Volume 4, Issue 3, March 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Metaheuristics for Communication Protocols: Overview and Conceptual Comparison

Manish Suroliya*, V. S. Dhaka, Ramesh C. Poonia
School of Engineering & Technology,
Jaipur National University, Jaipur - India

Abstract—In the recent years, Vehicular Ad hoc Networks (VANETs) became one of the most interesting research area in the field of Mobile Ad hoc Networks (MANET). Communication has become very important for exchanging information between people from and to anywhere at any time in any form. Metaheuristic algorithms play an important role in communication protocols configuration with the aim of optimizing transmission time. These algorithms are generic methods which offer good solutions, even global optimum, within a reasonable computing time. This paper is aimed at establishing the foundations needed to understand the metaheuristic algorithms used to address optimization problems and a brief classification of these techniques and their conceptual comparison are discussed.

Keywords— VANETs, PSO, Metaheuristics, SA, GA, ES, DE

I. INTRODUCTION

Optimization problems [1] consist of the search for a best configuration of a set of variables to get some goals. Generally, optimization problems can be seen as the search for the best solution, or at least good enough, to a given problem. Currently they are using optimization techniques in fields as diverse as networking, economics or logistics. Humans are constantly solving optimization problems, such as what route to take to get from one place to another, how to organize our schedule. As these problems have small dimension, we can process them easily by using our brain. However, when the optimization problems become larger and more complex it appears the necessity of using specialized tools and the computational power of computers. Metaheuristic optimization deals with optimization problems using metaheuristic algorithms. Optimization is essentially everywhere, from engineering design to economics and from holiday planning to Internet routing. As money, resources and time are always limited, the optimal utility of these available resources is crucially important. Most real-world optimizations are highly nonlinear and multimodal, under various complex constraints. Different objectives are often conflicting. Even for a single objective, sometimes, optimal solutions may not exist at all. In general, finding an optimal solution or even sub-optimal solutions is not an easy task.

To solve the optimization problem [2], efficient search or optimization algorithms are needed. There are many optimization algorithms which can be classified in many ways, depending on the focus and characteristics. If the derivative or gradient of a function is the focus, optimization can be classified into gradient-based algorithms and derivative-free or gradient-free algorithms. Gradient-based algorithms such as hill-climbing use derivative information, and they are often very efficient. Derivative-free algorithms do not use any derivative information but the values of the function itself. Some functions may have discontinuities or it may be expensive to calculate derivatives accurately, and thus derivative-free algorithms such as Nelder-Mead downhill simplex become very useful.

Obtaining optimal protocol configurations by using exact techniques is practically impossible due to the enormous number of possible configurations. Therefore, the use of automatic intelligent tools is required to face these problems. Metaheuristic algorithms emerged as efficient techniques able to tackle complex optimization problems. Indeed, these algorithms present successful performance dealing with multitude of engineering problems [3], [4]. Unfortunately, the use of metaheuristics in VANETs is still limited, and just a few a related approaches can be found in the literature

II. RELATED WORK

From the 1940s to 1960s, heuristic methods have been used in various applications, but the first landmark came with the advent of evolutionary algorithms. In 1963 Ingo Rechenberg and Hans-Paul Schwefel, then both at the Technical University of Berlin, developed evolutionary strategies while L. J. Fogel et al. developed evolutionary programming in 1966. Genetic algorithms were developed by J. Holland in the 1960s and 1970s, though his seminal book on genetic algorithms was published in 1975. The 1980s and 1990s were the most exciting time for metaheuristic algorithms. One big step was the development of simulated annealing (SA) in 1983, an optimization technique, pioneered by S. Kirkpatrick et al., and inspired by the annealing process of metals. Another important step was the development of artificial immune systems by Farmer et in 1986. The use of memory in metaheuristics was pioneered by Glover in Tabu search in the 1980s, though his seminal book on Tabu Search was published later in 1997. Search moves are recorded in a Tabu list, and future moves should try to avoid revisiting previous solutions. In 1992, Marco Dorigo finished his PhD

thesis on optimization and natural algorithms, in which he described his innovative work on ant colony optimization (ACO). This search technique was inspired by the swarm intelligence of social ants using pheromone as a chemical messenger. Later in 1995, another significant progress was the development of particle swarm optimization by James Kennedy and Russell C. Eberhart. Around 1996 and later in 1997, R. Storn and K. Price developed their vector-based evolutionary algorithm, called differential evolution (DE). This algorithm proved to be more efficient than genetic algorithms in many applications. At the turn of the 21st century, things became even more exciting. Firstly, Zong Woo Geem developed the harmony search (HS) algorithm in 2001, which is a music-inspired algorithm. Around 2002, a bacteria foraging algorithm was developed by Passino.

In 2004, S. Nakrani and C. Tovey proposed the honey bee algorithm and its application for optimizing Internet hosting centers, which was followed by the development of a novel bee algorithm by D. T. Pham et al. in 2005 and the artificial bee colony (ABC) by D. Karaboga in 2005. In 2008, the author of this article developed the firefly algorithm (FA). In 2009, Xin-She Yang and Suash Deb introduced an efficient cuckoo search (CS) algorithm (Yang and Deb, 2009; Yang and Deb, 2010), and it has been demonstrated that CS is far more effective than most existing metaheuristic algorithms including particle swarm optimization.

III. METAHEURISTIC ALGORITHMS CLASSIFICATION

A brief classification of these techniques is shown in Figure 1.The more general classification which divides these techniques into exact and approximate. The exact methods, which are based on the mathematical extraction of the optimal solution, or an exhaustive search until the optimum is found, guarantee the optimality of the solution obtained. These techniques present some drawbacks, however. The time they require, though bounded, is generally very large, especially for NP-complex problems. Furthermore, it is not always possible to find such an exact technique for every problem. This makes exact techniques not to be the right choice in many occasions, since both their time and memory requirements can become unreasonably high for large problems. Therefore, approximate methods have been often used by the research community in the last few decades. These methods sacrifice the guarantee of finding the optimum in favor of providing some satisfactory solution within reasonable time.

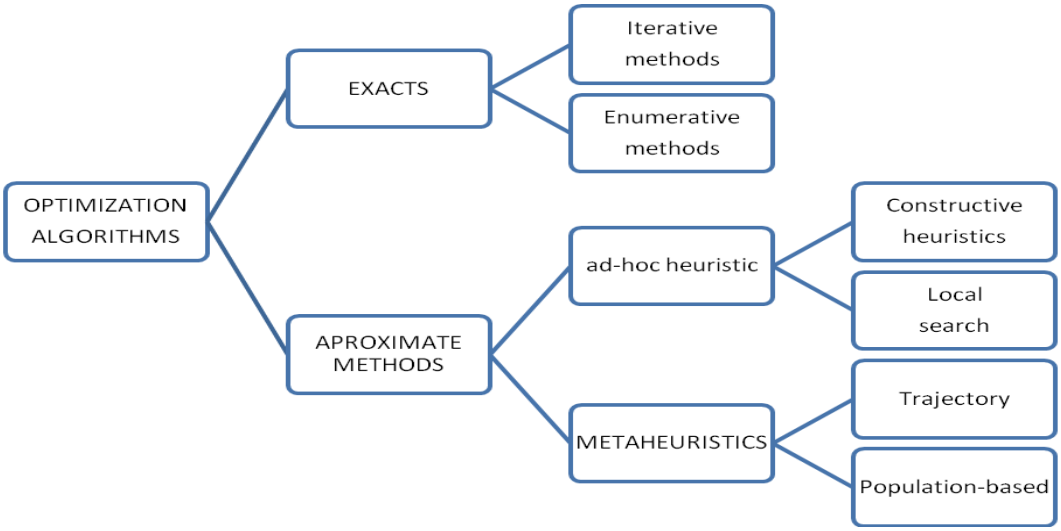

Figure 1: Optimization techniques classification.

Among approximate algorithms, one can find two types: ad hoc heuristics and metaheuristics. We focus this paper on the latter. Ad hoc heuristics can in turn be divided between constructive heuristics and local search methods. Constructive heuristics are usually the swiftest methods. They construct a solution from scratch by iteratively incorporating components until a complete solution is obtained, which is returned as the algorithm output. Finding some constructive heuristic can be easy in many cases, but the obtained solutions are of low quality. In fact, designing one such method that actually produces high quality solutions is a nontrivial task, since it mainly depends on the problem, and requires thorough understanding of it. For example, in problems with many constraints it could happen that many partial solutions do not lead to any feasible solution.

Finally, three decades ago came a new stream research within the approximate algorithms. It basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. These methods are called metaheuristics, but they were often called modern heuristics. To this type of algorithms belong the tabu search (TS), genetic algorithms (GA), differential evolution (DE), and simulated annealing (SA), among others. Since its emergence the term metaheuristic has been defined in different ways. Osman and Laporte proposed a widely used definition in 1996 [5].

There are different ways to define metaheuristics taxonomy. Depending on the characteristics selected to differentiate among them, several classifications are possible, each of them being the result of a specific point of view. Next, we briefly summarize the most important ways of classifying metaheuristics [6]:

- Nature inspired (bio-inspired) vs. non-nature inspired: Generally, it is the most natural way to classify metaheuristics, since it is based on the origins of the algorithm. It takes into account whether their models have been inspired by Nature or not. There are bio-inspired algorithms, like Genetic Algorithms (GA) and Ant Colony Algorithms (ACO), and non nature-inspired ones such as Tabu Search (TS) and Iterated Local Search (ILS). This classification is not very meaningful since the emergence of hybrid algorithms.
- Population-based vs. single point search (trajectory): In this case, the characteristic used for the classification is the number of solutions used at the same time. On the one hand, single point search algorithms work on a single solution describing a trajectory in the search space during the search process. They encompass local search-based metaheuristics, like Variable Neighborhood Search (VNS), Tabu Search (TS), and Iterated Local Search (ILS). On the other hand, population-based methods work on a set of solutions (points) called population. Figure 1.1 shows graphically some examples of this taxonomy.
- Static vs. dynamic objective function: The algorithms which keep the objective function given in the problem during the whole process are called metaheuristics with static objective function. However, there are other algorithms with dynamic objective function, like Guided Local Search (GLS), which modify the fitness function during the search, incorporating information collected during the search process to escape from local optimum.

Other characteristics can be used for classification of different metaheursitics, like the number of neighborhood structures and the use or not of memory. In the following subsections, we present briefly some of the most representative metaheuristics grouping them into trajectory or population based algorithms.

IV. METAHEURISTIC ALGORITHMS FOR OPTIMIZATION

Many metaheuristic algorithms exist in literature and some algorithms have been discussed in detail in other papers such as swarm intelligence, ant colony optimization and particle swarm optimization. This paper will briefly introduce the most popular metaheuristic algorithms for optimization.

4.1 **Simulated Annealing (SA)**

Simulated annealing (SA) is a generic probabilistic metaheuristic algorithm. It is used for the global optimization problem of locating a good approximation to the global optimum of a given function in a large search space. Simulated Annealing was one of the first metaheuristic algorithms with an explicit strategy to escape from local minimum. The basic idea is to allow movements to select solutions worse than the current solution. The probability of performing such a movement decreases during the search process. The operation of a typical SA is summarized in Figure 2.

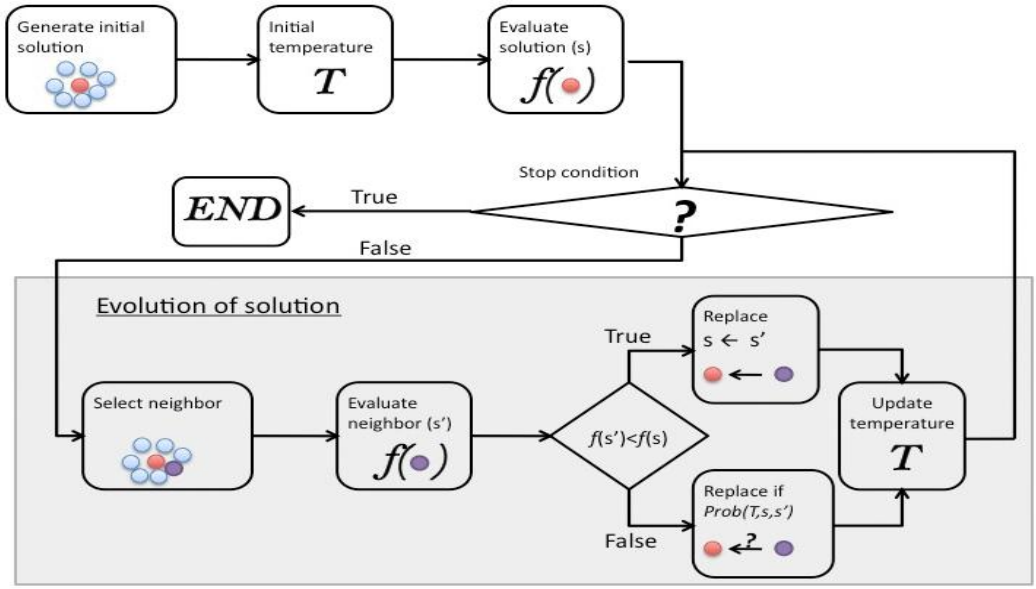

Figure 2: Operation of a typical SA.

The whole process starts by generating an initial solution s (randomly or using some heuristic) and starting the temperature parameter (T). The algorithm works iteratively keeping a single tentative solution s at any time. In every iteration, a new solution s' ϵ N(s) is generated from the previous one, and either replaces it or not depending on an acceptance criterion. The acceptance criterion works depending on the fitness values (f(s) and f(s')) and temperature T. s' replaces s as current solution if f(s) < f(s') (s' has better quality). However, in the case of f(s) ≥ f(s'), s' replaces it with probability prob (Equation 1.1). This probability depends on the difference between their quality (f(s') − f(s)) values and T. This acceptance criterion provides the way of escaping from local optima.

$$prob(T,s,s') = \frac{2}{1+e^{\frac{f(s')-f(s)}{T}}} \qquad (1.1)$$

The temperature value (T) is updated (generally decreasing) during the search. Thus, at the beginning the probability of accepting new solutions are high and it gradually decreases along the iterations, following a cooling schedule. The algorithm ends by selecting only those solutions that improve the current one (s). This process is based on annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions (a local optimum of the internal energy) and wander randomly through states of higher energy, the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. The algorithm is a result of the combination of two different processes: random step and iterative improvement. In the first phase, exploration of the search space becomes more random and erratic, producing movements to worst solutions. However, as it iterates, the erratic component gradually decreases the search converging to a optimum (local). The selection of solutions that do not improve the current one is controlled by two factors: the difference between the fitness functions and temperature (see Equation 1.1). First, if we fix the temperature, the higher the difference f(s') − f(s), less probability to move from s to s'. Moreover, the higher the temperature T, the more probability to accept s' as a new solution. The selection of a suitable cooling strategy is crucial for the behavior of the algorithm. The cooling strategy Q defines the temperature T for each moment k, $T_{k+1} = Q(T_k, k)$. The most used one is $T_{k+1} = \alpha \times T_k$, where $\alpha \in (0, 1)$. The cooling strategy and the initial temperature must be adapted to problems because the cost of escaping a local optimum depends on the structure of the search space. There are many ways leading to different variants of SA as: Fast SA (FSA), Very Fast SA (VFSA) or Adaptative SA (ASA) [7]. Ingber [8] proposed an empirical solution which starts with an initial random sampling of search space and T_0 is calculated from the average and the standard deviation of the values of the objective function. SA has been successfully applied to many optimization problems: the Quadratic Assignment Problem (QAP) [9] and the textitJob Shop Scheduling (JSS) [10]. In the field of wireless mobile networks, it has been used as an efficient method for the hand off mechanism in cellular networks [11] and problems of optimizing MANET routing protocols [12] or the clustering of the nodes [13].

4.2 Genetic Algorithm (GA)

Genetic Algorithms are the most widespread subclass of EAs. They were developed by John Holland [14]. As EA, it applies the natural selection principles to solve optimization problems. During successive generations, the solutions evolve to the optimum according to these principles. The evolution of these solutions depends largely on an adequate codification of them. GA work with a population of individuals, each of which represents a feasible solution to a given problem. Each individual is assigned a value associated with the goodness of this solution (fitness). Similarly, in Nature, the fitness could be seen as a value of the adaptation and the effectiveness to compete for certain resources for biological organisms. The higher individual adaptation to the problem, the grater the probability to be selected to breed, i.e., crossing its genetic material with another individual. This crossing produces new individual's descendants of the above ones, which share some characteristics of their parents. So, spread the genetic material of the best individuals in successive generations (see Figure 3). If the algorithm is designed properly, the population will converge to an optimum solution to the problem. The algorithm manipulates a collection p of individuals (the population), each of which comprises one or more chromosomes. These chromosomes allow each individual represent a potential solution for the problem under consideration. An encoding/decoding process is responsible for performing this mapping between chromosomes and solutions. Chromosomes are divided into smaller units termed genes. The different values a certain gene can take are called the alleles for that gene. The genes are encoded in a string of symbols (numbers or letters). There are various representations such as the use of binary numbers in BCD or Gray codes, the use of real or integer, etc. Most of the time, a correct coding is the key to a good resolution of the problem. Typically, codification is static, but in cases of numerical optimization, the number of bits dedicated to encode a parameter can vary.

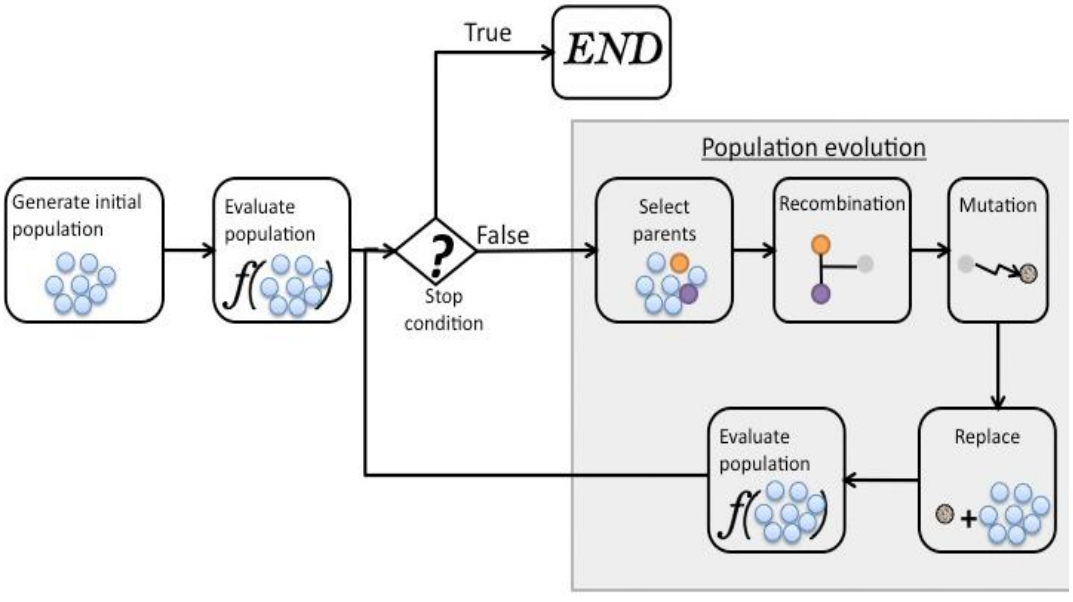

Figure 3: Operation of a typical GA.

The algorithm is an iterative process which starts by generating an initial population of solutions. This is typically addressed by randomly generating the desired number of solutions. When the alphabet used for representing solutions has low cardinality, this random initialization provides a more or less uniform sample of the solution space. Then, applying the genetic operators: selection, recombination, mutation, and replacement on this population.

4.3 Evolutionary Strategies (ES)

Evolutionary Strategies (ES) are metaheuristic algorithms designed to solve real parameterization optimization problems. They were presented in 1964 at the Technical University of Berlin (Germany) and developed in the 70s by Ingo Rechenberg and Hans-Paul Schwefel [15][16]. As common with evolutionary algorithms, they use mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions. The first purposed ES selected just one individual (parent) to create one new (individual) offspring (see Figure 4). This strategy is denoted (1 + 1) − ES. The new population accepts the best individuals (parent or offspring) according to the Equation 1.3. Thus, less fit individuals have zero chance of survival. In (1+1) −ES, the new individual is generated by the next equation:

$$y_i^g + y_i^g + N(0,\sigma) \qquad (1.2)$$

where N(0, #) is a vector of random numbers with a Gaussian distribution with mean zero and standard deviation σ. Figure 1.4 shows the whole process of (1 + 1) − ES.

$$x_i^{g+1} \leftarrow \begin{cases} y_i^g & si\ fitness\left(y_i^g\right) \le fitness\left(x_i^g\right), \\ x_i^g & Otherwise \end{cases} \qquad (1.3)$$

However, the mutation by itself does not have the ability to combine the genetic information of the best individuals, so it is also necessary to introduce crossover operators. Thus, the first type of a multimembered ES was proposed, the (μ+1)−ES or steady-state ES. There are μ parents at a time. Two of them are chosen at random and recombined to give life to an offspring, which also underlies mutation. The selection resembles extinction of the worst, may it be the offspring or one of the parents, thus keeping constant the population size.

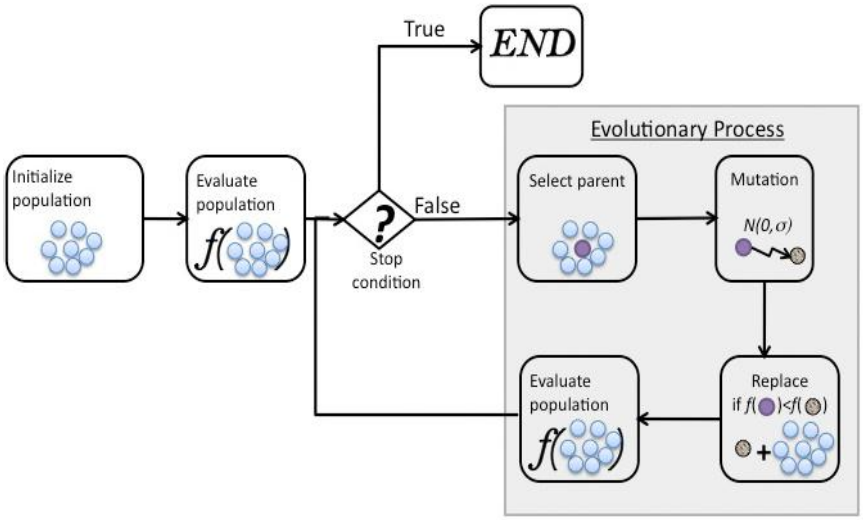

Figure 4: Operation of a typical ES.

Schwefel proposed the use of two multimembered strategies called (μ + λ) − ES and (μ, λ) − ES where μ indicates the population size and λ the offspring size [17]:

- in the (μ+ λ)−ES, not only one offspring is created at a time or in a generation, but λ ≥ 1 descendants, and, to keep the population size constant, the λ worst out of all μ + λ individuals are discarded,
- in the (μ, λ)−ES, the selection takes place among the λ offspring only, whereas their parents are forgotten no matter how good or bad their fitness was compared to that of the new generation. Obviously, this strategy relies on a birth surplus, i.e., on λ > μ in a strict Darwinian sense of natural selection.

Since the GAs described in Section 4.2, the ESs belong to EAs. Although they belong to the same family of algorithms there are some key differences between them [18]:

- ESs are applied to real parametric optimization problems, but GAs are mostly applied to optimization of attributes.
- The key operator of the GA is the recombination, however the most important operator in ES is the mutation.
- By default, the replacement in the GA is stochastic, in the ES is deterministic.
- GAs are global search mechanisms, that is, they have trouble on fine adjustment, just the opposite happens with ES.

The application of evolutionary strategies is closely linked to its nature of being a tool for parametric optimization of functions. However, they also have been applied to circuit partitioning problems [19], design of materials [20] or neural network training [21]. It has also been used in the field of mobile ad-hoc networks using them to improve the performance of AODV routing protocol [22] and Jamal Toutouh presents the OLSR routing protocol in VANETs by using Differential Evolution algorithm to search for energy-efficient configurations.

4.4 Particle Swarm Optimization (PSO)

Particle Swarm Optimization algorithm is a population based metaheuristic inspired by the social behavior of bird flocking and fish schooling. More precisely, PSO is a parallel evolutionary computation technique that provides a collaborative population-based search model. Individuals of the population called particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience and according to the experience of a neighboring particle, moving toward the best position encountered by itself or its neighbors. Thus, the PSO system combines local search methods (through self-experience) with global search methods (through neighboring experience), trying to balance exploration and exploitation [23].

Generally, a particle p_i is composed by three vectors and two fitness values:

- $x^i = (x_{i1}, x_{i2}, ..., x_{in})$ and $pBest^i = (p_{i1}, p_{i2}, ..., p_{in})$ store its current position and the best solution of the particle, respectively.
- $v^i = (v_{i1}, v_{i2}, ..., v_{in})$ is the gradient vector (direction) which defines the next movement of the particle p_i.
- The two fitness values are the current one (fitness_x_i) and the value of the best local solution found so far (fitness_pBest).

In this algorithm, each particle position x^i is updated each generation g by means of the Equation 1.4.

$$x_{g+1}^i \leftarrow x_g^i + v_{g+1}^i \qquad (1.4)$$

where factor x_{g+1}^i is the velocity of the particle and is given by

$$v_{g+1}^i \leftarrow \omega \cdot v_g^i \; \varphi_1 \cdot \left(p_g^i - x_g^i\right) + \varphi_2 \cdot (b_g - x_g^i) \qquad (1.5)$$

In this formula, p_g^i is the best solution that the particle i has stored so far, b_g is the best particle (also known as the leader) that the entire swarm has ever created, and ω is the inertia weight of the particle (it controls the trade-off between global and local experience). Finally, φ_1(cognitive component) and φ_2(social component) are specific parameters which control the relative effect of the personal and global best particles ($\varphi_1 = \varphi_2 = 2 \cdot$ UN(0, 1)).

PSO may be configured in different ways describing different variants of this algorithm. Kennedy in [24] defined four main variants depending on the influence of the values of the cognitive and social components ($\varphi_1$and φ_2). They are the following ones:

- Full Model: $\varphi_1, \varphi_2 > 0$. The movement is determined by both the cognitive and social component.
- Cognition-only Model: $\varphi_1 > 0$ y $\varphi_2 = 0$. Only the cognitive component is involved in the movement.
- Social-only Model: $\varphi_1 = 0$ y $\varphi_2 > 0$. Only the social component is involved in the movement.
- Exclusive Social Model: $\varphi_1 = 0$, $\varphi_2 > 0$ y $x_i \neq g_i$. The position of the particle itself cannot be the best in its environment.

According to M.G. Epitropakis [25] the sum of the values of the cognitive and social components of PSO ($\varphi_1$andφ_2) should be about 4.0, and common usage is to set them 2.05 each. This technique has been applied to problems of optimization of numerical functions [26] training neural networks [27], traveling salesman problem [28], and image processing [29]. In turn, it has been used as the basis for managing the links between nodes in VANET [30]. Ghassan Samara proposes a protocol Particle swarm optimization Beacon Power Control (PBPC), which makes dynamic transmission power control to adjust the transmission power of the safety periodic messages that have been aggressively sent by all vehicles on the road.

3.5 Differential Evolution (DE)

Differential Evolutions is appeared in 1994 when Kenneth Price tried to solve the Chebyshev polynomials [31]. Then, Price came up with the idea of using vector differences for perturbing the vector population. Since this seminal idea a lively discussion between Ken Price and Rainer Storm and endless ruminations and computer simulations on both parts yielded many substantial improvements which make DE the versatile and robust tool [32].

DE is a stochastic population based optimization algorithm. It is employed to solve problems in continuous (real) search spaces (real parameters, real valued functions). DE optimizes a problem by maintaining a population of N vectors v_i^g (where g is the generation number and i ϵ [1, N]) that represents the candidate solutions. The creation of new candidate solutions is carried out by combining existing vectors according to its simple formulae of differential-crossover and differential-mutation, and then keeping whichever candidate solution has the best fitness on the optimization problem at hand.

The differential-mutation generates new individuals by adding the weighted difference of two of the vectors of the population to the third, following the Equation 1.6:

$$w_{g+1}^i \leftarrow v_g^{r1} + \mu \cdot (v_g^{r2} - v_g^{r3}) \qquad (1.6)$$

Where r1, r2, r3 ϵ {1, 2, . . . , i − 1, i + 1, . . . ,N} are random integers mutually different, and also different from the index i, the mutation constant μ > 0 stands for the amplification of the difference between the individuals v_g^{r2} and v_g^{r3} , and it avoids the stagnation of the search process.

In order to increase even more the diversity in the population, each mutated individual undergoes a crossover operation with the target individual v_g^i, by means of which a trial individual u_{g+1}^i is generated. A randomly chosen position is taken from the mutant individual to prevent that the trial individual replicates the target individual.

$$u_{g+1}^i(\mathrm{j}) \leftarrow \begin{cases} w_{g+1}^i(j) & if\ r(j) \le Cr\ or\ j = j_r, \\ v_{g\,(j)}^i & otherwise. \end{cases} \quad (1.7)$$

As shown in Equation 1.7, the crossover operator randomly chooses a uniformly distributed integer value jr and a random real number r ϵ (0, 1), also uniformly distributed for each component j of the trial individual u_{g+1}^i. Then, the crossover probability Cr and r are compared just like j and jr. If r is less than or equal than Cr (or j is equal to jr) then we select the jth element of the mutant individual to be allocated in the jth element of the trial individual u_{g+1}^i. Otherwise, the jth element of the target individual v_g^i becomes the jth element of the trial individual. Finally, a selection operator decides the acceptance of the trial individual for the next generation if and only if it yields a reduction in the value of the evaluation function (also called fitness function f()), as shown by the following Equation (1.8):

$$v_{g+1}^i \leftarrow \begin{cases} u_{g+1}^i & if\ f\left(u_{g+1}^i\right) \le f\left(v_g^i\right), \\ v_{g\,(j)}^i & otherwise. \end{cases} \quad (1.8)$$

DE has been applied to various classical optimization problems with known global optimum [33], also problems of industrial processes optimization, task allocation [34], and to the design of wireless networks [35] or other problems such as the training of neural networks [36].

V. CONCLUSION

In the present work, we choose the five metaheuristic algorithms. Specifically, they are Simulated Annealing (SA), Genetic Algorithm (GA), Evolutionary Strategy (ES), Particle Swarm Optimization (PSO), and Differential Evolution (DE). We select these five algorithms because they are able to work on continuous (real valued) search spaces. Also, these techniques were selected with the aim of experimenting with different population structures, as well as different reproduction mechanisms. Thus, the analysis offers the possibility to get the quality-of-service improvements on a VANET's performance when metaheuristic algorithms are configured with vehicular data transfer protocol.

REFERENCES

[1] Christian Blum and Andrea Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison", *Metaheuristics International Conference*, *Kyoto*, August 2003.

[2] Xin-She Yang, "Metaheuristic Optimization," *Scholarpedia*, 6(8):11472, 2011.

[3] E. Alba, J. Garc´ıa-Nieto, J. Taheri, and A. Zomaya, "New research in nature inspired algorithms for mobility management in gsm networks," *in LNCS of the 5th European Workshop on the Application of Natureinspired Techniques to Telecommunication Networks and other Connected Systems, EvoWorkshops08*. Napoli Italy: Springer-Verlag, pp. 1–10, 2008.

[4] K. Parsopoulos and F. Vrahatis, "Unified particle swarm optimization for solving constrained engineering optimization problems*," in Advances in Natural Computation. Springer*, pp. 582–591, 2005.

[5] Osman, I. H., & Laporte, G., "Meta-heuristics: A bibliography", *Annals of Operations Research*, 63, 513–623, 1996.

[6] Roland Braune, Günther Zäpfel, *Metaheuristic Search Concepts: A Tutorial with Applications to Production and Logistics*, springer, 2010.

[7] Su, Xiao, "Global Optimization in Slope Analysis by Simulated Annealing", *project Report for Rocscience inc., 2008.*

[8] L. Ingber and B. Rosen, "Genetic algorithms and very fast simulation reannealing: a comparison*," Mathematical and Computer Modelling vol. 16, No. 11*, pp. 87–100, 2002.

[9] Zakir Hussain Ahmed, "A New Reformulation and an Exact Algorithm for the Quadratic Assignment Problem," *Indian Journal of Science and Technology*, vol. 6(4), pp. 4368–4377, 2013.

[10] A. Moraglio, H.M.M. Ten Eikelder, R. Tadei, "Genetic Local Search for Job Shop Scheduling Problem," Technical Report, *CSM-435, ISSN 1744-8050*.

[11] Vikas.M.N, Keshava.K.N, Prabhas.R.K and Hameem Shanavas.I, "Efficient Hand off using Fuzzy and Simulated annealing" *in I. J. Computer Network and Information Security*, Volume 1, Issue 2, 2012, pp. 17–23.

[12] N. Kumar, "Power Aware Routing Protocols in Mobile Adhoc Networks-Survey," *in International Journal of Advanced Research in Computer Science and Software Engineering*, Volume 2, Issue 9, September 2012, pp. 121–127.

[13] Nikos Dimokas, Dimitrios Katsaros and Yannis Manolopoulos, "Node Clustering in Wireless Sensor Networks by Considering Structural Characteristics of the Network Graph," in *International Conference on Information Technology (ITNG'07), IEEE computer society*, 2007.

[14] Tsunoda, D.F., Freitas, A.A. and Lopes, H.S., "MAHATMA: a genetic programming-based tool for protein classification", In Proc. Ninth International Conference on Intelligent Systems Design and Applications (ISDA-09), IEEE Press, 2009, pp 1136-1142
[15] Ingo Rechenberg., *Computer Methods in Applied Mechanics and Engineering"* ELSEVIER Press,Volume 186, Issues 2–4, June 2000, Pages 125–140.
[16] Hans-Paul Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons, 1995.
[17] Klaus Lucas, Peter Roosen, Emergence, *Analysis and Evolution of Structures: Concepts and Strategies Across Disciplines*," SPRINGER, 2009.
[18] Enrique Alba and Marco Tomassini, "Parallelism and Evolutionary Algorithms", *In Proc. IEEE Transactions on Evolutionary Computation*, Vol. 6, NO. 5, Oct. 2002, pp 443-461
[19] A.J. SOPER, C. WALSHAW and M. CROSS, "A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph-Partitioning," *in Journal of Global Optimization*, *Kluwer Academic Publishers*, Volume 29, Issue 9, 2004, pp. 225–241.
[20] Sivakumar,K. and Iyengar, N.G.R and Deb Kalyanmoy, "Optimization of composite laminates with cutouts using genetic algorithm variable metric and complex search methods", Eng. Opt.32, 2000, pp. 635 –657.
[21] Guifang Wu, Yumin Ren, Yan Li, Hoonsung Kwak and Seyoung Jang, "Research on Parameter Optimization of Neural Network", *In Proc International Journal of Hybrid Information Technology*, Vol. 2, No. 1, 2009, pp 81-89.
[22] Ajaii Sharma and Aditya Goel, "Performance Analysis of Mobile Ad-hoc Network Using AODV Protocol*", in Proc International Journal of Computer Science and Security (IJCSS)*, Volume (3): Issue (5), 2009, Pages 334–341.
[23] Da Ruan, *Computational Intelligence in Complex Decision Systems*, SPRINGER, 2010.
[24] Deal T. E. and Kennedy, A., *Corporate Cultures: The Rites and Rituals of Corporate Life*, Harmondsworth, Penguin Books, 2000.
[25] M.G. Epitropakis, V.P. Plagianakos, M.N. Vrahatis, "Evolving cognitive and social experience in Particle Swarm Optimization through Differential Evolution: A hybrid approach", *In Proc. ELSEVIER Information Sciences 216 2012*, pp 50-92
[26] Reza Akbari and Koorush Ziarati, "A Cooperative Approach to Bee Swarm Optimization*", In Proc. Journal of Information Science and Engineering* , Vol. 27, 2011, pp 799-818.
[27] James Kennedy, and Russell Eberhart, "Particle Swarm Optimization," *In Proc. IEEE Journal*, Vol. 0-7803-2768-3, 1995, pp 1942-1948.
[28] Xuesong Yan, Can Zhang, Wenjing Luo, Wei Li, Wei Chen and Hanmin Liu, "Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm", *In Proc. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 2, 2012*, pp 264-271.
[29] Venkatalakshmi, K and Shalinie, S.M.,"A customized Particle Swarm Optimization algorithm for image enhancement" *in Proc. IEEE International Conference on Communication Control and Computing Technologies (ICCCCT)*, ISBN 978-1-4244-7769-2, 2010, pp. 603 - 607.
[30] Ghassan Samara, Tareq Alhmiedat and Amer O. Abu Salem, "Dynamic Safety Message Power Control in VANET Using PSO", *In Proc. World of Computer Science and Information Technology Journal (WCSIT), Vol. 3, No 10, 2013*, pp 176-184.
[31] Mohammadreza Bonyadi, "A hybrid particle swarm with velocity mutation for constraint optimization problems", *In Proc of the fifteenth annual conference on Genetic and evolutionary computation conference(GECCO)*, ACM, 2013, pp 1-8, ISBN: 978-1-4503-1963-8.
[32] Ken Price, Rainer Storn, and Jouni Lampinen, "*Differential Evolution - A Practical Approach to Global Optimization", Springer*, 2008, ISBN: 3-540-20950-6.
[33] Sezimária de Fátima Pereira Saramago and José Laércio Doricio, "Optimum Design of 3R Robot Manipulator by using Improved Differential Evolution Implemented in Parallel Computation ", *in Proc. 22nd International Congress of Mechanical Engineering (COBEM)*, Brazil, 2013.
[34] Xianhui Zeng, Wai-Keung Wong and Sunney Yung-Sun Leung., "An operator allocation optimization model for balancing control of the hybrid assembly lines using Pareto utility discrete differential evolution algorithm*", in Proc. Computers and Operations Research*, Elsevier Science Ltd. Oxford, UK, Vol. 39 Issue 5, 2012 Pages 1145-1159.
[35] Okdem, S, Ozturk, C, and Karaboga, D, "A comparative study on Differential Evolution based routing implementations for wireless sensor networks", *In Proc. International Symposium on Innovations in Intelligent Systems and Applications (INISTA), 2012*, pp 1-5.
[36] Jarmo Ilonen,Joni-Kristian Kamarainen and Jouni Lampinen, "Differential Evolution Training Algorithm for Feed-Forward Neural Networks", *In Proc. Kluwer Academic Publishers, Netherlands, 2003*, pp 93-105.

