Annals of the University of Craiova, Mathematics and Computer Science Series
Volume 41(2), 2014, Pages 251–270
ISSN: 1223-6934

Survey on distributed approaches to swarm intelligence for graph search problems

Sorin Ilie

Abstract. In this paper a survey on existing distributed approaches to swarm intelligence approaches for graph search problems is presented. In particular we reviewed papers on Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO). The comparison criteria that have been used are computational efficiency and speedup. The conclusion of this study is that the coarse grained master-slave model is the most studied. However, we found a large amount of papers sustaining that the island approach offers better solution quality. The added flexibility of the communication strategy between the islands makes this model preferred by the most recent papers.

2010 Mathematics Subject Classification. Primary 68T01; Secondary 68R10.
Key words and phrases. swarm intelligence, distributed computing, ACO, BCO, parallel computing.

1. Introduction

Before going into the subject of this paper, we first introduce a few recurrent terms that will be used throughout the following sections.

SwarmIntelligence(*SI*) is the emergence of coherent functional global patterns from the collective behaviors of entities interacting locally [48]. Two families of SI algorithms are Ant Colony Optimization(ACO) and Bee Colony Optimization (BCO).

Entity is a general term used to refer to a unit of the Swarm Intelligence population, such as an ant, bee, particle, bird, fish etc.

Agraph is a set of objects and the pairwise relations between them. In [28] the objects are named vertices (singular: vertex) while the relations are called edges. Edges are actually considered to be connections between the vertices and are graphically represented as lines while vertices are represented as dots or ovals.

We define the term *computingnode* as an abstraction of a computer or machine, physical or virtual, having its own or allocated processor and memory. This term is used in the subject of distributed applications [65] and adopted by distributed computing support systems [106].

SI is inspired by natural bio systems consisting of populations of simple beings such as: ants, bees, birds, fish etc. The main characteristic of these is the inherently distributed way they solve problems. This suggests SI algorithms should allow straightforward mapping directly onto distributed software systems.

Based on our literature review we found that existing approaches rely on running multiple instances of sequential SI algorithms using parallel and highperformance

Received August 9, 2014.
This work was supported by the strategic grant POSDRU/159/1.5/2/133255. Project ID 133225(2014), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007- 2013.

computing architectures. SI computational approaches [48] are used to solve high complexity problems, such as NP-hard problems. These approaches are heuristic [47] by nature. Basically they provide an optimal solution, or one that is close to an optimum, to a high complexity problem in reasonable time. This is accomplished by using a population of entities that interact locally with each other and/or with the environment. There are two approaches to the interaction model:

1) the entities interact in the problem environment in order to guide each other to better solutions. For example, such a SI approach is Ant Colony Optimization [45], Bee Colony Optimization [116]

2) the entities interact in the solution space of the problem in order to improve existing initial solutions given as an output of an additional algorithm. For example, such a SI approach is Particle Swarm Optimization (PSO) [27], Cat Swarm Optimization (CSO) [24]. This type of SI is generally designed for continuous solution space that is given as input. Discrete adaptations exist [83], however, they require an additional algorithm, especially tailored to each problem to be solved, that outputs the vicinity of a solution which can be modeled as a graph vertex. Therefore this type of SI algorithm is not well suited for "graph search", the type of problem we are trying to solve.

There is a certain level of abstraction at which SI systems can be modeled as distributed computational systems composed of interacting artificial entities. Thus, we would expect distributed computing, including multiagent middleware, to have a lot of potential for the application of SI approaches.

The rest of this paper presents a survey on the subject of distributing SI and in particular ACO and BCO but also introduces the background knowledge required for the proper understanding of the said survey.

2. Graph Search Problems Formalization

SI algorithms using the first interaction model, mentioned previously in this paper, are very well suited for graph search problems. Consequently, these algorithms are usually tested on the benchmark problem of the NP-hard Traveling Salesman Problem (TSP), which falls in this category of problems. We now present a formal mathematical description of the problem type we are addressing. Let $G = (V, E)$ be a directed graph, where V is a set of vertices and E is a set of edges [59]. An edge is defined as an ordered pair of vertices (x, y). The size of the set of vertices is labeled $n = |V|$.

A path $p = (v1, v2, ..., vn), k > 0$ in the graph G is an ordered sequence of vertices such that for any two subsequent vertices vi and v_{i+1} there is an edge $(v_i, v_{i+1}) \in E$. We label the totality of paths in a graph with P. Any graph search problem can be formalized as minimizing or maximizing a function $f : P' \to \mathbb{R}$ where $P' \subseteq P$, i.e. P' is P restricted by some constraints. As an example, we now formalize the Traveling Salesman Problem (TSP). We first define a weight function w that assigns a real number to an edge of the graph. TSP can be formulated mathematically as the minimization of the function:

$$f : P' \to \mathbb{R}; f(p) = \sum_{i=1}^{n-1} w_{1,1+1} + w_{n,1} \tag{1}$$

where p is a path that is evaluated; $w_{i,i+1}$ is the weight of the edge $(i, i + 1)$; P' is the totality of Hamiltonian cycles.

Other examples of problems that can be described in this manner include: graph flow problems [59], generalized TSP [3], pickup and delivery problem [90] etc. Therefore each of these problems can be defined as the minimization of a function f over a subset of paths $P' \subseteq P$. An approach that can solve one of them can be adapted to solve all graph search problems.

2.1. Ant Colony Optimization(ACO). ACO [45] refers to a family of SI optimization algorithms that get their inspiration from the metaphor of real ants searching for food. When ants search for food, they secrete pheromone on their way back to the anthill. Other colony members sense the pheromone and become attracted by marked paths; the more pheromone is deposited on a path, the more attractive that path becomes.

The pheromone is volatile, i.e. disappears over time. Evaporation erases pheromone on longer paths as well as on the paths that are not of interest anymore. However, shorter paths are more quickly refreshed, thus having the chance of being more frequently explored. Intuitively, ants will converge towards the most efficient path, as that path gets the strongest concentration of pheromone.

In the SI approach called ACO, artificial ants are programmed to mimic the behavior of real ants while searching for food. The ants' environment is modeled as a graph and the path to the food becomes the solution to a given graph search problem. Artificial ants originate from the anthills that are vertices in the environment and travel between vertices to find optimal solutions, following ACO rules. When a solution is found, ants mark the solution with pheromone by retracing their path.

At the core of ACO algorithms there are rules that determine the amount of pheromone deposited on edges traversed by ants, the edge chosen by each ant on its way, and how fast the deposited pheromone evaporates. Ants randomly choose to travel across edges with a probability proportional to the pheromone-weight ratio. Ant a located at vertex i decides to move to vertex j with the probability $p_{i,j}$ computed as follows:

$$p_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\Sigma_j(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}} \tag{2}$$

where:

- α is a parameter to control the influence of $\tau_{i,j}$
- β is a parameter to control the influence of $\eta_{i,j}$
- j represents a node reachable from node i that was not visited yet
- $\tau_{i,j}$ is the amount of pheromone deposited on edge (i, j)
- $\eta_{i,j}$ is the desirability of edge (i, j) computed as the inverse of the edge weight, i.e. $1/w_{i,j}$
- N_i represents the set of neighbors of node i

Better solutions need to be marked with more pheromone. So whenever an ant k determines a new tour V_k of cost L_k the ant will increase pheromone strength on each edge of the tour with a value that is inversely proportional to the cost of the solution.

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/L_k \& \text{if edge } (i,j) \text{ belongs to found tour } V_k \\ 0 \& \text{otherwise} \end{cases} \tag{3}$$

where L_k is the cost of the k-th ant's tour.

As pheromone is volatile, if a real ant travels more, pheromone will have more time to evaporate, thus favoring better solutions to be discovered in the future. When an ant completes a solution it will retrace its steps marking the edges on the way with pheromone. The update will also take into account pheromone evaporation. Both

evaporation and pheromone updates are implemented as follows:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau_{i,j}^{k} \tag{4}$$

where ρ is the evaporation rate $0 \leq \rho < 1$.

All ants use formula 2 to probabilistically determine their next step. Therefore they will often choose the edge with the highest pheromone, i.e. the exploration of less probable edges is lower. The solution is to decrease the pheromone on edges chosen by ants, i.e. apply a local evaporation process. This has the effect of making them less desirable, increasing the exploration of the edges that have not been picked yet. Whenever an ant traverses an edge it applies local evaporation by updating pheromone as follows:

$$\tau_{i,j} = (1-\xi)\tau_{i,j} + \xi\tau_0 \tag{5}$$

where:

- ξ is the local evaporation rate $0 \leq \xi < 1$.
- τ_0 is the initial amount of pheromone on each edge

A good heuristics to initialize pheromone trails is to set them to a value slightly higher than the expected amount of pheromone deposited by the ants on a solution; a rough estimate of this value can be obtained by setting $\tau_0 = 1/(nC)$, where n is the number of nodes, and C is the tour cost generated by a reasonable solution approximation procedure [45]. For example we can set $C = nw_{avg}$ where w_{avg} is the average edge cost.

2.2. Bee Colony Optimization (BCO). Bee Colony refers to a family of SI optimization algorithms that get their inspiration from the metaphor of real bees searching for food. Bee colonies forage for pollen sources by moving randomly in the physical environment. When a bee finds a rich food source, starts to perform the so called "waggle dance" [46] upon its return to the hive. The purpose of this dance is to inform the other bees about the direction and distance to the food source [12]. The other bees from the swarm will then be inclined to explore the indicated location.

In the SI approach Bee Colony, artificial bees are programmed to mimic the behavior of real bees while searching for food. The bees environment is modeled as a graph while the path to the food becomes the solution to a given graph search problem. In [101] the authors present a survey of Bee Colony algorithms among them the basic "Bee Colony Optimization" (BCO) algorithm and its mathematical model which we will now detail. In BCO a bee randomly moves across edge (i,j) according to the transition probability formally defined in the following equation.

$$P_{i,j} = \frac{(p_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\Sigma_j (p_{i,j})^{\alpha}(\eta_{i,j})^{\beta}} \tag{6}$$

where:

- p it the fitness function of edge (i,j)
- α is a parameter to control the influence of $p_{i,j}$
- β is a parameter to control the influence of $\eta_{i,j}$
- j represents a node reachable from node i that was not visited yet

The fitness of a particular edge is calculated according to the "preferred path" suggested by other bees that found a solution. The edge (μ,σ) in the "preferred path" that originates in the current vertex μ where the bee is located has the fitness $p_{\mu,\sigma} = \lambda$. The other edges originating have the fitness $p_{\mu,\sigma} = (1-\lambda)/\phi$ where ϕ is the number of unvisited vertices apart from the "preferred path" next vertex σ. Concretely, if the bee has to choose between the unvisited vertices $(\nu_1, \nu_2, ..., \nu_k, \sigma)$ the probabilities of choosing each vertex will be $(\frac{1-\lambda}{\phi}, \frac{1-\lambda}{\phi}, ..., \frac{1-\lambda}{\phi}, \lambda)$ respectively.

However, if the preferred vertex σ has already been visited or a "preferred path" does not exist yet, the fitness of the edge (μ, σ) automatically becomes null, i.e. $p_{\mu,\sigma} = 0$. Intuitively, the bees will explore variations of the best solutions while notifying the other bees about the improved solutions found using "waggle dance".

2.3. Using Local Search To Improve Solutions. "Local Search" [62] is an algorithm used to improve suboptimal solutions to graph search problems found by heuristic approaches. This algorithm chooses k distinct edges of the path representing the solution and tries to reorganize them in such a way that the quality of the solution improves. In this algorithm k is a natural number smaller than the solution size that is given as a parameter.

Depending on the number k, Local Search algorithms are called $2 - opt$, $3 - opt$ and so on up to $k - opt$. Basically, k edges from the solution are replaced such that a lower cost solution is generated. We will now present a very general version of the k-opt local search algorithm LOCAL-SEARCH(solution, k) where:

- "solution" is the path that the algorithm operates on,
- k is the number of considered edges,
- CHOOSE-EDGES(solution, k) randomly chooses the k distinct edges,
- REORGANIZE(edges) generates a new solution by reorganizing the k edges,
- SOL-ACCEPTABLE(solution) returns true if the solution is improved by an acceptable amount
- SOL-IMPROVED(solution) returns true when the solution is improved by reorganizing the edges,
- MAX-ITERATIONS() returns true is a predefined number of iterations have been executed with no improvement.

The complexity of this algorithm depends on the parameter k and how fast an improvement is achieved, however it has been experimentally shown that 2-opt local search performs just as well as 3-opt or higher values for k, while using a lot less CPU time [62]. Therefore there is little or no incentive to use a value higher than $k = 2$.

```
LOCAL-SEARCH(solution, k)
1.   initialSolution=solution
2.   for all groups of k edges of solution
3.          edges=CHOOSE-EDGES(solution, k)
4.          for all solution=REORGANIZE(edges)
5.                if SOL-ACCEPTABLE(solution)return solution
6.                if SOL-IMPROVED(solution)
7.                       initialSolution=solution
8.          if MAX-ITERATIONS() return initialSolution
```

Using local search has been shown to improve the performance of SI algorithms [45, 48, 111]. Therefore, SI approaches sometimes use 2-opt local search on solutions before "advertising" it by using "waggle dance", pheromone deposits etc.

2.4. Distributed Approach Performance Measures. During our review of the literature on the topic we discovered several distributing models (independent runs, master-slave, island and hybrid), some general purpose distributed frameworks designed to distribute any heuristic algorithm, inherently distributed agent-based approaches and several isolated combinations of existing models..

The reviewed papers present various scalability experiments solving multitude of graph search problems. Performing a fair comparison of the different approaches to distributing various SI algorithms requires some common normalized measures to put side by side. For this reason we introduce here the notions of speedup and

computational efficiency. The speedup metric of a distributed application measures how much faster the application runs when more computing nodes are used as opposed to a single computing node. The computational efficiency metric normalizes the value of the speedup to the number of computing nodes. The equations for computing the two metrics are:

$$S_c = \frac{T_1}{T_c} \tag{7}$$

$$e_c = \frac{S_c}{c} \tag{8}$$

where

c is the number of computing nodes used
S_c is the speedup of the application
T_1 is the execution time of the application on a single computing node
T_c is the execution time of the application on c computing nodes
e_c is the efficiency of running the application on c computing nodes

3. State of the Art in Distributing ACO and BCO

A very recent overview and classification of parallel computing approaches to Ant Colony Optimization (ACO) was reported by [87]. The authors propose an interesting and novel classification scheme for parallel ACO algorithms. The classification includes: master-slave model, cellular model, independent runs model, island model and hybrid models. However, the authors do not include agent-based approaches which, they consider, are not specifically designed to take advantage of multi processor architectures.

The cellular model mentioned in [87] is actually the author's own work [86] which proposes splitting ACO search space into overlapping neighborhoods each one with its own pheromone matrix. The good solutions gradually spread from one neighborhood to another through diffusion. This approach requires the search space to be partitioned in such a way that each set contains a continuous part of the optimal solution. The problem with this is that if the search space is not partitioned in this manner, an optimal solution can never be reached. The authors are the only ones that have ever implemented and tested this model based on the cell diffusion approach.

In the case of BCO we could find four papers [88, 89, 102, 103] that collectively offer a good classification of BCO parallel approaches. Interestingly, these papers identify the same general taxonomy of approaches for BCO as [87] does for ACO: master-slave model, independent runs model, island model and hybrid models. Therefore in the following subsections we identify and describe the research done using master-slave model, independent runs model, island model, hybrid models and the agent-based models. But because these terms are not standardized we must first define what we mean by them.

3.1. The Basic Taxonomy of Distributed SI Approaches. The master-slave model of distributing SI requires a central "master" computing node that manages a global best-so-far solution while multiple "slave" computing nodes find candidate solutions.

This model can be further divided into:

- fine grained master-slave model, where the slaves do atomic actions such as move one ant or find one solution. This requires a large amount of communication with the master.

- coarse grained master-slave model, in which the slaves do complex actions such as find multiple solutions and communicate the best one to the master.

In the fine grained version the slaves have to communicate with the master after each tiny task they are asked to execute and then they wait for new requests from the master. This overloads the master computing node and introduces large wait times when many slaves are used. In the coarse grained version the slaves find whole solutions or even sets of solutions before reporting to the master. Since in most implementations that is all the slaves do, they actually do not need to be explicitly told what to do next. Evidently the coarse grained master-slave model puts less stress on the master and therefore is a lot more scalable.

The independent runs model uses many instances of the SI algorithm that solve the problem independently and the best solution is chosen. SI approaches are inherently heuristic therefore they cannot guarantee finding the optimal solution, however, running an SI algorithm multiple times increases the chances of getting a very good quality solution. The fact that no communication is used but more than one instance of the algorithm is executed entails three consequences:

- faster execution time and implicitly better efficiency than any other implementation that requires communication, but only if the same number of solutions are explored in all considered implementations,
- better quality of solutions than a single sequential run,
- less qualitative solutions than implementations that use communication.

The island model of distributing SI requires running a separate group of entities, referred to as an "island", on each available computing node. Each group has its own map of the search space, therefore can be considered a sequential implementation of the SI algorithm. At predetermined points in time, with fixed frequency, solutions are communicated and the best solution over all is marked as required by the SI algorithm. Solutions can also be communicated asynchronously. This type of collaboration between the islands sets it apart from the master-slave model and the independent runs model.

When implementing the island model there are at least two issues to consider: i) the communication topology of the islands where the unidirectional ring (on the left) and full mesh topology (on the right) are depicted, ii) the frequency with which the islands communicate solutions to each other and whether this should be done synchronously or asynchronously. When the ring communication topology is used, each island sends a single message containing a solution to a predefined island. When using the mesh topology, any given island can broadcast its solution to all other islands.

This model is also called the "multi-colony model" in the case of ACO and the "multi-hive model" for BCO. This model is even more scalable than the coarse-grained master-slave since there is no master computing node to cause a "bottleneck". The existence of a master decreases performance when too many slaves try to communicate at once.

The hybrid model is a combination of the island with the master-slave model. An example of the hybrid model could be an island model where each island is actually a master-slave model instead of a simple sequential implementation of the SI algorithm.

Before going into each model separately, we present a quick overview of the taxonomy papers that will be referred multiple times in the current subsection. The work introduced in [88] and continued in [89] differentiates between the master-slave model, hybrid model and the island model of BCO. Experiments were done with each model

implemented using the message passing interface (MPI) library [82], and tested on numerical benchmark functions: Rastrigin, Griewank and generalized Schaffer. The implementations were executed using 11 computing nodes for the master-slave approach and 4 computing nodes for the island and hybrid model. The authors concluded that the island approach offers better quality solutions in less time, closely followed by the master-slave approach. No scalability tests were done on any of the models, the network delay was not taken into account. Furthermore, in papers [102, 103] the authors identify the model of independent runs additionally to the master-slave model, hybrid model and the island model of BCO. The authors also identify two variants on the island model: i) communicating just one solution and ii) the complete solution matrix. Their experiments were run in a simulated environment. The conclusion was that the independent serial runs approach was the fastest, however, the models that used communication offered the best quality of solutions.

3.2. Fine Grained Master-Slave Model. Implementing a scalable fine grained master-slave model is not trivial, as the first attempts show. Paper [79] analyzes the amount of communication needed by the fine grained model of ACO, concluding that it is not feasible. The first successful attempt at achieving scalability by using a hierarchy of master computing nodes was published in [13], however, their approach did not scale very well. In paper [96] a fine grained master-slave model of ACO is analyzed and the maximum efficiency obtained, of 0.8, was when using only two processors. Bullnheiner et al. in [15] made an attempt at the same model. The authors concluded that "acceptable execution time can be achieved" for TSP problems larger than 200 vertices. In the papers [36, 37] the distributed approach from [15] is revisited using a shared memory machine in the experiments reaching the maximum efficiency when employing 8 nodes. In such an architecture the access to the memory is concurrent, i.e. there is a shared memory and each process uses it one at a time in order to centralize its knowledge. No other modifications have been made to the architecture. The authors continue their work in [38] by comparing their approach from [36] to implementations that use synchronized messages instead of a shared memory. The access to the shared memory is much faster than messaging, which also requires broadcasting the current solution. This allows the ants to benefit sooner from the improvements found by the other population members. The conclusions were that their approach offers better solutions in less time. The authors obtained a speedup of 5.45 using 16 nodes, therefore resulting in an efficiency of merely 0.34. The most recent approach we could find on the fine-grained master-slave model is [55]. The authors of this paper used up to 240 GPUs to solve TSP using ACO and they succeeded in obtaining speedups of up to 30. The small amount of research using this model and the conflicting opinions about its effectiveness suggest that it is being abandoned as a viable model and is not, in fact, the state of the art approach to distributing SI.

3.3. Coarse Grained Master-Slave Model. The coarse grained master-slave model, unlike the fine grained one, is not plagued by conflicting opinions about its effectiveness. [74] implemented the coarse grained master-slave model for BCO and obtained near optimal solutions surpassing the quality of the results offered by the sequential approach. In paper [104] the authors present their approach to the coarse grained master-slave model called "ANTabu" which offers high quality solutions to the Quadratic Assignment Problem (QAP) [14]. Peng et al. presented their experiments with the coarse grained model [91, 92] that offered better quality solutions than evolutionary algorithms in the case of two different problems. In paper [75] groups of ants are

used on each slave resulting in better performance than sequential ACO. However, no comparison was made with the case of using only one ant per slave. Although these experiments used more than one computing node, these papers did not study the computational efficiency of their approaches.

We found several papers describing scalable implementations of the coarse grained master-slave model. In the work started in [9] and continued in [10] the authors implement the coarse master-slave and the hybrid models for BCO and test their implementations on a cluster of computers. The best execution time was achieved by the master-slave while the hybrid model obtained better quality of solutions. In [69] the authors present experiments showing that the coarse grained master-slave model is more efficient than the independent runs model. Their approach obtained efficiency of 0.8 using up to 16 processors. Papers [70] and [56] present experimental results where this model offers a speedup of up to 1.72 on dual core computers when running the slaves on separate threads, which means 0.86 efficiency.

Very good results were obtained using graphical processing units (GPU processors) for the slaves. One example is the already mentioned paper [55]. Another would be [16] that implemented a coarse grained master-slave where the master executed on the CPU and the slaves executed on the 8 pixel processors of a graphics card. The execution time depended heavily on the number of ACO entities used.

We also researched the problem of synchronous versus asynchronous centralization to the master computing node. The authors of [6] have implemented their version of this model for BCO and concluded that asynchronous centralization more efficient. Paper [16] compares the two approaches to coarse grained master-slave model for ACO and found them to offer similar speedup. On the other hand papers [110] and [63] found the asynchronous approach to be much faster for their respective implementations of ACO. The synchronous implementation from [45] offers an efficiency of 0.7 using 8 computing nodes, while the asynchronous approach from [21] boasts 0.9 efficiency on 8 nodes. We can conclude that in most cases an asynchronous implementation has better performance.

3.4. Independent Runs Model. The independent runs model is found to be used rarely. Experiments in [95] confirm that the independent runs model is more efficient than the coarse grained approach and offers slightly better solution quality than sequential implementations. Papers [1] and [2] also concluded that it is more efficient than coarse grained master-slave, synchronous and asynchronous island model. However, at the same time they confirm the fact that the independent runs model offers less qualitative solutions than implementations that use communication. These results are confirmed in the case of BCO by the authors of [102, 103], described earlier in this subsection. Therefore the implementations that require communication on top of constructing and validating solutions will automatically have longer execution times. However, the models that use communication offer better solutions sooner than the independent runs model. Thus, it is our conclusion that better alternatives to the independent runs model have already been developed.

3.5. The Hybrid Model. The hybrid model was implemented and studied in just a few publications. In papers [37, 71] each colony becomes the master of multiple slaves. This type of approach was tested in [37] on a cluster of 9 heterogeneous machines. The papers reported modest speedup values and slight improvements in quality of solutions when compared to other models. On the other hand, papers such as [64, 99] propose that the solutions of an island model should be collected and processed by a master computing node. Both papers lack efficiency analysis.

3.6. The Island Mode. The island model is a very popular research topic. We will first present the research conducted on the main features to be considered in this model: (i) the communication topology and (ii) the synchronization intervals.

On the subject of the communication topology to be used in the island model, the authors of [111] thoroughly analyze multi-colony ACO algorithms applied on TSP, experimenting with different communication policies and variable synchronization frequency, used as input parameters. They tested the communication topologies in terms of execution time and solution quality, with focus on maximizing the latter. The authors of paper [111] found that a ring topology outperforms the sequential approach to ACO and other tested topologies, confirming through independent experimentation the results of [76, 94, 79, 25]. Although, [23] sustains that the star topology offers the best results, their conclusions are, by far, less rigorously supported by experimental data than those in paper [111]. In all of these approaches the colonies distribute their solutions synchronously.

Experiments concerning point (ii), the synchronization intervals for the island model can also be found in the literature. In papers [34, 35] the authors analyze implementations of BCO using: the independent run model, a fine grained, coarse grained master-slave model and an island asynchronous implementation, using a ring communication topology. The experimental results show that the island asynchronous implementation outperformed all other models in both execution time and solution quality. [18, 20] found that the island model can remain scalable up to 25 computing nodes with self-adaptation but efficiency is worse than without self-adaptation. Experiments in [73] present that asynchronous solution updating between the colonies offers better scalability. The authors experimented on a cluster of 72 dualcore machines using one thread per island. We can conclude that in most cases an asynchronous implementation and larger intervals between synchronizations offer better performance.

Scalability and effectiveness of the island model was also extensively studied. A distributed version of the island model is presented and compares it with the sequential version of their BCO algorithm in [6]. The conclusions drawn by the authors were that the distributed approach is superior in both the quality of solutions and the execution time. In paper [84] the island model for BCO is implemented. The authors use shared memory to synchronize the solutions obtained by the separate hives and boast execution time decrease when increasing the number of computing nodes. [120] presents experimental data on island model ACO tests on up to 64 computing nodes, however, they do show a dramatic decrease in speedup for more than 8 nodes. In papers [17, 19] and [20], although the results were less than ideal, good execution time was still obtained when increasing the number of CPUs up to 25. In [68] the island model experiments show better execution times and higher quality of solutions than genetic algorithms (GA) [58] and sequential models of ACO. In paper [25] the authors claim that multi-colony is faster than the master-slave model. Their approach was tested on up to 5 computing nodes. Paper [121] confirms that multi-colony is more efficient than sequential ACO when using up to 8 computing nodes. The experiments show that for given TSP problem sizes there is always a number of colonies above which efficiency decreases.

Many authors noted that island model offers an increase of solution quality compared to sequential implementations. In [111] it is stated that the island model consistently offers better solutions than the sequential implementations of the same

ACO algorithm. On the other hand, [122] found that the island model offers "competitive" solutions when compared with other approaches. In paper [100] the authors experiment with exchanging full pheromone matrices instead of just one best-so-far solution. Their conclusion was that this greatly affected the execution time of the approach but slightly improved the quality of the solutions. Given the large number of papers written on the subject and the outstanding results, it is clear that the island approach is the current state of the art model of distributing SI.

3.7. Variations of the Master-slave Model. Some approaches to the master-slave model use the slaves only to perform local search on the solutions generated by the master. In [72] this idea is implemented for ACO using one thread per slave on two dualcore machines. The authors obtained speedup in their experiments, however, the hardware used did not facilitate detailed testing of their approach. Paper [108] tested a multi-threading implementation of ACO on two quad-core computing nodes. The experiments showed significant improvement in execution time. Continuing their work in [109] using two dual core machines, they obtained better results than a synchronous communication island model and an independent parallel runs model. [115] experimented with this model of ACO on 47 non-dedicated machines, they concluded that larger problems generate better speedup values. In paper [29] the authors used their parallel framework for master-slave approaches to implement similarly asynchronous ACO. They achieved very good speedup values up to 25 computing nodes, however, efficiency decreased directly proportional to the resources used.

3.8. Agent-based Approaches to Distributing SI. Papers [112, 113, 114] present and improve an agent based approach to the Vehicle Routing Problem (VRP) [57] inspired by BCO called "BeeJamA". The authors use two types of bee agents to disseminate routing cost information using BCO rules. Their approach is evaluated only in terms of quality using a simulation in MATSim [77], no scalability or efficiency study was performed.

Paper [105] presents a potential application of BCO in road traffic implemented as a multi-agent system. The authors propose a reactive agent system that leads to good results through the nature of the BCO algorithm. This is a practical application with a high interest in quality of solutions and adaptability, however, not efficiency during distributed execution. The work in [101] describes a purely theoretical overview on multi-agent optimization based on BCO. The authors describe the bees formally as reactive agents. The paper consists of detailed descriptions on how to program the bee agents to respect different BCO algorithm rules. No real implementation is mentioned.

The authors of [98] present a multi-agent system [8] implementation of ACO applying their solution to TSP. In their approach, graph vertices and ants are implemented as JADE [8] agents. Ants centralize the information about pheromone deposits and vertices' best tour cost through a single message exchange per vertex. This procedure adds up to $2n$ messages per tour for each ant, where n is the number of vertices. Each ant has to notify the vertex about its next hop and the cost of its tour for the vertex to be able to update its pheromone levels. This generates other n messages. When an ant completes a tour, it compares the tour cost with the collected best tours from the vertices. A best tour synchronization is triggered for all the vertices if a better tour has been found. This brings an additional overhead of n messages. Hence this approach requires $4n$ messages per tour per ant. Unfortunately, the authors provide no experimental data to support their claim of "good results".

In [50] the authors compare a distributed form of ACS with the flooding algorithm applied to resource discovery problem using ns2 network simulation tool [85]. They show that ACS is the better approach in terms of: best success rate, least number of hops and least traffic. The detailed algorithm and ACO parameters are not presented in order to duplicate their approach for a realistic comparison. The main characteristics of their approach are: (i) resource queries are handled centrally at a single computing node, thus introducing single point of failure, (ii) they do not take edge weights into account as they are trying to solve the resource discovery problem, and (iii) ants are implemented as ns2 mobile agents. Moreover, in practice very often there is no need for code mobility as every ant is governed by the same behavioral rules.

In the article [33] the authors present a fully distributed approach to ACO that models both vertices and ants as agents. Their proposal uses multiple vertices that play the role of anthills, generating colored ants modeled as mobile agents. They test their proposal on the problem of load balancing in distributed systems and compare it with the workstealing approach. The article concludes that the proposed architecture is more efficient in terms of the number of busy vertices and the elapsed time until a load distribution of 50% is reached. No efficiency study is made on the approach and in this case too there is no need for code mobility either, since all ants are governed by the same rules.

In the article [22] the authors present an agent-based distributed approach similar to the fine grained master-slave model from [37] but implemented with agents with learning capabilities. This papers main goal is not to use the distributed architecture to improve the efficiency of [37] but to improve other performance measures by the use of a knowledge base. As a consequence, this approach inherits the efficiency shortcomings of its predecessor.

In paper [78] the authors present a hybrid ACO/PSO (Particle Swarm Optimization) control algorithm for distributed swarm robots applied on the resource finding problem. A virtual pheromone is used in the form of messages. The messages contain the two-dimensional coordinates of the resource, its size and quality reflected in the quantity of pheromone. All robots have to manage their own map and pheromone deposits. PSO is used to avoid convergence to a local optima. The experiments are done in a virtual sequential environment. No scalability studies have been made.

Paper [119] proposes a JADE-based [8] multi-agent environment for dynamic manufacturing scheduling that combines intelligent techniques of ACO and multi-agent coordination. There is no globally accessible map, hence each agent needs to manage their own map and pheromone deposits. However, the focus in [119] is to evaluate the impact of ACO intelligence on multi-agent coordination, rather than to utilize multi-agent middleware for improving ACO. Therefore ACO intelligence is embedded into job and machine agents that have the role of ants.

We are aware of only one agent-based approach that was studied in terms of computational efficiency: the ACODA approach from [123]. The authors model subsets of the search graph vertices as agents and execute them on separate computing nodes. Ants are modeled are software objects passed from one vertex to another via a local queue or agent message. The authors boast an e=0.98 efficiency value on 11 computing nodes.

3.9. General-purpose Distributed Frameworks. During our review of the work done on distributing Si we have also found a few general-purpose distributed frameworks designed for any algorithms .

Paper [32] proposes JABAT (JADE-Based A-Team) - a JADE [8] based middleware for collaborative problem solving using asynchronous teams known as A-teams. JABAT supports distributed implementation and collaboration of population-based optimization algorithms. JABAT agents represent sequential improvement algorithms that cooperate in order to solve a given problem in a distributed manner. Unfortunately, we could not find scalability studies of the JABAT distributed architecture. Earlier, a similar approach was introduced in [30, 31] using an object oriented approach [118] to run several sequential algorithms in parallel. In this case the authors impose a master-slave organization system between the processes. The framework is tested for image processing using ACO.

The scalable distributed constraint architecture called DisChoco is described in the paper [49]. In this approach, agents manage parts of the problem domain as well as the constraints specific to their partition. The agents propose, propagate or refuse partial solutions to each other.

Paper [61] proposes a purely theoretical framework for multi-agent systems. No experiments or implementations are mentioned in this work. The authors present a distributed form of ACO based on so called smart messages approach to multi-agent systems. Agent mobility is used to implement complex communication over dynamic networks. They use delegate multi-agent systems to manage these smart messages in order to design a multi-agent approach for ACO. No ways of modeling the environment, determining convergence or stopping condition of ACO experiments are presented.

The frameworks presented in papers [31, 7, 32] are designed for the coarse grained master-slave approach. The [61] approach is much better suited for the distribution of the search space then migrating entities in the form of messages, according to the particular SI rules. However, this does not motivate the use of code mobility since all entities are governed by the same behavioral rules.

3.10. Partitioned Search Space Approaches. Briefly mentioned at the beginning of this subsection, the cellular approach from [86] proposes splitting ACO search space into overlapping neighborhoods, each one with its own pheromone matrix. The good solutions gradually spread from one neighborhood to another through diffusion. The authors obtained the efficiency of 0.9 on 4 nodes.

A similarly interesting approach [81] to distributed ACO divides the search space between the available and merges the solutions offered by the slaves at a central computing node. The authors boast exponential decrease of execution time. Another approach that divides the search space called "D-ant" is introduced in [41] and continued in [42] which was reported to outperform a multi-colony and a coarse grained implementation of ACO. The efficiency of the approach peaked at 0.7 when using 8 computing nodes. However these approaches would need to be inherently synchronized, which has negative effects on performance of implementations with a central computing node. Most research conducted on the matter agrees that asynchronous messages offer better execution time than synchronous ones. Another criticism that can be brought to these implementations is the fact that they are not purely ACO since they are using a different algorithm to merge the partial solutions.

4. Conclusions

The independent runs model is rarely used by the scientific community. When many instances of the SI algorithm solve the problem independently and the best

solution is chosen in the end increases the chances of getting a higher quality solution. We draw two conclusions from the work on the subject: i) this model is faster, however, it offers worse solutions than any other distribution model and ii) it provides better quality of solutions than a single sequential run.

The hybrid models and other variations of the master-slave model are isolated attempts to improve the execution time of the existing approaches. Although they sometimes boast better results taking advantage of hardware architecture such as multiple GPUs they are not widely accepted as best approach to distributing SI.

In the case of fine grained master-slave model the communication overhead that is necessary to exchange solutions is too large. The small amount of research using this model and the conflicting opinions about its effectiveness suggests that it is being abandoned as a viable model and is not, in fact, the state of the art approach to distributing SI.

Most authors focus on the coarse grained master-slave model or the island approach. Obtaining top efficiency when distributing their implementations on 8 to 25 nodes distribution. The approaches that were tested on TSP generally used graphs from the benchmark library TSPLIB [97] consisting of up to 500 vertices. The exception is the work presented in [120], where an island model was tested on a 15 000 vertex TSP instance of TSPLIB. However, efficiency peaked at 8 nodes. The most scalable approach in all the considered papers was presented in [20], an island model that reached peak efficiency at 25 nodes. This was tested on a 318 vertex TSP instance. However, the peak efficiency was inferior to maximum efficiency we encountered of 0.9 , obtained in paper [21] using 8 computing nodes. We found significant research sustaining the superior asynchronous communication for the island model. It is our conclusion supported by the large number of papers written on the subject and the outstanding results, that the island approach is the current state of the art model of distributing SI. These distributing models of SI may be criticized for not being especially designed for SI approaches. Therefore they do not take advantage of the inherently distributed nature of the SI approaches they are using.

In the case of agent-based approaches to distributed SI, the authors model entities and vertices as agents. Although this type of modeling is fully distributed, the separation of SI entities and search environment results in a large amount of messaging. All the necessary actions are done through messages: visiting a vertex, current solutions update any other SI specific action. Agents can run on the same computing node or on separate nodes. In the latter messaging between agents is slower due to network connection delay.

The frameworks presented in papers [7, 32, 31] are designed for the coarse grained master-slave approach. On the other hand, [61] is much better suited for the implementation of a distributed environment that migrates entities in the form of "smart messages", according to the particular SI rules. However, again, this does not motivate the use of code mobility since all entities are governed by the same behavioral rules. [61] presents a purely theoretical design, on the other hand, the somewhat similar distributed constraint architecture called "DisChoco" from paper [49] has been tested and proved to be scalable.

References

[1] E. Alba, G. Leguizamon, and G. Ordonez, Analizing the behavior of parallel ant colony systems for large instances of the task scheduling problem, in *Proceedings of the 19th International Parallel and Distributed Processing Symposium*, IEEE Computer Society, page 14, 2005.

[2] E. Alba, G. Luque, J. Garcie-Nieto, G Ordonez, and G. Leguizamon, Two models of parallel ACO algorithms for the minimum tardy task problem, *International Journal of High Performance Systems Architecture* **1** (2007), no. 1, 74–85.

[3] D.L. Applegate, R.E. Bixby, V. Chvatal, and W.J. Cook, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, 2006.

[4] K. Arnold, J. Gosling, and D. Holmes, *The Java Programming Language*, Fourth Edition, Addison-Wesley Professional, 2005.

[5] C. Bădică, S. Ilie, and M. Ivanović, Optimizing Communication Costs in ACODA Using Simulated Annealing: Initial Experiments, In *Computational Collective Intelligence, Technologies and Applications*, Lecture Notes in Computer Science **7653**, Springer Berlin, Heidelberg, 2012, 298–307.

[6] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, Artificial bee colony algorithm on distributed environments, in *Nature and Biologically Inspired Computing* (NaBIC), 2010 Second World Congress on, IEEE, 13–18.

[7] D. Barbucha, I. Czarnowski, P. Jedrzejowicz, E. Ratajczak, and Iza Wierzbowska, Jade-based a-team as a tool for implementing population-based algorithms. In *Proc. 6th International Conference on Intelligent Systems Design and Applications*: ISDA2006, IEEE Computer Society, 2006, 144–149.

[8] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*, John Wiley & Sons Ltd, 2007.

[9] C.M. Vargas Benítez, and H.S. Lopes, Parallel Artificial Bee Colony Algorithm Approaches for Protein Structure Prediction Using the 3DHP-SC Model, In *Intelligent Distributed Computing IV, Studies in Computational Intelligence* **315** (2010), Springer Berlin / Heidelberg, 255–264.

[10] C.M. Vargas Benítez, R.S. Parpinelli, and H.S. Lopes, Parallelism, Hybridism and Coevolution in a Multi-level ABC-GA Approach for the Protein Structure Prediction Problem, *Journal of Concurrency and Computation: Practice and Experience* **24** (2012), no. 6, 635-646.

[11] D. Bertsimas, and J. Tsitsiklis, Simulated annealing, *Statistical Science* **8** (1993), no. 1, 10–15.

[12] J.C. Biesmeijer, and T.D. Seeley, The use of waggle dance information by honey bees throughout their foraging careers, *Behavioral Ecology and Sociobiology* **59** (2005), no. 1, 133–142.

[13] M. Bolondi, and M. Bondaza, Parallelizzazione di un algoritmo per la risoluzione del problema del comesso viaggiatore, Master's Thesis, Politecnico di Milano, Itally, 1993.

[14] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*, Society for Industrial and Applied Mathematics, 2009.

[15] B. Bullnheimer, G. Kotsis, and C. Strauss, Parallelization Strategies for the Ant System, In (*High Performance Algorithms and Software in Nonlinear Optimization*, Kluwer International Series in Applied Optimization **24**, (R. De Leone and A. Murli and P. M. Pardalos and G. Toraldo, Eds.), Kluwer Academic Publishers, Dordrecht, 1998, 87–100.

[16] M.S.F. Catalano, and F. Malucelli, Parallel randomized heuristics for the set covering problem, In *Practical parallel computing*, (M. Paprzycki, L. Tarricone, and L. T. Yang, Eds.), Nova Science Publishers, Inc., Commack, NY, USA, 2001, 113–132.

[17] L. Chen, and C. Zhang, Adaptive parallel ant colony optimization, In *Parallel and Distributed Processing and Applications*, (Y. Pan, D. Chen, M. Guo, J. Cao, and J. Dongarra, Eds.) Lecture Notes in Computer Science **3758**, Springer Berlin / Heidelberg, 2005, 275–285.

[18] L. Chen, and C. Zhang. Adaptive parallel ant colony algorithm, In *Proceedings of the 1st International Conference in Advances in Natural Computation*, Lecture Notes on Computer Science 3611, 1239 – 1249, 2005.

[19] Y. Chen, L. Chen, and L. Tu, Parallel ant colony algorithm for mining clasification rules, In *Proceedings of IEEE International Conference on Granular Computing*, IEEE Press, 2006, 85–90.

[20] L. Chen, H.-Y. Sun, and S. Wang, Parallel implementation of ant colony optimization on MMP, In *Proceedings of the international conference on Machine Learning and Cybernetics*, volume 2, IEEE Press, 2008, 981–986.

[21] J. Chintalapati , M. Arvind, S. Priyanka, N. Mangala, and J. Valadi, Parallel Ant-Miner (PAM) on High Performance Clusters, In *Swarm, Evolutionary, and Memetic Computing*, Lecture Notes on Computer Science, Springer Berlin/ Heidelberg, 2010, 270–277.

[22] C. Chira, C.-M. Pintea, and D. Dumitrescu, Learning sensitive stigmergic agents for solving complex problems, In *Computing and Informatics* **29** (2010), 1001–1020.

[23] S.-C. Chu, J.F. Roddick, and J.-S. Pan, Ant colony system with communication strategies, *Inf. Sci. Inf. Comput. Sci.* **167** (2004), 63–76.

[24] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Q. Yang, and G. Webb, Cat Swarm Optimization, *PRICAI 2006: Trends in Artificial Intelligence*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2006, 854–858.
[25] D. Chu, and A.Y. Zomaya, Parallel ant colony optimization for 3D protein structure prediction using the HP lattice model, In *Parallel Evolutionary Computations*, Studies in Computational Intelligence **22**, Springer, 2006, 177–198.
[26] H. H. Clark, *Using Language*, Cambridge University Press, 1996.
[27] M. Clerc, *Particle Swarm Optimization*, ISTE, 2006.
[28] T. H Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (third edition), MIT Press, 2009.
[29] M. Craus, and L. Rudeanu, Multi-level parallel framework, *International Journal of Computing*, IEEE Computer Society **3** (2002), no. 3.
[30] M. Craus, and L. Rudeanu, Parallel framework for ant-like angorithms, In *Proceedings of the 3rd International Symposium on Parallel and Distributed Computinf*, IEEE Computer Society, 2004, 36– 41.
[31] M. Craus, and L. Rudeanu, Parallel framework for cooperative processes, *Sci. Program.* **13** (2005), 205 – 217.
[32] I. Czarnowski, P. Jedrzejowicz, and I. Wierzbowska, Ateam middleware on a cluster. In *Proc.3rd KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*: KES-AMSTA2009, volume 5559, Springer-Verlag, 2009, 764 – 772.
[33] Al-Dahoud Ali, M.A. Belal, and M.B. Al-Zoubi, Load balancing of distributed systems based on multiple ant colonies optimization, *American Journal of Applied Sciences* **3** (2010), 428–433.
[34] T. Davidovic, D. Ramljak, M Selmic, and D. Teodorovic, MPI parallelization of bee colony optimization, *Proc. of the 1st International Symposium & 10th Balkan conference on Operational Research*, vol. 2, 193 –200, 2011.
[35] T. Davidovic, T. Jaksic, D. Ramljak, M Selmic and D. Teodorovic. MPI Parallelization Strategies for Bee Colony Optimization, *Special Issue of Optimization entitled "Advances in Discrete Optimization"*, dedicated to BALCOR 2011.
[36] P. Delisle, M. Krajecki, M. Gravel, and C. Gagne, Parallel implementation of an ant colony optimization metaheuristic with OpenMp. In *Proceedings of the Third European Workshop on OpenMP* (EWOMP01), 2001.
[37] P. Delisle, M. Gravel, M. Krajecki, C. Gagne, and W.L. Price, A shared memory parallel implementation of ant colony optimization. In *Proceedings of the 6th Metaheuristics International Conference*, Vienne, Autriche, 2005, p. 257.
[38] P. Delisle, M. Gravel, M. Krajecki, C. Gagne, and W.L. Price, Comparing parallelization of an aco: Message passing vs. shared memory. In *Hybrid Metaheuristics*, 2005, 1–11.
[39] S. Depickere, D. Fresneau, and J. Deneubourg. Effect of social and environmental factors on ant aggregation: A general response?, *Journal of insect physiology* **54** (2008), no. 9, 1349–1355.
[40] C. Detrain, and J.-L. Deneubourg, Self-organised structures in a superorganism: do ants "behave" like molecules?, *Physics of Life Reviews 3* **3** (2006), no. 3, 162–187.
[41] K. Doerner, R. Hartl, and M. Lucka, A parallel version of the D-ant algorithm for the vehicle routing problem, in *Proceedings of the International Workshop on Parallel Numerics*, 2005, 109–118.
[42] K. Doerner, R. Hartl, S. Benkner, and M. Lucka, Parallel cooperative savings based ant colony optimization- multiple search and decomposition approaches, *Parallel Processing Letters* **16** (2006), no. 3, 351–370.
[43] M. Dorigo, G. Di Caro, and L. Garambella, Ant algorithms for discrete optimization, *Artificial Life* **5** (1999), no. 2, 137–172.
[44] M. Dorigo, and G. Di Caro, *The ant colony Optimization Meta-Heuristic, New Ideas in Optimization*, McGraw-Hill, 1999, 11–36.
[45] M. Dorigo, and T. Stutzle, *Ant Colony Optimization*, MIT Press, 2004.
[46] F. C. Dyer, The biology of the dance language, *Annual Review of Entomology* **47** (2002), 917–949.
[47] S. Edelkamp, and S. Schroed, *Heuristic Search: Theory and Applications*, Elsevier, 2012.
[48] A. P. Englebrecht, *Fundamentals of computational swarm intelligence*, Wiley, 2005.
[49] R. Ezzahir, C. Bessiere, M. Belaissaoui, and E. Bouyakhf, Dischoco: A platform for distributed constraint programming, In: *Proceedings of the Eighth International Workshop on Distributed Constraint Reasoning* at IJCAI'07, 2007.

[50] S. M. Fattahi, and N. M. Charkari, Ant distributed acs algorithm for resource discovery in grid, *International Journal of the Computer, the Internet and Management* **17** (SP1), 2009.
[51] The Foundation for Intelligent Phisical Agents (FIPA) webpage. http://www.fipa.org/
[52] The Agent Communication Language (ACL) FIPA Standard webpage. http://www.fipa.org/specs/fipa00061/SC00061G.html
[53] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd Edition), Addison-Wesley Professional, 2003.
[54] The W3C File Transfer Protocol specifications http://www.w3.org/Protocols/rfc959/
[55] J. Fu, L. Le, and G.Zhou, A parallel ant colony optimization algorithm with gpu-acceleration based on all-in-roulette selection, In *Proceedings of the 3rd International Workshop on Advanced Computational Intelligence* (2010), 260–264.
[56] D. Gao, G. Gong, L. Han, and N. Li, Application of multi-core parallel ant colony optimization in target assignment problem, In *Proceedings of the international Conference on Computer Application and System Modeling* **3** (2010), 514–518.
[57] B.L. Golden, S. Raghavan, and E.A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges*, Springer, 2008.
[58] D. Goldberg, and K. Sastry, *Genetic Algorithms: The Design of Innovation*, Springer, 2012.
[59] J.L. Gross, and J. Yellen, *Graph Theory and Its Applications*, Chapman & Hall/CRC, 2006.
[60] B. Hendrickson and T.G. Kolda, *Graph partitioning models for parallel computing. Parallel Computing*, **26** (2000), 1519–1534.
[61] T. Holvoet, D. Weyns, and P. Valckenaers, Patterns of delegate mas. *International Conference on Self-Adaptive and Self-Organizing Systems* (2009), 1–9.
[62] H.H. Hoos, and T. Stutzle, *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann, 2005.
[63] S. Ibri, H. Drias, and M. Nourelfath, A parallel hybrid ant-tabu algorithm for integrated emergency vehicle dispatching and covering problem, *Internatonal Journal of Innovative Computing and Applications* **4** (2010), no. 2, 226–236.
[64] I. Iimura, K. Hamaguchi, T. Ito, and S. Nakayama, A study of distributed parallel processing for queen ant strategy in ant colony optimization, In *Proceedings of the 6th international Conference on Parallel and Distributed Computing Applications and Technologies*, IEEE Computer Society, 553–557, 2005.
[65] N. Santoro, *Design and Analysis of Distributed Algorithms*, Wiley Series on Parallel and Distributed Computing, Wiley-Interscience, 2006.
[66] R. Jovanovic, M. Tuba, and D. Simian, Analysis of parallel implementations of the ant colony optimization applied to the minimum weight vertex cover problem, in *Proceedings or the 9th World Scientific and Engineering Academy and Society International Conference on Simulation, Modeling and Optimization*, 254–259, 2009.
[67] R. Jovanovic, M. Tuba, and D. Simian, Comparison of different topologies for island-based multicolony ant algorithms for the minimum weight vertex cover problem, *WSEAS Transactions on Computers* **9** (2010), no. 1, 83–92.
[68] Z. Lee, C. Lee, and S. Su, Parallel ant colony optimizers with local and global ants, In *Proceedings or the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, IEEE Computer Society, 2009, 1655–1659.
[69] X. Li, X. Yu, and X. Luo, Parallel Implementation of Ant Colony Optimization for Vector Quantization Codebook Design, In *Proceedings of the Third International Conference on Natural Computation* (ICNC '07) **4**, 2007, IEEE Computer Society, Washington, DC, USA, 787–791.
[70] N. Li, D. Gao, G. Gong, and Z. Chen, Realization of parallel ant colony algorithm based on tbb multi-core platform, In *Proceedings of the international Forum on Information Technology and Applications*, 2010, 177 - 180.
[71] C. Liu, L. Li, and Y. Xiang, Research of multi-path routing protocol based on parallel ant colony algorithm optimization in mobile ad-hoc networks, In *Proceedings of the 5th International Conference on Information echnology: New Generations*, IEEE Computer Society, 2008, 1006–1010.
[72] M. Lopez-Ibanes, T. Prasad, and B. Paechter, Parallel optimization of pump schedules with a thread-safe variant of epanet toolkit, In *Geotechnical Special Publication* **187**, 2009, 462–471.
[73] M. Lucka, and S. Piecka, Parallel posix threads based on ant colony optimization using asynchronous communication, In *Proceedings of the 8th International Conference on Applied Mathematics* **2**, 2009, 229–236.

[74] R. Luo, T.-S. Pan, P.-W. Tsai, and J.-S. Pan, Parallelized Artificial Bee Colony with Ripple-communication Strategy, In *Proceedings of the 2010 Fourth International Conference on Genetic and Evolutionary Computing* (ICGEC '10), 2010, IEEE Computer Society, Washington, DC, USA, 350–353.

[75] Q. Lv, X. Xia, and P.Qian, A parallel ACO approach based on pheromone matrix, In *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence*, Lecture Notes on Computer Science **4150**, 2006, 332–339.

[76] M. Manfrin, M. Birattari, T. Stutzle, and M. Dorigo, Parallel ant colony optimization for the traveling salesman problem, In (M. Dorigo, L. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Sttzle, ed s) *Ant Colony Optimization and Swarm Intelligence*, Lecture Notes in Computer Science 4150, Springer Berlin / Heidelberg, 2006, 224–234.

[77] MATSim - Multi-Agent Transport Simulation Toolkit, official homepage at: http://www.matsim.org.

[78] Y. Meng, O. Kazeem, and J. C. Muller, A Hybrid ACO/PSO Control Algorithm for Distributed Swarm Robots, In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium* (SIS 2007), 2007, 273–280.

[79] M. Middendorf, F. Reischle, and H. Schmeck, Multi Colony ant algorithms, *Journal of Heuristics* **8** (2002), 305–320.

[80] A. Mocanu, S. Ilie and C. Bădică, Ubiquitous Multi-agent Environmental Hazard Management System, *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* Timisoara (SYNASC), Romania September 26 - 29, 2012, Workshop on Agents for Complex Systems ACSys, 2012 accepted paper.

[81] J. Mocholi, J. Martinez, and J. Canos, A grid ant colony algorithm for the orienteering problem, In *Proceedings of the IEEE congress of evolutionary computations*, IEEE Press, 2005, 942–949.

[82] MPICH2 website: www.mcs.anl.gov/research/projects/mpich2/

[83] S. Muthuswamy, and S. Lam, Discrete particle swarm optimization for the team orienteering problem, *Journal of Memetic Computing* **3** (2011), no. 4, 287–303.

[84] H. Narasimhan, Parallel artificial bee colony (PABC) algorithm, *World Congress on Nature & Biologically Inspired Computing*, NaBIC 2009, 306–311.

[85] The ns2 project, a network simulation tool, http://nsnam.isi.edu/nsnam/index.php/.

[86] M. Pedemonte, and H. Cancela, A cellular ant colony optimization for the generalized Steiner problem, *International Journal of Innovative Computing and Appllications* **2** (2010), no. 3, 188–201.

[87] M. Pedemonte, S. Nesmachnow, and H. Cancela, A survey on parallel ant colony optimization, *Applied Soft Computing* 11 (2011), no. 8), 5181–5197.

[88] R.S. Parpinelli, C.M.V. Benitez, and H.S. Lopes, Parallel Approaches for the Artificial Bee Colony Algorithm, (B.K. Panigrahi, Y. Shi, M.-H. Lim eds.) *Handbook of Swarm Intelligence*, Springer Berlin Heidelberg, 2010, 329–345.

[89] R.S. Parpinelli, C.M.V. Benitez, and H.S. Lopes, Parallel Approaches for the Artificial Bee Colony Algorithm, In *Handbook of Swarm Intelligence: Concepts, Principles and Applications*, Series: Adaptation, Learning, and Optimization, Springer, Berlin, 2011, 329–346.

[90] S.N. Parragh, K.F. Doerner, and R.F. Hartl, A survey on pickup and delivery problems, *Journal für Betriebswirtschaft*, 58 (2008), no. 2, 81 - 117.

[91] W. Peng, R. Tong, M. Tang, and J. Dong, Ant Colony Search Algorithms for Optimal Packing Problem, In *Proceedings Advances in Natural Computation*, First International Conference, Changsha, China, August 2729, ICNC (2), Springer, 2005, 1229–1238.

[92] W. Peng, R. Tong, G. Qian, and J. Dong, A Constrained Ant Colony Algorithm for Image Registration, *Computational Intelligence and Bioinformatics, International Conference on Intelligent Computing*, ICIC 2006, Kunming, China, August 16 - 19, Springer, 2006.

[93] R. Pinchuk, S. Ilie, T. Neidhart, T. Dalmas, C. Bădică, and G. Pavlin, Augmenting Semantics to distributed agents logs Enabling graphical after action analysis of federated agents logs, Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management 2011.

[94] D.A.L. Piriyakumar, and P. Levi, A new approach to exploiting parallelism in ant colony optimization, In Proceedings of 2002 International Symposium on Micromechatronics and Human Science, MHS 2002, 237–243.

[95] M. Rahoual, R. Hadji, and V. Bachelet, Parallel ant system for the set covering problem, In Proceedings of the 3rd International Workshop on Ant Algorithms, Lecture Notes on Computer Science **2463**, 2002, 262–267.

[96] M. Randall, and A. Lewis, A parallel implementation of ant colony optimization, *J. Parallel Distrib. Comput.* **62** (2002), 1421–1432.

[97] G. Reinelt, Tsplib - a traveling salesman library, *ORSA Journal on Computing* **3** (1991), 376–384 .

[98] E. Ridge, E. Curry, D. Kudenko, and D. Kazakov, Parallel, asynchronous and decentralised ant colony system, In *Proc.1st International Symposium on Nature-Inspired Systems for Parallel, Asynchronous and Decentralised Environments* (NISPADE2006), 2006.

[99] O. Roozmand, and K. Zamanifar, Parallel ant miner 2, In *Proceedings of the 9th International Conference on Artificial Intelligence and Soft Computing*, Lecture Notes on Computer Science **5097**, 2008, 681–692.

[100] A. Sameh, A. Ayman, and N. Hassan, Parallel ant colony optimization, *International Journal of research and reviews in Computer Science* **1** (2010), no. 2, 77–82.

[101] S. A. Subbotina, and Al. A. Oleinik, Multiagent Optimization Based on the Bee-Colony Method, *Cybernetics and Systems Analysis* **45** (2009), no. 2, 177–186.

[102] M. Subotic, M. Tuba, and N. Stanarevic, Parallelization of the artificial bee colony (ABC) algorithm. In *Proceedings of the 11th WSEAS international conference on nural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems* (NN'10/EC'10/FS'10). World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 191–196, 2010.

[103] M. Subotic, M. Tuba, and N. Stanarevic, Different approaches in parallelization of the artificial bee colony algorithm, *International Journal of Mathematical Models And Methods in Applied Sciences* **5** (2011), no. 4, 755–762.

[104] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard, Parallel ant colonies for the quadratic assignment problem, *Future Generation Computer Systems* **17** (2001), no. 4, 441–449.

[105] D. Teodorovic, Transport modeling by multi-agent systems: a swarm intelligence approach, *Transportation Planning and Technology* **26** (2003), no. 4, 289–312.

[106] D. Thain, T. Tannenbaum, and M. Livny, Distributed computing in practice: the condor experience, *Concurrency and Computation, Practice and Experience* **17** (2005) (2-4), 323–356.

[107] I. Trencansky, and R. Cervenka, Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS, *Journal Informatica* **29** (2005), 391–400.

[108] S. Tsetsui, Parallel ant colony optimization for the quadratic assignment problem with local search and parallelization, In *Proceedings of the 2nd International Conference on Pattern recognition and Machine Intelligence*, Lecture Notes in Computer Science **4815**, 2007, 269–278.

[109] S. Tsutsui, Parallel ant colony optimization for quadratic assignment problems with symmetric multi processing, In *Proceedings of the 6th International conference on Ant Colony Optimization and Swarm Intelligence*, Lecture Notes in Computer Science **5217**, 2008, 363–370.

[110] S. Tsutsui, and N. Fujimoto, Parallel ant colony optimization algorithm on a multi-core processor, In *Proceedings of the 7th International Conference on Swarm intelligence*, Lecture Notes on Computer Science **6234**, 2010, 488–495.

[111] C. Twomey, T. Stutzle, M. Dorigo, M. Manfrin, and M. Birattari, An analysis of communication policies for homogeneous multi-colony aco algorithms, *Information Sciences* **180** (2010), no. 12, 2390–2404.

[112] H.F. Wedde, M. Farooq, and Y. Zhang, Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *Ant Colony Optimization and Swarm Intelligence*, (M. Dorigo, ed.), Springer Berlin, 2004, 83–94.

[113] H.F. Wedde, S. Lehnhoff, B. van Bonn, Z. Bay, S. Becker, S. Bottcher, C. Brunner, A. Buscher, T. Furst, A.M. Lazarescu, E. Rotaru, S. Senge, B. Steinbach, F. Yilmaz, and T. Zimmermann, A novel class of multi-agent algorithms for highly dynamic transport planning inspired by honey bee behavior, *Conference on Emerging Technologies and Factory Automation* ETFA, IEEE, 25 - 28 Sept. 2007, 1157–1164.

[114] H. Wedde, S. Senge, S. Lehnhoff, F. Knobloch, T. Lohmann, R. Niehage, and M. StrasserBee, Inspired Online Vehicle Routing, in *Large Traffic Systems, in Proceedings of ADAPTIVE 2010 : The Second International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2010.

[115] G. Weiss, and A. Lewis, Using XMPP for ad-hoc grid computing - an application example using parallel ant colony optimization, In *Proceedings of the International Symposium on Parallel and Distributed Processing*, 2009, 1–4.

[116] L.-P. Wong, M.Y. Hean Low, and C. S. Chong, Bee Colony Optimization with Local Search for Travelling Salesman Problem, *International Journal on Artificial Intelligence Tools* **19** (2010), no. 3, 305–334.
[117] M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons Ltd, 2002.
[118] C. Thomas Wu, *An Introduction to Object-Oriented Programming with Java*, McGraw-Hill Higher Education, 2009.
[119] W. Xiang, and H. P. Lee, Ant colony intelligence in multi-agent dynamic manufacturing scheduling, *Engineering Applications of Artificial Intelligence* **21** (2008), no. 1, 73–85.
[120] J. Xiong, C. Liu, and Z. Chen, A new parallel ant colony optimization algorithm based on message passing interface, In *Proceedings of the international Symposium on Parallel and Distributed Processing*, 2008, 178–182.
[121] Z. Yang, B. Yu, and C. Cheng, A parallel ant colony algorithm for bus network optimization, *Computer-Aided Civil and Infrastructure Engineering* number **22** (2007), no. 1, 44-55.
[122] B. Yu, Z. Yang, and J. Xie, A parallel improved ant colony optimization for multi-depot vehicle routing problem, *Journal of the Operational Research Society* **62** (2011), no. 1, 183–188.
[123] S. Ilie, and C. Bădică, Multi-agent approach to distributed ant colony optimization, *Journal of Science of Computer Programming - SCP* 78 (2013), no. 6, 762–774.

(Sorin Ilie) Department of Computers and Information Technology, University of Craiova, 13 A.I. Cuza Street, Craiova, 200585, Romania
E-mail address: silie@software.ucv.ro