point [unit]**	**Grid size [unit]**	**Number of obstacles**
X=0 , Y=0	X=23 , Y=30	23 × 30	50

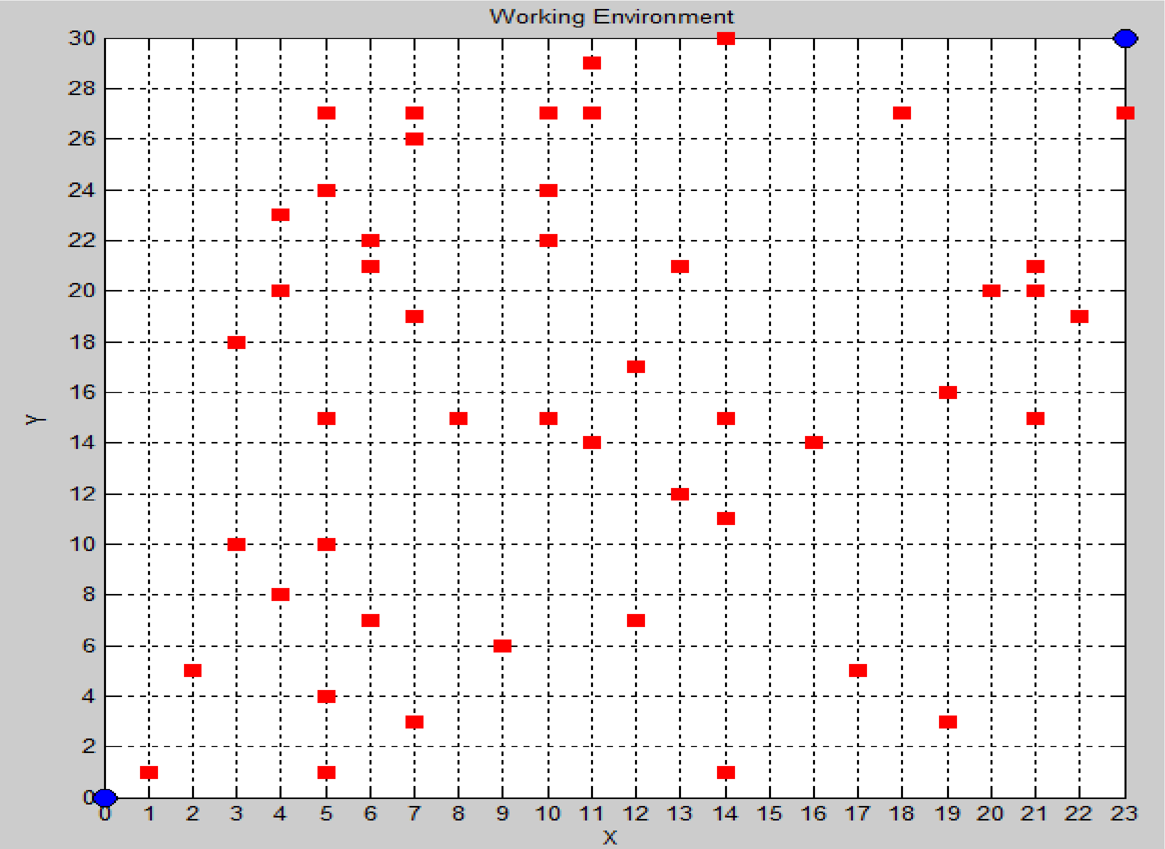

Figure 3.1: Working environment

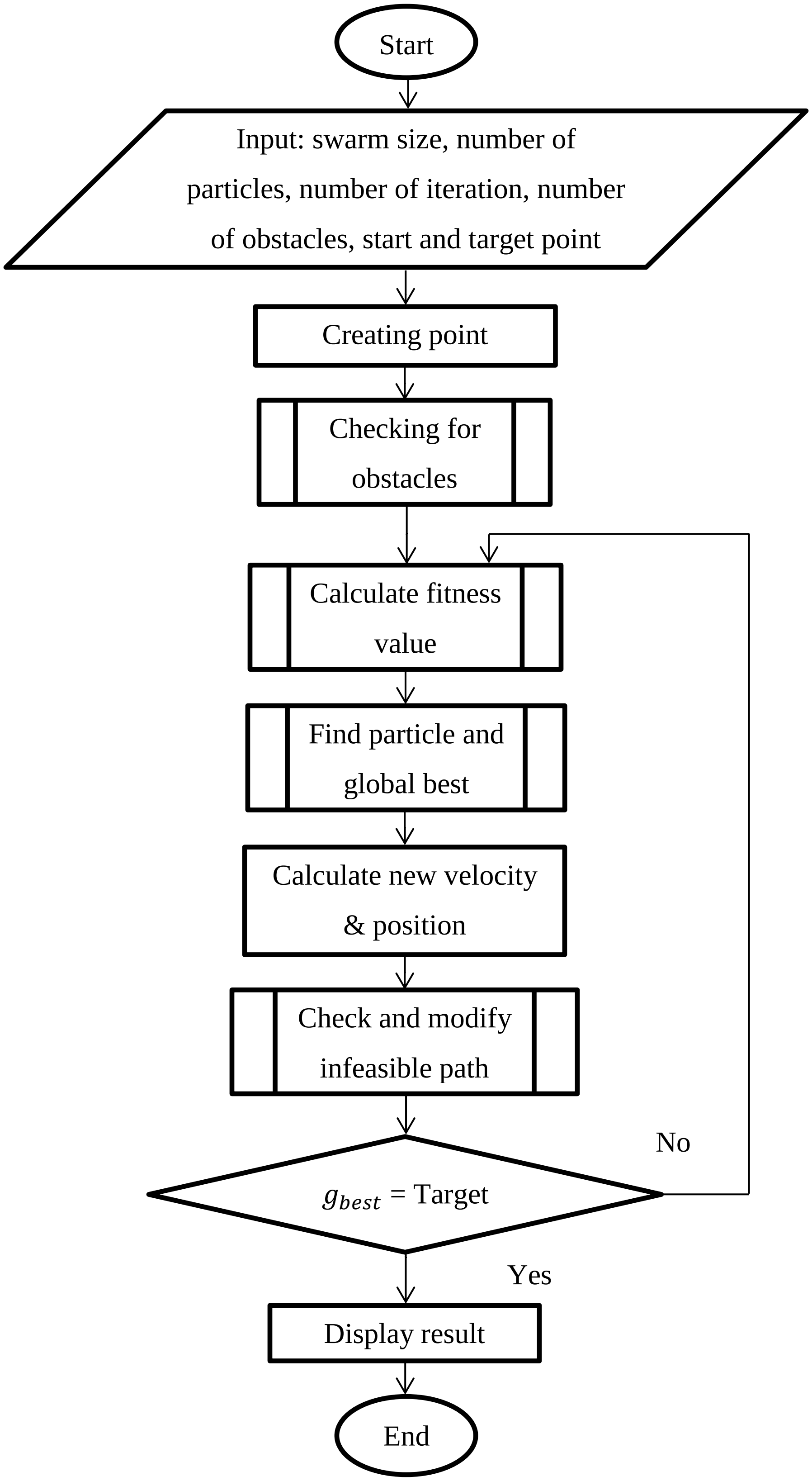
Start
Input: swarm size, number of particles, number of iteration, number of obstacles, start and target point
Creating point
Checking for obstacles
Calculate fitness value
Find particle and global best
Calculate new velocity & position
Check and modify infeasible path
g_{best} = Target
No
Yes
Display result
End

Figure 3.2: Main flow chart

Figure 3.2 is a flowchart which shows the detailed steps and process of how an SSO algorithm operates. A value is assigned for each variable swarm size which contain X and Y axis respectively (swarm X and swarm Y), while the values for both swarm size were initialize randomly. Furthermore, the number of particles signifies how many points the swarm have to pass from the initiation point to the target point. While number of alterations signifies how many iterations were assigned for the particle swarm to choose the best value among iterations.

The number of obstacles indicates how many obstacles are there within the working environment, while navigating from the initial point to the target point. Within the working environment, for the robot to navigate, there is an initial and target point for the navigation. Subsequently, after inputting the values and defined the variables, a point will be created for the robot to navigate and find its path within the working environment. A conditional statement will be use to check, weather there is an obstacle or not at each single point, if there is an obstacle, a new point has to be provided through the best neighborhood of X_direction (axis) and Y_direction (axis). The subsequent step is to calculate the distance between each point in which the robot passes through from the initial to the target point while navigating.

The distance between each point will be the sum of every iteration, while the distance from all iteration will be compared in order to find the list number for determining the best position (pbest) for the particle, and that best position with smallest number will be regarded as global best position (gbest) for swarm. However, there is a need to find a new position. In finding new positon, there is a need to find new velocity for X_direction and Y_direction which will determine the new position for each particle. The subsequent step is to check for infeasible path. This will determine whether the path is feasible for robot or not, this will help to find out either to increment or decrement X by one point, or increment or decrement Y by one point, vice versa. Furthermore, point addition will be made for both X and Y swarm. There will be continues iteration until global best position is equivalent to the target point. After that, the best vale will be displayed and that will end the process.

3.2 Explanation of Each Entity From Flowchart

3.2.1 Simplified swarm optimization

The initial locus of every particle is also the beginning stage of the path. The principal locus and haste of every particle is produced haphazardly, however constrained to the limits of the search boundaries. The coordinates of start and goal points are *xs*, *ys* and *xg*, *yg* respectively.

3.2.2 Checking for obstacles

While particles explores through the pursuit space from existing position to new position, a conditional statement is utilized to check if the following point is obstacles or not, the condition check, if the X_direction of point equal to XOPS and Y_direction of point equal to YOPS if the condition is true then change the position of particle and if condition false add point to swarm X and swarm Y, are shown in Figure 3.3.

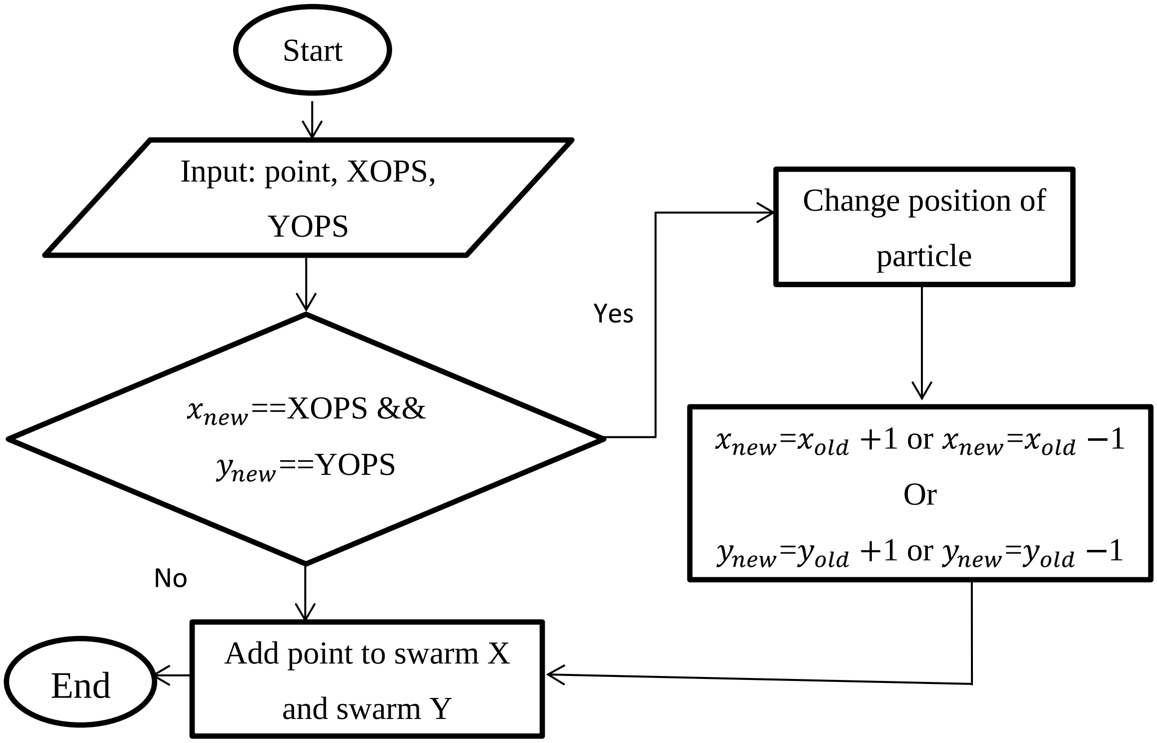
Start
Input: point, XOPS, YOPS
x_{new}==XOPS && y_{new}==YOPS
Yes
Change position of particle
x_{new}=x_{old} +1 or x_{new}=x_{old} −1
Or
y_{new}=y_{old} +1 or y_{new}=y_{old} −1
No
Add point to swarm X and swarm Y
End

Figure 3.3: Flowchart for checking obstacles

3.2.3 Evaluation fitness function

The path of the particles are viewed and surveyed inside each movement. Assessment capacity is used to give a measure of how particles have implemented in the issue based on space. In the SSO, dual parameters (the time and an amount space between two points) determined by the particle are utilized to compute the particle fitness ($p_{-fitness}$). Since the fitness should augment as the partition and time reduce the fitness capacity is assessed as:

$$p_{-fitness} = 1/(w_d * d(p_i, p_{i+1}) + w_t * t(p_i, p_{i+1})) \qquad (1.3)$$

From the equation; w_d and w_t are factor for distance in relation to weight and time of travel, $d(p_i, p_{i+1})$ signifies Euclidean distance concerning point p_i to then point p_{i+1} for same particle and $t(p_i, p_{i+1})$ denotes time taken by the particle to cover from p_i to p_{i+1} :

$$d(p_i, p_{i+1}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \qquad (2.3)$$

$$t(p_i, p_{i+1}) = d(p_i, p_{i+1})/vi \qquad (3.3)$$

Where x_i and y_i signifies particle's parallel and perpendicular locus, x_{i+1} and y_{i+1} are particle's next parallel and perpendicular locus, and vi means the speed of the particle when venturing out from p_i - p_{i+1}. The computation of fitness amount is assessed by flowchart appeared in Figure (3.4).

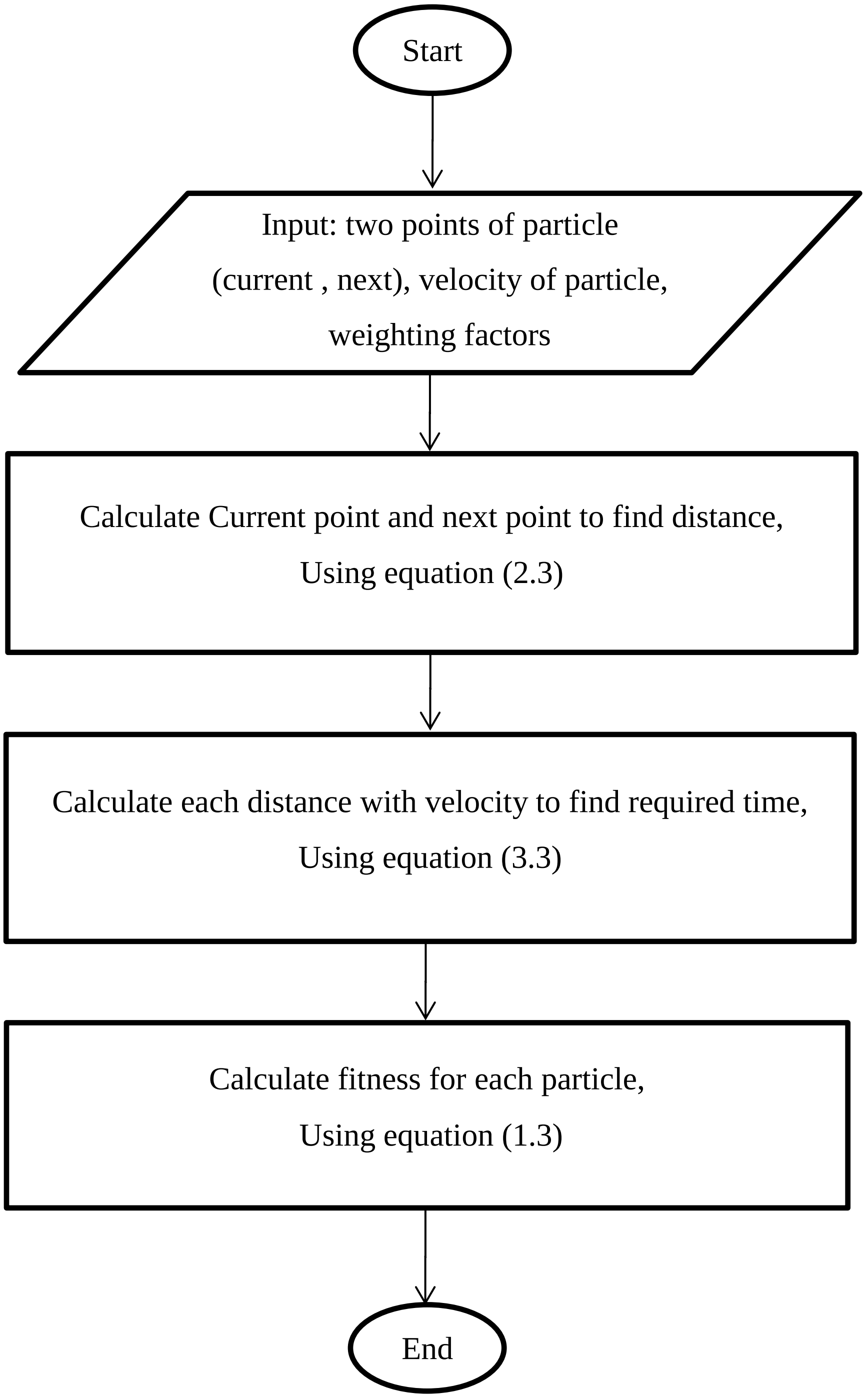
Start
Input: two points of particle
(current , next), velocity of particle,
weighting factors
Calculate Current point and next point to find distance,
Using equation (2.3)
Calculate each distance with velocity to find required time,
Using equation (3.3)
Calculate fitness for each particle,
Using equation (1.3)
End

Figure 3.4: Flowchart for fitness calculation

3.2.4 Finding the individual (particle) and global best location

The particle's sum of distance appreciation is considered at amongst present and new area at every iteration to decide the best position (p_{best}) for every particle and the particle with the best sum of distance appreciation is allotted as the universal best (g_{best}), as outlined by the flowchart appeared in Figure (3.5).

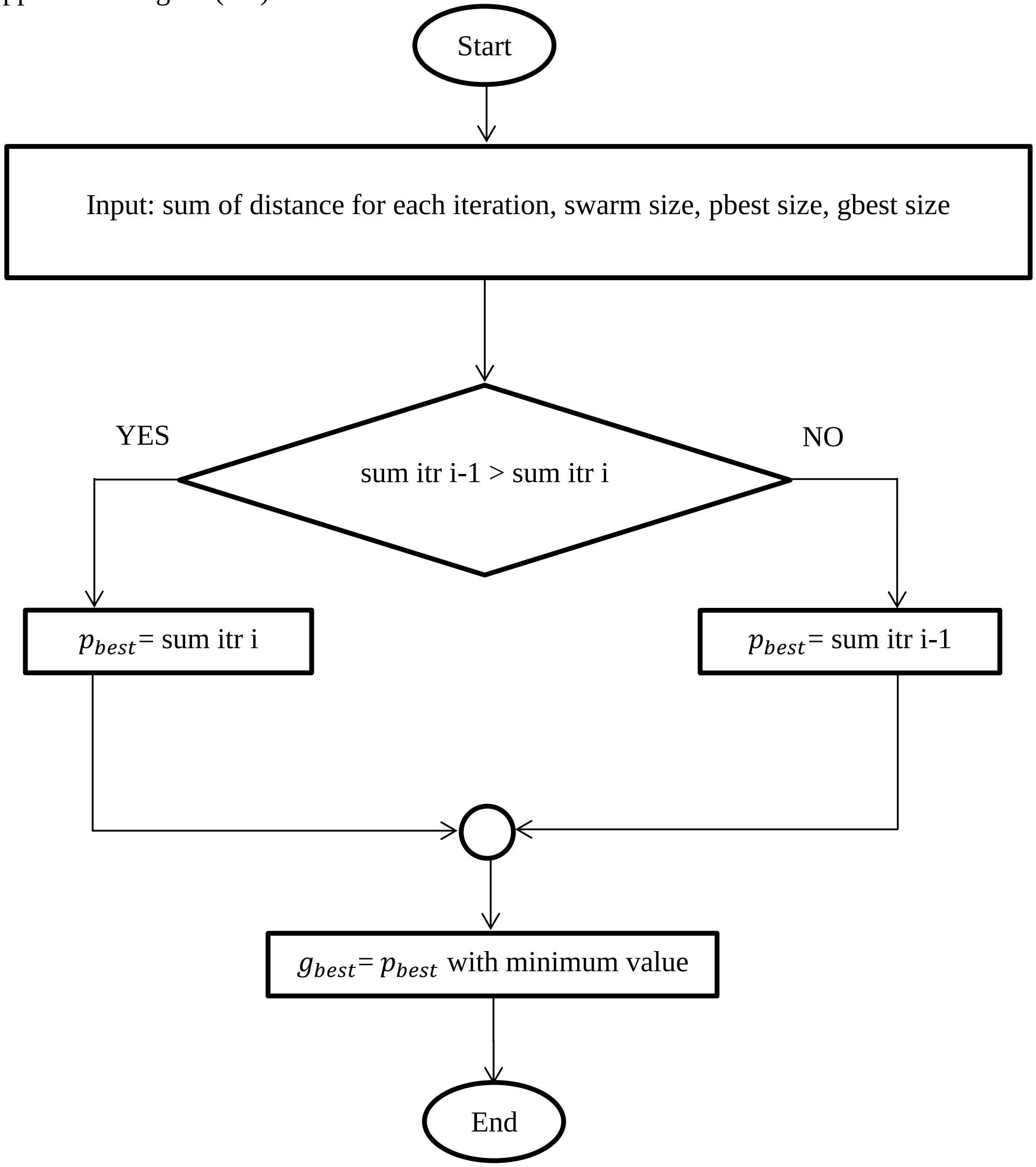
Start
Input: sum of distance for each iteration, swarm size, pbest size, gbest size
YES
NO
sum itr i-1 > sum itr i
p_{best}= sum itr i
p_{best}= sum itr i-1
g_{best}= p_{best} with minimum value
End

Figure 3.5: Finding the particle and global best location

3.2.5 Calculate new velocity and position

The algorithm of SSO was introduced with various particles and afterward search for ideal. The locus of a particle is affected by the finest locus went to without anyone else's input which is alluded to as particle best “p_{best}”, and the best locus in the entire swarm which is alluded to as the global best”g_{best}”. A particle overhauls its locus and velocity utilizing the supplementary Equations (4.3 & 5.3).

$$vx_2 = \mathrm{w} * \mathrm{vx}_1 + \mathrm{c}_1 * \mathrm{r}_1(\mathrm{p}_{best} - \mathrm{x}_1) + \mathrm{c}_2 * \mathrm{r}_2 * (g_{best} - x_1) \tag{4.3}$$

$$\mathrm{vy}_2 = \mathrm{w} * \mathrm{vy}_1 + \mathrm{c}_1 * \mathrm{r}_1(\mathrm{p}_{best} - \mathrm{y}_1) + \mathrm{c}_2 * \mathrm{r}_2 * (g_{best} - y_1) \tag{5.3}$$

From the equation (4.3 and 5.3), the paricle segment $(\mathrm{p}_{best} - \mathrm{x}_1)$ connotes the particles possess understanding as to where the best outcome is and the social segments, $(\mathrm{p}_{best} - \mathrm{y}_1)$ signifies the conviction of the whole swarm as to where the best arrangement is vx_2 and vy_2 are the effective particle's velocity. vx_1 , vy_1 and x_1 , y_1 are current particle’s velocity and position along x and y-direction respectively. Overhauling speeds is the way that empowers the particle to look around its particular best locus and global best locus.

Individual factor, c_1 and swarm certainty factor, c_2 makes particles have the function of personal instantaneous and figure out how to the best of swarm, and draw near to the best position of its own and also inside the swarm. The inertia weight (w) this term fills in as a memory of past velocity and it utilized to control the effect of the past velocity on the present velocity. The estimation of inertia weight (w) is in the scope of [0,1]. r_1 and r_2 arbitrary numbers inside the interval [0, 1]. Updated equation (4.3 and 5.3) for SSO to Equation (6.3 and 7.3).

$$vx_2 = \mathrm{w} * \mathrm{vx}_1 + \mathrm{c}_2 * \mathrm{r}_2(g_{best} - \mathrm{x}_1) \tag{6.3}$$

$$\mathrm{vy}_2 = \mathrm{w} * \mathrm{vy}_1 + \mathrm{c}_2 * \mathrm{r}_2(g_{best} - \mathrm{y}_1) \tag{7.3}$$

3.2.6 Check and modify infeasible path

After finding a new point a conditional statement is used to check all the points if the point is obstacles or not if condition true replaces point and path become feasible if condition is false then the path is feasible.

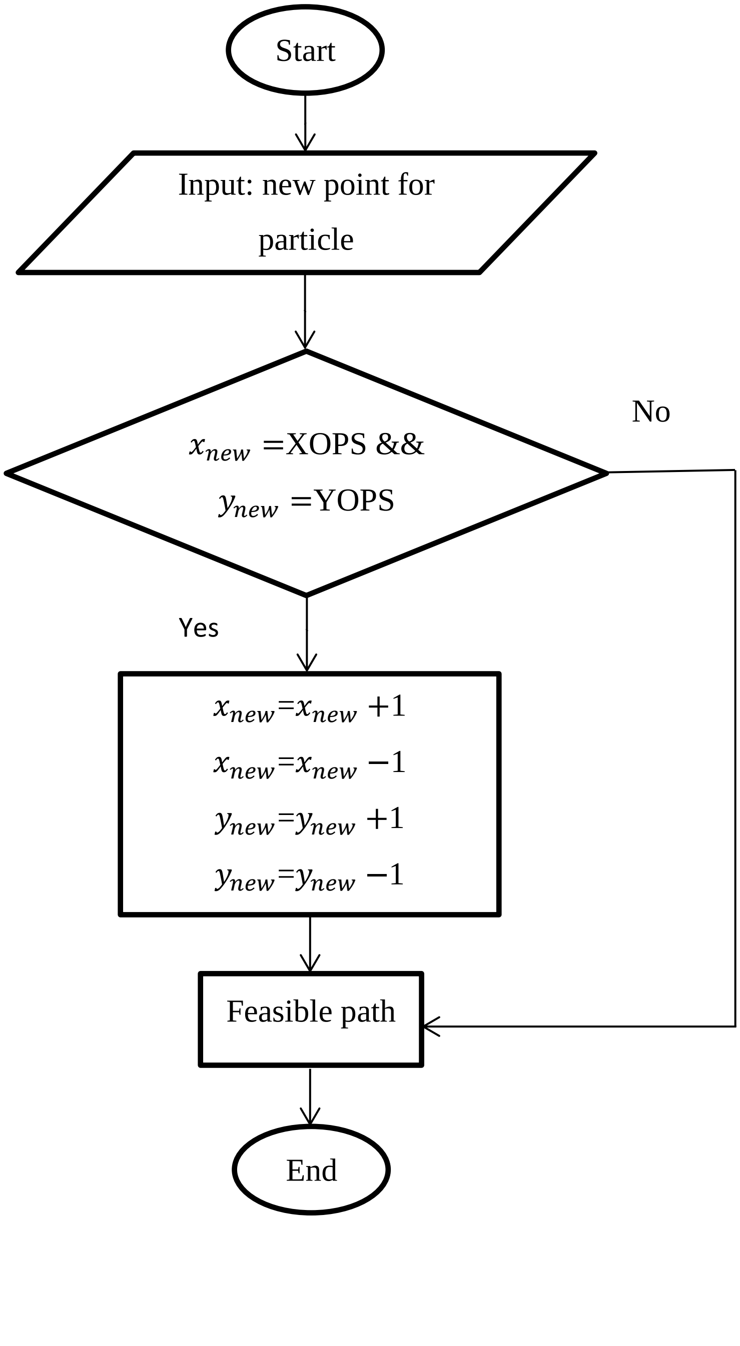

Figure 3.6: Flowchart for modify feasible path

3.3 Numerical Example of One Iteration

Swarm X

1	1	18	22	5	6
2					
20					

Swarm Y

1	6	11	29	14

Swarm size

(20 , 5)

	XOPS	YOPS
1	1	1
2	5	1
50	14	30

Number of

obstacles (50 , 1)

Start point = (0 , 0)

Target point = (23 , 30)

P1 = (1 , 1)	point of swarm x and swarm y
P2 = (18 , 6)	point of swarm x and swarm y
P3 = (22 ,11)	point of swarm x and swarm y
P4 = (5 ,29)	point of swarm x and swarm y
P5 = (6 ,14)	point of swarm x and swarm y

$p_{start} \rightarrow p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4 \rightarrow p_5 \rightarrow p_{target}$

P1 = (1 ,1)

x_1 = 1 , y_1 = 1 this point is obstacles

x_{1new} = x_1+1

x_{1new} = 1+1 = 2 then the point changes to (2 , 1)

Or

y_{1new} = y+1

y_{1new} = 1+1 = 2 then the point changes to (1 , 2)

Now P1 = (2 , 1)

$$d_{1(\mathrm{Pstart}\,,\mathrm{P1})} = \sqrt{(x_{p1} - x_{p\mathrm{start}})^2 + (y_{p1} - y_{p\mathrm{start}})^2}$$

$$d_{1(\mathrm{Pstart}\,,\mathrm{P1})} = \sqrt{(2-0)^2 + (1-0)^2} \;=\; 2.2$$

$$d_{2(\mathrm{P1}\,,\mathrm{P2})} = \sqrt{(x_{\mathrm{p2}} - x_{p1})^2 + (y_{p2} - y_{p1})^2}$$

$$d_{2(\mathrm{P1}\,,\mathrm{P2})} = \sqrt{(18-2)^2 + (6-1)^2} = 16.7$$

Using equation of distance to find distance between two points

$$d_{3(\mathrm{P2}\,,\mathrm{P3})} = \sqrt{(x_{\mathrm{p3}} - x_{p2})^2 + (y_{p3} - y_{p2})^2}$$

$$d_{3(\mathrm{P2}\,,\mathrm{P3})} = \sqrt{(22-18)^2 + (11-6)^2} = 6.4$$

$$d_{4(\mathrm{P4}\,,\mathrm{P3})} = \sqrt{(x_{\mathrm{p4}} - x_{p3})^2 + (y_{p4} - y_{p3})^2}$$

$$d_{4(\mathrm{P4}\,,\mathrm{P3})} = \sqrt{(5-22)^2 + (29-11)^2} = 24.7$$

$$d_{5(\mathrm{P5}\,,\mathrm{P4})} = \sqrt{(x_{\mathrm{p5}} - x_{p4})^2 + (y_{p5} - y_{p4})^2}$$

$$d_{5(\mathrm{P5}\,,\mathrm{P4})} = \sqrt{(6-5)^2 + (14-29)^2} = 15.03$$

$$d_{6(\mathrm{Ptarget}\,,\mathrm{P5})} = \sqrt{(x_{\mathrm{ptarget}} - x_{p5})^2 + (y_{\mathrm{ptarget}} - y_{p5})^2}$$

$$d_{6(\mathrm{Ptarget}\,,\mathrm{P5})} = \sqrt{(23-6)^2 + (30-14)^2} = 23.3$$

$$path_{dist} = \sum di = 88.33$$

Sum of distance between points

	Path_dist Itr i-1	Path_dist Itr i	Pbest
1	30	88.33	30
2	43.2	28.1	28.1
20	93	12	12

Gbest=12

Iteration, Particle best position, Global best position with size (20 , 1)

For Particle Swarm Optimization (PSO)

New velocity for X

$$v_{x1}new = v_{x1} + 2 * r_1(p_{best} - x_1) + 2 * r_2(g_{best} - x_1)$$

$$v_{x1}new = 1 + 2 * 0.2(30 - 2) + 2 * 0.3(12 - 2) = 18.2$$

New position for X

$$x_1new = x_1old + v_{x1}new$$

$$x_1new = 2 + 18.2 = 20.2$$

New velocity for Y

$$v_{y1}new = v_{y1} + 2 * r_1(p_{best} - y_1) + 2 * r_2(g_{best} - y_1)$$

$v_{x1}new = 1 + 2 * 0.2(30 - 1) + 2 * 0.3(12 - 1) = 19.2$

New position for Y

$y_1new = y_1old + v_{y1}new$

$y_1new = 1 + 19.2 = 20.2$

New point (x_1new , y_1new) = (20.2 , 20.2)

For Simplified Swarm Optimization (SSO)

New velocity for X

$v_{x1}new = v_{x1} + 2 * r_2(g_{best} - x_1)$

$v_{x1}new = 1 + 2 * 0.3(12 - 2) = 7$

New position for X

$x_1new = x_1old + v_{x1}new$

$x_1new = 2 + 7 = 9$

New velocity for Y

$v_{y1}new = v_{y1} + 2 * r_2(g_{best} - y_1)$

$v_{x1}new = 1 + 2 * 0.3(12 - 1) = 7.6$

New position for Y

$y_1new = y_1old + v_{y1}new$

$y_1new = 1 + 7.6 = 8.6$

New point (x_1new , y_1new) = (9 , 8.6)

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Algorithm Parameters

Table 4.1 below is displaying the defined parameters which are signifying the algorithms for the proposal of the approach.

Table 4.1: Algorithm variables

	Variables	**Size**
1	No. of iteration	100
2	Swarm X	5 x 20
3	Swarm Y	5 x 20
4	XOPS	1 x 50
5	YOPS	1 x 50
6	Path_dist	1 x 20
7	P_best	1 x 20
8	G_best	1 x 20

4.2 The Implementation of SSO in Path Planning

To look at the effect of different swarm dimension on SSO and their execution, several numeral of particle has been relating (such as 5, 6, 7, and 8). The particle's dimension has been picked on the base of testing, to explore the execution of SSO. The computation proposed has been connected with work environment which was showed in Figure 3.1 in order to discover the path that is perfect. The best gained re-enactment comes to different number of particles (swarm size) for the work environment which showed up in Table 4.2.

$$distance\ (s\ ,e) = distnce\ between\ start\ and\ end\ point$$

$(1,1) \longrightarrow (23,30)$

$distance\ (s\ ,e) = \sqrt{(30-1)^2 + (23-1)^2}$

$distance\ (s\ ,e) = 36.4$

$error = distance\ algorithm - distance\ (s\ ,e)$

Table 4.2: Simulation results for the working environment using PSO and SSO algorithm

NO. of particles	Type of algorithm	Distance	Error
5	PSO	67.1897	30.7897
5	SSO	36.4371	0.0371
6	PSO	88.0187	51.6187
6	SSO	36.4608	0.0608
7	PSO	80.2049	43.8049
7	SSO	36.6457	0.2457
8	PSO	109.0702	72.6702
8	SSO	37.2733	0.8733

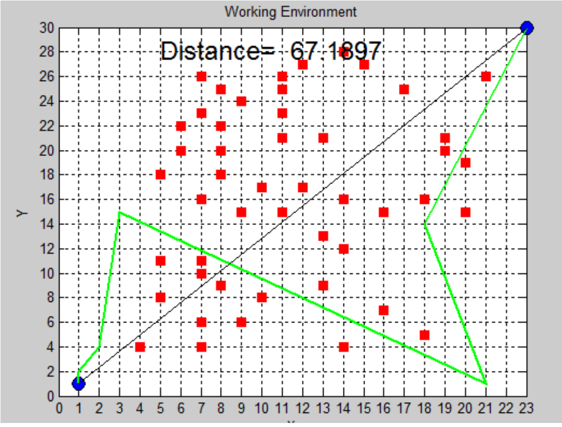

Figure 4.1: Using PSO, (5 point)

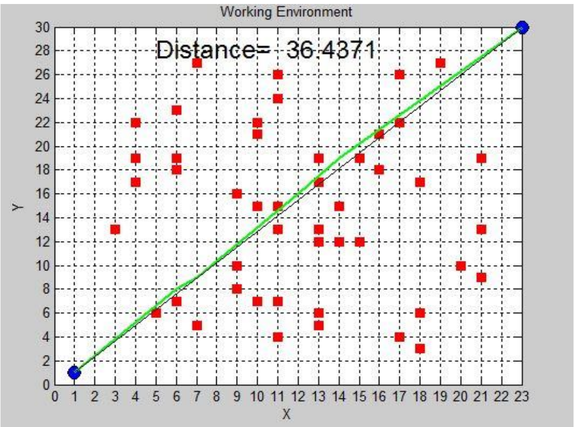

Figure 4.2: Using SSO, (5 point)

The figure 4.1 – 4.2 above are displaying the obtained solution after PSO and SSO algorithms were used for working environment with X axis (0,23) direction and Y axis (0,30) direction. The working environment was used as the search space, which was occupied by number of random obstacles. These obstacles were represented by red square from the graph. Similarly, blues circles represent the starting and target point of the robot. Black line represents the robots path distance from starting to target point while navigating in the working environment. Green line represents the path used by robots to navigate from starting to target point. Figure 4.1 is showing the PSO algorithm applied using 5 points and a distance of (67.1897). This algorithm is signifying that, the robot took 5 points to reach its target point from the starting point with the distance it took to navigate. In figure 4.2 an SSO algorithm was also applied with its 5 points and a distance of (36.4371). The SSO algorithm was determined to find the minimum path possible for the robot to navigate without obstacles while navigating, this provides less distance when compared with PSO algorithms. SSO algorithms was created to make an optimization algorithm which will be more capable for global path planning, SSO also will make infeasible paths problems feasible for robots to navigate efficiently and effectively.

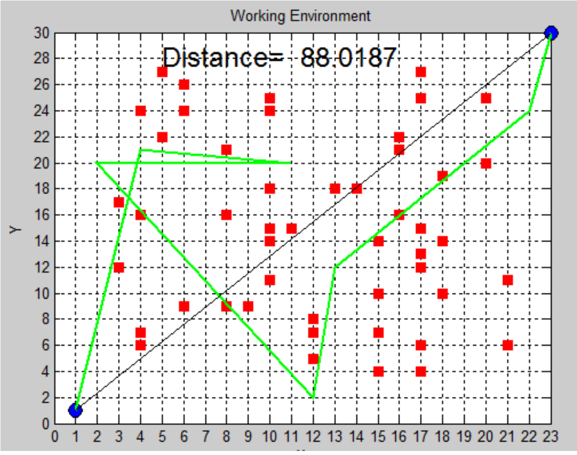

Figure 4.3: Using PSO, (6 point)

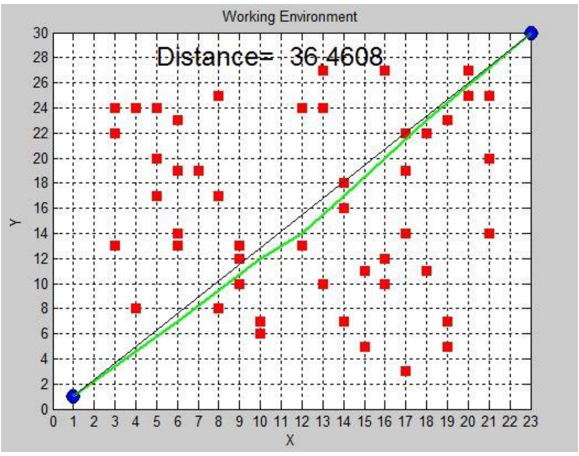

Figure 4.4: Using SSO, (6 point)

The figure 4.3 – 4.4 above are displaying the obtained solution after PSO and SSO algorithms were used for working environment with X axis (0,23) direction and Y axis (0,30) direction. The working environment was used as the search space, which was occupied by number of random obstacles. These obstacles were represented by red square from the graph. Similarly, blues circles represent the starting and target point of the robot. Black line represents the robots path distance from starting to target point while navigating in the working environment. Green line represents the path used by robots to navigate from starting to target point. Figure 4.3 is showing the PSO algorithm applied using 6 points and a distance of (88.0187). This algorithm is signifying that, the robot took 6 points to reach its target point from the starting point with the distance it took to navigate. In figure 4.4 an SSO algorithm was also applied with its 6 points and a distance of (36.4608). The SSO algorithm was determined to find the minimum path possible for the robot to navigate without obstacles while navigating, this provides less distance when compared with PSO algorithms. SSO algorithms was created to make an optimization algorithm which will be more capable for global path planning, SSO also will make infeasible paths problems feasible for robots to navigate efficiently and effectively.

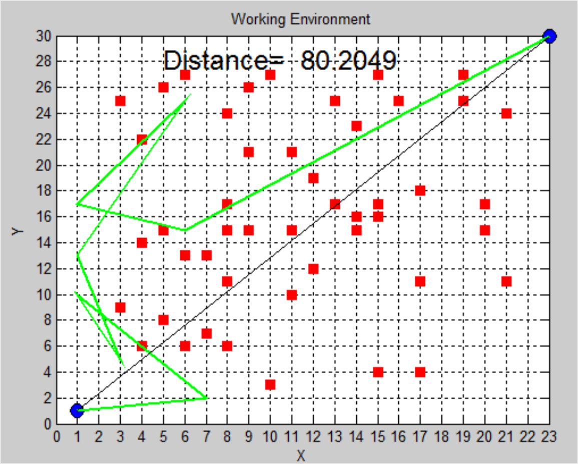

Figure 4.5: Using PSO, (7 point)

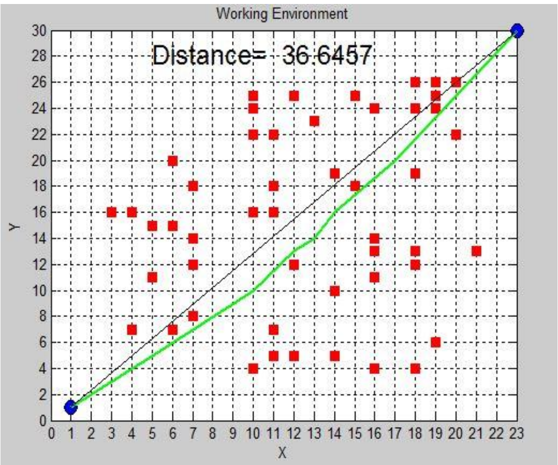

Figure 4.6: Using SSO, (7 point)

The figure 4.5 – 4.6 above are displaying the obtained solution after PSO and SSO algorithms were used for working environment with X axis (0,23) direction and Y axis (0,30) direction. The working environment was used as the search space, which was occupied by number of random obstacles. These obstacles were represented by red square from the graph. Similarly, blues circles represent the starting and target point of the robot. Black line represents the robots path distance from starting to target point while navigating in the working environment. Green line represents the path used by robots to navigate from starting to target point. Figure 4.5 is showing the PSO algorithm applied using 7 points and a distance of (80.2049). This algorithm is signifying that, the robot took 7 points to reach its target point from the starting point with the distance it took to navigate. In figure 4.6 an SSO algorithm was also applied with its 7 points and a distance of (36.6457). The SSO algorithm was determined to find the minimum path possible for the robot to navigate without obstacles while navigating, this provides less distance when compared with PSO algorithms. SSO algorithms was created to make an optimization algorithm which will be more capable for global path planning, SSO also will make infeasible paths problems feasible for robots to navigate efficiently and effectively.

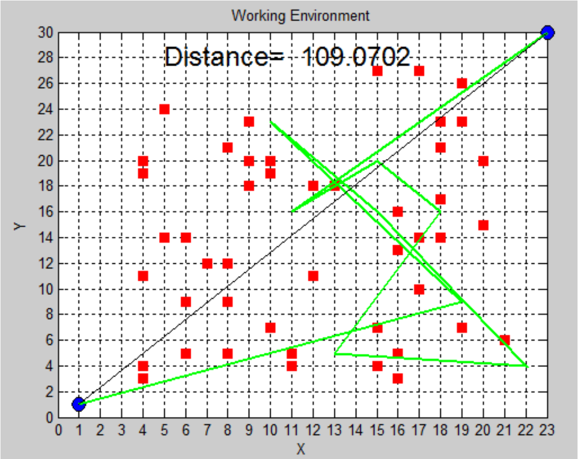

Figure 4.7: Using PSO, (8 point)

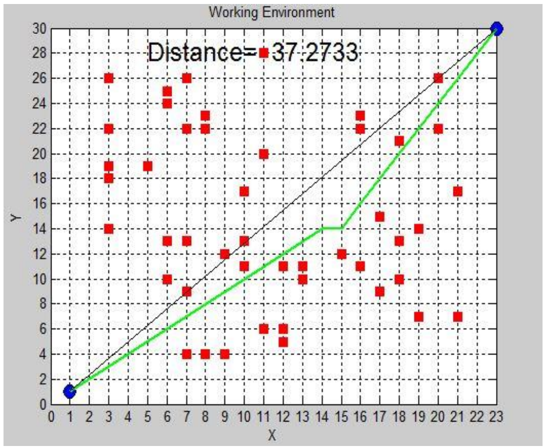

Figure 4.8: Using SSO, (8 point)

The figure 4.7 – 4.8 above are displaying the obtained solution after PSO and SSO algorithms were used for working environment with X axis (0,23) direction and Y axis (0,30) direction. The working environment was used as the search space, which was occupied by number of random obstacles. These obstacles were represented by red square from the graph. Similarly, blues circles represent the starting and target point of the robot. Black line represents the robots path distance from starting to target point while navigating in the working environment. Green line represents the path used by robots to navigate from starting to target point. Figure 4.7 is showing the PSO algorithm applied using 8 points and a distance of (109.0702). This algorithm is signifying that, the robot took 8 points to reach its target point from the starting point with the distance it took to navigate. In figure 4.8 an SSO algorithm was also applied with its 8 points and a distance of (37.2733). The SSO algorithm was determined to find the minimum path possible for the robot to navigate without obstacles while navigating, this provides less distance when compared with PSO algorithms. SSO algorithms was created to make an optimization algorithm which will be more capable for global path planning, SSO also will make infeasible paths problems feasible for robots to navigate efficiently and effectively.

Table 4.3: The simulation results for the working environment using SSO algorithm

No. of particles	11 iteration	55 iteration	99 iteration
4	36.5689	36.4204	36.4112
9	66.4912	56.5358	40.0703
40	171.4331	159.4229	141.9987
90	442.6258	423.9265	387.5449

Number of particles (4) with the iteration (11) the distance = 36.5689
Number of particles (4) with the iteration (55) the distance = 36.4204
Number of particles (4) with the iteration (99) the distance = 36.4112
Number of particles (9) with the iteration (11) the distance = 66.4912
Number of particles (9) with the iteration (55) the distance = 56.5358
Number of particles (9) with the iteration (99) the distance = 40.0703
Number of particles (40) with the iteration (11) the distance = 171.4331
Number of particles (40) with the iteration (55) the distance = 159.4229
Number of particles (40) with the iteration (99) the distance = 141.9987
Number of particles (90) with the iteration (11) the distance = 442.6258
Number of particles (90) with the iteration (55) the distance = 423.9265
Number of particles (90) with the iteration (99) the distance = 387.5449

The result shows that the distance will increase by increasing particle and the distance will decrease by increasing iteration.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In this study, path planning for mobile robot was investigated using Simplified Swarm Optimization (SSO). A SSO algorithm was utilized to discover ideal way for the mobile robot in workplace with irregular complications. Based on the result obtained from simulations, the fallowing conclusions were drawn:

- The algorithm for the (SSO) that was proposed was proficient of successfully managing a robot movement from initial locus to the final locus in complex environment at the same time finding a perfect and shortest path without affecting any obstacles.

- In the SSO algorithm errors element was verified, to guarantee the PSO unites together. These errors were utilized in altering the particle's trajectory to move toward the objective target.

- An adjusted strategy was implemented in the SSO in order to tackle the convenient path problems. A conditional statement was utilized when the particle path falls within a problematic environment; then it will transfer it to a locus neighborhood outside of the deterrent.

- The SSO computation does not take part in the enclosed environment in which there was no path between the initial and the goal point.

- SSO can legitimately be seen like better than average algorithm in light of its joining speed and efficient in global search.

5.2 Recommendations

Due to timeframe for this study, it is recommended that the study may consider the following for future development:

- Applying the proposed SSO algorithm on Real-Time environment at which the obstacles change their position with time.

- Applying the proposed SSO algorithm in dynamic environments.

- Comparing the suggested SSO algorithm with other algorithms, which include, Artificial Bee Colony (ABC), Ant Colony Optimization (ACO).

REFERENCES

Ahmadzadeh, S., & Ghanavati, M. (2012). Navigation of mobile robot using the PSO particle swarm optimization. *Journal of Academic and Applied Studies (JAAS)*, 2(1), 32-38.

Atyabi, A., Phon-Amnuaisuk, S., & Ho, C. K. (2010). Navigating a robotic swarm in an uncharted 2D landscape. *Applied soft computing*, 10(1), 149-169.

Clerc, M. (2010). Particle swarm optimization (Vol. 93). John Wiley & Sons.

Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. G. (2008). Particle swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on evolutionary computation*, 12(2), 171-195.

Dorigo, M., de Oca, M. A. M., & Engelbrecht, A. (2008). *Particle swarm optimization Scholarpedia*, 3(11), 1486.

Dos Santos, R. P. B., Martins, C. H., & Santos, F. L. (2011). Simplified particle swarm optimization algorithm-doi: 10.4025/actascitechnol. v34i1. 9679. *Acta Scientiarum. Technology*, 34(1), 21-25.

Du, K. L., & Swamy, M. N. S. (2016). Particle swarm optimization. In *Search and Optimization by Metaheuristics* (pp. 153-173). Springer International Publishing.

Dutta, S. (2010). Obstacle avoidance of mobile robot using PSO-based neuro fuzzy technique. *International Journal of Computer Science and Engineering*, 2(2), 301-304.

Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. *In Micro Machine and Human Science*, 1995. MHS'95., Proceedings of the Sixth International Symposium on (pp. 39-43). IEEE.

Han, K. M. (2007). *Collision free path planning algorithms for robot navigation problem*, Master Thesis, University of Missouri-Columbia.

Higashi, N., & Iba, H. (2003). Particle swarm optimization with Gaussian mutation. *In Swarm Intelligence Symposium*, 2003. SIS'03. Proceedings of the 2003 IEEE (pp. 72-79). IEEE.

Ismail, A. T., Sheta, A., & Al-Weshah, M. (2008). A mobile robot path planning using genetic algorithm in static environment. *Journal of Computer Science*, 4(4), 341-344.

Jatmiko, W., Mursanto, P., Kusumoputro, B., Sekiyama, K., & Fukuda, T. (2008). Modified PSO algorithm based on flow of wind for odor source localization problems in dynamic environments. *WSEAS Transaction on System*, 7(3), 106-113.

Jiang, Y., Hu, T., Huang, C., & Wu, X. (2007). An improved particle swarm optimization algorithm. *Applied Mathematics and Computation*, 193(1), 231-239.

Kennedy, J. (2011). Particle swarm optimization. In Encyclopedia of machine learning (pp. 760-766). Springer US.

Konar, A. (1999). Artificial intelligence and soft computing: behavioral and cognitive modeling of the human brain. CRC press.

Li, P., Huang, X., & Wang, M. (2010). A New Hybrid Method for Mobile Robot Dynamic Local Path Planning in Unknown Environment. *JCP*, *5*(5), 773-781.

Mohanty, P. K., & Parhi, D. R. (2013). Controlling the motion of an autonomous mobile robot using various techniques: a review. *Journal of Advance Mechanical Engineering*, 1(1), 24-39.

Moradi, M. H., & Abedini, M. (2012). A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems. *International Journal of Electrical Power & Energy Systems*, 34(1), 66-74.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1), 33-57.

Raja, P., & Pugazhenthi, S. (2009, October). Path planning for mobile robots in dynamic environments using particle swarm optimization. *In Advances in Recent Technologies in Communication and Computing*, 2009. ARTCom'09. International Conference on (pp. 401-405). IEEE.

Raja, P., & Pugazhenthi, S. (2012). Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences*, 7(9), 1314-1320.

Raquel, C. R., & Naval Jr, P. C. (2005, June). An effective use of crowding distance in multiobjective particle swarm optimization. *In Proceedings of the 7th annual conference on Genetic and evolutionary computation*, (pp. 257-264). ACM.

Sariff, N., & Buniyamin, N. (2006, June). An overview of autonomous mobile robot path planning algorithms. *In Research and Development*, 2006. SCOReD 2006. 4th Student Conference on (pp. 183-188). IEEE.

Saska, M., Macas, M., Preucil, L., & Lhotska, L. (2006, September). Robot path planning using particle swarm optimization of Ferguson splines. *In Emerging Technologies and Factory Automation*, 2006. ETFA'06. IEEE Conference on (pp. 833-839). IEEE.

Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *In evolutionary computation*, 2001. Proceedings of the 2001 Congress on (Vol. 1, pp. 81-86). IEEE.

Shiltagh, N. A., & Jalal, L. D. (2013). Path planning of intelligent mobile robot using modified genetic algorithm. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(2), 31-36.

Van Den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information sciences*, 176(8), 937-971.

Velagic, J., Lacevic, B., & Osmic, N., (2006). Efficient path planning algorithm for mobile robot navigation with a local minima problem solving, *In Proceedings of IEEE International Conference on Industrial Technology*, IEEE, Mumbai, pp. 2325–2330.

Wang, Y. Q., & Yu, X. P. (2012, January). Research for the robot path planning control strategy based on the immune particle swarm optimization algorithm. *In Intelligent System Design and Engineering Application (ISDEA)*, 2012 Second International Conference on (pp. 724-727). IEEE.

Yun, S. C., Ganapathy, V., & Chong, L. O. (2010, December). Improved genetic algorithms based optimum path planning for mobile robot. *In Control Automation Robotics & Vision (ICARCV)*, 2010 11th International Conference on (pp. 1565-1570). IEEE.

Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H. (2009). *Adaptive particle swarm optimization. IEEE Transactions on Systems*, Man, and Cybernetics, Part B (Cybernetics), 39(6), 1362-1381.

Zhao, Y., & Zu, W. (2009, April). Real-time obstacle avoidance method for mobile robots based on a modified particle swarm optimization. *In Computational Sciences and Optimization*, 2009. CSO 2009. International Joint Conference on (Vol. 2, pp. 269-272). IEEE.

Zhihua, C., & Jianchao, Z. (2011). Particle swarm optimization algorithm.

APPENDIX : Source Code

```
close all;
clc;
clear;

%===========================
% Initialization
%===========================
N=50;                                %Number of iteration
line_th=0.0;
swarm_size=20;                       %Swarm size
number_of_points=6;                  %Number of particles
ob=50;                               %Number of obstacles
x_start = 1;
y_start = 1;
x_end = 23;
y_end = 30;
X=0:1:23;
Y=0:1:30;

XS=round(23*rand(swarm_size,number_of_points));
YS=round(30*rand(swarm_size,number_of_points));
XYOb=zeros(ob,1);
XYOb(:,1)=round(18*rand(ob,1))+3;
XYOb(:,2)=round(25*rand(ob,1))+3;
save('XY_Ob','XYOb')
load('XY_Ob.mat','XYOb');

%===========================
% Working Environment
%===========================
for m1=1:ob
hold on
plot(XYOb(m1,1),XYOb(m1,2),'--rs','MarkerEdgeColor','r',
'MarkerFaceColor','r','MarkerSize',6);
end
plot(1,1,'ok','MarkerEdgeColor','k',
'MarkerFaceColor','b','MarkerSize',10);
hold on
plot(23,30,'ok','MarkerEdgeColor','k',
'MarkerFaceColor','b','MarkerSize',10);
axis([0,23,0,30])
grid on
box on
```

```
set(gca,'YTick',0:2:30)
set(gca,'XTick',0:1:23)
title('Working Environment');
xlabel('X');
ylabel('Y');


%===================================
% obstacles checking for feasibility
%===================================
redo=1;
while redo==1
redo=0;
for i4=1:ob
xc=abs(XS-XYOb(i4,1));
yc=abs(YS-XYOb(i4,2));
xyc=xc+yc;
xycn=round(xyc./(xyc+0.01));
[ro,co]=find(xycn==0);
for i5=1:size(ro,1)
XS(ro(i5),co(i5))=round(23*rand(1));
YS(ro(i5),co(i5))=round(30*rand(1));
redo=1;
end
end
end


%Feasibility of line segments%
%----------------------------
redo2=1;
while redo2==1
redo2=0;
for i6=1:swarm_size

%start segment
slope=(YS(i6,1)-1)/(XS(i6,1)-1);
for k1=1:ob
if abs(slope)==inf
if XYOb(k1,1)==XS(i6,1)
XS(i6,1)=round(23*rand(1));
YS(i6,1)=round(30*rand(1));
redo2=1;
end
else
onl=abs((XYOb(k1,2)-1)-slope*(XYOb(k1,1)-1));
```

```
if onl<line_th
XS(i6,1)=round(23*rand(1));
YS(i6,1)=round(30*rand(1));
redo2=1;
end
end
end

% in between segments
for j=1:number_of_points-1
slope=(YS(i6,j+1)-YS(i6,j))/(XS(i6,j+1)-XS(i6,j));
for k2=1:ob
if abs(slope)==inf
if XYOb(k2,1)==XS(i6,j)
XS(i6,j+1)=round(23*rand(1));
YS(i6,j+1)=round(30*rand(1));
redo2=1;
end
else
onl=abs((XYOb(k2,2)-YS(i6,j))-slope*(XYOb(k2,1)-XS(i6,j)));
if onl<line_th
XS(i6,j+1)=round(23*rand(1));
YS(i6,j+1)=round(30*rand(1));
redo2=1;
end
end
end
end

% end segment
slope=(30-YS(i6,number_of_points))/
(23-XS(i6,number_of_points));
for k3=1:ob
if abs(slope)==inf
if XYOb(k3,1)==XS(i6,number_of_points)
XS(i6,number_of_points)=round(23*rand(1));
YS(i6,number_of_points)=round(30*rand(1));
redo2=1;
end
else
onl=abs((XYOb(k3,2)-30)-slope*(XYOb(k3,1)-23));
if onl<line_th
XS(i6,number_of_points)=round(23*rand(1));
YS(i6,number_of_points)=round(30*rand(1));
redo2=1;
end
```

```
end
end
end
end
for i=1:swarm_size
hold on
plot ([1,XS(i,:),23],[1,YS(i,:),30])
end

pbest_X=XS;
pbest_Y=YS;
gbest_X=pbest_X(1,:);
gbest_Y=pbest_Y(1,:);
D=inf*ones(swarm_size,1);
D_pbest=D;
D_gbest=inf;

c1=2;
c2=2;
w_min=0.2;
w_max=0.9;
w=w_max-(w_max-w_min)*(0:1:N-1)./(N-1);   %inertia weight
v_X=zeros(swarm_size,number_of_points);
v_Y=zeros(swarm_size,number_of_points);

%---- Constraints -------------------
v_X_max=2;
v_X_min=-2;
v_Y_max=2;
v_Y_min=-2;

X_min=1;
Y_min=1;
X_max=23;
Y_max=30;

hold off

%MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
%$$$$$$  main loop of algorithm
%MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
for it=1:N
for ks=1:swarm_size
dis=0;
dis=dis+sqrt((1-YS(ks,1))^2+(1-XS(ks,1))^2);
for k4=1:number_of_points-1
```

```
dis=dis+sqrt((YS(ks,k4)-YS(ks,k4+1))^2+
(XS(ks,k4)-XS(ks,k4+1))^2);
end
dis=dis+sqrt((30-YS(ks,number_of_points))^2+
(23-XS(ks,number_of_points))^2);
D(ks)=dis;
if D(ks)<D_pbest(ks)
D_pbest(ks)= D(ks);
pbest_X(ks,:)=XS(ks,:);
pbest_Y(ks,:)=YS(ks,:);
end
end
[min_D_pbest,idx]=min(D_pbest);
if min_D_pbest<D_gbest
D_gbest=min_D_pbest;
gbest_X=pbest_X(idx,:);
gbest_Y=pbest_Y(idx,:);
end

%UPDATE V and P
% ** Update the velocity and Postion vectors_START_ ***
for ks2=1:swarm_size
r2=rand(1);
v_X(ks2,:)=w(it)*v_X(ks2,:)+(c2.*r2.*(gbest_X-XS(ks2,:)))
v_Y(ks2,:)=w(it)*v_Y(ks2,:)+(c2.*r2.*(gbest_Y-YS(ks2,:)))

% *** Check the velocity constraint_START
for kv=1:number_of_points

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
if v_X(ks2,kv)<v_X_min
v_X(ks2,kv)=v_X_min;
else
if v_X(ks2,kv)>v_X_max
v_X(ks2,kv)=v_X_max;
end
end

%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
if v_Y(ks2,kv)<v_Y_min
v_Y(ks2,kv)=v_Y_min;
else
if v_Y(ks2,kv)>v_Y_max
v_Y(ks2,kv)=v_Y_max;
end
end
```

```
end

% ... Check the velocity constraint_END

% ** Update the position vector
XS(ks2,:)=round(XS(ks2,:)+v_X(ks2,:));
YS(ks2,:)=round(YS(ks2,:)+v_Y(ks2,:));
% in the searching range
%|||||||||||||||||||||||||||||||||||||||||||
%|||||Feasibility TEST||||||||||||||||||||||
%----------------------------
redo=1;
while redo==1
redo=0;
for i4=1:ob
xc=abs(XS-XYOb(i4,1));
yc=abs(YS-XYOb(i4,2));
xyc=xc+yc;
xycn=round(xyc./(xyc+0.01));
[ro,co]=find(xycn==0);
for i2=1:size(ro,1)
rx=round(4*(rand(1)-0.5));
XS(ro(i2),co(i2))=XS(ro(i2),co(i2))+rx;
ry=round(4*(rand(1)-0.5));
YS(ro(i2),co(i2))=YS(ro(i2),co(i2))+ry;
redo=1;
end
end
end

redo2=1;
while redo2==1
redo2=0;

% start segment
slope=(YS(ks2,1)-1)/(XS(ks2,1)-1);
for k3=1:ob
if abs(slope)==inf
if XYOb(k3,1)==XS(ks2,1)
XS(ks2,1)=XS(ks2,1)+1;
YS(ks2,1)=YS(ks2,1)+ry;
redo2=1;
end
else
onl=abs((XYOb(k3,2)-1)-slope*(XYOb(k3,1)-1));
if onl<line_th
```

```
rx=round(4*(rand(1)-0.5));
XS(ks2,1)=XS(ks2,1)+rx;
ry=round(4*(rand(1)-0.5));
YS(ks2,1)=YS(ks2,1)+ry;
redo2=1;
end
end
end

% in between segments
for j=1:number_of_points-1
slope=(YS(ks2,j+1)-YS(ks2,j))/(XS(ks2,j+1)-XS(ks2,j));
for k3=1:ob
if abs(slope)==inf
if XYOb(k3,1)==XS(ks2,j)
XS(ks2,j+1)=XS(ks2,j+1)+1;
redo2=1;
end
else
onl=abs((XYOb(k3,2)-YS(ks2,j))-slope*(XYOb(k3,1)-  XS(ks2,j)));
if onl<line_th
rx=round(4*(rand(1)-0.5));
XS(ks2,j+1)=XS(ks2,j+1)+rx;
ry=round(4*(rand(1)-0.5));
YS(ks2,j+1)=YS(ks2,j+1)+ry;
redo2=1;
end
end
end
end

% end segment
slope=(30-YS(ks2,number_of_points))/
(23-XS(ks2,number_of_points));
for k3=1:ob
if abs(slope)==inf
if XYOb(k3,1)==XS(ks2,number_of_points)
XS(ks2,number_of_points)=XS(ks2,number_of_points)+1;
redo2=1;
end
else
onl=abs((XYOb(k3,2)-30)-slope*(XYOb(k3,1)-23));
if onl<line_th
rx=round(4*(rand(1)-0.5));
XS(ks2,number_of_points)=XS(ks2,number_of_points)+rx;
ry=round(4*(rand(1)-0.5));
```

```
YS(ks2,number_of_points)=YS(ks2,number_of_points)+ry;
redo2=1;
end
end
end
end

% *** Check the X and Y Bounds____START
for kp=1:number_of_points

%XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
if XS(ks2,kp)<X_min
XS(ks2,kp)=X_min;
else
if XS(ks2,kp)>X_max
XS(ks2,kp)=X_max;
end
end

%YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
if YS(ks2,kp)<Y_min
YS(ks2,kp)=Y_min;
else
if YS(ks2,kp)>Y_max
YS(ks2,kp)=Y_max;
end
end
end
% *** Check the X and Y Bounds_____END

% ...Update the velocity and Postion vectors_END_...
end


%EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
disp(['generation= ',num2str(it)]);
disp(['min_D_pbest= ',num2str(min_D_pbest),
'  at index = ',num2str(idx)]);
disp('__________________')
disp(['D_gbest= ',num2str(D_gbest)]);
disp(['gbest:  ',num2str(gbest_X)]);
disp('=======================');
disp('@@@@@@@@@@@@@@@@@@@@@@@');
ax=[1,gbest_X,23];
by=[1,gbest_Y,30];
clf
```

```
hold off

%===========================
% Working Environment Plot
%===========================
for m1=1:ob
hold on
plot(XYOb(m1,1),XYOb(m1,2),'--rs','MarkerEdgeColor','r',
'MarkerFaceColor','r','MarkerSize',6);
end
plot(1,1,'ok','MarkerEdgeColor','k'
'MarkerFaceColor','b','MarkerSize',10);
hold on
plot(23,30,'ok','MarkerEdgeColor','k',
'MarkerFaceColor','b','MarkerSize',10);
axis([0,23,0,30])
grid on
box on
set(gca,'YTick',0:2:30)
set(gca,'XTick',0:1:23)
title('Working Environment');
xlabel('X');
ylabel('Y');
plot(ax,by,'LineWidth',2);
axis([0,23,0,30]);
text(5,28,['Distance=  ',num2str(D_gbest)],'FontSize',18);
pause(0.001);
end
hold on
plot(ax,by,'g','LineWidth',2);
axis([0,23,0,30]);
hold on
plot([1,23],[1,30],'k')
```