Computing and Informatics, Vol. 29, 2010, 337–356

LEARNING SENSITIVE STIGMERGIC AGENTS FOR SOLVING COMPLEX PROBLEMS

Camelia Chira, Dumitru Dumitrescu

Department of Computer Science
Babes-Bolyai University
Cluj-Napoca 400084
Romania
e-mail: {cchira, ddumitr}@cs.ubbcluj.ro

Camelia-Mihaela Pintea

George Cosbuc N. College
Cluj-Napoca 400083
Romania
e-mail: cmpintea@yahoo.com

Manuscript received 3 December 2007; revised 10 June 2009
Communicated by Ivana Budinská

Abstract. Systems composed of several interacting autonomous agents have a huge potential to efficiently address complex real-world problems. Usually agents communicate by directly exchanging information and knowledge about the environment. The aim of the paper is to develop a new computational model that endows agents with a supplementary interaction/search mechanism of stigmergic nature. Multi-agent systems can therefore become powerful techniques for addressing NP-hard combinatorial optimization problems. In the proposed approach, agents are able to indirectly communicate by producing and being influenced by pheromone trails. Each stigmergic agent is characterized by a certain level of sensitivity to the pheromone trails. The non-uniform pheromone sensitivity allows various types of reactions to a changing environment. For efficient search diversification and intensification, agents can learn to modify their sensitivity level according to environment characteristics and previous experience. The resulting system for solving complex problems is called Learning Sensitive Agent System (LSAS). The proposed LSAS model is used for solving several NP-hard problems such as the Asymmetric and

Generalized Traveling Salesman Problems. Numerical experiments indicate the robustness and the potential of the new metaheuristic.

Keywords: Stigmergy, agents, ant colony systems, combinatorial optimization, learning

Mathematics Subject Classification 2000: 68T20

1 INTRODUCTION

Many optimization problems are NP-hard and therefore cannot be solved within polynomial computation times. NP-hard problems arise in many and diverse domains including network design, scheduling, mathematical programming, algebra, games, language theory and program optimization. Metaheuristics are powerful strategies to efficiently find high-quality near optimal solutions within reasonable running time for problems of realistic size and complexity.

A metaheuristic combining stigmergic behaviour and agent direct communication called *Learning Sensitive Agent System (LSAS)* is proposed. The LSAS model involves several two-way interacting agents [3, 5] endowed with learning capabilities that allows them to explore the search space more efficiently. LSAS agents can communicate by directly exchanging messages using an Agent Communication Language [13, 14, 16, 19]. The information directly obtained from other agents is very important in the search process and can become critical in a dynamic environment (where the latest changes in the environment can be instantly made available to other agents).

The LSAS model is engaged in solving various instances of the Asymmetric and Generalized Traveling Salesman Problems. Numerical results indicate a competitive performance of the proposed system compared to related state-of-the-art methods.

The paper is organized as follows: the idea of stigmergy is described and the Ant Colony Optimization metaheuristic is shortly presented; the notions of agent and multi-agent system are introduced with a focus on agent communication; the LSAS model is presented describing the communication mechanisms (direct and stigmergic) of agents, the concept of pheromone sensitivity for stigmergic agents and the learning mechanism engaged in the model; the LSAS numerical results and comparisons with other methods are presented; the conclusions of the paper and directions for future research are given.

2 STIGMERGY. ANT COLONY OPTIMIZATION

Metaheuristics inspired from nature represent a powerful and robust approach to solve NP-difficult problems. Biology studies emphasize the remarkable solutions

that many species managed to develop after millions of years of evolution. Self-organization [1] and indirect interactions between individuals make possible the identification of intelligent solutions to complex problems. These indirect interactions occur when one individual modifies the environment and other individuals respond to the change at a later time. This process refers to the idea of *stigmergy* [10].

The bio-inspired Ant Colony Optimization (ACO) model [6, 7] simulates real ant behavior to find the minimum length path between the ant nest and the food source. An ant algorithm is essentially a system based on agents that simulate the natural behavior of ants including mechanisms of cooperation and adaptation. This approach induces the development of a new metaheuristic that has been successfully used to solve combinatorial optimization problems.

Ant algorithms are based on the following main ideas:

- Each path followed by an ant is associated with a candidate solution for a given problem.
- When an ant follows a path, the amount of pheromone deposited on that path is proportional to the quality of the corresponding candidate solution for the target problem.
- When an ant has to choose between two or more paths, the path(s) with a larger amount of pheromone has(have) a greater probability of being chosen. As a result, ants eventually converge to a short path which hopefully represents the optimum or a near-optimum solution for the target problem.

Well known and robust algorithms include Ant Colony System [6, 9] and MAX-MIN Ant System [17].

3 COMMUNICATION. AGENTS AND MULTI-AGENT SYSTEMS

Autonomous agents have been the focus of researchers and developers from disciplines such as AI, object-oriented programming, concurrent object-based systems and human-computer interface design [3, 16]. Although there is no universally accepted agent definition, researchers and scientists generally agree that an agent acts on behalf of its user, is situated in an environment and is able to perceive that environment, has a set of objectives and takes actions so as to accomplish these objectives and is autonomous [3]. The main properties of an agent can be summarised as follows [2, 3, 13, 14, 16, 18]:

Autonomy: The ability to operate on its own without the intervention of humans or other systems.

Reactivity: The ability to perceive its environment and to respond to changes that occur in it.

Pro-activeness: The ability to take the initiative in order to pursue its individual goals (goal-directed behaviour).

Cooperation (or social ability) The capability of interacting with other agents and possibly humans via an agent-communication language. Involves the ability of an agent to dynamically negotiate and coordinate.

Learning: The ability to learn while acting and reacting in its environment. Learning can increase performance of an agent over time.

Mobility: The ability to move around a network in a self-directed way.

Furthermore, some researchers identify more properties associated with the notion of agency including temporal continuity, personality, veracity, benevolence and rationality.

Characterized by computational efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility and reuse, *multi-agent systems (MAS)* promote conceptual clarity and simplicity of design [2, 3]. A multi-agent approach to developing complex systems involves the employment of several agents capable of interacting with each other to achieve objectives [13]. The benefits of such an approach include the ability to solve large and complex problems, interconnection and interoperation of multiple existing legacy systems and the capability to handle domains in which the expertise is distributed [2, 3, 19].

Interoperation among autonomous agents of MAS is essential for the successful identification of a solution to a given problem. Agent-oriented interactions span from simple information interchanges to planning of interdependent activities for which cooperation, coordination and negotiation are fundamental.

Coordination is necessary in MAS because agents have different and limited capabilities and expertise. Agents have to coordinate their activities in order to determine the organizational structure in a group of agents and to allocate tasks and resources. Furthermore, interdependent activities require coordination (the action of one agent might depend on the completion of a task for which another agent is responsible).

Negotiation is essential within MAS for conflict resolution and can be regarded as a significant aspect of the coordination process among autonomous agents [2, 15].

Agents need to communicate in order to exchange information and knowledge or to request the performance of a task as they only have a partial view over their environment [2]. Considering the complexity of the information resources exchanged, agents should communicate through an agent communication language (ACL). Standard ACLs designed to support interactions among intelligent software agents include the Knowledge Query and Manipulation Language (KQML) proposed by the Knowledge Sharing Effort consortium [8] and FIPA ACL defined by the FIPA organization [11]. Both KQML and FIPA ACLs are designed to be independent of particular application vocabularies.

4 DIRECT AND STIGMERGIC INTERACTIONS

The proposed *Learning Sensitive Agent System (LSAS)* combines stigmergic behaviour and agent direct communication. The LSAS model involves several two-way

interacting agents [4, 5]. Agents are endowed with stigmergic behaviour similar to that of Ant Colony Systems [6, 7]. This means that each agent is able to produce pheromone trails that can influence future decisions of other agents. LSAS agents are characterized by a certain level of sensitivity to the pheromone trail allowing various types of reactions to a changing environment [5]. Furthermore, LSAS agents are endowed with learning capabilities that allow them to explore the search space more efficiently.

4.1 Direct Communication in LSAS

LSAS agents are able to exchange different types of messages in order to share direct knowledge and support interoperation. The content of the messages exchanged refers to environment characteristics and partial solutions obtained. The information about dynamic changes in the environment is of significant importance in the search process. The content of the messages exchanged depends highly on the problem being solved.

Furthermore, the LSAS model inherits agent properties such as autonomy, reactivity, learning, mobility and pro-activeness used in multi-agent systems. The agents that form the system have the ability to operate without human intervention, can cooperate to exchange information and can learn while acting and reacting in their environment. The learning mechanism used in the proposed model is detailed in Section 5.

4.2 Stigmergic Communication in LSAS

LSAS agents are endowed with the ability to produce pheromone trails that can influence future decisions of other agents within the system. The stigmergic behaviour of the LSAS agents is similar to that of the ants in the bio-inspired Ant Colony Optimization metaheuristic [6, 7, 9].

Let us consider that agents solve problems by finding a path from an intial state to a final state. Each state in the search space is represented by a node in a graph and a transition rule has to be specified.

If an agent is sensitive to stigmergic information, stronger pheromone trails are preferred and the most promising paths receive a greater pheromone trail after some time. The result of the algorithm is a sequence of states – corresponding to the optimal (or a near-optimal) solution of the given problem.

Let β be a parameter used for tunning the relative importance of edge length in selecting the next node. Let us denote by ${J^k}_i$ the unvisited successors of node i by agent k and $u \in {J^k}_i$. q is a random variable uniformly distributed over $[0, 1]$ and q_0 is a parameter similar to the temperature in simulated annealing, $0 \leq q_0 \leq 1$.

If $q > q_0$ the probability p_{iu} of choosing $j = u$ as the next node from the current node i is defined as [6, 7]:

$$p_{iu}(t) = \frac{[\tau_{iu}(t)][\eta_{iu}(t)]^{\beta}}{\Sigma_{o \in J^k{}_i}[\tau_{io}(t)][\eta_{io}(t)]^{\beta}}, \tag{1}$$

where

- $\tau_{iu}(t)$ refers to the pheromone trail intensity on edge (i, u) at time t, and
- $\eta_{iu}(t)$ represents the visibility of edge (i, u).

If $q \leq q_0$ the next node j is chosen according to the following rule [6, 7]:

$$j = argmax_{u \in J_i^k}\{\tau_{iu}(t)[\eta_{iu}(t)]^{\beta}\}. \tag{2}$$

4.3 Pheromone Sensitivity

Within the LSAS model each agent is characterized by a pheromone sensitivity level denoted by PSL which is expressed by a real number in the unit interval [0, 1]. Extreme situations are:

- If PSL = 0 the agent completely ignores stigmergic information (the agent is 'pheromone blind');
- If PSL = 1 the agent has maximum pheromone sensitivity.

Low PSL values (below a specified threshold a, which is a parameter of the algorithm) indicate that the agent tends to make decisions based on information received from other agents. Unpromising states in a list generated by direct communication are avoided in random decision making. However, if stigmergic infomation is considered in the decision making process, only very high pheromone marked moves will be considered (as the agent has reduced pheromone sensitivity). These agents are more independent and can be considered as environment explorers. They have the potential to autonomously discover new promising regions of the solution space. Therefore, search diversification can be sustained.

Agents with high PSL values (PSL $> a$) can choose any pheromone marked move. Agents of this category are able to intensively exploit the promising search regions already identified. In this case the agent's behaviour emphasizes search intensification.

The role of the pheromone sensitivity index is twofold. On one hand, an agent discriminates between direct and stigmergic interaction based on its PSL value. On the other hand, in the case of stigmergic behavior the agent's PSL influences the state transition mechanism. The LSAS specific transition mechanism is described in Section 5.

The PSL value for each agent in the proposed model is of major importance. There is no universal mechanism for setting an efficient value for this parameter for

each agent in the system. A natural idea is to update the PSL by using a learning procedure.

5 THE LEARNING SENSITIVE AGENT SYSTEM MODEL

This section further specifies the LSAS model by defining the renormalized transition probabilities and the learning rules used by agents. Furthermore, the corresponding algorithm for the LSAS model is described.

5.1 Renormalized Transition Probabilities in LSAS

LSAS agents that decide to use stigmergic information (pheromone trails) in choosing the path use transition probabilities similar to those described above but strongly influenced by the PSL value.

A measure of randomness proportional to the level of PSL is introduced in the decision process. It is proposed to achieve this by modifying the transition probabilities using the PSL values in a renormalization process. Consider $p_{iu}(A,t)$ as the probability for agent A of choosing the next node u from current node i (as given in (1) above).

Let us denote by $sp_{iu}(A,t)$ the renormalized transition probability for agent A (influenced by PSL) used in the LSAS model. In the proposed LSAS approach renormalization is accomplished via the following equation:

$$sp_{iu}(A,t) = p_{iu}(A,t)\mathrm{PSL}(A,t), \tag{3}$$

where $\mathrm{PSL}(A,t)$ represents the PSL value of agent A at time t.

It should be noted that if

$$\mathrm{PSL}(A,t) \neq 1 \tag{4}$$

then for each node i we have

$$\sum_u sp_{iu}(A,t) < 1. \tag{5}$$

In order to associate a standard probability distribution to the system, a *virtual state* denoted by *vs* – corresponding to the 'lost' probability – is introduced.

The transition probability associated to the virtual state *vs* is defined as

$$sp_{i,vs}(A,t) = 1 - \sum_u sp_{iu}(A,t). \tag{6}$$

Therefore, for agent k at moment t we may write

$$sp_{i,vs}(A,t) = 1 - \mathrm{PSL}(A,t)\sum_u p_{iu}(A,t), \tag{7}$$

and thus

$$sp_{i,vs}(A,t) = 1 - \mathrm{PSL}(A,t). \tag{8}$$

The renormalized probability $sp_{i,vs}(A,t)$ can be correlated to the system heterogeneity at time t.

The LSAS approach has to specify the action associated with the virtual state introduced. If the selected state is vs then the agent selects one of the following two strategies in a random manner:

- An accessible state is chosen randomly with uniform probability.
- The available states are ranked based on the information received from other agents and the knowledge base. The most promising state has the highest probability of being selected by the agent.

Proposed LSAS approach ensures the increasing of randomness in the selection process with the decreasing of the pheromone sensitivity level PSL for an agent that makes decisions based on stigmergic information.

5.2 Learning in LSAS

LSAS agents can learn to adapt their PSL according to the environment characteristics (and based on previous experience) facilitating an efficient and balanced exploration and exploitation of the solution space. The initial PSL values are randomly generated.

During their lifetime agents can potentially improve their performance by learning. This process translates to modifications of the pheromone sensitivity. The PSL value can increase or decrease according to the search space topology encoded in the agent's experience.

Low sensitivity of agents to pheromone trails encourages a good initial exploration of the search space. High PSL values emphasize the exploitation of previous search results.

Several learning mechanisms can be engaged at individual or global level. A simple reinforcing learning mechanism is proposed in the current LSAS model. According to the quality of the detected solution, the PSL value is updated for each agent after a complete solution is created.

Agents with high PSL value (above a specified threshold γ) are environment exploiters and they will be encouraged to further exploit the search region by increasing their PSL value each time a good solution is determined. Agents with small PSL value are good explorers of the environment and good solutions will be rewarded by decreasing agent PSL value (emphasizing space exploration). In the current paper, the most natural choice of threshold γ is considered, $\gamma = a$ (where a refers to the PSL threshold modulating the agent behavior). More elaborated schemes for setting the parameter γ as a function of a may induce a more sophisticated behavior.

$\mathrm{PSL}(A,t)$ represents the PSL value of the agent A at iteration t and $S(A,t)$ is the solution detected. The best solution determined by the system agents (until

iteration t) is denoted by *Best*(t). The proposed learning mechanism defines complementary approaches depending on the current value of $\mathrm{PSL}(A,t)$. If $\mathrm{PSL}(A,t) > \gamma$ the following rules apply:

- If $S(A,t)$ is better than *Best*(t) then A is rewarded by increasing its PSL value according to the following learning rule:

$$\mathrm{PSL}(A,t+1) = \min\left(1, \mathrm{PSL}(A,t) + e^{-\frac{\mathrm{PSL}(A,t)}{(t+1)^2}}\right). \tag{9}$$

- If $S(A,t)$ is worse than *Best*(t) then A is 'punished' by decreasing its PSL value according to the following learning rule:

$$\mathrm{PSL}(A,t+1) = \max\left(0, \mathrm{PSL}(A,t) - e^{-\frac{\mathrm{PSL}(A,t)}{(t+1)^2}}\right). \tag{10}$$

If $\mathrm{PSL}(A,t) \leq \gamma$ the following rules are engaged:

- If $S(A,t)$ is better than *Best*(t) then A is rewarded by decreasing its PSL value according to the learning rule (10).
- If $S(A,t)$ is worse than *Best*(t) then agent A is 'punished' by increasing its PSL value according to the learning rule (9).

It should be emphasized that the proposed learning mechanism is efficient for scenarios such as the following one:

1. Consider an exploiter agent obtaining low-quality solutions for several tours – the agent is 'punished' by decreasing its PSL value repeatedly;
2. if the PSL value decreases below the threshold value γ and the agent starts producing better solutions, the learning rule of decreasing the PSL value is applied as a rewarding mechanism in this case;
3. the performance of the agent will enhance with the agent's exploiting capabilities in this scenario.

LSAS agents learn the characteristics of the search space via a dynamic change in the PSL values. Good explorers of the solution space will be encouraged to further explore the environment more aggressively. Promising solutions already identified will be further exploited by rewarding the corresponding agent.

5.3 The LSAS Computational Model

The LSAS model is initialized with a population of agents that have no knowledge of the environment characteristics. Each agent deposits pheromone on the followed path and is able to communicate to the other agents in the system the knowledge it has about the environment after a full path is created or an intermediary solution

is built. The infrastructure evolves as the current agent that has to determine the shortest path is able to make decisions about which route to take at each point in a sensitive stigmergic manner.

Agents with small PSL values normally choose only paths with very high pheromone intensity or alternatively use the knowledge base of the system to make a decision. These agents can easily take into account ACL messages received from other agents. The information contained in the ACL message refers to environment characteristics and is specific to the problem that is being solved. On the other hand, agents with high PSL values are more sensitive to pheromone trails and easily influenced by stronger pheromone trails. However, this does not exclude the possibility of additionally using the information about the environment received from other agents.

These considerations can be summarized by the following algorithm:

Learning Sensitive Agent System (LSAS)
Begin
Set parameters pheromone trails
Initialize pheromone trails
Initialize knowledge base
While stop_condition is false
Activate a set of agents with various PSL
Place each agent in search space
Do for each agent
Apply a state transition rule to incrementally build a solution.
Determine next move (stigmergic strategy / direct communication).
Apply a local pheromone update rule.
Propagate learned knowledge.
Until all agents have built a complete solution
Update PSL value for each agent using proposed learning mechanism.
Apply a global pheromone update rule.
Update knowledge base (using learned knowledge).
End While
End

---

6 LSAS FOR SOLVING NP-HARD PROBLEMS

The Traveling Salesman Problem (TSP) is a well-known NP-hard problem in which a Hamiltonian path must be identified. Let $G = (V, E)$ be an n-node undirected graph whose edges are associated with non-negative costs. Let us assume that G is a complete graph (if there is no edge between two nodes, it can be added to it with an infinite cost). The aim of the problem is therefore to find a minimum-cost tour passing through all nodes of the graph exactly once.

The LSAS model is engaged in solving one version of TSP and a generalization of TSP. The first problem addressed is the *Asymmetric Traveling Salesman Problem (ATSP)* characterized by the fact that the cost of edge (i, j) is not the same as the cost for the inverse edge (j, i).

The other problem considered is the *Generalized Traveling Salesman Problem (GTSP)*. Let $V_1, \ldots, V_p$ be a partition of V from graph G into p subsets called *clusters*. The cost of an edge $(i, j) \in E$ is $c(i, j)$. GTSP refers to finding a minimum-cost tour H spanning a subset of nodes such that H contains exactly one node from each cluster V_i, $i \in \{1, \ldots, p\}$.

The implemented LSAS model works similarly for both TSP considered problems. Agents deposit pheromone on the followed path. Unit evaporation takes place each cycle. This prevents unbounded intensity trail increasing. In order to stop agents visiting the same node in the same tour a *tabu list* is maintained.

LSAS is implemented using sensitive stigmergic agents with initial randomly generated PSL values. Sensitive-explorer agents autonomously discover new promising regions of the solution space to sustain search diversification. Each generation the PSL values are updated according to the reinforcing learning mechanism described in Section 5. The learning rule used in LSAS ensures a meaningful balance between search exploration and exploitation in the problem solving process.

The LSAS model for solving TSP works as follows:

LSAS algorithm for solving TSP

Step 1. Initially the agents are placed randomly in the nodes of the graph. The PSL value of each agent is randomly generated.

Step 2. Each LSAS agent moves to a new node based on its current PSL value:

(i) PSL $\leq a$ – direct communication: random decision making influenced by information received from other agents or found in the knowledge base,
(ii) PSL $> a$ – stigmergic strategy: next node is selected with a probability based on the distance to that node and the amount of trail intensity on the connecting edge.

If the virtual state is selected, the agent randomly chooses on of the strategies:

(A) An accessible state is chosen randomly with uniform probability; and
(B) The probability of selection is based on the state rank generated using the information received from other agents and the knowledge base.

Step 4. The agent sends an ACL message to the other agents containing the latter edge formed and its cost.

Step 5. The trail intensity is updated.

Step 6. The PSL value for each agent is recalculated using the LSAS learning rule.

Step 7. Only agents that generate the best tour are allowed to globally update the virtual pheromone and the knowledge base. The global update rule is applied to the edges belonging to the best tour.

Remark. A run of the algorithm returns the shortest tour detected.

Sections 7 and 8 present the numerical results of this algorithm for a set of ATSP and GTSP instances.

7 LSAS IN SOLVING ATSP. NUMERICAL EXPERIMENTS

The LSAS model for solving ATSP is tested for several instances of the problem considered. The performance of the proposed LSAS model in solving ATSP is compared to the results of standard Ant Colony System (ACS) technique [9] and the Min-Max Ant System (MMAS) [17].

7.1 Numerical Results

Several problem instances from TSP library [12] are considered for numerical experiments. TSPLIB provides the optimal objective values (representing the length of the tour) for each problem.

In each algorithm, ten ants are used and the average of the best solutions is calculated for ten successive runs. The termination criteria are given by limiting the length of running time for each algorithm (i.e. maximum ten minutes of running time). The thresholds a and γ are both set to 0.5 in all experiments.

The proposed LSAS model detects a near-optimal or optimal solution for all problems engaged in the numerical experiments. Figure 1 presents the average deviation of the mean best values obtained by the compared models for the ATSP instances considered. It can be observed that the LSAS model performs better compared to the other models considered for all problems.

These test results indicate that the proposed LSAS model is able to obtain better results compared to standard ant-based models emphasizing the potential of learning stigmergic agent models.

7.2 Statistical Analysis

The Expected Utility Approach [10] technique is employed for statistical analysis purposes. The results of the test are presented in Table 1.

Let x be the percentage deviation of the heuristic solution and the best known solution of a particular heuristic on a given problem:

$$x = \frac{\text{heuristic solution-best known solution}}{\text{best known solution}} \times 100.$$

The expected utility function euf can be expressed as:

$$euf = \gamma - \beta(1 - \overline{b}t)^{-\overline{c}}, \tag{11}$$

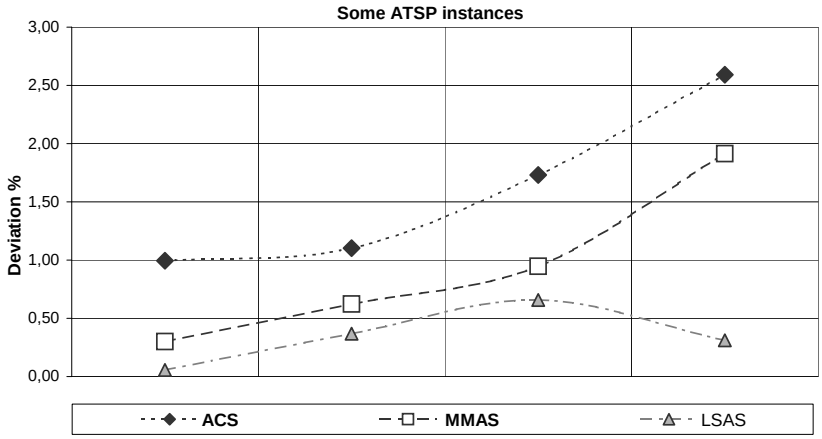


Fig. 1. Average deviation of mean best values obtained by ACS, MMAS and LSAS in solving ATSP instances Ry48p, Ft70, Kro124 and Ftv170

where $\gamma = 500$, $\beta = 100$ and $t = 0.025$. $\overline{b}$ and $\overline{c}$ are the estimated parameters of the Gamma function.

Because four problems have been used for testing, the following notations are used in Table 1:

$$\overline{x} = \frac{1}{4}\sum_{j=1}^{4} x_j, s^2 = \frac{1}{4}\sum_{j=1}^{4}(x_j - \overline{x})^2, \overline{b} = \frac{s^2}{\overline{x}}, \overline{c} = (\frac{\overline{x}}{s})^2. \tag{12}$$

As indicated in Table 1, the proposed LSAS model obtains Rank 1 (the last column in Table 1). This result emphasizes that LSAS is more accurate compared to ACS and MMAS techniques in detecting ATSP solutions for the considered problem instances (the mean values of each algorithm have been engaged for the statistical analysis).

Heuristic	$\overline{x}$	s^2	$\overline{b}$	$\overline{c}$	*euf*	Rank
ACS	1.6047	0.4038	0.2516	6.3771	391.58	3
MMAS	0.9458	0.3651	0.3860	2.4501	395.11	2
LSAS	0.3484	0.0457	0.1312	2.6561	398.23	1

Table 1. Statistical analysis results for compared models in solving ATSP

8 LSAS IN SOLVING GTSP. NUMERICAL EXPERIMENTS

The LSAS computational model is used for solving GTSP. The performance of the proposed model in solving GTSP is compared to the results of the ACS technique [6,

17], the *Nearest Neighbor (NN)* algorithm, the GI^3 composite heuristic [18] and *Random Key Genetic Algorithm (rkGA)* [20].

8.1 Compared Models

The algorithm *Ant Colony System* for GTSP [17] is based on the ACS [6, 7] idea of simulating the behavior of a set of agents that cooperate to solve a problem by means of simple communications.

In the *Nearest Neighbor* algorithm the rule is always to go next to the nearest as-yet-unvisited location. The corresponding tour traverses the nodes in the constructed order.

The composite heuristic GI^3 is composed of three phases: the construction of an initial partial solution, the insertion of a node from each non-visited node-subset, and a solution improvement phase [18].

The *Random Key Genetic Algorithm* combines a genetic algorithm with a local tour improvement heuristic. Solutions are encoded using random keys, which circumvent the feasibility problems encountered when using traditional GA encodings [20].

8.2 Numerical Results

The data set of Padberg-Rinaldi city problems (TSP library [12]) is considered for numerical experiments.

The parameters of the LSAS algorithm are similar to those of ACS: ten ants are used and the average of the best solutions is calculated for ten successive runs. The termination criteria are given by the maximum of 300 trials and 100 tours (in this case, the execution time required by each algorithm can also be analysed). The thresholds a and γ are both set to 0.5.

Figures 2 and 3 present the average deviation of the best mean values obtained by the compared models for the GTSP instances considered (i.e. 16PR76, 22PR107, 22PR124, 28PR136, 29PR144, 31PR152, 46PR226, 53PR264, 60PR299, 88PR439). The LSAS results are depicted in Figure 3.

The proposed LSAS model obtains the optimal solutions for 7 out of the 10 problem instances engaged in the numerical experiments. For two other problem instances, the solutions reported by the proposed model are very close to the optimal values and they are better than those detected by the methods considered.

8.3 Statistical Analysis

A statistical analysis test is performed using the Expected Utility Approach [10] technique to determine the most accurate heuristic. The results of the test are presented in Table 2.

The expected utility function euf is the same with the one used for the statistical analysis on ATSP results (see Equation (11)). The parameters for this function are

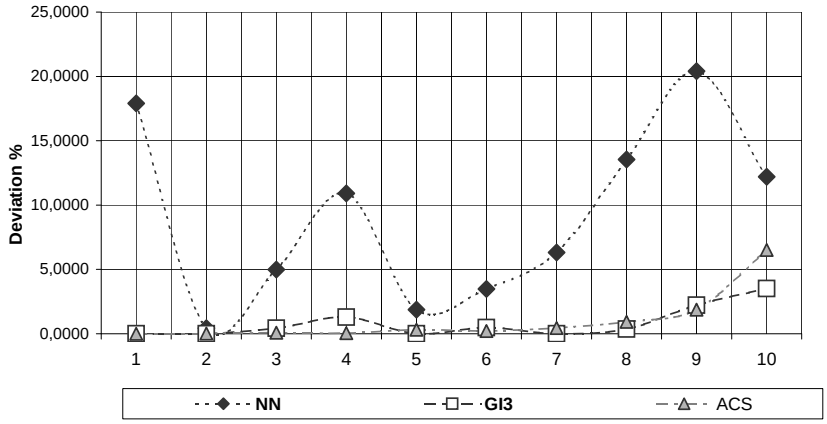


Fig. 2. Average deviation of mean best values obtained by NN, GI^3 and ACS in solving GTSP

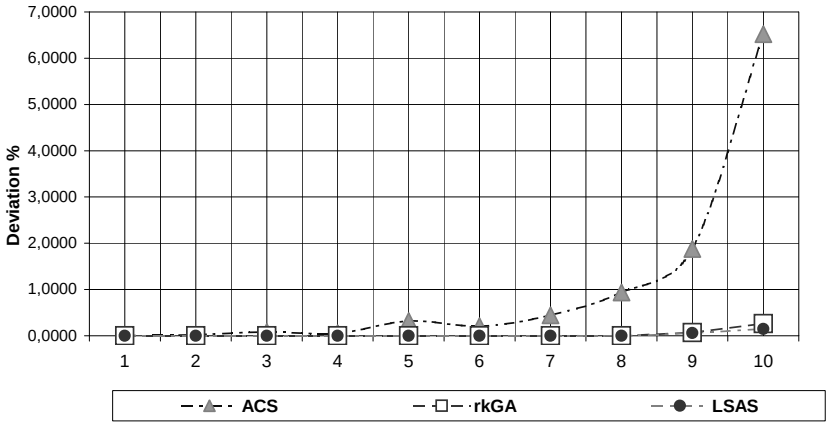


Fig. 3. Average deviation of mean best values obtained by ACS, $rkGA$ and $LSAS$ in solving GTSP

the following: $\gamma = 500$, $\beta = 100$ and $t = 0.025$. The values of $\overline{x}$, s^2, $\overline{b}$ and $\overline{c}$ required for computing the expected utility function are those given by Equations (12) but calculated against the ten GTSP problem instances considered.

The last column in Table 2 provides the rank 1 to 5 of the entries. It can be observed that LSAS has Rank 1 and represents the most accurate algorithm from the set of compared models.

Heuristic	$\overline{x}$	s^2	$\overline{b}$	$\overline{c}$	*euf*	Rank
NN	0.0920	0.0042	0.0458	2.0093	399.7695	5
GI^3	0.0083	0.0001	0.0153	0.5431	399.9793	3
ACS	0.0105	0.0060	0.5766	0.0181	399.9737	4
rkGA	0.0003	0.0059	17.6890	0.000019	399.9989	2
LSAS	0.00021	0.0059	28.4208	0.000007	399.9991	1

Table 2. Statistical analysis results for compared models in solving GTSP

8.4 Analysis of Algorithm Execution Times

In terms of running time of the algorithm, LSAS reports competitive performance compared to the other models considered. Figures 4 and 5 depict the running times for the ten GTSP instances considered (first seven in Figure 4 and last three in Figure 5). The LSAS numerical results have been obtained using a Java implementation of the algorithm running on a AMD Athlon 2600+, 333 Mhz with 2 GB memory.

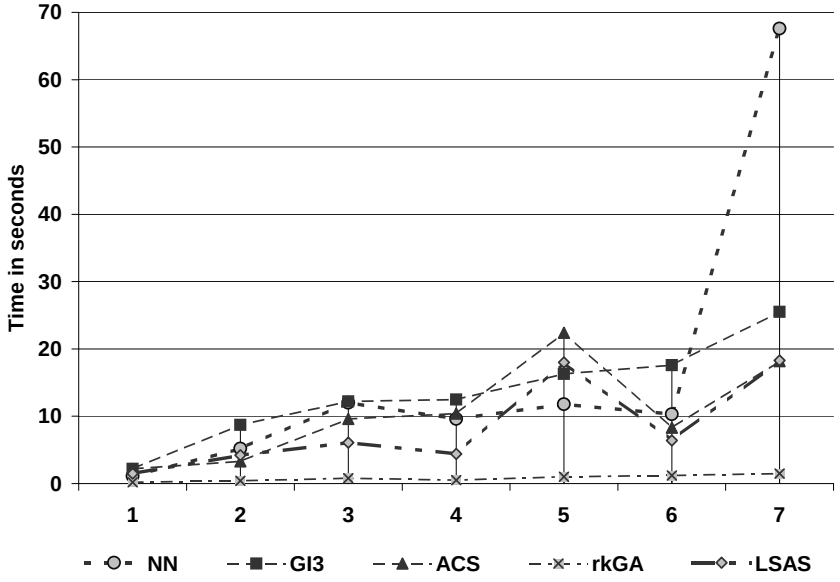


Fig. 4. Running times for the 16PR76 (1), 22PR107 (2), 22PR124 (3), 28PR136 (4), 29PR144 (5), 31PR152 (6) and 46PR226 (7) GTSP instances obtained by NN, GI^3, ACS, rkGA and LSAS

The LSAS model can be improved in terms of execution time (particularly compared to the rkGA model). Potential improvements regard the parameter values, an efficient combination with other algorithms or enhancing the agents with the capability of fully working in parallel on the inner loop of the algorithm. It should

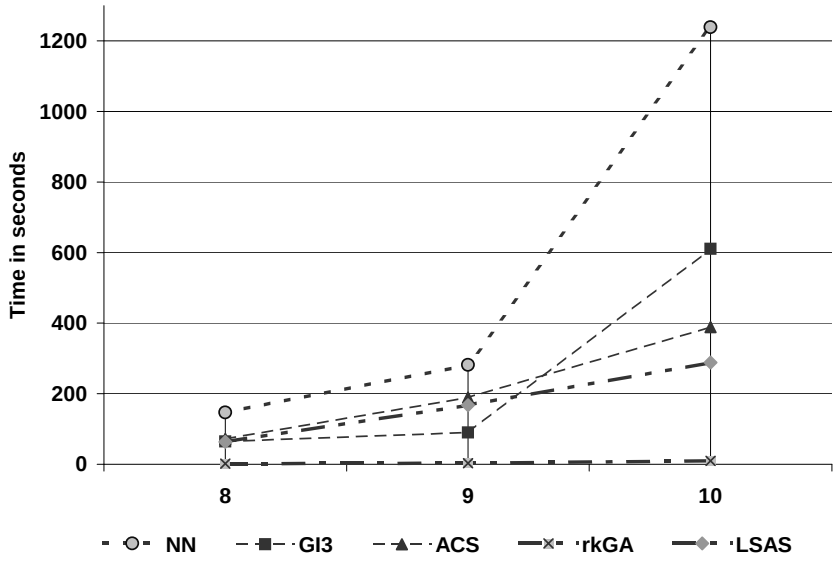


Fig. 5. Running times for 53PR264 (8), 60PR299 (9) and 88PR439 (10) GTSP instances obtained by NN, GI^3, ACS, rkGA and LSAS

be noted that LSAS reports better running times compared to the ACS model suggesting the benefits of system heterogeneity in the search process. Diversification of behavior modulated by various PSL values combined with the learning mechanism in LSAS results in higher-quality solutions obtained faster compared to ACS.

The numerical experiments and comparisons emphasize the potential of the proposed hybrid approach to address complex problems and facilitate further connections between multi-agent systems and nature inspired computing.

9 CONCLUSIONS AND FUTURE WORK

Solving large complex problems represents a challenging task. The idea explored in the current paper refers to combining two different complementary techniques in order to address different facets of complexity.

In the proposed approach to address complex problems, agents adopt a stigmergic behaviour (being able to produce pheromone trails) as a local search mechanism. Agents use local search to identify problem solutions. Direct communication enables agents to share knowledge about the environment. Each stigmergic agent is characterized by a certain level of sensitivity to the pheromone trails. The non-uniform pheromone sensitivity allows various types of reactions to a dynamic environment. During their lifetime, agents are able to learn by modifying their sensitivity level in order to maintain a good balance between search diversification and intensification.

This approach results in a new metaheuristic called LSAS (Learning Sensitive Agent System) able to address problems that involve very complex search spaces. Within LSAS, solutions are incrementally built by agents. Numerical experiments indicate the effectiveness and the potential of the proposed LSAS technique.

Future research directions focus on the use of agents with sensitive stigmergy for solving real-world problems in non-stationary environments. Stigmergic agents can share information concerning dynamic changes in the environment (e.g. node or edge removing in a dynamic graph, cost modification of an edge, introduction of new nodes or new edges) improving the quality of the search process. The LSAS approach is potentially useful for addressing large problems concerning routing, job assignement and schedulling. Other problems of interest refer to communication in mobile systems and dynamic location problems.

Acknowledgments

This work was supported by CNCSIS UEFISCSU, project number PNII – IDEI 508/2007 *New Computational Paradigmes for Dynamic Complex Problems.*

REFERENCES

[1] Camazine, S.—Deneubourg, J. L.—Franks, N. R.—Sneyd, J.—Theraulaz, G.—Bonabeau, E.: Self Organization in Biological Systems. Princeton Univ. Press 2001.

[2] Bradshow, J. M.: An Introduction to Software Agents. In J. M. Brandshow: Software Agents, MIT Press 1997.

[3] Chira, O.—Chira, C.—Tormey, D.—Brennan, A.—Roche, T.: An Agent-Based Approach to Knowledge Management in Distributed Design. Special issue on E-Manufacturing and web-based technology for intelligent manufacturing and networked enterprise interoperability. Journal of Intelligent Manufacturing, Vol. 17, 2006, No. 6.

[4] Chira, C.—Pintea, C. M.—Dumitrescu, D.: Stigmergic Agents for Solving NPdifficult Problems. Proceedings of Bio-Inspired Computing: Theory and Applications Conference, Evolutionary Computing Volume, Wuhan, China 2006, pp. 63–69.

[5] Chira, C.—Pintea, C. M.—Dumitrescu, D.: Sensitive Stigmergic Agent Systems. Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS), Maastricht, The Netherlands, MICC Technical Report Series, No. 07-04, Karl Tuyls, Steven de Jong, Marc Ponsen, Katja Verbeeck (Eds.), ISSN 0922-8721, pp. 51–57, 2007.

[6] Dorigo, M.—di Caro, G.—Gambardella, L. M.: Ant Algorithms for Discrete Optimization. Artificial Life, 5, pp. 137–172, 1999.

[7] Dorigo, M.—Blum, C.: Ant Colony Optimization Theory: A Survey. Theoretical Computer Science, Vol. 344, No. 2-3, pp. 243–278, 2005.

[8] Finin, T.—Labrou, Y.—Mayfield, J.: Kqml as an Agent Communication Language. In B. M. Jeffrey (Ed.): Software Agents, MIT Press 1997.

[9] Gambardella, L. M.—Dorigo, M.: Solving Symmetric and Asymmetric TSPs by Ant Colonies. International Conference on Evolutionary Computation, 1966, pp. 622–627.

[10] Golden, B. L.—Assad, A. A.: A Decision-Theoretic Framework for Comparing Heuristics. European J. of Oper. Res., Vol. 18, pp. 167–171, 1984.

[11] Grasse, P. P.: La Reconstruction du Nid et Les Coordinations Interindividuelles Chez Bellicositermes Natalensis et Cubitermes sp. La Théorie de la Stigmergie: Essai d'Interprtation du Comportement des Termites Constructeurs. Insect Soc., Vol. 6, pp. 41–80, 1959.

[12] `http://www.fipa.org`, Foundation for Intelligent Physical Agents.

[13] `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`.

[14] Jennings, N. R.: An Agent-Based Approach for Building Complex Software Systems. Comms. of the ACM, Vol. 44, No. 4, pp. 35–41, 2001.

[15] Nwana, H. S.: Software Agents: An Overview. Knowledge Engineering Review, Vol. 11, pp. 1–40, 1996.

[16] Nwana, H. S.—Lee, L.—Jennings, N.: Coordination in Software Agent Systems. BT Technology Journal, Vol. 14, 1996, No. 4, pp. 79–88.

[17] Pintea, C. M.—Pop, C. P.—Chira, C.: Reinforcing Ant Colony System for the Generalized Traveling Salesman Problem. Proc. BIC-TA, Vol. Evolutionary Computing, pp. 245–252, 2006.

[18] Renaud, J.—Boctor, E. F.: An Efficient Composite Heuristic for the Symmetric Generalized Traveling Salesman Problem. European Journal of Operational Research, Vol. 108, pp. 571–584, 1998.

[19] Russell, S.—Norvig, P.: Artificial Intelligence: A Modern Approach. 2/E, ed., 2003.

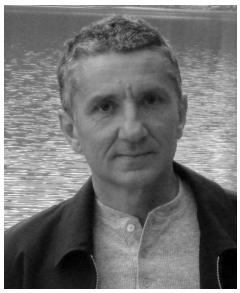
[20] Snyder, L. V.—Daskin, M. S.: A Random-Key Genetic Algorithm for the Generalized Traveling Salesman Problem. European Journal of Operational Research, pp. 38–53, 2006.

[21] Stützle, T.—Hoos, H. H.: The Max-Min Ant System and Local Search for the Travelling Salesman Problem. IEEE International Conference on Evolutionary Computation, Piscataway, T. Bck, Z. Michalewicz and X. Yao (Eds.), IEEE Press, pp. 309–314, 1997.

[22] Wooldridge, M.: Intelligent Agents. An Introduction to Multiagent Systems. G. Weiss Ed.; 1999.

[23] Wooldridge, M.—Dunne, P. E.: The Complexity of Agent Design Problems: Determinism and History Dependence. Annals of Mathematics and Artificial Intelligence, Vol. 45, No. 3-4, pp. 343–371, 2005.

Camelia Chira received the B. Sc. degree in computer science from Babes-Bolyai University, Cluj-Napoca, Romania in 1998. From 2000 to 2005 she was engaged in full-time research at Galway-Mayo Institute of Technology, Ireland focusing in the area of agent-based systems and ontologies for distributed collaborative design environments. She received the M. Sc. and Ph. D. degrees from the same institution in 2002 and 2005, respectively. Since 2006, she is a researcher at the Centre for the Study of Complexity, Babes-Bolyai University, Romania. Her main research interests include evolutionary computing, swarm intelligence, complex systems and networks, multi-agent systems and bioinformatics.



Dumitru Dumitrescu received the B. Sc., M. Sc. and Ph. D. degrees from Babes-Bolyai University, Cluj-Napoca, Romania (B. Sc. in physics – 1971, B. Sc. in mathematics – 1979, M. Sc. in theoretical physics – 1972, M. Sc. in mathematics – 1980 and Ph. D. in mathematics, 1990). He has been a university academic within the Babes-Bolyai University from 1972 and became a full Professor in 1994. During 1998–2000 he also was Visiting Professor at Pisa University, Italy. He is currently a Professor in the Department of Computer Science, Babes-Bolyai University. His current research interests include natural computing, complexity, and computational game theory.

Camelia-M. Pintea has been teaching informatics since 1992. She received a B. Sc. degree in mathematics, a M. Sc. in computer science and completed a Ph. D. in computer science at Babes-Bolyai University, Cluj-Napoca, Romania. Her research interests include applied mathematics, combinatorial optimization, meta-heuristics, natural computing, artificial intelligence, bioinformatics and computer graphics. She has been involved in several scientific projects and served on various scientific committees of journals and conferences.