Research Article

Nachamada Vachaku Blamah*, Aderemi Adewumi Oluyinka, Gregory Wajiga, and Yusuf Benson Baha

MAPSOFT: A Multi-Agent based Particle Swarm Optimization Framework for Travelling Salesman Problem

https://doi.org/10.1515/jisys-2020-0042
Received Apr 24, 2020; accepted Oct 01, 2020

Abstract: This paper proposes a Multi-Agent based Particle Swarm Optimization (PSO) Framework for the Traveling salesman problem (MAPSOFT). The framework is a deployment of the recently proposed intelligent multi-agent based PSO model by the authors. MAPSOFT is made up of groups of agents that interact with one another in a coordinated search effort within their environment and the solution space. A discrete version of the original multi-agent model is presented and applied to the Travelling Salesman Problem. Based on the simulation results obtained, it was observed that agents retrospectively decide on their next moves based on consistent better fitness values obtained from present and prospective neighborhoods, and by reflecting back to previous behaviors and sticking to historically better results. These overall attributes help enhance the conventional PSO by providing more intelligence and autonomy within the swarm and thus contributed to the emergence of good results for the studied problem.

Keywords: Multi-agent system, neighborhood, retrospective, topology, Belief-Desire-Intention, space/model

2020 Mathematics Subject Classification: 68T42, 68T20, 68T05, 68T37

1 Introduction

A multi-agent system (MAS) is made up of multiple interacting agents, with each agent having a set of goal(s) to accomplish. The agents are, to some extent, capable of autonomous actions, deciding for themselves what they need to do in order to satisfy their design objectives, and they normally engage in interaction with other agents in order to bring about the desired result. They are typically equipped with sufficient cognitive abilities to reason about a domain, make certain types of decisions, and perform the associated actions. Environments that are dynamic, uncertain, or complex are very good candidates for MAS, and communication is the basis for interactions and social organizations which enables the agents to cooperate and coordinate their actions [4, 46, 55].

Particle Swarm Optimization (PSO) is inspired by the social foraging nature of animals such as the flocking behaviors of birds moving towards an optimal goal [10], and it is in the class of technique based around

***Corresponding Author: Nachamada Vachaku Blamah:** Department of Computer Science, University of Jos, Jos, Nigeria; School of Computer Science, University of KwaZulu-Natal, Durban, South Africa; Email: blamahn@yahoo.com; blamahn@unijos.edu.ng
Aderemi Adewumi Oluyinka: School of Computer Science, University of KwaZulu-Natal, Durban, South Africa; Email: adewumi@ukzn.ac.za
Gregory Wajiga: Department of Computer Science, Moddibo Adama University of Technology, Yola, Nigeria; Email: gwajiga@gmail.com
Yusuf Benson Baha: Department of Information Technology, Moddibo Adama University of Technology, Yola, Nigeria; Email: bybaha@yahoo.com

the study of collective behavior in decentralized, self-organized systems. Each particle agent is subject to a movement in a multidimensional space that represents the ***belief*** space. Particle agents have memory, thus retaining part of their previous state, and there is no restriction for particles to share the same point in ***belief*** space, but in any case their individuality is preserved [20].

The PSO was first developed by [30] as an optimization method for continuous functions, and variants of this technique [29, 45, 49] have been invented with improvements on various features of the original model. The main notion about this scheme is that particles move towards more suitable members of the swarm and generally bias their movements towards historically good areas of the environment and they try to achieve the optimal goal by cooperating with their neighbors in addition to taking independent decisions and actions.

[38] applied PSO algorithm to solve Flexible job-shop scheduling problem (FJSP) aimed at minimizing the maximum completion time criterion. In the study, various benchmark data taken from literatures were tested and the experimental results proved that the developed PSO is effective and efficient to solve the FJSP. The research also proposed and compared two multi-agents based approaches using different benchmark instances to study the distribution of the PSO solving method for future implementation on embedded systems that can make decisions in real time according to the state of resources and any unplanned or unforeseen events. [3] presented a framework for co-operative path planning and task assignment of a multi-agent system to find an optimal path for the multi-agent system working under certain prescribed tasks. Simulation from the study revealed a reliable and robust way of solving co-operative path planning problems. [37] presented a broad review on Swarm Intelligence dynamic optimization focused on several classes of problems, such as discrete, continuous, constrained, multi-objective and classification problems, and real-world applications. It also focused on the enhancement strategies integrated in SI algorithms to address dynamic changes, the performance measurements and benchmark generators used in swarm intelligence dynamic optimization. The research identified several aspects for future directions as experimental protocols, benchmark generators, avoiding change detection, modeling real-world scenarios, consideration of other prospectives, choice of algorithms, and theoretical development. [24] proposed a hybrid approach based on PSO and twin support vector regression (TSVR) for forecasting a wind speed. TSVR was utilized to enhance the forecasting accuracy of the wind speed, and the optimal setting of TSVR parameters were optimized carefully by PSO. Experimental results revealed that the optimized PSO-TSVR approach is able to forecast wind speed with an accuracy of 98.996. The computational results proved that the proposed approach achieved better forecasting accuracy and outperformed the comparison algorithms.

[56] presented a generalized predictive PID control for main steam temperature based on improved PSO algorithm, with simulation results that show faster response speed, smaller overshoot and control error, better tracking performance, and reduced the lag effect of the control system over previous systems.

Although the conventional PSO and other similar algorithms are computationally attainable, they are influenced and controlled by a unified algorithm thereby rendering the swarm to have regulated intelligence with limited autonomy. By utilizing the inherent features of multi-agent systems, we present a Multi-Agent based Particle Swarm Optimization Framework for the Traveling salesman problem (MAPSOFT), in which the swarm particles *execute autonomously* and are capable of *asynchronous communications* with *improved learning* capabilities, thereby making the approach more superior. Similarly, scalability is a great attribute desired in many swarm systems since the populations change dynamically. In this paper, agents are given the abilities of moving from neighborhood to neighborhood that may be different in their sizes.

The remainder of the paper is organized as follows: in section 2, we present the Travelling Salesman Problem (TSP). This is followed by section 3 which presents the rationale for a multi-agent based PSO for the TSP. Section 4 looks at the intelligent PSO framework while section 5 relates PSO and MAS implementations. The model formulation for our scheme is presented in section 6 while the major algorithm follows in section 7. The computational results and discussion are then presented in section 8 and finally section 9 presents the conclusion and future work.

2 The Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is a well-known optimization problem that has been studied extensively and is often used for testing new algorithms [9, 23]. This problem is concerned with finding an optimum path of travel among N cities, and it is briefly described as follows: Given N cities on a map, find the shortest route that visits all cities once, and starts and ends at the same city. While Particle Swarm Optimization (PSO) was designed as a method for continuous problems, it has been transformed and applied to many discrete problems, among which is the TSP [50, 69]. Although there are existing models of multi-agent based TSP [62, 67], we apply a Multi-Agent based Particle Swarm Optimization Framework to the TSP (MAPSOFT) in order to test the appropriateness of our scheme. More details on TSP can be found in [13, 16, 22, 33, 35, 63, 68].

In an attempt to compare the performance of swarm intelligence algorithms, [47] used four common swarm intelligence algorithms to solve TSP, which include genetic algorithm, particle swarm optimization, ant colony optimization, and artificial bee colony. For each swarm intelligence, the various parameters and operators were tested, and the best values were selected for it. Experiments over some benchmarks from TSP LIB show that artificial bee colony algorithm is the best one among the four swarm intelligence based methods to solving routing problems like TSP.

Different authors have looked at divers optimization approaches to solve this computational problem. In [70], an improved version of the imperialist competitive algorithm for solving TSP was proposed. This was targeted at solving discrete combinatorial optimization problems, which changes the way of forming initial empires from that of the traditional approach.

Various authors have applied Genetic Algorithms (GA) to solve the TSP. In their work, [15] proposed an improved GA with initial population strategy, particularly to address the symmetric TSP. [11] looked at a parallel GA version to propose a solution for the TSP optimization, where the processes within the GAs were parallelized for the purpose of speeding up the execution. [18] presented a GA with trajectory prediction for solving the dynamic TSP. [64] modeled a high-performance GA and then used the TSP as a case.

Flower Pollination Algorithms (FPA), Ant Colony Optimization (ACO), African Buffalo Optimization (ABO), Bat Algorithm (BA) and Particle Swarm Optimization (PSO) approaches have also been proposed in literature for the TSP; [36] applied the ACO with local search to the dynamic TSPs, while [65] used a modified ACO for TSP and [54] developed an optimal solution for large scale traveling transportation problems using FPA. [40] applied the ABO technique to solve the TSP. They demonstrated that buffalos are able to ensure excellent exploration and exploitation of the search space through regular communication, cooperation, and good memory of the previous personal exploits as well as tapping from the herd's collective exploits. [41] present a discrete version of the BA to solve both the symmetric and asymmetric TSP. They further proposed an improvement in the basic structure of the classic BA. [28] proposed a new ant supervised-PSO variant applied to the TSP. In their approach, ACO played the role of the main optimization procedure while PSO was used to detect the optimum values of certain ACO parameters. Similarly, [19] developed solution of Multiple TSP using PSO-based Algorithms.

Even though these approaches achieved various degrees of successes in their models and results, they however implement highly centralized algorithms that coordinate the search processes, thereby limiting the autonomy of the individual agents and making the communications more synchronous.

3 Multi-Agent based PSO for TSP

From the belief-desire-intension (BDI) model's perspective, implementing the Particle Swarm Optimization (PSO) scheme from the conventionally *supervised* approach does not possess the autonomy such a system merits; we notice that the particles reach their goals by executing *elaborate program*, with random variables that imply autonomy. They are narrowed in their behaviors as a result of the *uniform algorithms* they execute, which restrict intelligence and self-sufficiency. PSO naturally falls within categories of systems in which the particles are given some targets in some high level manner, and the particles autonomously choose how to accomplish the targets in the best way.

In addition, there are versions of the PSOs in which the total population can shrink or grow dynamically [2, 21, 39, 42], depending on the behaviors and strengths of particles in the system. For this reason, scalability is desirable, and this is *hard* to achieve in a system that is cohesive and monolithic in nature. Therefore, agents should be given the abilities of moving from neighborhood to neighborhood that may be different in their sizes. This characteristic is available in Multi Agent System (MAS), as each agent is normally treated in isolation.

It is obvious that complex patterns of communication arise among the particles in a typical PSO setting such that communications are synchronous, which reduces the systems performance, particularly if the size of the swarm is high. In a MAS's implementation, there exist asynchronous communication naturally, since the agents execute autonomously with inherent concurrency. More so, population-based algorithms that are implemented without concerns for isolated treatment of each member are very complex. Therefore, the performance of the system will be enhanced when each member of the swarm is treated separately.

Other features of PSO [52] that qualify it to be modeled as MAS can easily be identified; Natural algorithm: the PSO is built on the behavior of real fish/birds which are real agents; Distributed and parallel algorithm: the swarm is a group of agents which move independently, simultaneously and without a leader or supervisor; Cooperative agents: each particle chooses a new position somewhat based on the information received from other particles.

Some of these issues have been addressed in literatures as MAS-based PSO [1, 32, 34, 48] with emphasis on implementations, modeling and fault tolerance/load balancing. In this research, an intelligent and robust PSO framework is presented, which is founded on the BDI model of MAS, and this is an extension of the works in [6, 7]. We incorporate learning abilities into the agents so that they can adjust their behaviors in terms of optimality and performance, and the model is tested on the Travelling Salesman Problem. These are outlined in details in [6].

4 Intelligent Particle Swarm Optimization Model

A good excerpt [12] that presents the idea of the particle swarm optimization (PSO) as an intelligent system states:

> The PSO algorithm is a population-based algorithm, where a set of potential solutions ***evolves*** to approach a convenient solution for a problem, and the aim is to find the global optimum of the fitness function defined in a given search space. The agents that are part of a society hold an ***opinion*** that is part of a "***belief space***" (the search space) shared by every possible agent. Agents may modify this "***opinion state***" based on either the ***knowledge*** of the environment (its fitness value), the individual's previous ***history of states*** (its memory), or the previous ***history of states*** of the individual's neighborhood.
>
> Following certain rules of interaction, the individuals in the population adapt their scheme of ***belief*** to the ones that are more successful among their social network. Over the time, a culture arises, in which the individuals hold ***opinions*** that are closely related.

This captures the idea of the model of agency where agents start operating within their environments with initial set of beliefs that eventually transforms into intentions that they get committed to.

Intelligence is linked with the way *reasoning* is done so as to arrive at a deduction. It refers to the capacity to come to correct conclusions about what is real or true, and about how to resolve problems [14]. Reasoning is a broad subject matter that denotes the capacity to make a sense of things, to establish and verify facts, and to justify or change practices and *beliefs* [31].

Practical reasoning model to agency [44, 66] are suitable in representing part of intelligent actions of the particle agents within the system. There are two major components that make up practical reasoning [66]: Deliberation (which indicates the state of affairs to be achieve, which turns out to be the *intention* of the agent) and Means-Ends Reasoning (which is an indication of how to accomplish the intentions, which produces a *plan*). The intentions in this context are the future directed intentions, which represent the agent's state of mind prior to taking any action.

We can model deliberation as the process of option generation and filtering, which are described as follows:

- The current beliefs and intentions are described as the option generation function, which produce the agent's set of *desire*. Thus,

$$option : 2^{Bel} \times 2^{Int} \to 2^{Des} \tag{1}$$

- Next, we obtain the *intentions* to be committed to by filtering and selecting the best options using the following filter relationship:

$$filter : 2^{Bel} \times 2^{Des} \times 2^{Int} \to 2^{Int} \tag{2}$$

The beliefs of an agent is updated by the use of a belief review function, which is described as:

$$brf : 2^{Bel} \times Per \to 2^{Bel} \tag{3}$$

where *Per* represents set of percepts within the operating environment.

A particle agent within the system will practically reach an alternative by deliberating on the available options, and the decisions will be taken by selecting the most promising alternatives. Agents keep on refining their percepts, and as time progresses, the perceptions and beliefs about the entire swarm may be refined, and the particle's intentions and desires may also be redefined in order to reflect the changes in the database of belief.

When an agent completes its deliberations and obtains commitments to some intentions, the agent will then need a plan on how to achieve the intentions based on the present environment's state (agent's belief) and the available actions. This is the means-ends reasoning.

Therefore, an agent will perceive its operating environment and adjusts its belief database accordingly, facilitates the production of the intentions. From there, practical reasoning will be applied in order to take an action that modifies the swarm, which adjusts the system towards a possible solution. This idea is modeled and represented by Figure 1, and incorporated in Algorithm 1.

A host of recent researches on efficient PSO based systems have been proposed in literature. Notable among these include the work done by [57], where they designed a SVM predictive control for calcination zone temperature in lime rotary kiln with improved PSO algorithm, and a work on short-term wind speed prediction based on improved PSO algorithm [59]. Others include a work by [51] who developed an enhanced partial search to PSO for unconstrained optimization, a work by [17] who applied an improved PSO to vehicle crashworthiness, and that of [53] who also designed an improved convergence PSO algorithm with random sampling of control parameters. Other specific applications of PSO in literature include modified PSO algorithms for the generation of stable structures of carbon clusters [26] and the application of improved PSO in vehicle depot overhaul shop scheduling [25].

5 Particle Swarm Optimization and Multi-Agent System

Within our implementation of Multi-Agent System (MAS) – based Particle Swarm Optimization (PSO), the particle agents consider problems by weighing opposing considerations for and against competing options, where the relevant considerations are provided by what the agent desires or values, and what it believes [8, 66].

A particle takes action by first considering and deliberating *what* state of affairs to achieve from existing options, which represents its *Intentions* that modify its state of mind. From there, the agent reasons on *how* to accomplish the selected state of affairs, which yields a plan of how best to realize the option. This is how intelligence is incorporated into the entire system

We look at the following specific issues in this paper:

1. The concept of neighborhood as a more complex information-sharing scheme among particles was introduced because the inertia weight PSO model gets trapped easily in local minima, especially in complex problems, where the swarm easily collapses due to complete diversity loss [43].
   In the neighborhood approach to PSO implementation, there is a reduction of the overall global information exchange structure to a local one, where information is broadcast in small parts of the population only, within each iteration. Each particle adopts a set of other particles to be its neighbors and, as each iteration passes, it communicates the values of its best position only to these agents. Therefore, the needed information about the overall best position within the swarm is firstly communicated to the neighborhood of the best agent only, and then successively to the remaining agents through their neighbors.
   Looking at this method, an agent is tied to a fixed neighborhood for interaction with each iteration without *planning* ahead and foreseeing better fitness figures with other neighborhoods within the same iteration; so when a particle initially part of a neighborhood, it does not directly share neighborhood best information with other particles outside its direct neighborhood within a single iteration to see if such interaction will produce better fitness values.
   In our implementation, we increase diversity and cohesion in the search space so as to avoid *blind commitment* which normally gets agents stuck in certain local minima. The particles use the Belief-Desire-Intension (BDI)-like reasoning and alternate neighbors dynamically (regardless of the neighborhood topology) within the search process. Within each iteration, a particle calculates several fitness values, all in parallel (based on neighborhood bests from the main neighborhood, and other neighborhoods), and saves the history record in a database of belief. With time, the particle will stick to neighbors that produce better values of fitness, depending on the best results obtained. As the agents interact in this manner, the best global behavior emerges.
2. A belief database is created in order to keep the particles' experiences. This is because overtime, the particles keep refining their beliefs and learn more within the swarm.
3. In the conventional PSO, a particle's action is influenced by both personal cognitive and social components. These intrinsic features of PSO make it good candidates for design as MAS; the social component is modeled as part of the multi-agents' implementation as the agents interact, while the cognitive component is designed as part of the individual agent's execution. The social component is implemented through communicators, a notion that we will explain later.
   These qualities as described above make the PSO agents more intelligent and autonomous.
   Other multi-objective models geared towards optimizations of real-life systems are demonstrated by [58, 60].

6 MAPSOFT Model Formulation

Let us consider the original Particle Swarm Optimization (PSO) model represented as:

$$V_i^{t+1} = \omega V_i^t + C_1 R_1 \left(pBest_i^t - X_i^t \right) + C_2 R_2 (lBest_i^t - X_i^t) \tag{4}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{5}$$

where t denotes the iteration counter, R_1 and R_2 are random numbers distributed within [0,1], C_1 and C_2 are weighting factors representing the cognitive and social parameters respectively, $pBest_i^t$ is the best position known so far by the particle i, $lBest_i^t$ is the neighborhood best, V_i^{t+1} is the velocity of particle i in iteration $t+1$, and X_i^{t+1} is the particle's position in iteration $t+1$.

Variants of the PSO [39, 42] demonstrate that the population within the swarm can grow or shrink, which means in reality, the particles in a swarm, within each iteration, can interact with as many agents as possible, ranging from 1 to $n-1$, n being the size of the swarm. This therefore inspires us to build the system such that each agent within the multi-agent-based PSO is able to communicate to various neighborhoods, and will not always limit its communication to a fixed number of agents. Therefore, the system assigns few other prospective neighborhoods to each agent apart from the primary neighborhood. Within every iteration, the $lBest_i^t$ values of these prospective neighborhoods are passed to the agent together with the $lBest_i^t$ value from the primary neighborhood, and the agent will process both main and prospective *positions*, *velocities* and *fitness* values in parallel. The particle will then store the prospective values in a database of belief (called prospective neighborhood vector hereafter), but uses the primary fitness in order to maintain its affiliation with the primary neighborhood. The agent reflects back on its execution history after a set time-stamp has elapsed [6]. The neighborhoods within the system are then reformulated according to fitness values that appear to be more promising.

At this point, we introduce discrete TSP [50, 69] representation that we used to formalize our model, and this will be followed by the description of the model. We will specifically adopt the models in [6, 7] and then redefine them to align to the TSP.

Suppose there are N cities to be visited, with x_i representing the set of edges. We consider symmetric TSP such that edge $(x,\ y)$ is the same with edge $(y,\ x)$.

Definition 1: $P(x,\ y)$ represents the probability P of connecting cities x and y, where $(x,\ y)$ represents an edge in the search space. When updating the Hamilton cycle $H(edges)$, a random number Q, $0 \le Q \le 1$ is generated such that

$$H(edges) \bigcup \begin{cases} choose\ (x,\ y) & Q \le P \\ discard\ (x,\ y) & otherwise \end{cases} \tag{6}$$

Definition 2: Velocity V is a set of elements $P(x,\ y)$, represented as

$$V = \{P_1(x_1,\ y_1),\ P_2(x_2,\ y_2),\ldots,\ P_n(x_n,\ y_n)\} \tag{7}$$

Definition 3: If N is a real number and $S = \{P_1(x_1,\ y_1),\ P_2(x_2,\ y_2),\ldots,\ P_n(x_n,\ y_n)\}$ is the set of edges, a multiplication between N and S is defined as:

$$N \star \{P_1(x_1,\ y_1),\ P_2(x_2,\ y_2),\ldots,\ P_n(x_n,\ y_n)\} = \{N \star P_1(x_1,\ y_1),\ N \star P_2(x_2,\ y_2),\ldots,\ N \star P_n(x_n,\ y_n)\} \tag{8}$$

where $P_1, \ldots, P_n$ represent the probabilities of the corresponding edges.

Definition 4: The subtraction between two position vectors x_i and x_j is the set of edges which exist in x_i but not in x_j with a 100% probability added to the edges to make it uniform as velocity.

If the position vector $v_1 = \{(x_1, y_1),\ldots,\ (x_n, y_n),\ (x_{n+1}, y_{n+1}),\ldots,\ (x_{n+k}, y_{n+k})\}$ and $v_2 = \left\{(x_1, y_1),\ (x_2, y_2),\ldots,\ (x_n, y_n),\ \left(x',\ y'\right)\right\}$, (where $\left(x',\ y'\right)$ are all pairs of x's and y's that are not present in v_1,) then

$$v_1 - v_2 = \{1(x_{n+1}, y_{n+1}),\ 1(x_{n+2}, y_{n+2}),\ldots,\ 1(x_{n+k}, y_{n+k})\} \tag{9}$$

Definition 5: in order to add two velocities v_1 and v_2, we perform vector additions on them. In order to prevent the velocity from resulting into a huge set, each city is limited to appear at most a certain number of times, say 5 times in one velocity (depending on the number of cities in the TSP) with any overflow discarded.

Therefore if $v_1 = v'$ and $v_2 = v''$, then

$$v_1 + v_2 = v' \oplus v'' \tag{10}$$

In the remaining part of this section, we present the multi-agent based schemes (Definitions 6 to 12) that describe the MAPSOFT model.

Definition 6: The environment state in which the search space of the agent may be, is defined as follows:

$$E = \{P_1,\ P_2,\ldots,\ P_N\} \tag{11}$$

where $P_i = (P_{i1}, P_{i2}, \dots, P_{in})^T$ is the best positions ever visited by each particle agent, representing the present state ($i = 1, 2, \dots, N$; N being the population size and n the current iteration counter).

We are going to describe more concepts in order to aid us to define an agent's action.

Each particle agent has a range of actions at its disposal, which are the consequences of agent's invocation. In order to accommodate more neighborhoods and prospective positions, we slightly modify the velocity and position formulae of equations (4) and (5) by simply converting them to k-dimensional vectors.

$$V_i^{k(t+1)} = \omega V_i^{k(t)} + C_1 R_1 \left(pBest_i^{(t)} - X_i^{k(t)}\right) + C_2 R_2 \left(lBest_i^{k(t)} - X_i^{k(t)}\right) \tag{12}$$

$$X_i^{k(t+1)} = X_i^{k(t)} + V_i^{k(t+1)} \tag{13}$$
$$1 \le k \le \eta$$

Each agent uses only one particle's best $pBest$ within each iteration, but the velocity V, neighborhood best $lBest$, and position X, all become k-dimensional vectors, where k ranges between 1 and the total number of neighborhoods η in the system.

So in practical implementation, the value of k depends on the number of neighborhoods each agent takes concurrently while executing. If $k = 1$, it becomes the conventional PSO; if $k = 2$, it means each particle considers 2 neighborhoods – its present neighborhood and 1 prospective neighborhood; if $k = 3$, then each particle considers 3 neighborhoods – its present neighborhood and 2 other prospective neighborhoods, and so on.

Let the actions in equations (12) and (13) be represented by the set

$$V_x = \{V, X\} \tag{14}$$

Let all the operations represented by equations (8), (9) and (10) be called basic operations. We will represent this as:

$$B_o = \{Sub, Mul, Add\} \tag{15}$$

Definition 7: If we generally define the set of actions at the disposal of agents as $Ac = \{\alpha, \alpha', \dots\}$, then specifically, the i^{th} particle agent in the system has these actions [6]:

$$Ac_i = \{V_x, C, \sigma, B_o\} \tag{16}$$

where V_x is the set of velocity and position functions described by equation (14), C is a communicator [5] which the agent uses to communicate with other agents, thereby separating the social activities of the Multi-Agent System (MAS) from the individual agent's activities, σ is a recap function that permits an agent to appraise its history from the last time stamp in order to decide whether or not to change neighborhood within the set G (explained in equation (20)).

Particle agents in the search space have single-minded commitments, because an agent continues to maintain an intension of improving the fitness values within a particular neighborhood until it believes either that the intension has been achieved, or else it is no longer a more feasible option to remain in that neighborhood, in which case it is rational for the agent to move away to a more promising neighborhood.

In our model, we assume that the size of neighborhoods can vary because there may be increase or decrease in population; agents can move from one neighborhood to another, or any other factor may occur that can alter the population size [39].

Definition 8: A run, r, of an agent in an environment is a sequence of interleaved environment states and actions. If we let R be the set of all such runs, then we have:

$$R = \{r, r', \dots\} \tag{17}$$

Let R^{Ac} be the subset of these that end with an action and R^E be the subset of these that end with an environment state.

Definition 9: When a particle agent invokes an action on an environment, it transforms the environment state, and this effect is modeled by the state transformer function defined as:

$$\tau : R^{Ac} \rightarrow 2^{E} \tag{18}$$

where 2^E is the power set of E.

This means that from runs which end with actions taken by a particle agent, the system will always end up in a particular environment state; taking an action by a particle agent on a previous environment state moves the environment to another state.

Definition 10: In a multi-agent system, the agents drive the system. The state of the environment emerges as a result of the actions by the agent – that is, based on the behaviors and interactions among the agents. As agents produce these actions when they execute in the system, agents are modeled as a function of execution, which yield actions (whose effect is the state transformer function). Thus, a particle agent is defined as:

$$A : R^{E} \rightarrow Ac \tag{19}$$

So if an action, say the position update function $X\epsilon\ Ac$, is desired of a particle agent A', the agent produces this action by executing on an existing run ending with environment state, say r', which is its current position, as follows:

$$X = A'(r')$$

This leaves the run to end with an action. The effect of taking this action, which is modeled by the state transformer function, τ, is to produce a new environment state.

Definition 11: We define an environment as a tuple:

$$\xi = \langle E,\ e_0,\ \tau,\ T,\ G\,\rangle \tag{20}$$

where E is the set of environment states described by equation (11), $e_0 \in E$ is an initial state, τ is a state transformer function described by equation (18), T is the active topology in the environment, and $G = \{g_1,\ g_2, \ldots, g_\eta\}$ is a set of η neighborhoods such that:

$$\varnothing \notin G,$$

$\bigcup G = S$, S being the swarm, and

$$\forall g_i, g_j \in G \ni g_i \neq g_j \Rightarrow g_i \bigcap g_j = \varnothing,\ 1 \leq i, j \leq \eta.$$

Definition 12: The Swarm S is defined to be the set of all agents, as follows:

$$S = \{A_1,\ A_2, \ldots,\ A_n\} \tag{21}$$

where $\bigcup G = S$. The Swarm System is thus defined as $R(S,\ \xi)$, where R is the set of all runs.

7 MAPSOFT Algorithm

Having established the definitions and the relationships above, we now describe the algorithms that aid particle agents to execute within the swarm.

Particle agents need to *plan* ahead and envisage better fitness values with other neighborhoods within the same iteration by sharing information with agents outside the main neighborhood. The particle agents thus get committed to achieving better fitness values by reasoning and dynamically alternating neighbors in the search process.

Based on the model of agency described in section 4, the behavior of *each* particle agent towards the choice of neighborhood is represented by Figure 1, which centrally realizes the recap action σ. This controls the switch (or otherwise) of neighborhoods for the particles. The numbers on the arrows in this figure only indicate the order of execution of the thought process of particles, and where the same numbers exist, it means they occur concurrently. So "*Option*", for example, is concurrently obtained from the "*Beliefs*" and the "*Intentions*".

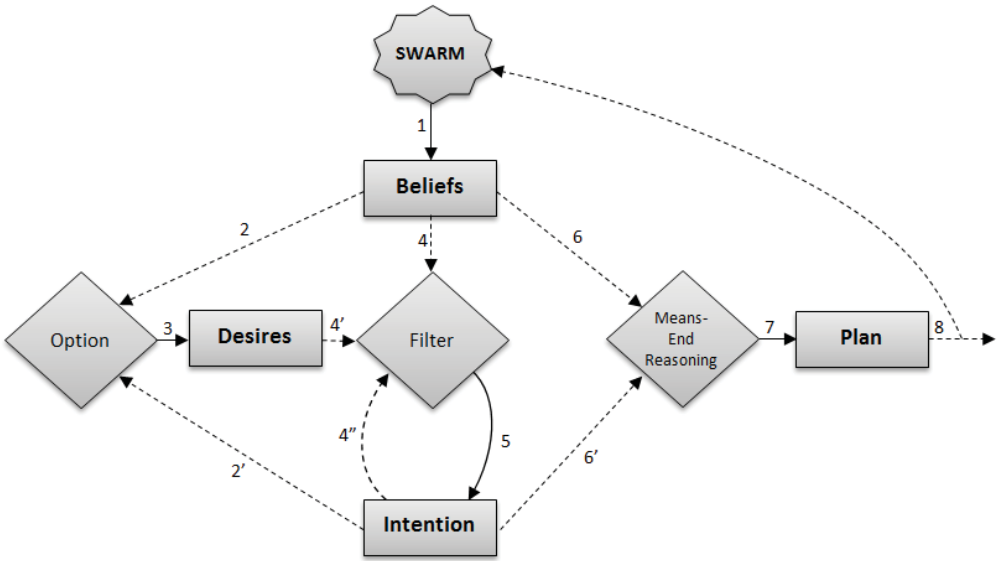

Figure 1: Particle's Thought Process for Choice of Neighborhood.

Each particle perceives its environment by using its previous belief, and then communicating with other agents through the communicator component C of equation (16) to get the current $lBest$ value.

The Belief update process is obtained using equation (3). Deliberation is then carried out by option generation and filtering using equations (1) and (2) respectively, to produce the corresponding Desires and Intentions. Having obtained these values, a particle reasons and formulates a plan by using the new values of beliefs and intentions, which is then executed.

The belief space is the search space, where each agent believes that there is a solution in the search space. This belief is theoretically kept by each particle without taking any action – they only have the knowledge of the existence of a solution. The particles further believe that any neighborhood has a solution, and when the population is initialized, the values of $pbest$ and $lbest$ are predefined for each agent. The environment state as initially perceived by particle agents, denoted by equation (11), represents the initial belief of the agents.

The desire of an agent is to be in any of the neighborhood $g_1, \; g_2, \ldots, \; g_\eta$ that gives more promising fitness value. If the best position of particle A_i in iteration t is represented by $pbest_i\,(t)$, then the desire of an agent is described as:

$$\forall A_i \in S, \; \exists g_j \in G: \; (A_i : R^E \to \sigma) \Lambda (A_i \in g_j) \Rightarrow pbest_i\,(t)\,,$$

$1 \le i \le n;\; 1 \le j \le \eta$, n is the swarm size and η is the total number of neighborhoods.

The initial intentions of each particle are predefined such that each agent is committed to its present neighborhood. But with time, each agent keeps to the prospective neighborhood which yields a better fitness value after the elapse of the chosen time stamp. This will eventually yield the global best of the fitness function.

With the initial beliefs and the initial intentions, an agent is executed using Algorithm 1.

In addition to the basic particle swarm optimization (PSO) processes, this algorithm handles the choice of prospective neighborhood between lines 4 and 8, and line 18 increments the timestamp. Note that line 6 executes the plan based on Figure 1.

Algorithm 1 MAPSOFT Process

```
1:  Randomly initialize the whole swarm
2:  Initialize $timeStamp = 0$; $threshold = thresholdVal$;
3:  While ( termination criteria is not met) {
4:      if ($timeStamp = threshold$) {
5:          for ( $i = 0$; $i < swarmsize$; $i++$)
6:              $Execute(plan)$; //based on Figure 1, and select a better neighborhood
7:          $timeStamp = 0$;
8:      }
9:      for ( $i = 0$; $i < swarmsize$; $i++$) {
10:         $Evaluate\ f(x_i)$;//where $x_i$ is the position of particle $i$
11:         if ($f(x_i) > f(pbest_i)$) $pbest_i = x_i$;
12:         if ($f(x_i) > f(lbest)$) $lbest = x_i$;
13:     }
14:     for ( $i = 0$; $i < swarmsize$; $i++$) {
15:         $calculate\ (v_i)$;//using equation (12)
16:         $update\ (x_i)$;//using equation (13)
17:     }
18:     $timeStamp++$;
19: }
```

8 Complexity Analysis

We consider Algorithm 1 in order to perform the complexity analysis of MAPSOFT. This involves the time for the choice of prospective neighborhood (T_p), initialization (T_i), evaluation (T_e), and velocity and position update (T_u) for each particle. If we assume that n is the population of the swarm, which represents the dimension of the search space, the time complexity of MAPSOFT can be estimated as:

$$T(n) = T_p + T_i + (T_e + T_u) = p + 3n$$

Now the size of the prospective neighborhood p is always less than the population size n, *i.e.*, $p < n$ Therefore, $O(n)$ is the time complexity of MAPSOFT.

9 Simulation and Computational Results

A simulation of the new algorithm was performed in order to illustrate the applicability of the entire scheme. The algorithm was implemented using java 7 SDK, and the machine for the execution was a hp Pavilion dv6 Notebook PC, 350 GB HDD, Intel (R) Core i3 CPU, M350@ 2.27 GHz and 4GB of RAM. The parameter settings for the Particle Swarm Optimization (PSO) were: C1 = 1.7, C2 = 1.7, w = 0.715, rand1(0, 1), rand2(0, 1), as recommended in [27]. With 10 agents initially grouped into 2 neighborhoods, each agent was assigned only 1 prospective neighborhood. The simulation was executed in 50 iterations (only 30 iterations shown in Table 1 for space) over a Travelling Salesman Problem (TSP) of 10 cities, and the threshold value was set to 6.

The result obtained is presented in Table 1, which shows two neighborhoods g_1 and g_2, each initially containing agents A_{11}, A_{12} and A_{21}, A_{22} respectively. Each of the agents has two results for fitness functions represented by f' and f''. f' is computed using $lbest$ of present neighborhoods, while f'' is computed based on the $lbest$ of the prospective neighborhoods.

We notice the behavior of A_{12} from g_1 that, after the elapse of the second timestamp, a switch of neighborhood was made to g_2 based on consistent better fitness values obtained from this prospective neighborhood, and this neighborhood eventually emerged with the overall $lbest$.

Figure 2 shows the overall behavior of the agents within the system. The un-shaded region with broken line represents all searches made by agents within neighborhood g_1, and the un-shaded region with solid line represents searches made within neighborhood g_2. The intersection of g_1 and g_2 represents where the fitness values from the two regions coincided, and the searches made by A_{12} for both f' and f'' span the two regions, which is indicated by the dot-shaded region with double solid line. A switch from g_1 to g_2 gave a better fitness due to the agent's ability to reflect back to previous behaviors and stick to historically better results.

By utilizing the inherent features of multi-agent systems, the agents in MAPSOFT *execute autonomously* and are capable of *asynchronous communications* with *improved learning* capabilities. Moreover, scalability is a great attribute desired in swarm systems since the populations change dynamically. The agents in MAPSOFT are given the abilities of moving from neighborhood to neighborhood robustly, as illustrated in Table 1.

Table 1: Fitness values and Neighborhood switch by agent A_{12}

		Neighbourhood g_1				*Neighbourhood* g_2			
t	σ	A_{11}		A_{12}		A_{21}		A_{22}	
		f'	f''	f'	f''	f'	f''	f'	f''
1	1	179	162	169	146	160	170	145	150
2	2	179	162	169	146	160	170	145	150
3	3	179	162	169	146	160	170	145	150
4	4	179	162	169	131	160	170	145	150
5	5	168	162	166	128	154	170	145	150
6	6	168	162	166	120	154	166	145	150
7	1	168	152	166	113	154	166	145	150
8	2	168	152	166	107	154	166	142	150
9	3	168	152	166	107	140	166	142	150
10	4	168	152	160	105	140	155	142	150
11	5	168	140	160	100	140	155	142	120
12	6	166	140	160	100	140	155	121	120
13	1	166	140	160	100	140	155	121	120
14	2	166	140	150	91	130	155	121	120
15	3	166	140	150	91	120	155	121	120
16	4	150	140	146	91	120	155	121	120
17	5	150	135	146	91	120	135	121	120
18	6	150	135	125	91	120	135	115	120
19	1	150	135	125	91	110	135	115	120
20	2	120	135	120	91	100	135	115	120
21	3	120	135	115	91	100	135	115	118
22	4	120	120	115	91	90	135	115	118
23	5	118	120	115	84	75	135	115	118
24	6	118	120	115	80	75	135	115	118
25	1	118	120	115	76	75	135	115	118
26	2	118	115	115	76	72	118	115	118
27	3	118	115	115	76	72	118	115	116
28	4	118	115	109	76	72	110	98	116
29	5	115	100	109	65	60	110	98	116
30	6	115	100	109	65	60	110	98	116

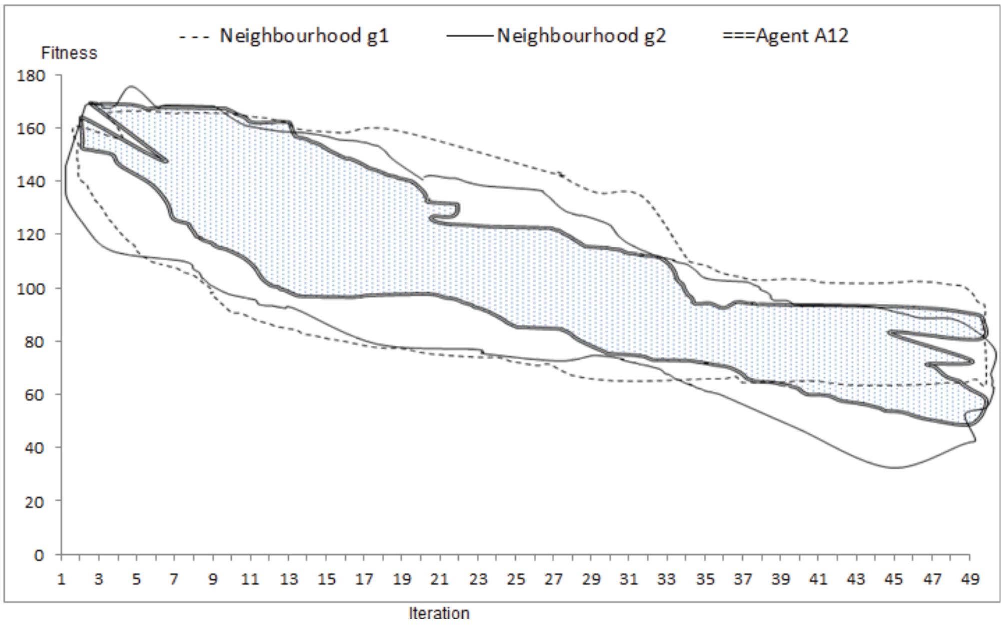

Figure 2: Fitness values within Neighborhoods g_1 and g_2 for all agents.

10 Conclusion and Future Work

The techniques of Particle Swarm Optimization (PSO) are generally made up of a group of particle agents that interact with their environments, and also with other particle agents, with the singular aim of obtaining the best solution within the environment. This offers a natural way of implementing the system as a multi-agent system. A novel multi-agent based scheme for PSO was designed and applied to the Travelling Salesman Problem (TSP). Learning capabilities were built into the agents so that they can adjust their behaviors dynamically, as they search for optimal solutions. Communicators were used in order to achieve autonomy within the system; the communicators separate agents' personal operations from that of the swarm. The traditional PSO algorithm was modified in order to incorporate the desired learning capabilities into the agents. The agents were designed to make parallel computations of fitness values based on the *lbest* of many neighborhoods and keep the result in the belief database for future use, and retrospectively stick to historically better fitness values. This scheme has enhanced the conventional PSO by providing for more intelligence and autonomy within the swarm.

The key defect of the MAPSOFT scheme is the additional overhead of maintaining information about the various neighborhoods by the agents in the system. As part of our future work, we intend to design the scheme to quickly explore more alternate neighborhoods as the search progresses, thereby making it more robust. This will then be applied to real-life scenario and compared with existing test results. Other improved search algorithms [61] can be applied to improve on the global search.

References

[1] Ahmad, R., Lee, Y., Rahimi, S., and Gupta, B. (2007). A Multi-Agent Based Approach for Particle Swarm Optimization, *In IEEE Publications*, 1-4244-0945-4/07, Paper 17.
[2] Bell, W.J., Roth, L., and Nalepa, C. (2007). *Cockroaches: Ecology, Behavior, and Natural History*, The Johns Hopkins University Press.
[3] Biswas, S., Anavatti, S. G., and Garratt, M. A. (2017). Particle Swarm Optimization Based Co-operative Task Assignment and Path Planning for Multi-Agent System. IEEE, 978-1-5386-2726-6/17

[4] Blamah, N. V. and Adewumi, A. O. (2012). A multi-agent-based model for distributed system processing, *International Journal of the Physical Sciences*, 7(34), pp. 5297-5303.

[5] Blamah, N. V., Adewumi, A. O. and Olusanya, M. O. (2013). A Secured Agent-Based Framework for Data Warehouse Management. *Proceedings of IEEE International Conference on Industrial Technology (ICIT)*, pp1840-1845, Cape Town.

[6] Blamah, N. V., Adewumi, A. O., Wajiga, G. M. and Baha, B. Y. (2013). An Intelligent Particle Swarm Optimization Model based on Multi-Agent System, IEEE, The African Journal of Computing and ICTs, 6(2), pp1-8. www.ajocict.net

[7] Blamah, N. V. (2013). A Reflective Swarm Intelligence Algorithm, Journal Of Computer Engineering, 14(4), pp. 44-48

[8] Bratman, M.E. (1990). *What is Intention?* In Cohen, P.R., Morgan, J.L., and Pollack, M.E., editors, Intentions in Communication, The MIT Press: Cambridge, MA.

[9] Brezina, I and Čičková, Z. (2011). Solving the Travelling Salesman Problem Using the Ant Colony Optimization. Management Information Systems, *6 (4), pp. 010-014*

[10] Brownlee, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*. 1st ed., Lulu. http://www.CleverAlgorithms.com

[11] Cakir, M. and Yilmaz, G. (2015). Traveling salesman problem optimization with parallel genetic algorithm, *IEEE 23nd Signal Processing and Communications Applications Conference (SIU) Malatya*, 2015.

[12] Confort, M and Meng, Y. (2008). Reinforcing Learning for Neural Networks using Swarm Intelligence, *IEEE Swarm Intelligence Symposium*, St. Louis MO, USA, Sept., 21-23, 2008.

[13] Davendra, D. (2010). “Traveling Salesman Problem, Theory and Applications,” InTech, ISBN 978-953-307-426-9.

[14] Davidson, H. (1992). *Alfarabi, Avicenna, and Averroes, on Intellect*, Oxford University Press,

[15] Deng, Y., Liu, Y. and Zhou, D. (2015). An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP, *Mathematical Problems in Engineering*, 2015, Hindawi Publishing Corporation.

[16] Dorigo, M. and Gambardella, L. (1997). “Ant Colonies for the Travelling Salesman Problem,” Bio Systems, Vol. 43, pp. 73-81.

[17] Gao, D., Li, X. and Chen, H. (2019). Application of Improved Particle Swarm Optimization in Vehicle Crashworthiness, Hindawi, Mathematical Problems in Engineering Volume 2019.

[18] Groba, C., Sartal, A. and Vazquez, X (2015). Solving the Dynamic Traveling Salesman Problem using a Genetic Algorithm with Trajectory Prediction. *Computers and Operations Research*, 56(C): 22-32, Elsevier Science.

[19] Gulcu, S. D. and Ornek, H. K. (2019). Solution of Multiple Travelling Salesman Problem using Particle Swarm Optimization based Algorithms, International Journal of Intelligent Systems and Applications in Engineering, 7(2), 72-82.

[20] Gupta, P., Sharma, K., and Singh, P. (2012). A Review of Object Tracking using Particle Swarm Optimizatio, *VSRD, International Journal of Electrical Electronics & Comm. Engg. 2(7)*.

[21] Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tache, F., Said, I., Durier, V., Canonge, S., Amé, J.M., Detrain, C., Correll, N., Martinoli, A., Mondada, F., Siegwart, R., and Deneubourg, J.L. (2007). Social integration of robots into groups of cockroaches to control self-organized choices, *Science*, November, 318(5853), pp.1155–1158.

[22] Hjertenes, M. O. (2002). “A Multilevel Scheme for the Travelling Salesman Problem,” University of Bergen.

[23] Hlaing, Z. C. S. S. and Khine, M. A. (2011) An Ant Colony Optimization Algorithm for Solving Travelling Salesman Problem. *International Conference on Information Communication and Management IPCSIT vol.16, IACSIT Press, Singapore*.

[24] Houssein, E. H. (2017). Particle Swarm Optimization-Enhanced Twin Support Vector Regression for Wind Speed Forecasting. J. Intell. Syst; https://doi.org/10.1515/jisys-2017-0378, *pp 1-10*

[25] Huang, M., Zhao, Z. and Liang, X. (2019). Application of Improved Particle Swarm Optimization in Vehicle Depot Overhaul Shop Scheduling, 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, pp. 103-106.

[26] Jana, G., Mitra, A., Pan, S., Sural, S. and Chattaraj, P.K. (2019). Modified Particle Swarm Optimization Algorithms for the Generation of Stable Structures of Carbon Clusters, Cn (n = 3–6, 10).

[27] Jiang, M, Luo, Y. P. and Yang, S. Y. (2007).Particle Swarm Optimization – Stochastic Trajectory Analysis and Parameter Selection. *In Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, I-Tech Education and Publishing, Vienna, Austria.

[28] Kefi, S., Rokbani, N., Kromer, P. and Alimi, A. M. (2016). A New Ant Supervised-PSO Variant Applied to Traveling Salesman Problem, *Hybrid Intelligent Systems*, Advances in Intelligent Systems and Computing, 420, Springer Int. Pub.

[29] Kennedy, J. (1999). Small worlds and mega-minds: Effects of Neighborhood Topology on Particle Swarm Performance. In Proceedings of the 1999 Congress on Evolutionary Computation.

[30] Kennedy, J., and Eberhart, R. (1995). Particle Swarm Optimisation. In: Proceedings of the IEEE Conf. on Neural Networks, Perth.

[31] Kompridis, N. (2000). So We Need Something Else for Reason to Mean, *International Journal of Philosophical Studies*, 8(3), pp. 271-295.

[32] Lee, K. C., Lee, N. and Li, H. (2009). A particle swarm optimization-driven cognitive map approach to analyzing information systems project risk, Journal of the American Society for Information Science and Technology, 60(6), 1208-1221

[33] Lin, S. (1965). “Computer Solutions of the Traveling Salesman Problem,” Bell System Technical Journal, Vol. 44, pp. 2245-2269.

[34] Lorion, Y., Bogon, T., Timm, I.J., and Drobnik, O. (2009). An Agent Based Parallel Particle Swarm Optimization – APPSO, *In IEEE Publications*, 978-1-4244-2762-8/09

[35] Matai, R., Mittal, M. L., and Singh, S. (2010). *Traveling salesman problem: An overview of applications, formulations, and solution approaches*: In Tech Open Access Publisher.

[36] Mavrovouniotis, M., Muller, F. P. and Yang, S. (2016). Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems. *IEEE Transactions on Systems, Man, and Cybernetics,* Part B: *Cybernetics*.

[37] Mavrovouniotis, M., Li, C., and Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. Preprint submitted to Journal of Swarm and Evolutionary Computation.

[38] Nouiri, M., Bekrar, A., Jemai, A., Niar, S., and Ammari, A. C. (2015). An effective and distributed particle swarm optimization algorithm for flexible jo-shop scheduling problem. J Intell Manuf, Springer Science and Business Media, New York, DOI 10.1007/s10845-015-1039-3,

[39] Obagduwa, I.C. (2012). Cockroaches Optimization Algorithms, Seminar paper presented at the Progress Seminar, March 3rd, UKZN, Durban.

[40] Odili, J. B. and Kahar, M. N. M. (2016). Solving the Traveling Salesman's Problem Using the African Buffalo Optimization, *Computational Intelligence and Neuroscience*, 2016, Hindawi Publishing Corporation.

[41] Osaba, E., Yang, X., Diaz, F., Lopez-Garcia, P. and Carballedo, R. (2016). An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems, *Engineering Applications of Artificial Intelligence*, 48 (1), 59-71.

[42] Parpinelli, R.S., and Lopes, H.S. (2011). New inspirations in swarm intelligence: a survey, *International Journal of Bio-Inspired Computation*, 3(1), pp.1–16.

[43] Parsopoulos, K.E., and Vrahatis, M.N. (2010). *Particle Swarm Optimization and Intelligence: Advances and Applications*, Information Science Reference, Hershey, (NY).

[44] Petrie, H.G. (1971). *Philosophy & Rhetoric* © 1971, Penn State University Press.

[45] Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle Swarm Optimization, An Overview. *Swarm Intelligence*, 1, pp.33–57.

[46] Reza G, Amin MF, Hamid T, Mahdi S (2008). Evolutionary Query Optimization for Heterogeneous Distributed Database Systems, *World Academy of Science, Engineering and Technology, 43, pp.43-49*.

[47] Sabet, S., Shokouhifar, M., and Farokhi, F. (2016). A comparison Between Swarm Intelligence Algorithms for Routing Problems. Electrical & Computer Engineering: An International Journal (ECIJ), Vol. 5, No. 1, *pp. 17-33*

[48] Shangxiong, S. (2008). A Particle Swarm Optimization (PSO) Algorithm Based on Multi-Agent System, In IEEE International Conference on Intelligent Computation Technology and Automation, 978-0-7695-3357-5/08

[49] Shi, Y., and Eberhart, R.C. (1998). A Modified Particle Swarm Optimizers. In Proceedings of the IEEE International Conference on Evolutionary Computation, pp.69–73.

[50] Shi, X.H., Liang, Y. C. Lee, H.P. Lu, C. and Wang, Q.X. (2007). Particle swarm optimization-based algorithms for TSP and generalized TSP, Elsevier, *Information Processing Letters 103 (2007) 169–176*

[51] Shu-Kai, S. F. and Chih-Hung, J. (2019). An Enhanced Partial Search to Particle Swarm Optimization for Unconstrained Optimization, Mathematics, 2019, 7, 357.

[52] Sivanandam, S.N., and Deepa, S.N. (2008). *Introduction to Genetic Algorithms*, Springer-Verlag Berlin Heidelberg

[53] Sun, L., Song, X. and Chen, T. (2019). An Improved Convergence Particle Swarm Optimization Algorithm with Random Sampling of Control Parameters, Hindawi, Journal of Control Science and Engineering Volume 2019.

[54] Suwannarongsri, S. and Puangdownreong, D. (2019). Optimal Solving Large Scale Traveling Transportation Problems by Flower Pollination Algorithm, WSEAS Transactions on Systems and Control, vol. 14.

[55] Taghezout N, Adla A, Zarate P (2009). A Multi-agent Framework for Group Decision Support System: Application to a Boiler Combustion Management System (GLZ). *International Journal of Software Engineering and Its Applications, 3(2), pp.9-20*

[56] Tian, Z.D., Li, S.J. and Wang, Y.H. (2017). Generalized predictive PID control for main steam temperature based on improved PSO algorithm. Journal of Advanced Computational Intelligence and Intelligent Informatics, 21(3):507–517.

[57] Tian, Z.D., Li, S.J. and Wang, Y.H. (2018a). SVM Predictive Control for Calcination Zone Temperature in Lime Rotary Kiln with Improved PSO Algorithm. Transactions of the Institute of Measurement and Control, 40(10): 3134-3146.

[58] Tian, Z.D., Li, S.J. and Wang, Y.H. (2018b). The multi-objective optimization model of flue aimed temperature of coke oven. Journal of Chemical Engineering of Japan, 51(8): 683-694.

[59] Tian, Z.D., Ren, Y. and Wang, G. (2019). Short-term Wind Speed Prediction Based on Improved PSO Algorithm Optimized EM-ELM. Energy Sources, Part A: Recovery, Utilization, and Environmental Effects, 41(1):26-46.

[60] Tian, Z.D., Wang, G. and Ren, Y. (2019). AMOAIA: Adaptive Multi-objective Optimization Artificial Immune Algorithm. IAENG International Journal of Applied Mathematics, vol. 49, no.1, pp14-21.

[61] Tian, Z.D. and Zhang, C. (2018). An Improved Harmony Search Algorithm and its Application in Function Optimization. Journal of Information Processing Systems, 14(5): 1237-1253.

[62] Tiejun, Z., Yihong, T. and Lining, X. (2006). A multi-agent approach for solving travelling salesman problem, Wuhan University Journal of Natural Sciences, 11(5), 1104-1108

[63] Tsai, H. K., Yang, J. M., Tsai, Y. F. and Kao, C. Y. (2004). "An Evolutionary Algorithm for Large Traveling Salesman Problems," IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 34(4), pp. 1718-1729.

[64] Tsai, C., Tseng, S., Chiang, M., Yang, C. and Hong, T. (2014). A High-Performance Genetic Algorithm: Using Traveling Salesman Problem as a Case, *Scientific World Journal*, 2014, Hindawi Publishing Corporation.

[65] Wei, X., Han, L. and Hong, L. (2014). A Modified Ant Colony Algorithm for Traveling Salesman Problem, *Int. J. of Comp., Comm. & Control.* 9(5): 633-643.

[66] Wooldridge, M. (2009), *An Introduction to MultiAgent Systems*. 2nd ed., John Wiley & Sons Ltd, Chichester.

[67] Xie, X. and Liu, J. (2009). Multiagent Optimization System for Solving the Travelling Salesman Problem (TSP), IEEE Transactions on Systems, Man, and Cybernetics-Part B:Cybernetics, 39(2), 489-502

[68] Yan, X., Zhang, C., Luo, W., Li, W., Chen, W. and Liu, H. (2012). "Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm," IJCSI International Journal of Computer Science Issues, Vol. 9(6), pp. 264-271.

[69] Zhong, W., Zhang, J. and Chen, W. (2007). A Novel Discrete Particle Swarm Optimization to solve Travelling Salesman Problem. *In Proceedings of IEEE Congress on Evolutionary Computing (CEC 2007)*.

[70] Zhang, X., Chen, X., Xiao, H. and Li, W. (2016). A new imperialist competitive algorithm for solving TSP problem. *Control and Decision*, 31(04): 586-592.