Journal of Information Assurance and Security
ISSN 1554-1010 Volume 11 (2016) pp. 369-384
© MIR Labs, www.mirlabs.net/jias/index.html

GPU PSO and ACO Applied to TSP for Vehicle Security Tracking

Olfa Bali[1], Walid Elloumi[1], Ajith Abraham[2,3] and Adel M. Alimi[1]

[1] REGIM-Lab: Research Groups on Intelligent Machines,
University of Sfax, National Engineering School of Sfax (ENIS)
BP 1173, Sfax, 3038, Tunisia
bali.olfa@gmail.com
{walid.elloumi, adel.alimi}@ieee.org

[2] Machine Intelligence Research Labs (MIR Labs), P.O. Box 2259, Auburn, WA 98071-2259, USA
[3]VSB - Technical University of Ostrava, Ostrava, Czech Republic
ajith.abraham@ieee.org

***Abstract*: The Travelling Salesman Problem (TSP) is a well-known benchmark problem for many meta-heuristic algorithms, including security traffic optimization problems. TSP is known as NP hard complex. It was investigated using classical approaches as well as intelligent techniques using Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and other meta-heuristics. The Graphic Processing Units (GPU) is well suited to the execution of nature and bio-inspired algorithms due to the rapidity of parallel implementation of GPUs. In this paper, we present a novel parallel approach to run PSO and ACO on GPUs and applied to TSP (GPU-PSO&ACO-A-TSP) for security tracking vehicles in road traffic. Both algorithms are implemented on the GPUs. Results show better performance optimization when using GPUs compared to results using sequential CPU implementation.**

***Keywords*:** PSO, ACO, TSP, GPU, CUDA, Optimization, Security.

I. Introduction

In the field of Engineering, the optimum solution of a problem is defined using optimality criteria. Mathematical equations and numerical interpretations are used to quantify optimality. Numerical interpretations should be fixed with respect to the problem characteristics, specificities and constraints [1]. While employing optimization techniques, we discriminate probabilistic and deterministic algorithms; ACO, Ant Colony Optimization [3], [18] algorithms are probabilistic methods, PSO particle swarm optimization [2].

Travelling Salesman Problem is a powerful optimization tool that permits to find the best path using particles and to track the paths of the vehicles in road traffic monitoring system. In our system, the vehicles travelling across roads represent the particles. Such system allows to offer best roads for the vehicles and to prevent road accidents by tracking vehicles during their traffic. GPS devices and electronic displays assure communication between the monitoring center and vehicles across the road.

Swarm intelligence techniques are bio-inspired methods, where group comportment is used to solve a problem based on the individualities of its members. Studies on insects' social behaviors have been applied in the field of optimization. Faced with the threats and dangers of nature, insects represent complex communication systems, and can show a great resistance. An individual insect may have only a few hundred-brain cells but a group of insects can have much more and is able to represent a well-organized architecture with better communication and resistance skills. In this case, we can speak of a complex social group that improves individual intelligence skills to more developed interactions and relationships between these groups. The flocks of birds, ants or fish strips promote a set of skills in artificial intelligence.

The central problem in the socio-biology of insects is the development of group's behaviors from the behaviors of individual ants. Provided that the behavior of a single ant is almost random, with a stochastic tendency to gravitate towards paths that have been trodden by other ants; the achievements of swarms of ants are most incredible. The behavior of an isolated ant quickly results in the demise of the individual, but the mass behavior of a colony of ants provides sustenance and defensive protection for the entire population [4].

An optimality criterion is generally expressed by a mathematical expression looking for a minimum value; these formulas are known as objective functions based on the concept of fitness. The heuristic optimizers are used in order to get a group of solutions that satisfy these functions.

Kennedy and Eberhart [5] began to develop PSO field that is based on a population approach using the intelligence of each particle. Through a flock of birds, they tried to simulate its behavior, like reaching an unknown destination as birds' goal (according to fitness), for example searching for food while flying (space research) [6]. Inspired by nature, the particles communicate and interact with each other [21]–[23]. Despite its simplicity, in nature, a swarm proceeds naturally with multi-objective optimization in all its activities, MOPSO,

Multi-objective PSO is a PSO with some multi-objective optimum criteria. Therefore, solving multi-objective problems, MOPs, can be considered as the mixture of both searching and decision-making approaches [7], [19], [20].

Examinations on ants' behavior showed that if ants have food located at some distance from the nest, with two unequal path length leading to it, ants will try both paths then will choose at the end the swarm having the shortest route. Using real ants, experiments have shown that when several ants move the nest to the food with different paths that are uneven; these ants will find the shortest path through the pheromones deposited by ants. If an obstacle is added on a path, the ants are able to move and to choose the shortest path in a definite time. If two sources are equal for example if both paths are of equal distance, the ant will choose a path arbitrarily [8].

Recently many works on intelligent security systems have been published. In [31], authors propose a system of vehicle anonymity enhancement in vehicular Ad-Hoc networks that consist in updating the pseudonym of vehicles regularly in order to preserve their privacy.

A hybrid security system based on naïve Bayes and decision trees used for network intrusion detection is presented in [29].

In [30], the authors developed a security intrusion detection program based on the use of evolutionary algorithms, especially Genetic Algorithm.

The biological studies on natural ant's shows that ants are able to produce specific pheromone and to identify the chemicals emitted substances and the glands that emit them [9]. Dorigo et al. also have painfully identified the fixed action answers to each of the various pheromones. They found out that pheromone involves a medium for communication among the ants, allowing fixed action collaboration; the result shows that a group behavior is adaptive while the individuals' behaviors are not. Initially the AS-PSO was proposed by Elloumi et al, 2009 in [10].

Several recent and important works that used the GPU to accelerate optimization problems, we can find PSO [26]–[28].

In this paper, we investigate an aspect of GPU particle swarm optimization and ant colony optimization applied to TSP (GPU-PSO&ACO-A-TSP), focusing only on the implementation of GPUs on the two algorithms mentioned above. The remaining of this paper is organized as follows: In section 2, we review the PSO, ACO algorithms and TSP problem. Section 3 is dedicated to the presentation of CUDA programming model. A GPU model is presented in section 4. Our approach, GPU Particle Swarm Optimization and Ant Colony Optimization Applied to Travelling Salesman Problem, is illustrated in Section 5. Section 6 includes experimental results with discussions of our approach. The paper is ended by a conclusion and further works openings in Section 7.

II. Variants of PSO, ACO and TSP

A. *Particle swarm Optimization*

This optimization technique is based on cooperation between individuals within a social organization. The group has to face and resolve the problems, by applying the capabilities of each individual as well sub-groups capacities. It belongs to a family of heuristic methods, called; swarm optimizers that include also the algorithm of ant colony optimization. ACO also relies on the concept of self-organization [11]. The idea is that a collection of individuals, who have a small amount of intelligence, each one, can produce a complex global organization. Thus, through simple rules of movement (in the space of solutions), the particles in the group can progressively converge to a local minimum.

PSO is fundamentally a method of parallel multi-agent research. The locations and velocities of particle creation, the updating speed, search of local and global optimums and the updating location are the four stages of PSO.

The role of PSO in our system is to act as a security supervisor that permits to identify accidents across the roads, prevent traffic jam and offer different roads for vehicles.

The research problem can be solved by using the position vector of a particle with respect to the search space. After a number of iterations, one can find the best global position of the particle that is about to fly with a certain speed [12].

For each iteration, each particle can update its velocity attributes using its best local position (p_{best} - particle best) and the best position of its global neighbors (g_{best} - Global Best), and then calculate the new location of each "particle" that is about to fly. Through this algorithm, one can find the global optimum, based on the trajectory and the behavior of each particle relative to its neighbors.

The communication between PSO particles is employed to assure the traffic security. An Ad-Hoc network of vehicles is used to exchange messages from a vehicle to another. The best global vehicles send messages to inform its neighbor's vehicles about traffic security information in order to prevent road accident and improve traffic quality.

The route of the particle swarm is calculated using all the adjacent particles and its past attributes.

During the optimization process, the MOPSO approach suffer from a loss of diversity, in this case MOSPO have some problems to maintain the balance between exploitation and exploration [13].

MOPSO allows use of the principle of density estimator for choosing the best overall particle and remove particles from the external archive. In this case, the non-dominated solutions are ranked in descending order, and a particle will be selected in a random manner [19]. We have presented our approach Fuzzy Ant Colony Optimization (FACO) and Multi-objective Particle Swarm Optimization (MOPSO) [24].

In PSO, the formulation of the velocity and the position, is given by equations (1) and (2)

$$V_{i+1,j+1}(t+1) = wV_{i,j}(t) + c_1 * rand() * (x_{lbest}(t) - x_{i,j}(t)) + c_2 * rand() * (x_{Gbest}(t) - x_{i,j}(t)) \quad (1)$$

$$x_{i+1,j+1}(t+1) = x_{i,j}(t) + V_{i+1,j+1}(t+1) \quad (2)$$

The inertia weight is represented by w ; i = 1, 2,. . ., N : indicates the number of population particles (swarm); t = 1,

2, . . . t_{max} indicates the number of iterations, $V_{i,j}$ is the symbol of the speed of the particle i, j^{th} , C_1 and C_2 are positive constants that modify the particle velocity while taking into account local particle x_{lbest} and global particle x_{Gbest} .

The algorithm starts by dispersing the particles randomly in search space. Then, the particles form a "bench" and explore the search space while maintaining cohesion between them and gathering around the optimum. They no longer run away from this optimum. Depending on the configuration of the algorithm, the particles end up in the same spot, which highlights a global trend to move towards the optimum.

B. Ant Colony Optimization

When moving, ants leave pheromone marks on the way, which disappear with time and distance. The ants follow the strongest path having the most amounts of pheromones, which facilitates for other ants to follow the shortest path [14].

Using the concept of pheromone by ants, it can solve the Travelling Salesman Problem [15]. The ant moves from the nest to the food and use the ant colony optimization to find the shortest path around obstacles.

ACO could be very useful for road traffic enhancement, even in case of obstacles blocking the roads, ACO continue to find other secure paths.

In ACO [8], [9] based on the amount of pheromone deposed on the path, we noticed that the ants follow the path having the densest amount of pheromone. Where $\tau_{i,j}$ represents the amount of pheromone of $ant_{i,j}$, which depends on the following probability:

$$f(j \in \Omega_i)$$

$$p_{i,j}^{(n)} = \frac{\left(\tau_{i,j}{}^{(n-1)}\right)^{\alpha}\left(\eta_{i,j}\right)^{\beta}}{\sum_{j\in\Omega_i}\left(\tau_{i,j}{}^{(n-1)}\right)^{\alpha}\left(\eta_{i,j}\right)^{\beta}} \quad (3)$$

With Ω_i is the neighborhood nodes for a given $ant_{i,j}$. The constants α and β denote the visibility of magnitude when switching from one path to another, which allows to move from node (i) to the node (j). $\eta_{i,j}$ represents the inverse distance between node (i) and node (j). In the ACO process, the update matrix of the amount of pheromone is necessary for the measurement; this update corresponds to the equation (4), ρ is the evaporation rate, while B_t represents the fastest lap.

$$if\,(i,j) \in B_t \longrightarrow \tau_{i,j}^{(n-1)} = (1-\rho)\tau_{i,j}^{(n-1)} + \rho\Delta_{i,j}^{(k)}$$
$$else \longrightarrow \tau_{i,j}^{(n-1)} = \tau_{i,j}^{(n-1)} \quad (4)$$

In best tour, B_t, determination several policies could be applied, the first one, and the simplest, consist in considering the best solution found from the beginning of the processing. The second alternative is to define a limited, B_t, that evaluates only the current procedure or a set of limited samplings to precedent B_t. The Combination of these two proposals with a moderation rates are also possible, in this paper the B_t, is selected using the first strategy.

According to the evaporation, another update of the amount of pheromone is performed when colonies displacement as shown in the equation (5).

$$\tau_{i,j}^{(n)} = (1-\rho).\tau_{i,j}^{(n-1)} + \rho\tau_{i,j}^{(0)} \quad (5)$$

C. The TSP problem

TSP, the traveling salesman problem, which may be defined as follows: at first we initialize (n) cities that must be visited. Initially we start from a city chosen randomly and then returns to the starting city. The objective is to determine the overall distance and visit every city just once with respect to fixed start/end locations [16].

In case of traffic accident, TSP optimization allows to inform other vehicles about the accident in order to follow better paths to reach destination safely and quickly.

An illustration of this problem with five cities is given in figure1; it shows two possible solutions, one in red and the other in green color. The two routes do not have the same length. A travelling salesman will choose the shortest path to reduce the cost of the travel. However, the TSP is said np-complete. In fact, for n cities the number of possible route is equal to (n − 1)!/2.

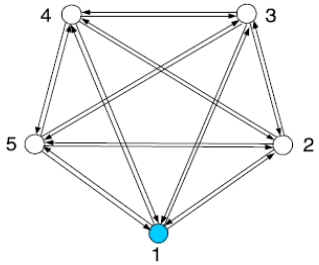

Figure 1. TSP possible solutions for a simplified cities representation, here the number of cities is limited to five

III. CUDA Programming model

A. What is CUDA?

CUDA is a parallel computing design developed by NVIDIA to multiply the system computing performance by coupling the power of graphics processors (GPU).

While millions of GPU compatible with CUDA were sold, thousands of software developers, scientists and researchers use CUDA in a wide range of areas, including the processing of images and videos, computational fields and more.

B. Parallel processing with CUDA

Data processing has evolved from the exclusive processing CPU to co-processing capabilities offered by the combination of GPU and CPU. To enable this new data processing paradigm, NVIDIA designed the CUDA parallel processing architecture, today included in GEFORCE and Tesla thus

offering important material base for application developers.

On the side of the public, most of the major applications of video processing are or soon will be accelerated by CUDA. The existing fleet of Tesla GPUs, providing significant capacity for GPU computing, allows gauging the success of CUDA. More than 700 GPU clusters are today active worldwide. The most important companies in the world have adopted CUDA.

IV. GPU model

A. GPU computing

The GPU computing consists to use the graphics processor (GPU) in parallel with the CPU to accelerate business applications of science, analysis, engineering, production and business. Launched in 2007 by NVIDIA GPU accelerators have established themselves as an industry standard. Worldwide, most of the low-energy data centers are used, both in government and university laboratories in small and medium enterprises. NVIDIA GPUs to accelerate many applications on supports as varied as Smartphone's, tablets, cars, drones and robotic systems.

B. How to accelerate applications with GPU

The GPU computing allows parallelizing jobs and offering high performance in several applications. The GPU accelerates the time-consuming parts of code in computational tasks, the rest of the application remains assigned to the CPU, which improves applications computational time.

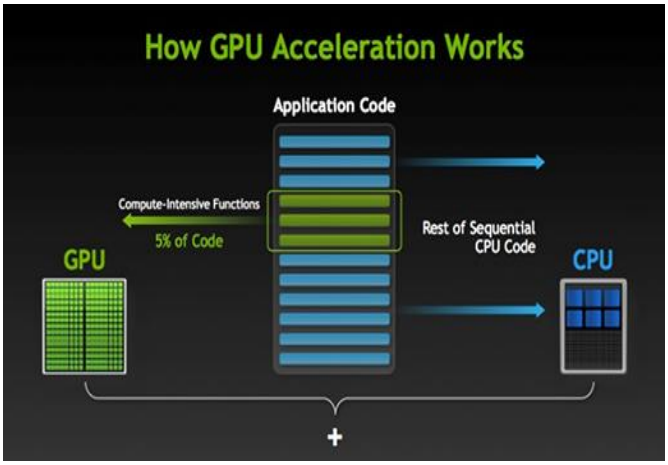

Figure 2. How GPU Acceleration Works

To understand the fundamental differences between a CPU and a GPU, just compare their treatment of each operation. The CPUs include a limited number of cores optimized for serial processing, whereas the GPU integrate thousands of cores designed to efficiently handle many simultaneous tasks.

V. GPU-PSO&ACO-A-TSP

In this section, we will study our approach having two essential parts. The first part explain GPU PSO-A to TSP while the second part explain GPU ACO-A to TSP.

The vehicle security tracking system based on PSO and ACO optimization require high CPU computation resources. That's explain the necessity to use GPU accelerated system.

A. GPU PSO

Meta-heuristic have emerged along with the paradigm itself, they are gaining popularity because they have worked well in some hard optimization problems such as travelling salesman, vehicle routing problem or the Hamiltonian path.

To design an evolutionary or a swarm solver, the main difficulties that arise are to determine the individual behavior, the environment and the social dynamic that govern the operation of the system to produce the desired collective response. Genetic algorithms, differential evolution, neural networks, PSO, ACO, or Bee colony optimization, were successfully applied to hard optimization problem, algorithms have shown potential capabilities at producing solutions, while the quality of the solutions depends on the heuristic parameters fittings. Classical heuristics used a set of user-designed parameters, while adaptive ones are trying to overcome this problem, given to the heuristic a capacity of parameters self-tuning.

In fact, the main idea that underlies the design of hierarchic heuristics is simple: for a given optimization problem, we have two algorithms, each with its strengths and weaknesses, the first one is used as a solver while the second is used to optimize to first solver. Classical ACO are used for discrete optimization while PSO is employed for continuous optimization problems.

Separately, PSO and ACO showed great potential in solving a wide range of optimization problems. We use both to solve optimization problems this is what Elloumi et al. have tried to do in [10]. The idea is to allow PSO to optimize the optimizer (ACO), knowing that ACO is used for discrete problems and PSO generally for continuous problems, and considering their strengths and their weaknesses. The figure 3 shows roughly the process of Graphical Process Unit Particle Swarm Optimization Applied to Travelling Salesman Problem.

The use of GPU acceleration in PSO security supervisor allows in case of traffic blocking to choose quickly other path for vehicles and to prevent from accidents.

Our goal is to cover all cities (designated nodes) once (if the particle passes through the city i to j it does not cross the town in the other direction, from j to i). Finally, the particle returns to the starting city, so we get a cycle.

The "gpuArray()" function allows copying data from the memory of the CPU to the GPU memory brings us to manipulate the table on the GPU memory.

Afterwards, we take the overall particle represented by x_{Gbest} and local particle referenced x_{lbest} each particle speed and positions are changed respectively in Equation 1 and Equation 2.

We had to repeat these steps until reaching the maximum number of iterations; it is assigned to each node. This allows us to obtain an archive, according to the latter; we can make a comparison between the different obtained paths. We choose the best way in terms of its execution time. Finally, we return

the GPU data to the CPU through the control "gather()" (see Figure 3).

B. GPU ACO

The heuristic is directly related to the physical problem and try to solve it, while the meta-heuristic adjusts the parameters of the heuristics.

The running of the classical ACO is based on parameters that are often set by the user of the algorithm. Thus, to found parameters that are appropriate for a problem, the user needs to perform many tests.

We have proposed an Ant colony algorithm on GPU applied to TSP.

The purpose is identical to that shown in PSO. We begin by initializing the ACO parameters; the number of iterations, the number of Ant particle is initialized every particle of ants on a start node randomly selected. The use of "gpuArray()" function allows us to transfer data to the CPU to GPU.

Each ant is chosen the next node according to probability calculated in Equation 3, if there are other nodes to visit we return to the previous step that is the second step.

The use of GPU acceleration in ACO helps to quickly suggest new paths for vehicles circulating on the roads to use these secure paths in case of obstacles.

Otherwise, it returns to the chief starting city and changing the amount of pheromone in a cycle. Assign each node a thread and repeat these steps until reaching the maximum number of iteration. If we arrive at stopping criterion the concept of pheromone placing procedure that includes data on the efficiency previous balance sheet, guides the building procedure to each thread. Solutions to the intermediate partial problems are seen that we display the best lap, otherwise we return in step 2.

The main steps above are designated for non-natural ants, simple calculation of individual agents and iterative used to build solutions to the problem that was been modelled as a graph.

The particles of Ants travel visiting the nodes, which are connected by arcs. Solutions of the problematic represent an ordered sequence of nodes. The research procedure is performed in parallel on multiple computational practical threads.

The dynamic memory structure that is inspired by the movement of the ant k at each iteration of the algorithm, (See Figure 4).

Distinctly, PSO and ACO showed a high resolution for optimization problems. To solve the ongoing problems in an efficient manner, this article introduces a new approach GPU PSO and ACO algorithm.

The proposed algorithm reduces the likelihood that minimizes falling into local optimum and improves the capacity and research accuracy. Definite steps that were followed to the end to keep the strength of our approach.

First, we start by resetting the search space, then the number of iterations for PSO and its cost function. The second step is to initialize the particle ant, the cost of this function. Thereafter we use the function "gpuArray()". Then follows the steps of the ACO or PSO. We try to change the position and speed of PSO particles, and Ant positions too. If the conditions are fulfilled, we arrive at our goal, which is the end, otherwise we return to the second step.

With GPU, displays the processing steps of the two algorithms ACO & PSO algorithm in which we can identify the various stages of operation of the algorithm.

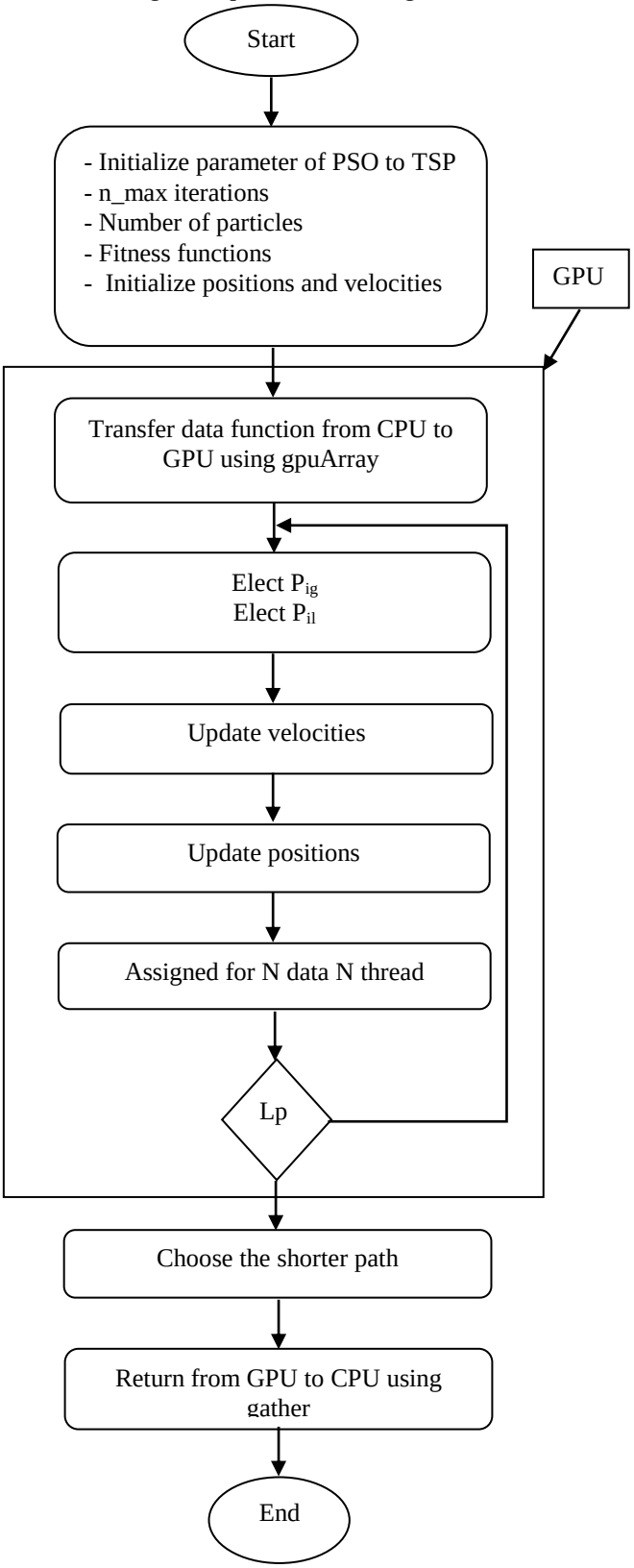

Figure 3. The process of GPU-PSO-A-TSP

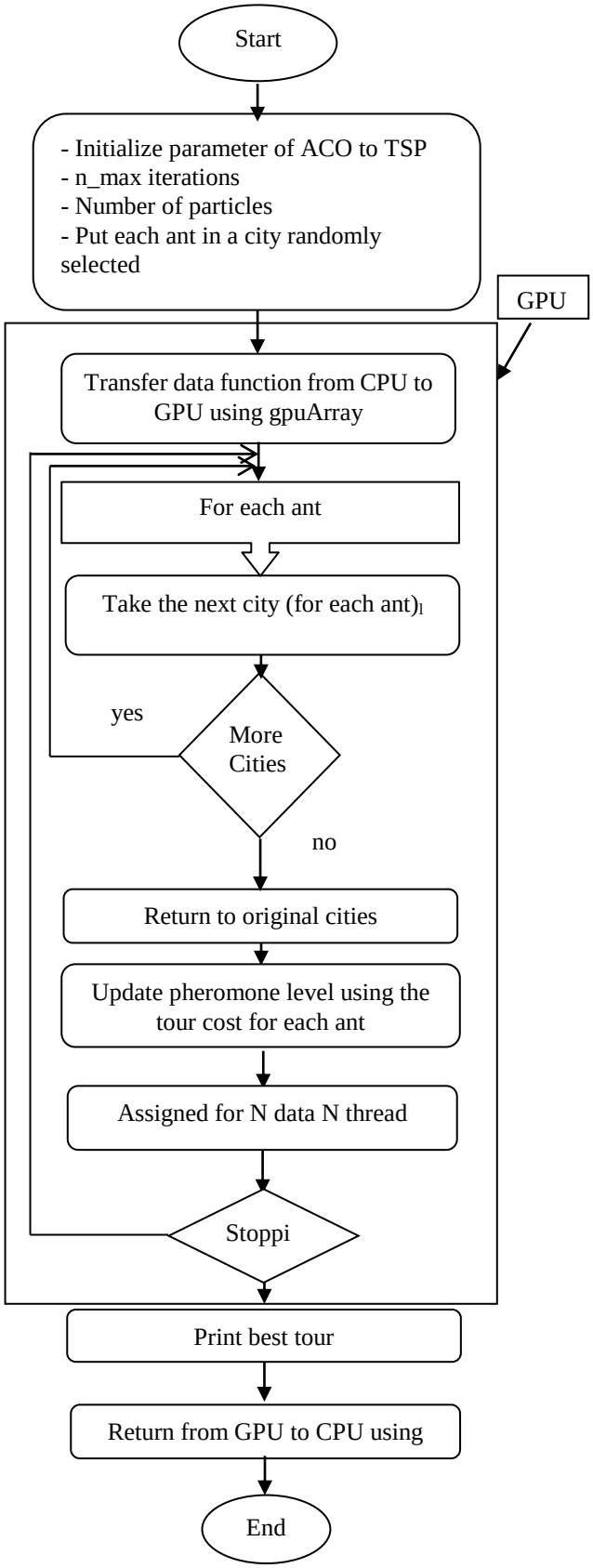

Figure 4. The process of GPU-ACO-A-TSP

The proposed algorithm reduces the likelihood that minimizes falling into local optimum and improves the capacity and research accuracy. Definite steps that were followed to the end to keep the strength of our approach.

First, we start by resetting the search space, then the number of iterations for PSO and its cost function. The second step is to initialize the particle ant, the cost of this function. Thereafter we use the function "gpuArray()". Then follows the steps of the ACO or PSO. We try to change the position and speed of PSO particles, and Ant positions too. If the conditions are fulfilled, we arrive at our goal, which is the end, otherwise we return to the second step.

With GPU, displays the processing steps of the two algorithms ACO & PSO algorithm in which we can identify the various stages of operation of the algorithm.

VI. Experimental Results

In our approach, we begin by resetting the parameters feature of PSO applied to the TSP; to say the number of nodes contained in a graph, a weight of every particle and the coefficients of acceleration. In the proposed vehicles tracking system security, the nodes represent the departure and arrival cities and the particles represent the vehicles in roads. The maximum number of iterations in this case was taken as 1000 iterations. The number of used particles depends on the graph. For example, for a graph with 22 cities, the possible number of particles of the population to use in this first test is 22, the second test is 70 for the third test is 100.

We had 1000 iteration because we have tests by using 2000 and 3000 iterations.

Our method consists in using 1000 iterations because the number and the time of the cycle are proportional among iterations to avoid the wasting

Our approaches GPU Optimization Swarm of particles and Ant Colony Optimization applied to the TSP (GPU-PSO ACO-A-TSP) is coded in Matlab 2014b and executed on a processor Intel ® Core T i7-4700MQ (6MB cache memory, 3.40 GHz) PC with memory of 12GB and NVIDIA GeForce 480M and Windows 7. There is many parameters used for our approach. The size of the population, which we are three times going to increase, is the number of knots of the social and cognitive probability, having c1 and c2, defined as $c1 = c2 = 2$. The mass of inertia w is taken as 0.9 and the maximum of the speed live taken as 100 and the dimension of the space as 10. Every cycle of TSP is executed during five replications and 1000 iterations. Both has and ß control the relative importance of pheromone trail and the distance between cities TSP where has $\alpha = 1.5$, $ß = 2$ Refers to the speed of pheromone evaporation $\rho = 0.7$. Each test TSP is performed for 5 replications iterations and 1000.

A. TSP solved by PSO and GPU PSO

To solve the TSP using PSO, the following processing can steps can be used. A particle is the position of a node, the nodes represent a city, and the route is the path between city (i) and city (j). The fitness functions are the distances between nodes; and finally, the ultimate multi-objective purpose, which is to minimalize the distances of the paths that relate and time [17].

Because of the large number of vehicles on the road and the large number of cities, the use of GPU acceleration with TSP is essential to delivery of messages to vehicles in real time to assure their security.

In Table 1, N refers to the number of nodes, CPU-T.PSO

refers to the best time for PSO (per seconds) and CPU-L.PSO refers to CPU-PSO the length for PSO. The same indices for GPU.

In the same table, we have tried to represent the different numbers of nodes, after that, we have attempted to increase the population of PSO keeping the same number of nodes. Thus, It was found from number 22 to 70 nodes, that the route of the shortest path decreases when the number of PSO population increases. Therefore, when the execution time increases, the number of (CPU-GPU) PSO population increases too. Therefore, we found it necessary to use TSP-ACO to improve results.

B. Comparative study with CPU and GPU for PSO

In Figure 5, the x-axis indicates the size of the population of PSO, The y-axis represents the execution time per seconds.

When selecting the number of nodes 22, 29, 30, 48, 52, 70 and the number of the population size PSO 22, 70, 100, it has been proven that the computation time decrease when executing the GPU-PSO.

In Figure 6, the x-axis shows the number of nodes and population, the y-axis shows the execution time per seconds. There after the implementation rate ((GPU T.PSO) / (CPU T.PSO)) *100 is very important, to choose for example, nodes 22 and 22 population, the rate is 60.31%.

In Figure 7, the x-axis shows the number of nodes and population, the y-axis shows the execution time per seconds.

While selecting the number of nodes as equal to the number of population, it was found that the execution time of the T.PSO -GPU compared to T.PSO-CPU decreases.

In Figure 8, the x-axis shows the number of nodes and population, the y-axis shows the execution time per seconds. The rate of execution ((GPU T.PSO) / (CPU T.PSO)) *100 is very important, as an example, for 29 nodes and 29 population the rate is 56, 11%.

In Figure 9, the x-axis shows the nodes, the y-axis shows the execution time per seconds.

While selecting the number of nodes 22, 29, 30, 48, 52, 70 and the number of the size of population of PSO 70, 100, it has been verified that execution of the GPU PSO decreases compared to the execution time of CPU PSO.

In Figure 10, the x-axis shows the nodes, the y-axis shows the execution time per seconds. The rate of execution ((GPU PSO) / (CPU PSO)) * 100 is very important, as an example, for 22 nodes and 70 population the rate is 61, 26%.

C. TSP solved by ACO and GPU ACO

To solve the TSP using ACO, the following processing can be used. A particle is the position of a node, the nodes represent a city, and the route is the path between node (i) and (j). The adapting functions are the distances between nodes; and the ultimate multi-objective function is to minimize the distances of the paths that are related to time.

In Table 2, N refers to the number of nodes, T.ACO refers to the best Time for ACO (per seconds), L.ACO is the best Length for ACO. We have not used the length of GPU in Table 2, because the same values are replicated.

In order to accelerate finding secure roads path for vehicles and to reduce road accident we propose to use TSP based GPU ACO.

In Table 2, we have tried to represent the different numbers of nodes. After that we have tried to increase the number of people of ACO keeping the same number of nodes. In this Table it was found that the route of the shortest path decreases when the number of ACO population increases. When the execution time increases the number of population increases too. The execution time depends on the complexity of the TSP as well.

Now comparing the results of the two tables we notice that the results of the shortest path of (CPU-GPU) ACO are better compared to the (CPU-GPU) PSO, but the best performance is that of (CPU-GPU) PSO compared to that of (CPU-GPU) ACO time.

D. Comparative study with CPU and GPU for ACO

In Figure 11, the x-axis indicates the size of the population of ACO, The y-axis represents the execution time per seconds.

When selecting the number of nodes 22, 29, 30, 48, 52,70 and the number of the population size ACO 22, 70, 100, it has been proven that when executing the GPU ACO a time decreases.

The results in Figure11 show that the running time of the GPU ACO less than the time of the execution by CPU ACO. For a number of nodes 22 and population size 30, the CPU-T. ACO is 0, 3724 while the GPU-T. ACO is 0.2933. These results prove that we have a very import gain time using the GPU.

E. TSP solved by our approach

Figure 12, present a set of probable results by using our method, giving to the similar number of nodes and the configuration of the problem. This figure represents the best TSP for our approach chosen.

In Figure 13, the x-axis indicates the size of the population of ACO, The y-axis represents the execution time per seconds.

When selecting the number of nodes 100 and the number of the population size of ACO 22, 29, 30, 48, 52, 70, it has been proven the Average length ACO of decrease

The results in figure13 show that average length ACO (100) less than the average length by PSO. For a number of nodes 30 and population size, the Average length ACO is 508.8037 Km while the average length PSO is 563.9448 Km.

In Table 3 we prove that the average length of ACO is below the average length of PSO.

VII. Conclusions, Discussions and further works

In this paper was given two approaches the GPU-PSO–A-TSP and GPU-ACO–A-TSP. Our system is applied to vehicles traffic to increase road security and reduce accidents. PSO is a heuristic continuous ACO is a discrete, MOPSO is PSO multi-objective, all these algorithms belong to the heuristics family while GPU-PSO is meta-heuristic applied to a set of TSP configurations at 22 to 70 nodes, ACO has a shorter length compared to PSO. In 22 cities, the average length of CPU-PSO (90.20), while CPU-ACO gives (77.33) represented

in table 3, the same constraints are made for different size nodes (see Table 1 and 2 in accordance with the nodes 22-70).

According to the results explained above in the two tables (table 1 and table 2) we note that the time of GPU-PSO and better than the GPU-ACO time in all cases. But, for the length that is the opposite that is to say that GPU ACO length is smaller relative to the length of GPU PSO. For this reason, we will try to achieve hybridization between PSO and ACO using the GPU based on [25].

Execution time of the proposed system is critical for the safety of vehicles that why we have proposed to use GPU acceleration to assure finding secure paths and sending messages to vehicles in appropriate time.

In a typical scheme, GPU is applied repeatedly with the same settings and the same initialization of ants. Then the best test results are supposed to be the solution of the problem. With GPU-PSO the execution time is improved compared to the execution time with GPU-ACO.

Acknowledgment

The authors would like to acknowledge the financial support of this work by grants from General Direction of Scientific Research (DGRST), Tunisia, under the ARUB program.

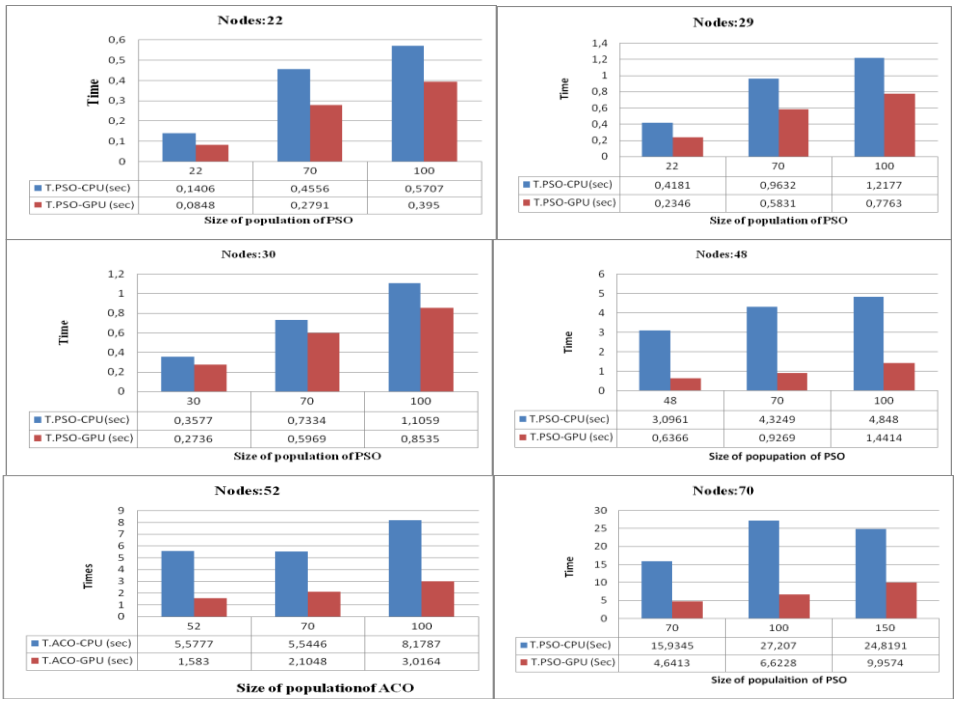

Figure 5. CPU-PSO and GPU-PSO comparative for different nodes (22, 29, 30, 48, 52 and 70)

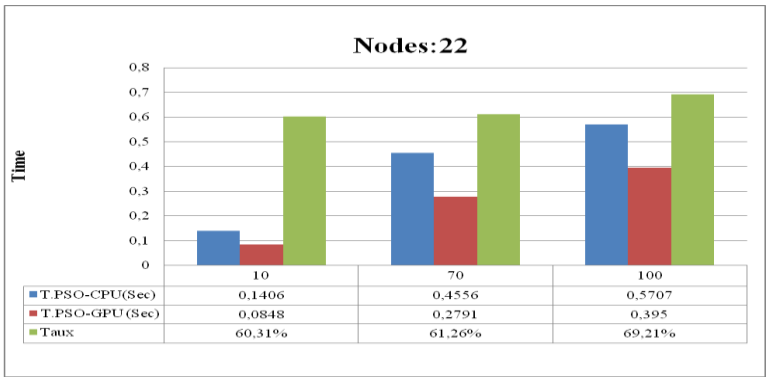

Figure 6. Taux= ((GPU T.PSO) / (CPU T.PSO))*100 for number nodes 22 and number population (22, 70, 100)

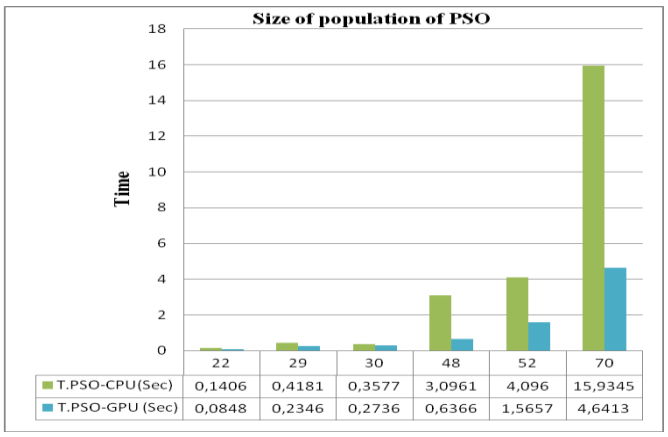

Figure 7. CPU PSO and GPU PSO comparative for different the number of nodes = the number of size of population of PSO

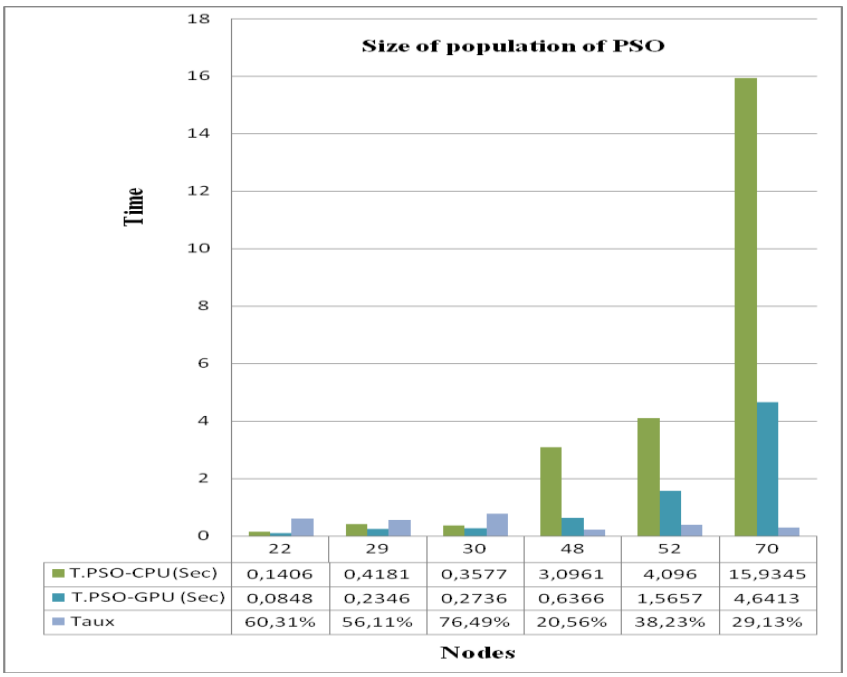

Figure 8. Taux= ((GPU T.PSO) / (CPU T.PSO))*100 for number nodes 22, 29, 30, 48, 52 and 70

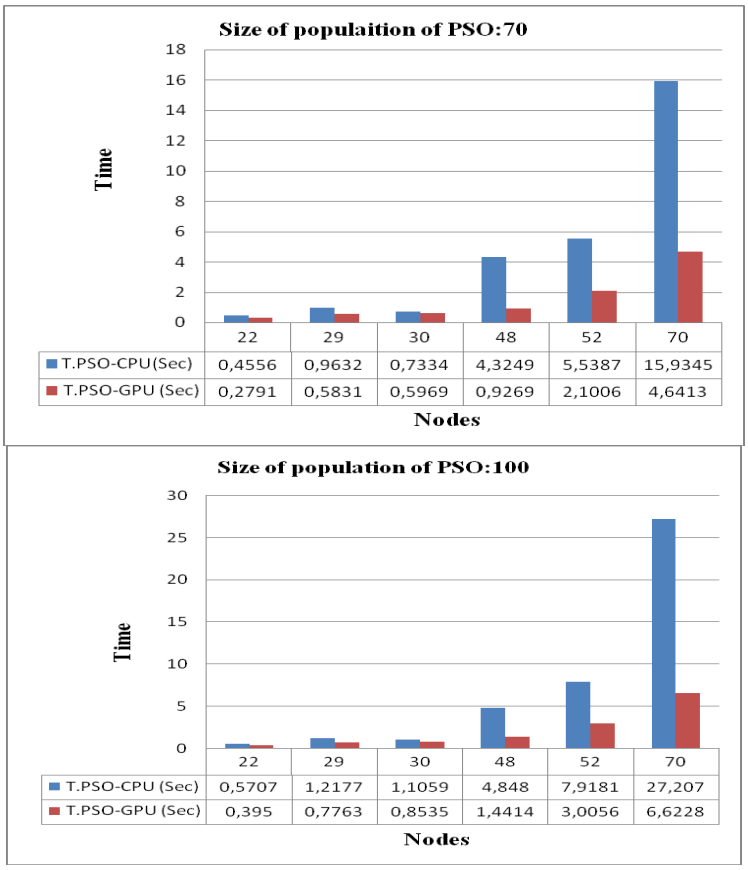

Figure 9. CPU T.PSO and GPU T.PSO comparative for different number of size of population of PSO (70 and 100)

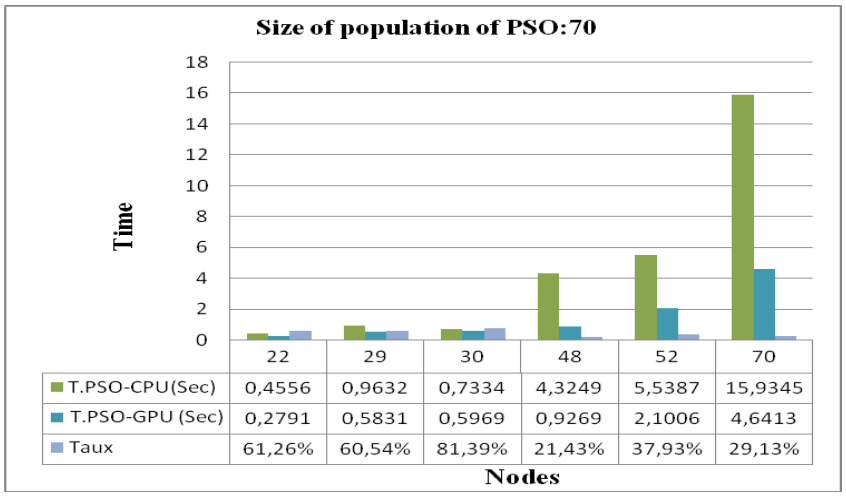

Figure 10. Taux= ((GPU T.PSO) / (CPU T.PSO))*100 for number nodes 22 and number population 70

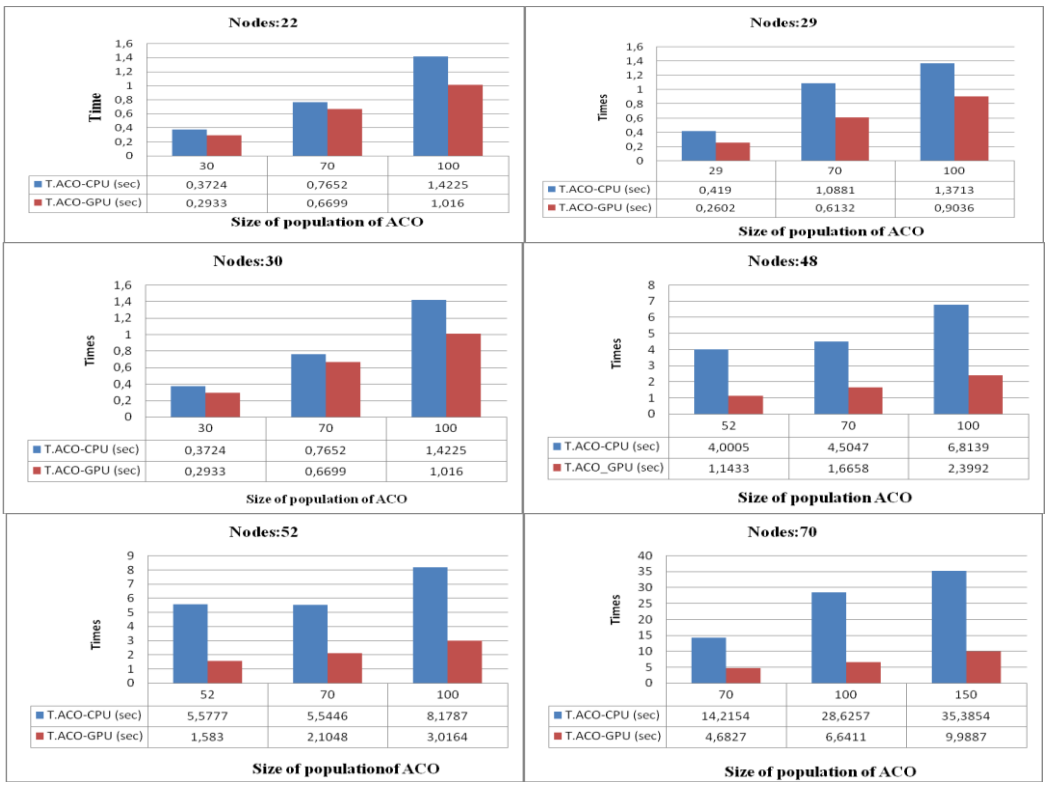

Figure 11. CPU ACO and GPU ACO comparative results for nodes number of 22, 29, 30, 48, 52 and 70

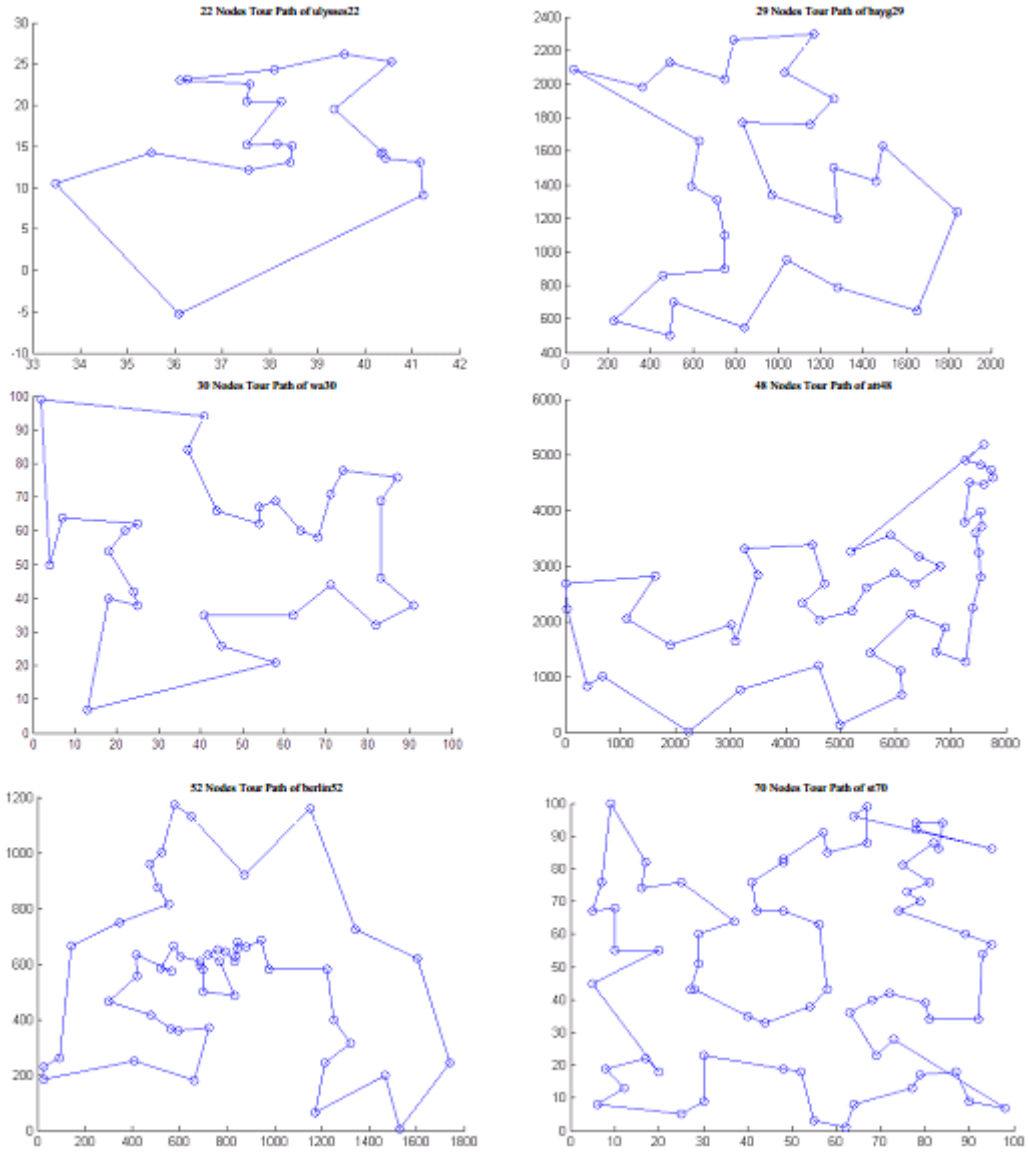

Figure 12. Results of best GUP-PSO and GPU-ACO applied to TSP

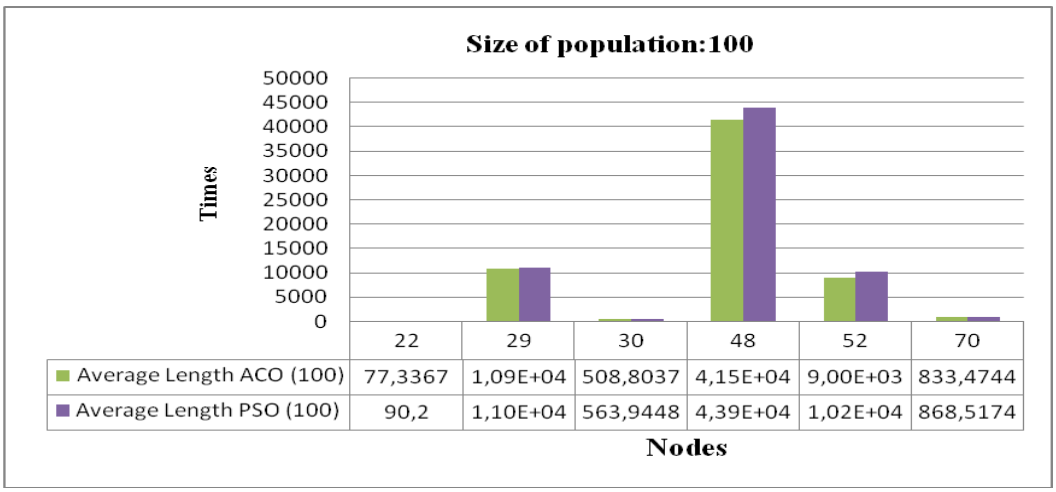

Figure 13. Average length ACO (100) and average length PSO (100) comparative results for 22, 29, 30, 48, 52 and 70.

Table 1. PSO and GPU PSO for TSP

		CPU		GPU	
N	Size of population of PSO	T.PSO (sec)	L.PSO (Km)	T.PSO (sec)	L.PSO (Km)
22	22	0.1406	90.6884	0.0848	90.6884
	70	0.4556	90.4220	0.2791	90.4220
	100	0.5707	89.4898	0.3950	89.4898
29	29	0.4181	1.1761e+004	0.2346	1.1761e+004
	70	0.9632	1.0900e+004	0.5831	1.0900e+004
	100	1.2177	1.0472e+004	0.7763	1.0472e+004
30	30	0.3577	584.0341	0.2736	584.0341
	70	0.7334	562.0160	0.5969	562.0160
	100	1.1059	545.7844	0.8535	545.7844
48	48	3.0961	4.5973e+004	0.6366	4.5973e+004
	70	4.3249	4.4654e+004	0.9269	4.4654e+004
	100	4.8480	4.1158e+004	1.4414	4.1158e+004
52	52	4.0960	1.0404e+004	1,5657	1.0404e+004
	70	5.5387	1.0191e+004	2,1006	1.0191e+004
	100	7.9181	1.0099e+004	3,0056	1.0099e+004
70	70	15.9345	910.6275	4,6413	910.6275
	100	27.2070	848.9480	6,6228	848.9480
	150	24.8191	845.9768	9,9574	845.9768

Table 2. ACO and GPU ACO for TSP

		CPU		GPU
N	Size of population of ACO	T.ACO (sec)	L.ACO (km)	T.ACO (sec)
22	22	0.2438	77.8000	0.0930
	70	0.4969	77.1834	0.2964
	100	0.6866	76.1212	0.4223
29	29	0.4190	1.1621e+004	0.2602
	70	1.0881	1.0530e+004	0.6132
	100	1.3713	1.0432e+004	0.9036
30	30	0.3724	537.9874	0.2933
	70	0.7652	495.5985	0.6699
	100	1.4225	491.7651	1.0160
48	48	4.0005	4.2086e+004	1,1433
	70	4.5047	4.1420e+004	1,6658
	100	6.8139	4.0585e+004	2.3992
52	52	5.5777	9.0578e+003	1,5830
	70	5.5446	9.0068e+003	2,1048
	100	8.1787	8.9425e+003	3,0164
70	70	14.2154	855.4880	4.6827
	100	28.6257	834.5480	6,6411
	150	35.3854	810.3874	9,9887

Table 3. Comparative for a need number of (100)

N	Size of population	Average Length ACO (100)	Average Length PSO (100)
22	100	77.3367	90.2000
29	100	1.0910 e+004	1.1044e+004
30	100	508.8037	563.9448
48	100	4.1502e+004	4.3928e+004
52	100	9.0023e+003	1.0231e+004
70	100	833.4744	868.5174

References

[1] N. Rokbani and A. M. Alimi. "IK-PSO, PSO Inverse Kinematics Solver with Application to Biped Gait Generation.International", Journal of Computer Applications, p 33-39, November 2012.

[2] Y. Shi and R. Eberhart. "A modified particle swarm optimizer", In Proc of the IEEE World Congress on Computational Intelligence and IEEE International Conference on Evolutionary Computation, p 69-73, 1998.

[3] M. Dorigo, M. Birattari, and T. Stutzle. "Ant colony optimization", IEEE Computational Intelligence Magazine, p 28-39, 2006.

[4] K. Vittori, G. Talbot, J. Gautrais, V. Fourcassié, A. F. Araujo, and G. Theraulaz. "Path efficiency of ant foraging trails in an artificial network", Journal of heoretical Biology, p 507-515, 2006.

[5] R. C. Eberhart and J. Kennedy. "A New Optimizer Using Particle Swarm Theory", In Proceedings of International Symposium on Micro Machine and Human Science, p 39-43, 1995.

[6] S. S. Kim, J. H. Byeon, H. Liu, A. Abraham and S. F. McLoone. "Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization", Soft Computing, p 867-882, 2013.

[7] M. Ali, M. Pant and A. Abraham. "Unconventional initialization methods for differential evolution", Applied Mathematics and Computation, p 4474-4494, 2013.

[8] M. Dorigo, V. Maniezzo and A. Colorni. "The ant system: Optimization by a colony of cooperating agents", IEEE Transaction System Man Cybern B, Cybern, p 29-41, 1996.

[9] S. Nonsiri and S. Supratid. "Modifying Ant Colony Optimization", IEEE Conference on Soft Computing in Industrial Applications, 2008.

[10] W. Elloumi, N. Rokbani and A. M. Alimi. "Ant supervised by PSO", In Proc of International symposium on Computational Intelligence and Intelligent Informatics, Egypt, cairo, p 161-166, 2009.

[11] P. Winker, P and M. Gilli. "Applications of optimization heuristics to estimation and modelling problems", Computational Statistics & Data Analysis, p 211-223, 2004.

[12] B. Yue, H. Liu and A. Abraham. "Dynamic Trajectory and Convergence Analysis of Swarm Algorithm", Computing and Informatics, p 371-392, 2012.

[13] B. Iglesia, A. Reynolds, and V. J. R. Smith. "Developments on a multi-objective metaheuristic (momh) algorithm for finding interesting sets of classification rules", In: C. A. C. Coello, A. Hern´andez Aguirre and E. Zitzler (eds.) EMO, Springer, Heidelberg, p 826-840, 2005.

[14] T. Vigneswari, M. Mohamed. "Optimal Grid Scheduling Using Improved Artificial Bee Colony Algorithm ", International Journal of Computer, Electrical, Automation, Control and Information Engineering, p 2055-2063, 2014.

[15] M. Dorigo and L. M. Gambardella. "Ant colony system: A cooperative learning approach to the travelling salesman problem", IEEE Transactions on Evolutionary Computation, 1997.

[16] B. Gavish and S. C. Graves. "The travelling salesman problem and related problems", 1978.

[17] W. Elloumi and A. M. Alimi."Combinatory Optimization of ACO and PSO", International Conference on Metaheuristique and Nature Inspired Computing, Tunis, Hammamet, p 1-8, 2008.

[18] M. Reimann and M. Laumanns. "A hybrid aco algorithm for the capacitated minimum spanning tree problem", Proceedings of first international workshop on hybrid metaheuristics, 2004.

[19] W. Elloumi and A. M. Alimi. "A More Efficient MOPSO for Optimization", The eight ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Hammamet, p 1-7, 2010.

[20] C. Coello. "Evolutionary multiobjective optimization: A historical view of the field", IEEE Computational Intelligence Magazine, p 28-36, 2006.

[21] R. Thangaraj, M. Pant, A. Abraham and P. Bouvry. "Particle Swarm Optimization: Hybridization Perspectives and Experimental Illustrations", Applied Maths and Computation, Elsevier Science, Netherlands, p 5208-5226, 2011.

[22] Y. Maheshkumara, V. Ravi and A. Abraham. "A Particle Swarm Optimization Threshold Accepting Hybrid Algorithm for Unconstrained Optimization", Neural Network World, 2013.

[23] H. Liu, A. Abraham, V. Snasel and S. McCloone. "Particle Swarm Scheduling for Work-Flow Applications in Distributed Data-Intensive Computing Environments", Information Sciences, Elsevier Science, Netherlands, p 228-243, 2012.

[24] W. Elloumi, N. Baklouti, A. Abraham and A. M. Alimi. "The Multi-Objective Hybridization of Particle Swarm Optimization and Fuzzy Ant Colony Optimization", Journal of Intelligent and Fuzzy Systems, 27(1) p 515-525, 2014.

[25] W. Elloumi, H. ElAbed, A. Abraham and A. M. Alimi. "A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP". Journal Applied Soft Computing 25: 234-241, 2014.

[26] P. Krömer, J. Platoš·and V. Snášel. "Nature-Inspired Meta-Heuristics on Modern GPUs: State of the Art and Brief Survey of Selected Algorithms". International Journal of Parallel Programming, p 681-709, 2014.

[27] P. Krömer, V. Snášel, J. Platoš and A. Abraham. "Many-threaded implementation of differential evolution for the CUDA platform". Proceedings of the 13th annual conference on Genetic and evolutionary computation, p 1595-1602, 2011.

[28] P. Krömer, J. Platos and V. Snášel. "Differential evolution for the linear ordering problem implemented on CUDA". IEEE Congress on Evolutionary

Computation (CEC), p 796-802, 2011.
[29] M. Panda, A. Abraham and M. R. Patra, "Hybrid Intelligent Systems for Detecting Network Intrusions". Journal of Security and Communication Networks, 8(16): 2741-2749, 2015.
[30] A. Abraham, C. Grosan and C. M. Vide. "Evolutionary Design of Intrusion Detection Programs", International Journal Network Security, p 328-339, 2007.
[31] B. K. Chaurasia, S. Verma, G. Tomar and A. Abraham. "Optimizing Pseudonym Updation in Vehicular Ad-Hoc Networks", Transactions on Computational Science, Special Issue on Security in Computing, Springer Berlin / Heidelberg, p 136-148, 2009.

Author Biographies

Olfa Bali Received the BS degree in computer science from the Faculty of Sciences of Sfax-Tunisia (FSS) in 2005 and Master's degrees in Computer Science from the National School of Engineers of Sfax - Tunisia (ENIS), in 2007. In September 2008, she joined the Sfax University (USS), where she was Assistant Professor Computer analyst at the Higher Institute of Nursery Sciences of Sfax, Tunisia.

Walid Elloumi Received the B.S. degree in computer science from The Faculty of Sciences of Sfax-Tunisia (FSS) in 2003, and the Master degrees in Computer Science from the National Engineering School of Sfax - Tunisia (ENIS), in 2005. In September 2005, he joined the Sfax University (USS), where he was an assistant professor in the Department of Computer science of the Preparatory Institute being studied of Engineers of Sfax (IPEIS). Between 2006 and 2009 he was an assistant professor in the Department of Computer science of Institute Higher Electronic Communication of Sfax (ISECS) and for the year 2009 2010 he served in National Engineering School of Sfax (ENIS).

In 2010 he joined Gabes University where he is currently an Assistant professor on Computer science on the High Institute of Industrial Systems of Gabes (ISSIG), Tunisia, Department of Computer science. He is member of the REsearch Group on Intelligent Machines (REGIM). His research interests include Computer Vision, pattern recognition and Swarm Intelligence. These research activities are centered now on Swarm Intelligence. He is an IEEE member. He is member of the Organizing Committee of ACIDCA-ICMI'05, he participated in: IEEE/CIS Distinguished lecturers 2009, Training in English "Writing Reasearch Articles" 2008, Training in English "Listening For Academic settings" 2007, School of Spring on the Systems of Management of Quality for Teaching, Industry and the Services (SYMAQ) 2007, School of Winter on the Valorisation of the Innovation by the Patents (EHVIB) 2006, School of UNIX formation/LINUX 2006, School of Winter on wavelets 2005, Cycle of formation teaching 2005, University spring (time) of teaching 2004.

Ajith Abraham is the Director of Machine Intelligence Research Labs (MIR Labs), a Not-for-Profit Scientific Network for Innovation and Research Excellence connecting Industry and Academia. The Network with Head quarters in Seattle, USA has currently more than 1,000 scientific members from over 100 countries. He also works as a Research Professor in VSB-Technical University of Ostrava, Czech Republic.

As an Investigator / Co-Investigator, he has won research grants worth over 100+ Million US$ from Australia, USA, EU, Italy, Czech Republic, France, Malaysia and China. Dr. Abraham works in a multi-disciplinary environment involving machine intelligence, cyber-physical systems, Internet of things, network security, sensor networks, Web intelligence, Web services, data mining and applied to various real world problems. In these areas he has authored / coauthored more than 1,000+ research publications out of which there are 100+ books covering various aspects of Computer Science. One of his books was translated to Japanese and few other articles were translated to Russian and Chinese. About 800+ publications are indexed by Scopus and over 450 are indexed by Thomson ISI Web of Science. Some of the articles are available in the ScienceDirect Top 25 hottest articles. He has 700+ co-authors originating from 40+ countries. Dr. Abraham has more than 20,000+ academic citations (h-index of 70 as per google scholar). He has given more than 100 plenary lectures and conference tutorials (in 20+ countries). For his research, he has won seven best paper awards at prestigious International conferences held in Belgium, Canada Bahrain, Czech Republic, China and India. Since 2008, Dr. Abraham is the Chair of IEEE Systems Man and Cybernetics Society Technical Committee on Soft Computing (which has over 200+ members) and served as a Distinguished Lecturer of IEEE Computer Society representing Europe (2011-2013. Under his direct academic supervision, 8 students received Ph.D. degrees and is currently supervising 10 Ph.D. students in different Universities in Europe, USA, Africa and India. He has examined over 100 Ph.D. theses. Currently Dr. Abraham is the editor-in-chief of Engineering Applications of Artificial Intelligence (EAAI) and serves/served the editorial board of over 15 International Journals indexed by Thomson ISI. He is actively involved in the organization of several academic conferences, and some of them are now annual events. Dr. Abraham received Ph.D. degree in Computer Science from Monash University, Melbourne, Australia (2001) and a Master of Science Degree from Nanyang Technological University, Singapore (1998). More information at: http://www.softcomputing.net/

Adel M. Alimi He graduated in Electrical Engineering in 1990. He obtained a Ph.D. and then an HDR both in Electrical Computer Engineering in 1995 and 2000, respectively. He is full Professor in Electrical Engineering at the University of Sfax, ENIS, since 2006.

Prof. Alimi research interests include:

- iBrain (Evolutionary computing, Swarm intelligence, Artificial immune systems, Fuzzy Sets, Uncertainty analysis, Fractals, Support vector machines, Artificial neural networks, Case Based, Reasoning, Wavelets, Hybrid intelligent systems, Nature inspired computing techniques, Machine learning, Ambient intelligence, Hardware implementations, Multi-Agent Systems, iRobotics, Multi-Robots, Autonomous Robots)
- iPerception (Classification, Classifiers Combination, Features Extraction, High Dimension Classification Problems, Handwriting Recognition, Handwriting Modeling, Motor Control, Perception, OCR, Arabic Script, Historical Documents Analysis, iDocument, iVision, Remote Sensing, Affective Computing) & iData (Big Data, iWeb, Biometry, Data Mining, Web Mining, Cloud Computing).