Context Discovery in Mobile Environments: A Particle Swarm Optimization Approach

Christos Anagnostopoulos and Stathes Hadjiefthymiades

Department of Informatics and Telecommunications, University of Athens, Greece
{bleu,shadj}@di.uoa.gr

Abstract. We introduce a novel application of Particle Swarm Optimization in the mobile computing domain. We focus on context aware applications and investigate the context discovery problem in dynamic environments. Specifically, we investigate those scenarios where nodes with context aware applications are trying to (physically) locate up-to-date context, captured by other nodes. We establish the concept of context quality (an ageing framework deprecates contextual information thus leading to low quality). Nodes with low quality context cannot capture such information by themselves but are in need for "fresh" context in order to feed their application. We assess the performance of the proposed algorithm through simulations. Our findings are quite promising for the mobile computing domain and context awareness in specific. We assess two different strategies for the PSO-based context discovery framework.

Keywords: Context-awareness, context-discovery, distributed systems, swarm intelligence, particle swarm optimization.

1 Introduction

Mobile and distributed computing has become increasingly popular during the last years. Many mobile applications exhibit self-organization in dynamic environments adopted from multi-agent, or *swarm*, research. The basic paradigm behind swarm systems is that tasks can be more efficiently dispatched through the use of multiple, simple autonomous agents instead of a single, sophisticated one. Such systems are much more adaptive, scalable and robust than those based on a single, highly capable, agent.

A swarm system can generally be defined as a decentralized group (swarm) of autonomous agents (particles) that are simple with limited processing capabilities. Particles must cooperate intelligently to achieve common tasks. We investigate a mechanism that exploits the collaborative behavior of the agents in order to deal with the Context Discovery Problem (CDP). Specifically, in CDP an agent (e.g., mobile node) needs to discover, locate and track the source that generates the required contextual information – *context* (e.g., environmental parameters like temperature, humidity, situations like fire outbreak) for the executing context-aware, mobile application (e.g., the control of a group of robots).

Swarm Intelligence (SI) introduces a powerful new paradigm for building fully distributed systems in which overall system functionality is attained by the interaction of

A.V. Vasilakos et al. (Eds.): AUTONOMICS 2009, LNICST 23, pp. 160–175, 2010.

individual agents with each other and with their environment. Such agents coordinate using decentralized control and self-organization. Swarm systems are intrinsically highly parallel and exhibit high levels of robustness and reliability:

1. A SI-driven distributed system does not have hierarchical command and control structure and thus no single failure point or vulnerability. Agents are often very simple and the overall swarm is intrinsically fault-tolerant since it consists of a number of identical units operating (sensing context) and cooperating (sharing context) in parallel. In contrast, a conventional complex distributed system requires considerable design effort to achieve fault tolerance.
2. The key central concept in a swarm system is the simplicity of the agents -an agent can be a mobile phone carrying sensors. Simply increasing the number of agents assigned to a task (e.g., sensing context) does not necessarily improve the system's performance (i.e., efficiency and reliability). Agents collaborate by exchanging useful information in order to obtain the required context.
3. In a totally distributed environment agents collaborate for discovering context with certain validity (e.g., related to time and/or space constraints). Context periodically turns obsolete and has to be regularly determined and discovered. Moreover, the resources of simple agents are limited in terms of (1) memory; agents remember the history of their operation up to a certain extent, (2) sensing capabilities; for agents moving around, the sensing radius can be small enough relatively to the coverage area once possible neighboring agents can provide analogous local information, and (3) communication resources; communication among agents is intended solely to convey information on the swarm.

The above-mentioned points lead to the question: "Is the SI paradigm suitable for application in the CDP?" The aim of this paper is to address that question.

Many research efforts have examined multi-agent systems inspired by biology, e.g., flocking models [1, 2], emphasizing in fault tolerance [3], cooperative hunting [4] and ant colony optimization [5] for solving problems in distributed environments. Below we report some typical applications: 'covering' (explore enemy terrain), 'patrolling' (guarding a museum against theft), 'self-assembling' (reconfigurable robots), 'localization' (improvement of positioning accuracy) and 'environment manipulation' (transportation control). In addition, significant research effort has been invested in the design of swarm system for searching areas, either known or unknown, which is most relevant to our work. Specifically, in most previous works the targets, i.e., nodes with valuable information (e.g., sensor nodes) are assumed to be static. However, only a few works examine a swarm system in dynamic environments dealing with the mobility of agents [6, 7, 8] and with information validity constraints. One of the first studies in the application of PSO to dynamic environments came from [21]. The work in [9] considers dynamic targets but does not deal with certain validity issues as required in the CDP. A significant SI adaptive mechanism to detect and respond to dynamic systems is reported in [23]. The involved agents in such mechanism cannot be fully applied to mobile nodes as long as the inherent communication load and efficiency are not taken into consideration, especially when dealing with real context-aware applications. Therefore, we adopt same ideas from [23] regarding the response strategies to various changes. To the best of our knowledge, there is no prior work based on SI in order to deal with the CDP. This motivated us to define, model and propose a solution (algorithm and strategies) for the CDP.

The structure of the paper is as follows: Section 2 presents the basic idea of SI, while in Section 3 we introduce certain issues for the CDP. In Section 4 we propose an algorithm for the CDP adopting concepts from SI. We assess our algorithm in Section 5 and Section 6 concludes the article.

2 Swarm Intelligence

The *Particle Swarm Optimization* (PSO) incorporates swarm behaviors observed in flocks of birds, swarms of bees, or human social behavior, from which the idea is taken [10]. The main strength of PSO is its fast convergence, which compares favorably with many global optimization algorithms (e.g., Genetic Algorithms and Simulated Annealing). The PSO model consists of a swarm of N particles, which are initialized with a population of random candidate solutions (particles). They move iteratively through a d-dimension problem space $\Re^d$ to search new optima. f: $\Re^d \rightarrow \Re$ is a *fitness* function that takes a particle's solution in $\Re^d$ and maps it to a single decision metric; the CDP deals with the geometrical space of two dimensions, i.e., $d = 2$, as will be discussed bellow. Each particle indexed by i has a position represented by a vector x$i \in \Re^d$ and a velocity represented by a vector v$i \in \Re^d$, i=1, …, N. Each particle "remembers" its own *best* position so far in a vector xi# = [xij#]. The best position vector among the swarm so far is then stored in a vector x* = [xj*]. During the iteration (time) t, the velocity update is performed as in Eq(1). The new position is then determined by the sum of the previous position and the new velocity in Eq(2).

$$u_{ij}(t + 1) = wu_{ij}(t) + c_1r_1(x_{ij}^{\#}(t) - x_{ij}(t)) + c_2r_2(x_j^{*}(t) - x_{ij}(t)) \quad (1)$$

$$x_{ij}(t + 1) = x_{ij}(t) + u_{ij}(t + 1) \quad (2)$$

w is an *inertia* factor. The r_1, r_2 random numbers are used to maintain the diversity of the population and are uniformly distributed in the interval [0, 1] for the jth dimension of the ith particle. c_1 and c_2 are positive constants called *self-recognition* and *social* component, respectively. They interpret how much the particle is directed towards good positions. That is, c_1 and c_2 indicate how much the particle's private knowledge and swarm's knowledge on the best solution is affected, respectively. The time interval between velocity updates is often taken to be unit, thus, omitted (the Equation (2) is dimensionality inconsistent). From Equation (1), a particle decides where to move at the next time considering its own experience, which is the memory of its best past position and the experience of the most successful particle in the swarm (or in a neighboring part of swarm). The inertia w regulates the trade-off between the *global* (wide-ranging) and *local* (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration, i.e., searching new areas, while a small value facilitates local exploration, i.e., fine-tuning the current search area –exploitation [18].

The PSO algorithm is presented in Algorithm 1. The *end criterion* (line 2) may be the maximum number of iterations, the number of iterations without improvement, or the minimum objective function error between the obtained objective function and the best fitness value w.r.t. a pre-fixed anticipated threshold. Particles are started at random positions with zero initial velocities and search in parallel. What is needed is some *attraction*, if not to the absolutely best position known, at least towards a position close to the particle where the fitness is better than the fitness a particle has currently determined. All particles exploit at least one *good* position already found by some particle(s) in the swarm (line 7). Hence, particles adjust their own position and velocity based on this good position (line 9). Often, the position that is exploited is the *best* position yet found by any particle (line 5). In this case, all particles know the currently best position found and are attracted to this position. This, obviously, requires communication between particles and some sort of collective memory to the current *global best* (*gbest*). The $\mathbf{x}^*$ vector in Equation (1) represents the *gbest* position of the swarm (line 5).

Algorithm 1. Particle Swarm Optimization Algorithm

1 **Initialize** randomly the positions and zero velocities.
2 **While** (the *end criterion* is not met) **Do**
3 $t \leftarrow t + 1$;
4 **Calculate** the fitness value f of each particle;
5 $\mathbf{x}^* = \arg min^{N}_{i=1}\{f(\mathbf{x}^*(t-1)), f(\mathbf{x}_1(t)), \ldots, f(\mathbf{x}_i(t)), \ldots, f(\mathbf{x}_N(t))\}$;
6 **For** $i = 1{:}N$
7 $\mathbf{x}_i^{\#}() = \arg min^{N}_{i=1}\{f(\mathbf{x}_i^{\#}(t-1)), f(\mathbf{x}_i(t))\}$;
8 **For** $j = 1{:}d$
9 **Update** the jth dimension of $\mathbf{v}_i$ and $\mathbf{x}_i$ w.r.t. (1), (2);
Next j
10 **Next** i
11 **End While**

Alternatively, a particle i can experience an attraction back to the best place yet found by it. The *personal best* (*pbest*) position for particle i results in its independent exploration without any input of the other particles. The *pbest* position for the ith particle is $\mathbf{x}_i^{\#}$ (line 7).

An idea for triggering a particle to direct to an attracted area is to balance the movement between the *gbest* and *pbest* positions by defining a *local* neighborhood around it. All N_i particles within an actual physical distance form the neighborhood of the ith particle. Each particle in N_i shares its fitness value with all other particles in that neighborhood. Hence, neighboring particles experience an attraction to the *local best* (*lbest*). The problem with *lbest* (not so critical as in *gbest*) is that, neighborhoods need to be calculated frequently and, thus, the computational cost for this operation has to be considered. The particles adjust their current velocity based on current *pbest* and prior knowledge derived from *gbest* and *lbest*. Based on *gbest*, particles have to communicate with the whole swarm for locating and maintaining information on the global best solution. In this case the best particle acts as an attractor pulling all the particles towards it. Eventually, all particles will converge to this position. Based on *lbest*, particles are required to check for any better solution appeared in adjacent particles.

In order to avoid the inherent communication cost in CDP due to the information exchange among particles for estimating *gbest* and the premature convergence obtained from *gbest*, we relate the social component c_2 in (1) to the *lbest* approach, i.e., the $\mathbf{x}^*$ vector in (1) represents the *lbest* position of a given particle. c_2 indicates the willingness of a particle to be attracted by any probable neighbor. We also adopt random relative weights for combining *lbest* and *pbest*. The continuous movement toward a position of better fitness (w.r.t. *pbest*) biases the selection of particles with even better fitness than the existing one. The discovery process, which is based on *pbest*, dramatically improves the average fitness of the positions explored. Evidently, this may result in exploration stopping at a local optimum. But, with a number of different local neighborhoods in use, there is a very good probability that the whole swarm will not get so trapped, and that any trapped particle will escape, especially if the *lbest* approach is also simultaneously in use. We adopt both approaches together with r_1, r_2 factors to set the relative influences of each.

3 The Context Discovery Problem

We firstly define the notions of *context* and *quality of context* and then map the parameters of the CDP into PSO.

3.1 Context Representation and Quality of Context

Context refers to the current values of specific parameters that represent the activity / situation of an entity and environmental state [11]. Let $\mathbf{Y} = [\mathrm{Y}_1, \ldots, \mathrm{Y}_m]$ be a m-dimensional vector of parameters, which assumes values y_l in the domain $\mathrm{Dom}(\mathrm{Y}_l)$, $l = 1, \ldots, m$. A parameter Y_l is considered instantiated if at time t some y_l value is assigned to Y_l. *Context* $\mathbf{y}$ is the instantiated $\mathbf{Y}$, i.e., $\mathbf{y} = [\mathrm{y}_1, \ldots, \mathrm{y}_m]$. For each instantiated Y_l, a function v: $\mathrm{Y}_l \times \mathrm{T} \rightarrow [0, a)$, $a > 0$, is defined denoting whether the value y_l is valid at time t after the Y_l instantiation; T is the time index and a is a real positive number. The value y_l is valid at time t for a context-aware application that is executed on node i if $v(\mathrm{y}_l, t) < \theta_{il}$ for a given threshold $\theta_{il} \in (0, a)$, which is application specific. A value of θ_{il} close to a means that y_l is not valid for the ith node. v can be any increasing function F with time t, i.e., $v = F(t)$. For simplicity reasons we can assume that F is the identity function (i.e., $v = t$). The value a is set w.r.t. application specification. For instance, in our case a is the maximum time from the sensing time of Y_l in which its value is not deprecated. A value of θ_{il} close to 0 means that y_l is of high importance. The indicator v increases over time from the sensing time of y_l. Hence, a value of v denotes the freshness of y_l, i.e., y_l refers to either an up-to-date (fresh) or obsolete measure. It should be noted that, v refers only to the temporal validity of a value. Evidently, other validity functions can be defined referring to quality indicators like spatial scope (value is usable within certain geographical boundaries), the source credibility, the reliability of the measurement, and other objective or subjective indicators [12].

We introduce the *quality of context* indicator g: $\mathbf{Y} \times \mathrm{T} \rightarrow [0, a)$ for context $\mathbf{y}$ at time t denoting whether the values of the parameters of $\mathbf{y}$ are valid or not with respect to a certain threshold. The value of g is the minimum indicator of the values, that is $g(\mathbf{y}, t) = \min_{l=1}{}^{m}\{v(\mathrm{y}_l, t)\}$ with threshold $\theta_{i\mathbf{y}} = \min_{l=1}{}^{m}\{\theta_{il}\}$. A value of $\theta_{i\mathbf{y}}$ close to a denotes invalid context, i.e., obsolete context, while a value of $\theta_{i\mathbf{y}}$ close to 0 denotes fresh context. Context $\mathbf{y}$ turns obsolete once some parameter turns also obsolete.

Each node i attempts to maximize the time period Δt in which $g(\mathbf{y}, t + \Delta t) < \theta_{i\mathbf{y}}$ for some t. That is, each node attempts to maintain fresh context as much time as possible. It is worth noting that a node i evaluates the quality of $\mathbf{y}$ differently from a node j, i.e., $g_i(\mathbf{y}, t) \neq g_j(\mathbf{y}, t)$. This means that, $\mathbf{y}$ may be of value for node i but not for node j at the same time. Without loss of generality we assume that all nodes evaluate the quality of context with the same $\theta_{i\mathbf{y}}$. That is, all nodes assess context with the same criteria / quality indicators. This does not imply that all nodes obtain context of the same quality. Instead, all nodes are interested in the same quality of context. This does not undermine the generality of the problem. In fact, if there are groups of nodes that assess context differently then groups of nodes will be formed and, consequently, each group will assess context with the same $\theta_{i\mathbf{y}}$.

3.2 Mapping Swarm Intelligence to Context Discovery

Let us assume discrete time and consider a square terrain of dimension L. Consider a group of N mobile nodes that maps to a swarm of particles and a set of M mobile sources (i.e., sensors that sense context) that correspond to the possible solutions in PSO. Each source regularly generates fresh context meaning that each source measures context with a given frequency-sensing rate q. Each sensed value is time stamped at the source. Every node needs to move to an area with at least a source that carries fresh context. Alternatively, a node attempts to locate areas where other nodes carry fresh context or context of better quality than the context currently available in them. In addition, a node does not know the existence of a source in a certain area and the swarm does not know the number of sources. This evidently denotes that the nodes continue searching until all sources are located or all nodes carries fresh context. However, the nodes have to adopt a mechanism in order to maintain context as fresh as possible as long as the validity fades over time.

The considered CDP is a 2-dimensional problem space in PSO ($d = 2$). It refers to the 2D location information (longitude and latitude) of the sources / nodes that carry fresh context. The exact 2D location information of a node is not known. Hence, we assume that all nodes are capable of detecting any neighboring node in a region with given transmission range equal to R. The physical presence of a node in a neighborhood can be detected thus such node is assumed to be located in the corresponding neighborhood. Moreover, a node i moves towards to a neighboring node j, which carries fresher context than node i.

The value of $g_i(\mathbf{y}, t)$ denotes the willingness of node i to seek for fresh, or at least of better quality (more up-to-date) context than the existing context. The quality of context indicator gi(y, t) resembles the fitness function f in PSO. A node i attempts to:

- minimize the value of $g_i(\mathbf{y}, t)$ at time t, and,
- maximize the duration in which it maintains fresh context, i.e., $g_i(\mathbf{y}, t) < \theta_{\mathbf{y}}$.

It should be noted that $g_i(\mathbf{y}, t)$ depends on time once the indicators for each parameter increase over time (w.r.t. sensing time). This means that a node i has to regularly update its fitness by dynamically adjusting its decision regarding the next movement w.r.t. *pbest* and *lbest*. Let us calculate the *pbest* and *lbest* so that node i decides in which direction to move. Let N_i be the indices of the neighboring nodes of node i at time t. The

$\mathbf{x}_i^{\#}$ vector at time t is the position $\mathbf{x}_j$ of the neighbor j, which carries fresher context $\mathbf{y}$ than that of node i and the freshest context among all neighbors of node i, i.e.,

$$\mathbf{x}_i^{\#} = \mathbf{x}_j: j = \arg min_{l \in \{Ni\}}\{g_l(\mathbf{y}, t) \wedge (g_i(\mathbf{y}, t) > g_l(\mathbf{y}, t))\}.$$

$\mathbf{x}_i^{\#}$ is currently the best position found at time t to which the node i adjust its next movement at time $t + 1$ assuming the *pbest* fitness value $g_i^{\#}(\mathbf{y}) = g_j(\mathbf{y}, t)$. The vector $(\mathbf{x}_i^{\#} - \mathbf{x}_i)$ refers to the self-recognition vector for node i that is attracted by the node j.

Furthermore, the node i can exploit its past knowledge. Based on *pbest* the node i locates the current best node j and moves towards it with a factor $r_1 \cdot c_1$. In addition, node i exploits the average fitness of all neighbors at time t that is

$$g_{N_i}(\mathbf{y},t) = \frac{1}{|N_i|}\sum_{i=1..k} g_i(\mathbf{y},t), k \in N_i - \{i\}$$

The proposed $g_{Ni}(\mathbf{y}, t)$ value refers to a *local fitness* of the neighborhood of node i. Node i can obtain a clear view of its neighborhood meaning that: if $g_{Ni}(\mathbf{y}, t) < \theta_{i\mathbf{y}}$ (i.e., fresh context) then the node i might not decide to move far away from this neighborhood hoping that it will probably be within an area where nodes carry fresh context. Similarly to $g_i^{\#}(\mathbf{y})$, we define the *lbest* $g_i^{*}(\mathbf{y})$ indicator that is an estimate for the freshness of $\mathbf{y}$ at time t obtained by the neighborhood of node i. If it holds true that $g_{Ni}(\mathbf{y}, t) < g_i^{*}(\mathbf{y})$ then the *lbest* $\mathbf{x}_i^{*}$ is the current $\mathbf{x}_i$ of node i at time t and the *lbest* fitness value $g_i^{*}(\mathbf{y})$ equals to $g_{Ni}(\mathbf{y}, t)$. However, it may hold true that $g_{Ni}(\mathbf{y}, t) > g_i(\mathbf{y}, t)$ but this does not imply that there might not be a neighboring node j that carries more fresh context than node i. In this case, the *lbest* position is not updated contrary to the *pbest* position. Instead, the node i adjusts its next movement by combining a movement towards the current *pbest* $\mathbf{x}_i^{\#}$ and previous *lbest* $\mathbf{x}_i^{*}$. Based on the $g_i^{*}(\mathbf{y})$ and $g_i^{\#}(\mathbf{y})$ indicators, the node i self-controls its decision on the next movement at time $t + 1$. The vector $(\mathbf{x}_i^{*} - \mathbf{x}_i)$ refers to the social vector component for node i denoting the attraction of node i to its neighborhood. Moreover, $g_i^{*}(\mathbf{y})$ increases over time thus node i has to regularly update and check *lbest*. That is because as long as a previous neighborhood has maintained fresh context as a whole, at the next time the *lbest* position may not refer to the same neighborhood even with the same value of $g_{Ni}(\mathbf{y}, t)$. The node i sim ply

Table 1. Mapping Between CDP & PSO

PSO concepts	Time variant		CDP concepts
swarm of N particles	no	no	group of N mobile nodes
Particle i	-	-	node i
problem space	no	yes	context $\mathbf{y}$
global optimum solution $\mathbf{x}_i$	no	yes	source positioned at $\mathbf{x}_i$
local optimum solution $\mathbf{x}_i$	no	yes	node with fresh context positioned at $\mathbf{x}_i$
number of optima	no	no	number of sources M
Fitness f	no	yes	quality of context $g_i(\mathbf{y}, t)$
pbest $\mathbf{x}_i^{\#}$	no	yes	position of neighboring node j that maximizes $g_j(\mathbf{y}, t)$
lbest $\mathbf{x}_i^{*}$	no	yes	position of node i whose neighborhood maximizes $g_{Ni}(\mathbf{y}, t)$

stores the previously visited *lbest* position assigned to $g_i^*(\mathbf{y})$. Hence, the node i has the option to move towards to a previous visited position as a last resort.

Table 1 depicts the mapping between CDP and PSO. It should be noted that the fitness function f in PSO depends only on the solution vector $\mathbf{x}_i$ and is not time dependent. The same holds true for the *pbest* and *lbest* positions in PSO. In CDP the corresponding fitness $g_i(\mathbf{y}, t)$ depends on time t as long as the invalidity of $\mathbf{y}$ increases over time. Furthermore, the $g_i^\#(\mathbf{y})$ and $g_i^*(\mathbf{y})$ indicators increase over time as well.

4 The Proposed Algorithm

We propose an algorithm in which nodes search for areas where better quality of context is obtained. In other words nodes attempt to find, locate and / or follow neighboring nodes (targets) that carry context of high value. The dynamic behavior of a mobile system means that the system changes state in a repeated manner. In our case the changes occur frequently, that is, both the location of a leader and the value of the optimum (context validity) vary[1] [22]. We propose several strategies for the CDP in order to (i) experiment with the required time for finding and maintaining high quality context, (ii) reduce the inherent network load that is used to automatically detecting and tracking various changes of the context validity and (iii) effectively respond to a wide variety of changes in context validity. The network load derives from the intercommunication among nodes. In addition, several constraints that refer to the temporal validity of context are taken into account. Therefore, the best solution of CDP is time dependent (context turns obsolete over time).

The proposed behaviors indicate the intention of a node in discovering and maintaining fresh context based on its mobility and other characteristics explained below. Specifically, a node transits between three states in order to discover context. In each state, the node decides on certain actions. A state k_i of node i can be *Obsolete* (O), *Partially satisfied* (P), or *Satisfied* (S) as depicted in Figure 1. In state O, a node either carries obsolete context (or is in need of) i.e., $g_i(\mathbf{y}, t) > \theta_\mathbf{y}$. In the S state, a node carries fresh context i.e., $g_i(\mathbf{y}, t) < \theta_\mathbf{y}$. If context $\mathbf{y}$ turns obsolete then node i transits into O. In the P state, a node chooses to carry less obsolete context than the existing context as long as this is the current best solution it achieves (local optimum). This means that the node i has found a neighbor j with fresher context i.e., $g_i(\mathbf{y}, t) > g_j(\mathbf{y}, t) > \theta_\mathbf{y}$. The node i escapes from the P state once another node k, which carries more fresh context, is located i.e., $g_i(\mathbf{y}, t) > \theta_\mathbf{y} > g_k(\mathbf{y}, t)$. We assume that all nodes adopt the same threshold for assessing the quality of context ($\theta_{i\mathbf{y}} = \theta_\mathbf{y} = \theta$, $i = 1, \ldots, N$).

4.1 Foraging for Context

A node i in state O initiates a foraging process for context acting as follows: The node i moves randomly ($\mathbf{v}_i \sim \mathrm{U}(\mathbf{v}_{min}, \mathbf{v}_{max})$) in the swarm and intercommunicates with neighbors till to be attracted by a neighbor j. The node j is then called *leader*. The leader j either carries objectively fresh context i.e., $g_j(\mathbf{y}, t) < \theta_\mathbf{y}$ or carries context that is more fresh than the context carried by node i i.e., $g_i(\mathbf{y}, t) > g_j(\mathbf{y}, t) > \theta_\mathbf{y}$ (see obsolete state in Figure 1). In the former case, the node i transits directly to state S. In the latter

[1] This dynamic environment refers to Type III environment ([18]).

case, the node j does not carry context of the exact quality that node i expects but such context is preferable than that of node i. Hence, node i can either follow node j hoping that it approaches areas (neighborhoods) with more fresh context -thus transiting to P state- or, alternatively, ignores such opportunity and continues moving at random -thus remaining at state O. In state P, node i settles with lower quality of context. This does not imply that node i stops communicating with other neighbors while moving. Instead, it continues exchanging information related to context quality with the purpose of locating another leader with more fresh context. The P state is an intermediary state between the O and S states (see partially satisfied state in Figure 1). The node is moving among neighborhoods carrying context of better quality and continues exploring areas. This policy reflects the idea of exploring the solution space even if a solution has already been reached (possibly a local optimum).

Node i attempts to retain fresh context for as long as possible. However, the $v(\mathrm{y}_l, t)$ indicator for a sensed parameter Y_l increases over time t until that value turns obsolete after some Δt, i.e., $v(\mathrm{y}_l, t + \Delta t) > \theta_l$. Hence, Y_l has to be regularly determined / sensed, with frequency at least $1/\Delta t$. In order for the node i to obtain up-to-date context $\mathbf{y}$, it follows leaders or sources that regularly generate objectively fresh context.

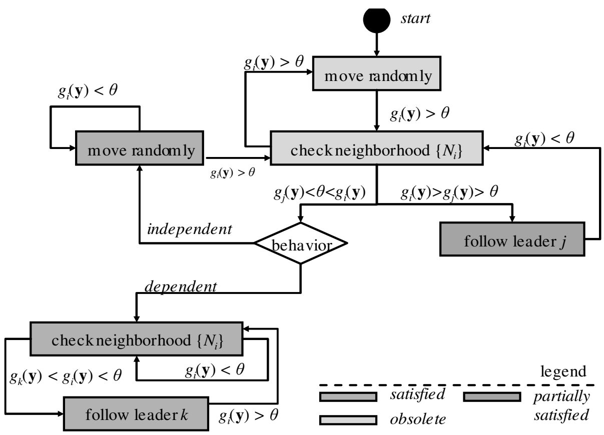

Fig. 1. A state transition in CDP

It should be noted that a localization system is needed in order to determine the solutions $\mathbf{x}_i^{\#}$ and $\mathbf{x}_i^*$, and the way node i is directed to its leader. Specifically, a node i carried by an agent (possbily a human) is directed to its leader once a GIS application displays directional information of the leader obtained, for instance, by a compass-based mechanism [16] (or other techniques, e.g., the time-of-flight technique that adopts radio frequency and ultrasound signal to determine the distance between nodes [15]). However, a non-human node i (e.g., a robot), without localization mechanisms, can "blindly" follow its leader by adjusting its direction / velocity through small improvement steps w.r.t. the signal quality [14]. Imagine for example a WLAN user trying to determine the best signal quality in a certain room by stepping around without knowing the exact location of the access point. This local-searching blind technique is not as efficient as the previously discussed method [17].

4.2 Maintaining Fresh Context

A node i, in S state, acts as follows (see the satisfied state in Figure 1): it either continues communicating with leaders (*dependent* behavior) or re-starts moving at random (*independent* behavior) with $\mathbf{v}_i \sim U(\mathbf{v}_{min}, \mathbf{v}_{max})$. In the former behavior, it is likely that node i constantly follows leaders (a.k.a. tracking optima [19]). The advantage of such behavior is that: in case node i's context turns obsolete, node i will easier find some leader provided that the latter might be yet reachable (or not far away). By adopting the independent behavior node i has no information in which direction to move towards once context turns obsolete.

Once a neighbor node k, of a node i, in S state obtains better context $\mathbf{y}$ (i.e., $\theta_{\mathbf{y}} > g_j(\mathbf{y}, t) > g_k(\mathbf{y}, t)$) then node i may choose to abandon the existing leader and follow the new leader node k. Specifically, by adopting the dependent behavior, the node i communicates with neighbors with the intention of finding a node k that carries more fresh context than the objectively fresh obtained context. Hence, the node i switches constantly between among leaders. In addition, the node i never transits to state O since its leader is a source (global optimum). However, the main objective in CDP is to enable nodes to minimize the communication load and discover as many sources and leaders as possible escaping from local optima. If all nodes adopted the dependent behavior then they would attach to sources resulting in large communication effort for the sources (sources would have to communicate with a large number of nodes) but carrying objectively fresh context. It is of high importance to take into account the inherent efficiency for both behaviors.

4.3 The CDP Algorithm

A node i in state O either transits only to state S once a leader with objectively fresh context is found or transits to the immediate state P once a leader with better context is found. As long as a leader carries objectively fresh context then node i transits from state P to S. In state S node i adopts either the independent or the dependent behavior. In this paper, we present the CDP algorithm in Algorithm 2, in which node i, in state O, transits to states P and/or S, and, in state S, it adopts the dependent behavior. Initially, all N nodes in the swarm are in state O and are randomly distributed in a given terrain with random velocities in $[\mathbf{v}_{min}, \mathbf{v}_{max}]$. The inertia w is used to controlling the exploration and exploitation abilities of the swarm and eliminating the need for velocity clamping (i.e., if $|\mathbf{v}_i| > |\mathbf{v}_{max}|$ then $|\mathbf{v}_i| = |\mathbf{v}_{max}|$). The inertia is very important to ensure convergent behavior; large values for w facilitate exploration with increased diversity while small values promote local exploitation. We adopt a dynamically changing inertia values, i.e., an initially value decreases nonlinearly to a small value allowing for a shorter exploration time (due to context validity rate) with more time spent on refining optima [20]. That is,

$$w(t+1) = \frac{(w(t) - 0.4)(t_0 - t)}{t_0 + 0.4}$$

$w(0) = 0.9$, t_o is the maximum number of iterations. In case a node transits to state O then it re-sets w to its initial value. The randomly moving M sources generate context with sensing rate q (in samples/second, Hz) and the thresholds $\theta_l = a$ for the properties Y_l are set. The c_1 and c_2 constants denote how much the *lbest* and *pbest* solutions influence the movement of the node; usually $c_1 = c_2 = 2$ ([18]). The $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ are two random vectors with each component be a uniform random number in [0, 1]. In each iteration, a node i in O, or P, adjusts its movement (lines 18, 19) w.r.t. *pbest* and *lbest* (lines 20-28) once interactive communication takes place. If the node i in S adopts:

- dependent behavior then it adjusts its movement w.r.t lines 21-28,
- independent behavior then it randomly moves with $\mathbf{v}_i$ in $[\mathbf{v}_{min}, \mathbf{v}_{max}]$ (omit lines 17-28).

The end-criterion of the algorithm can be the number of iterations, the time needed to find fresh context a given portion of nodes, or energy consumption constraints. In our case the end-criterion is time dependent since the validity of context depends on the sensing rate q. Nodes adopting the independent behavior stop searching as long as they obtain fresh context and re-start foraging once context turns obsolete. The end-criterion for the dependent behavior is the minimum mean value $g_+(t)$ for the fitness function g. We require that $g_+(t)$ be as low as possible w.r.t. the a threshold that is, maximize $d(t) = (g_+(t) - a)^2$. The $d(t)$ value denotes how much fresh is context. In other words, it reflects the portion of time needed for context to turn obsolete as long as $g_+(t)$ is greater than a. For instance, let two nodes, i and j, carry context $\mathbf{y}$ with $g_i(\mathbf{y}) = a/2$ and $g_j(\mathbf{y}) = a/4$. Objectively, both nodes carry fresh context w.r.t. a. Therefore, node j carries fresher context than node i since node j will carry fresh context for longer time than node i. The convergence $g_+(t_o)$ value denotes a state in which some nodes obtain fresh context for $t \geq t_o$ and depends highly on a: a high value of a denotes a little time for context to turn obsolete. In that case, the nodes may stay for a long in S state. On the contrary, a low value of a (i.e., nodes are interested only for up-to-date context) results in values of $g_+(t)$ close to a; context turns obsolete with a high rate. It is of high interest to examine the efficiency of each behavior.

Algorithm 2. The Context Discovery Problem Algorithm

1. **Set** c_1, c_2, N, M, q
2. **Set** random $\mathbf{x}_i(t)$, threshold $\theta_i\mathbf{y} = \theta\mathbf{y}$, $t \leftarrow 0$
3. **For** $i = 1: N$
4. $\mathbf{v}_i(t) \sim \mathrm{U}(\mathbf{v}_{min}, \mathbf{v}_{max})$, $k_i \leftarrow \mathrm{O}$
5. $g_i^*(\mathbf{y}) \leftarrow (g_1(\mathbf{y}, t) + \cdots + g_{|Ni|}(\mathbf{y}, t)) / |N_i|$
6. $\mathbf{x}_i^* \leftarrow \mathbf{x}_i(t)$
7. $g_i^\#(\mathbf{y}) \leftarrow max_l\{g_l(\mathbf{y}, t)\}),\ l \in \{N_i\} \cup \{i\}$
8. $\mathbf{x}_i^\# \leftarrow \mathbf{x}_e$: $e = \arg max_l\{g_l(\mathbf{y}, t)\}),\ l \in \{N_i\} \cup \{i\}$, $leader_i \leftarrow e$
9. **Next** i
10. **While** (the *end criterion* is not met) **Do**
11. $t \leftarrow t + 1$;
12. **For** $i = 1: N$
13. **Calculate** $g_i(\mathbf{y}, t)$
14. **Validate** $g_i^*(\mathbf{y})$, $g_i^\#(\mathbf{y})$ *//increase validity indicators*
15. **Next** i
16. **For** $i = 1: N$
17. **Set** random unary vectors $\boldsymbol{r}_1, \boldsymbol{r}_2$
18. $\mathbf{x}_i(t) \leftarrow \mathbf{x}_i(t-1) + \mathbf{v}_i(t)$
19. $\mathbf{v}_i(t) \leftarrow \mathbf{v}_i(t-1) + c_1\boldsymbol{r}_1 (\mathbf{x}_i^* - \mathbf{x}_i(t\text{-}1)) + c_2\boldsymbol{r}_2 (\mathbf{x}_i^\# - \mathbf{x}_i(t\text{-}1))$
20. $g_{Ni}(\mathbf{y}, t) \leftarrow (g_1(\mathbf{y}, t) + \cdots + g_{|Ni|}(\mathbf{y}, t)) / |N_i|$
21. **If** $g_i^*(\mathbf{y}) < g_{Ni}(\mathbf{y}, t)$ **Then**
22. $\mathbf{x}_i^* \leftarrow \mathbf{x}_i(t)$
23. $g_i^*(\mathbf{y}) \leftarrow g_{Ni}(\mathbf{y}, t)$
24. **End**
25. **If** $g_i^\#(\mathbf{y}) < max_l\{g_l(\mathbf{y}, t)\}),\ l \in \{N_i\}$ **Then**
26. $\mathbf{x}_i^\# \leftarrow \mathbf{x}_e$: $e = \arg max_l\{g_l(\mathbf{y}, t)\}),\ l \in \{N_i\}$, $leader_i \leftarrow e$
27. $g_i^\#(\mathbf{y}) \leftarrow max_l\{g_l(\mathbf{y}, t)\}),\ l \in \{N_i\}$
28. **End**
29. **If** $g_i^\#(\mathbf{y}) < g_i(\mathbf{y}) < \theta\mathbf{y}$ **Then** $k_i \leftarrow \mathrm{O}$
30. **If** $g_i(\mathbf{y}) < g_i^\#(\mathbf{y}) < \theta\mathbf{y}$ **Then** $k_i \leftarrow \mathrm{P}$
31. **If** $\theta\mathbf{y} < g_i^\#(\mathbf{y})$ **Then** $k_i \leftarrow \mathrm{S}$
32. **Next** i
33. **End While**

5 Performance Evaluation

In this section we assess the proposed behavior for the CDP. Our objective is to enable nodes to discover and maintain fresh context. However, the fact of locating sources and leaders in an attempt to carry fresh context is at the expense of the inherent network load due to communication of nodes. We define as efficiency $e(t)$ of a certain behavior the portion of nodes $n(t)$ being in state S out of the communication load $l(t)$ among neighboring nodes exchanging information about context quality, i.e., $e(t) = n(t) / l(t)$. We require that $e(t)$ assumes high values minimizing the load $l(t)$ and maximizing $n(t)$ w.r.t. the adopted behavior.

The parameters of our simulations are: a swarm of $N = 100$ nodes, $M = 2$ sources, $a = 100$ time units, a terrain of $L = 100$ spatial units, transmission range $R = 0.01L$, the *random waypoint* model for mobility behavior in $[\mathbf{v}_{min}, \mathbf{v}_{max}] = [0.1, 2]$ ([13]), and 1000 runs of the algorithm. Context turns obsolete every a time units and is sensed by the sources with q ranging from $(2/a)$ Hz to 1Hz. We require that $g_{+}(t)$ be lower than a as time passes or, at least, lower than a between consequent intervals of a time units.

Figure 2 depicts the $g_{+}(t)$ value (in time units –t.u.) when all nodes in the swarm adopt the dependent behavior for different values of q. It is observed that all nodes rapidly locate leaders and then carry fresh context denoting CDP algorithm convergence. The $g_{+}(t)$ value converges to $g_{+}(t_o)$ ranging from 14.633t.u. to 40.882t.u. for q ranging from 1Hz to 0.02Hz, respectively. It is worth noting that, for q = 1Hz, the $g_{+}(t_o)$ is 14.633t.u. i.e., 14.633% of the validity threshold a indicating that most nodes can process context for 85.367% of the sensing time before it turns obsolete. Moreover, as q assumes low values (e.g., 0.02Hz), which means that the sources sense context every 50t.u., the value of $g_{+}(t)$ swings around the 40.882t.u. This indicates that, nodes locate sources whose context turns obsolete after 50t.u. For that reason, the $g_{+}(t)$ value for such nodes exhibits that behavior. On the other hand, once q assumes high values (e.g., 1Hz), the sources constantly carry up-to-date context. Consequently, nodes that locate sources carry fresh context ($g_{+}(t)$ converges). The achieved maximum value for $d(t_o)$ is $0.6952.10^4$ for q = 1Hz, as depicted in Figure 3, compared to $0.3854.10^4$ w.r.t. independent behavior, as discussed later. Evidently, by adopting the dependent behavior, a large portion of the swarm follows leaders and/or sources carrying objectively fresh context. However, such behavior requires that nodes communicate

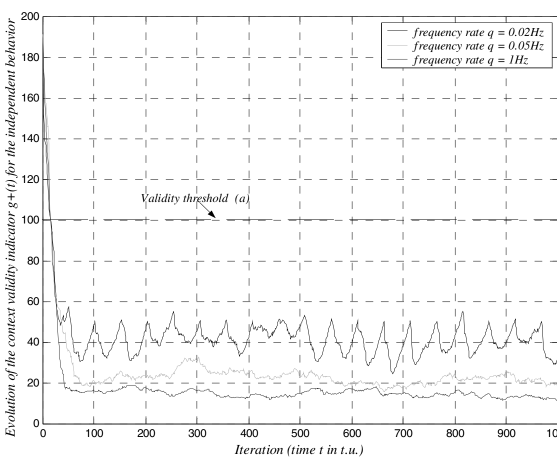

Fig. 2. The $g_{+}(t)$ value of the dependent behavior for sensing rate q = 0.02Hz, q = 0.05Hz and q = 1Hz

continuously in order to locate sources and leaders with more fresh context even if nodes are in state S for maximizing $d(t)$. That leads to additional communication load thus keeping the efficiency to 50% as depicted in Figure 4. Specifically, Figure 4 depicts the value of $e_d(t)$ for the dependent behavior for q = 1Hz. Obviously, the inherent communication load of such behavior is high since a large portion of nodes attempts to carry fresh context.

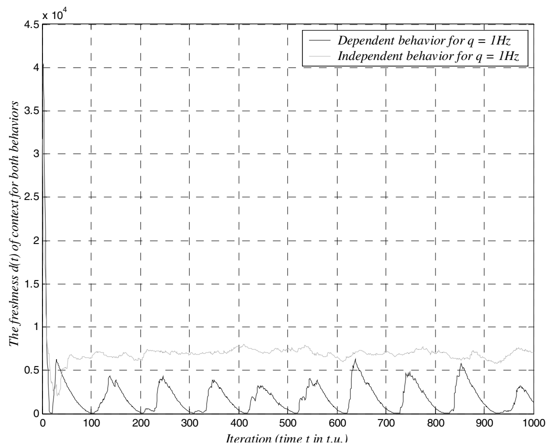

Fig. 3. The $d(t)$ value of the dependent and independent behavior with sensing rate q = 1Hz

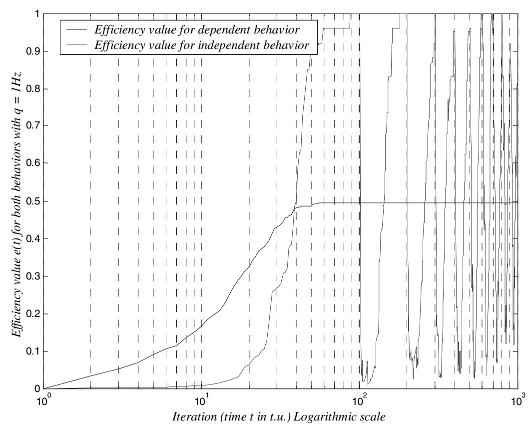

Fig. 4. The values of $e_d(t)$ and $e_i(t)$ efficiency in logarithmic scale for q = 1Hz

Figure 5 depicts the $g_+(t)$ value of nodes adopting the independent behavior. We illustrate $g_+(t)$ for sensing rates q = 1Hz, q = 0.05Hz and q = 0.02Hz. Evidently, nodes seek for fresh context only when the existing context turns obsolete. This is indicated by the sharp bend of $g_+(t)$ between intervals of a time units for q = 1Hz. The periodic behavior of $g_+(t)$ reflects the idea of the independent behavior denoting that a node is about to seek for context only when needed. Hence, between intervals of a time units nodes that are in S state save energy as long as they do not exchange information with others. When context turns obsolete, nodes re-start foraging but having the *pbest* solution as a candidate starting point. This means that, each time context turns obsolete nodes adjust its movement based on the last known *pbest* solution. Hence, they start moving "blindly" as long as their first direction might be the *pbest* position indicating "prolific" neighborhood. For that reason, the maximum value of $g_+(t)$ is close to a in each "period" as depicted in Figure 5. Moreover, the $g_+(t)$ value ranges from 40 t.u. to 100 t.u. compared to the convergence value of $g_+(t_o)$ = 14.633t.u. in case of the dependent behavior for q = 1Hz. It is worth noting that the value of $g_i^*(\mathbf{y})$ for *pbest* must denote valid context, otherwise node i has to move entirely at random. Moreover, consider the $g_+(t)$ value having $q = 2/a = 0.02$Hz. Specifically, $g_+(t)$ assumes the minimum value every (a/2) = 50 t.u., which is greater than the minimum value of $g_+(t)$ achieved for q = 1Hz every at.u. In the former case nodes re-start foraging sooner than in the latter case (practically two times more), thus, the adoption of the *pbest* solution seems more prolific. For that reason, $g_+(t)$ assumes higher minimum values in the former case even though the sensing rate is lower. In such cases, the adoption of the *pbest* solution is of high importance.

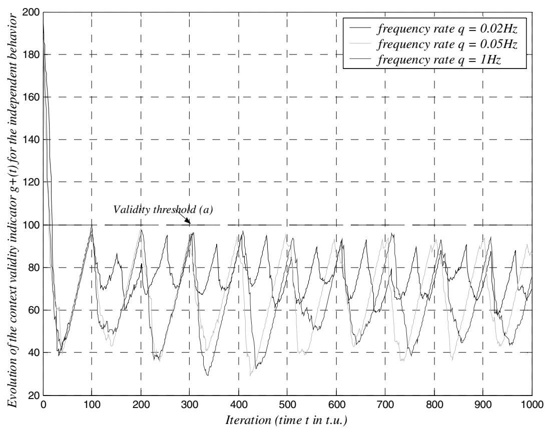

Fig. 5. The g+(t) value of the independent behavior for sensing rates q = 1Hz, q = 0.05Hz and q = 0.02Hz

Figure 3 depicts also the $d(t)$ value for the independent behavior. The $d(t)$ assumes the maximum value (therefore lower than in the case of the dependent behavior) only when a large portion of nodes carry fresh context. In addition, $d(t)$ assumes zero value regularly every a time units denoting the time that all nodes carry obsolete context. The mean value of $d(t)$ is $0.3854.10^4$, that is 44.56% lower than the convergence value of $d(t_o)$ in the case of the dependent behavior (for the same sensing rate q = 1Hz). Hence, the adoption of the independent behavior for CDP results in 44.56% lower quality of context than that achieved by dependent behavior.

By adopting the independent behavior we can achieve high values of efficiency $e_i(t)$ during intervals in which nodes carry fresh context. This is due to the fact that in such intervals nodes stop communicating with each other thus reducing the load $l(t)$.

However, when context turns obsolete then $e_i(t)$ assumes a very low value (mean value lower than 0.1) as long as a large portion of nodes do not carry fresh context thus reducing $n(t)$. In Figure 4 the behavior of $e_i(t)$ for sensing rate q = 1Hz is also illustrated. We can observe that $e_i(t)$ ranges from 0.069 (mean value) to 0.93 (mean value) compared to the convergence value of $e_d(t)$ = 0.5.

Each behavior can be applied on a context-aware application considering the specific requirements of the application. Once the application needs critically up-to-date context then the adoption of the dependent behavior is preferable. On the other hand, once we are interested in saving energy then nodes can adopt the independent behavior. However, a hybrid scheme combining both behaviors can be adopted. For instance, a portion of nodes can adopt the independent behavior for reducing energy consumption and the rest nodes adopt the dependent behavior maintaining up-to-date information. Another combination refers to the adoption of the dependent behavior for rapidly locating sources and leaders followed by the adoption of the independent behavior till context turns obsolete.

6 Conclusions

PSO is a simple algorithm with a wide application range on different optimization problems. We deal with the CDP by adopting the decentralized control and self-organization of SI. We provide the mapping between PSO and CDP, and study how SI-inspired computing can facilitate context discovery. We introduce the time-variant context quality indicator g that refers to the fitness function f in PSO. Hence, each particle-node attempts to carry and maintain fresh context w.r.t. the g indicator. We propose the independent and dependent foraging behaviors (strategies) for mobile nodes The use of such behaviors in conjunction to the local fitness of the neighborhood enables node to discover sources and/or leaders that provide up-to-date contextual information. The proposed algorithm for the CDP supports such behaviors provided that context turns obsolete over time in a dynamic environment. We evaluated the efficiency and the effectiveness of each behavior. The adoption of each behavior relies on the context-aware application itself: for critically up-to-date context constrained applications the dependent behavior is preferable while, once energy savings are of high importance then the independent behavior exhibits satisfactory results. Our simulation results indicate the applicability of SI in context discovery and the proposed foraging behaviors provide useful tools in mobile computing. In addition, the adoption of SI for transmission range / power adjustment, so that context-aware nodes control their energy consumption, is a future work that can be considered in the CDP.

References

1. Deneubourg, J., Goss, S., Sandini, G., Ferrari, F., Dario, P.: Self-Organizing Collection and Transport of Objects in Unpredictable Environments. In: Symposium on Flexible Automation, pp. 1093–1098 (1990)
2. Drogoul, A., Ferber, J.: From Tom Thumb to the Dockers: Some Experiments With Foraging Robots. In: 2nd Int. Conference on Simulation of Adaptive Behavior, pp. 451–459 (1992)

3. Parker, L.E.: ALLIANCE: An Architecture for Fault-Tolerant Multi-Robot Cooperation. IEEE Transactions on Robotics and Automation 14(2), 220–240 (1998)
4. LaVille, S.M., Lin, D., Guibas, L.J., Latombe, J.C., Motwani, R.: Finding an Unpredictable Target in a Workspace with Objects. In: IEEE Int. Conf. on Robotics and Automation, pp. 737–742 (1997)
5. Dorigo, M., Gambardella, L.M.: Ant Colony Systems: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans. on Evolutionary Computation 1(1), 53–66 (1997)
6. Pasino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M., Baum, M.: Cooperative Control for Autonomous Air Vehicles. In: Murphy, R., Pardalos, P. (eds.) Cooperative Control and Optimization. Kluwer Academics Publishers, Boston (2002)
7. Polycarpou, M., Yang, Y., Pasino, K.: A Cooperative Search Framework for Distributed Agents. In: IEEE Int. Symposium on Intelligent Control, pp. 1–6 (2001)
8. Stone, L.D.: Theory of Optimal Search. Academic Press, New York (1975)
9. Vincent, P., Rubin, I.: A Framework and Analysis for Cooperative Search Using UAV Swarms. In: ACM Symposium on Applied Computing (2004)
10. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco (2001)
11. Anagnostopoulos, C., Tsounis, A., Hadjiefthymiades, S.: Context Awareness in Mobile Computing Environments, Special Issue on Advances in Wireless Communications Enabling Technologies for 4G. Wireless Personal Communication Journal 2(3), 454–464 (2007)
12. Anagnostopoulos, C., Sekkas, O., Hadjiefthymiades, S.: Context Fusion: Dealing with Sensor Reliability. In: IEEE Int. Workshop on Information Fusion and Dissemination in Wireless Sensor Networks - IEEE Int. Conference on Mobile Ad-hoc and Sensor Systems, pp. 1–7 (2007)
13. Bettstetter, C., Hartenstein, H., Pérez-Costa, X.: Stochastic properties of the random waypoint mobility model. ACM/Kluwer Wireless Networks: Special Issue on Modeling and Analysis of Mobile Networks 10(5) (September 2004)
14. Matthias, O., Hanspeter, A.: Biomimetic robot navigation. Robotics and Autonomous Systems 30(1-2), 133–153 (2000)
15. Savvides, A., Han, C., Srivastava, M.: Dynamic Fine-grained Localization in Ad-hoc Networks of Sensors. In: ACM Mobicom, July 2001, pp. 166–179 (2001)
16. Anjum, F., Mouchtaris, P.: Security for Wireless Ad Hoc Networks. Wiley-Interscience, Hoboken (2007)
17. Zhao, F., Guibas, L.: Wireless Sensor Networks - An Information Processing Approach. Elsevier Science, Amsterdam (2004)
18. Engelbrecht, A.P.: Computational Intelligence: An Introduction, 2nd edn. Wiley, Chichester (2007)
19. Eberhart, R.C., Shi, Y.: Tracking and Optimizing Dynamic Systems with Particle Swarms. In: IEEE Cong. on Evolutionary Computation, vol. 1, pp. 94–100 (2001)
20. Peram, T., Veeramachaneni, K., Mohan, C.K.: Fitness-Distance-Ration based Particle Swarm Optimization. In: IEEE Symp. Swarm Intelligence, pp. 174–181 (2003)
21. Carlishe, A., Dozier, G.: Adapting Particle Swarm Optimization to Dynamic Environments. In: Intl. Conf. Artificial Intelligence, pp. 429–434 (2000)
22. Hu, X., Eberhart, R.C.: Tracking Dynamic Systems with PSO: Where's the Cheese? In: Proc. Workshop on Particle Swarm Optimization, pp. 80–83 (2001)
23. Hu, X., Eberhart, R.: Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. In: IEEE Congress on Evolutionary Computation, USA, pp. 1666–1670 (2002)