Towards Failure-Resistant Mobile Distributed Systems Inspired by Swarm Intelligence and Trophallaxis

Arles Rodríguez¹, Jonatan Gómez¹ and Ada Diaconescu²

¹ALIFE Research Group, Universidad Nacional de Colombia
²Telecom ParisTech, LTCI CNRS
{aerodriguezp,jgomezpe}@unal.edu.co, ada.diaconescu@telecom-paristech.fr

Abstract
Designing distributed algorithms for mobile ad-hoc sensor systems is difficult, not at least because of their asynchronous communication, mobility, absence of shared memory and high risk of failures. To deal with these challenges, some techniques like replication and consensus are proposed in the literature. However, techniques like consensus depend on a leader election, and this leader can fail. In this paper, we present some advances inspired from nature for the design of a decentralized, scalable way for getting and synchronizing information among components in a point-to-point way. To achieve this, we address the problem of getting and synchronizing information by defining a Distributed System as a swarm of agents (termites), which look for information. Termites are designed with the task of exploring a simulated environment, sensing some desired data distributed throughout the space, and sharing their local knowledge regarding the environment with other nestmates only if they are neighbors. However, when failure rates increase it is less probable than a termite completes the entire task by itself before all termites fail. In order to allow at least one termite to gather the complete information from the environment, several solution approaches are proposed, like sequential exploration with one agent as reference, random movements with local information exchanges, Levy walks and a pheromone-based exploration algorithm inspired by Ant Colony System. This algorithm allows a termite to explore data in a world and enables a termite to search other nestmates with more information by using a trace and by defining a search status given the amount of local information than a termite has. Results show how swarms manage to collect and replicate information from the entire space even when failures occur. By local interactions, almost all the termites get complete information from a defined world before failing, without a central control and with simple local rules.

Introduction
A Distributed System consists of a collection of components (e.g. processes, agents, robots and nodes) connected via a network, that coordinate their activities and share system resources, offering different services to users and appearing to them as a single system (Tanenbaum and Steen, 2006). Coordination and cooperation between processes are necessary and communication is a key point of Distributed Systems from a design point of view because each process has partial knowledge of the environment, acts in a local way and can fail (Raynal, 2013).

Distributed systems must be scalable. It means being adaptable to changes in terms of size (to provide an easy way to add or delete resources), geographical localization of components and provide easy management independently of its size (Tanenbaum and Steen, 2006). Scalability involves the design and implementation of decentralized algorithms dealing with components that can fail, have no complete information, take decisions in a local way and use point-to-point communication because broadcast is not possible. Lack of scalability usually implies loss of performance of a system while a system grows up (Tanenbaum and Steen, 2006).

Autonomic Computing addresses complexity with the idea of a computer system that adapts to changes without human intervention (Lalanda et al., 2013). Inspired by nature, the idea is that systems elements manage themselves while also providing their services (Kephart and Chess, 2003). The internal behavior of an autonomic element and the set of relationships with other elements are based on goals that a designer has embedded in it and on goals incorporated by other systems through subcontracts with other elements with its tacit or explicit consent (Kephart and Chess, 2003). A set of Self-* properties are required to achieve from the Self-management goal: adapting to the addition or deletion of components (Self-configuration), detecting and recovering from failures without disruption in the system operation (Self-healing), finding improvements in the efficiency of a system (Self-optimization), and anticipating and preventing of threats (Self-protection) (Lalanda et al., 2013; Kephart and Chess, 2003).

On the other hand, replication is proposed as an essential component of failure tolerance in distributed systems. A well known mechanism of replication in Distributed Systems is consensus. By consensus, it is expected that replicas have agreement over a value given local information in each process (Aguilera, 2010). Different areas of application includes state and log machine replication in databases (Agiilera, 2010; Ongaro and Ousterhout, 2013), motion planning,
alignment problems (Nedic et al., 2010) and information processing in sensor networks (computing averages of local observations) (Ozdaglar and Nédic, 2007).

Some existing consensus algorithms like Paxos (Lamport, 1998) and Raph (Ongaro and Ousterhout, 2013) propose distributed consensus algorithms based on a leader election process. This leader receives information and replicates it to other processes. However, natural systems do not require a leader directing them and self-organisation relies on collective behaviors that emerge from local interactions as happens in insect colonies where collaboration emerges from Stigmergy (Doursat et al., 2012).

An important challenge to deal with failures is to collect, replicate and synchronize information in a fast way based on the development of adaptable ways of point-to-point communication. In this paper, we propose a solution to the sharing information problem that deals with some of the challenges introduced before. A simulated world is defined with some data of interest distributed in the environment. To get information in this world, we define agents called termites with the task of exploring and obtaining this desired data. A termite can move, sense data in its current location and can share its local information with other neighbors. However, termites are unreliable and can crash with a given probability $p_f$. Given these conditions, the problem is how to get at least one termite to collect whole information of a world before they all crash.

Agents are called “termites” because they follow social organization and their feeding is carried out via trophallaxis, meaning that food is stored in their stomach and it is transferred among nestmates through mouth-to-mouth feeding (Rodriguez and Gomez, 2011). In this paper, local information is also inside each agent and local exchange of information is performed between neighbors that look for more information present in other nestmates using stigmergy. Trophallaxis is important in nutritional dynamics and communication of many social insects, individual foragers return from a food resource and transfer a portion of their gut material to one or several nestmates. These recipients subsequently become donors to others, and the process continues (Suárez and Thorne, 2000).

This paper uses stigmergy for guiding termites in their search of new information enabling termites to explore a terrain, to get data information from other nestmates and to synchronize local information with others. Our approach inspired by trophallaxis and swarms allows termites to get the whole data before they crash in a decentralized way. A swarm features self-organization that allows it to continue working even if some termites die looking for new information. Inspired by this feature, by adding failures to termites we aim to determine if our proposal can achieve terrain exploration and failure resistance in a simple way inspired by nature; and also establish limits. The remaining of this paper is organized as follows: next section presents a detailed description of the problem, the following section shows some approaches to solve the problem including our proposal of sharing information based on stigmergy and trophallaxis. Finally a result analysis is performed and some conclusions are drawn.

The Problem: World exploration

A world is a bidimensional toroidal space defined as a matrix of properties $Props_{width \times height}$. $Props$ is a collection $Props = \{\tau_w, data\}$, defined with the following values: amount of pheromone in the world $\tau_w : \tau_w \in \mathbb{R} \land \tau_w \in [0, 1]$ and data : data $\in \mathbb{R} \land data \in [0, 1]$. data represents some information of interest in this world (e.g. temperatures, altitudes, distance to a determined objective). By now, data values do not change because the main objective is to find a way to get all data information and share it in a fast way. At the beginning all the world positions have a pheromone value of 0.5 and data is generated for each location in a random fashion.

Termites Definition

Multi-agent systems define agents capable of independent actions with the ability of interacting with others. In order to achieve their tasks they are required to cooperate, coordinate and negotiate which each other (Balaji and Srinivasan, 2010). Each termite senses information from the environment and acts by using actuators (Russell and Norvig, 2004). Agents by now are reactive. It means, that each termite operates to respond to changes and to satisfy its design objectives (Russell and Norvig, 2004). Objectives are defined in a termite program which determines the action to be executed by an agent according to its local knowledge. The main thread of each agent looks like algorithm 1. While an agent is alive (status != Action.DIE) this agent senses its environment, then chooses an action based on its perceptions and finally the action has an effect on the environment.

```
while (status != Action.DIE) {
    Percept p = environment.sense(this);
    Action action = compute(p);
    environment.act(this, action);
}
```

Algorithm 1: Termite main program

In this paper, each termite is designed with the objective of getting data information in a simulated world of size $width \times height$. data values are represented as a matrix of continuous values $\mathbb{R}^{width \times height}$. The idea is that termites cooperate and coordinate among them to get the global data information starting from local perceptions. The main idea is that each agent looks for and senses new data locally and shares its collected data at the same time.
Different perceptions have been defined for termites $\text{Percept} = \{\text{pheromone}, \text{data}, \text{socialStatus}, \text{neighbor}, \text{msg}, \text{loc}\}$. $\text{pheromone}$ is a vector $\mathbb{R}^n$ with values in $[0, 1]$ representing the amount of pheromone that a termite has in its vicinity (Moore neighborhood $r = 1$ with center in the termite location (Gray, 2002)); $\text{data}$ is the information in the current location of the termite, $\text{socialStatus} = \{\text{SEEKER, CARRIER}\}$ indicates the status of a termite, $\text{neighbor}$ returns the id of a nestmate randomly selected from its Moore neighborhood if one exists, $\text{msg}$ stores new messages received from other nestmates and $\text{loc}$ returns the current termite location as a matrix position ($\text{row}, \text{column}$).

A termite has the following actions $\text{Actions} = \{\text{none, down, left, right, up, upleft, upright, downright, downleft, Die, Collect, Send, Receive}\}$. First nine actions make reference to movements in the world and $\text{Die}$ stops an agent’s thread to simulate failure. $\text{Collect}$ is performed in each round and means to store the $\text{data}$ collected in the local memory of termite. A termite $s$ sends its $\text{data}$ collected $I_s$ to a termite $r$ in a $\text{msg}$ encoded as a collection $\text{msg} = [I_s]$ in a process defined as $\text{Send}(r, \text{msg})$. Sending this information is inspired by traditional Asynchronous Distributed Systems where there are FIFO communication channels (Messages received first are processed first by each agent (Raynal, 2013), local communication (each agent receives and sends messages only to its neighbors), and there is no timing assumptions regarding message delay, clock drift or time taken to send a message. The control returns back to the invoking process after the data is copied in the buffer of the process that receives (Chandra and Toueg, 1996; Kshemkalyani and Singhal, 2008). To implement this mechanism there is a FIFO queue mailbox in the proposed model, a crash is equivalent to the death of a termite. This kind of failure is implemented by defining a probability of failure ($p_f$) for the agents, see Algorithm 2. For example, $p_f = 0.1$ means that a process has a probability of failure in 1 of 10 rounds.

### Solution Approaches

In this section, some approaches are proposed to solve the aforementioned problem. First, a sequential solution is defined given one termite. After that, more termites that communicate among themselves are added into the environment featuring a random strategy of exploration. Finally a solution based on stigmergy is proposed. For experiments, a fixed size of the environment is defined ($width = 50$, $height = 50$) and several values of $p_f$ are defined. The idea is to estimate the amount of information that agents can get from the environment.

### Sequential Exploration with One Termite

Sequential exploration with one termite is modeled as point of reference given the definition of the world as a matrix. Sequential exploration is a good strategy to explore the world because it implies exploring each location in the world only one time. The termite $i$ knows its current location as $\text{loc}_i = (x, y)$ where $x$ corresponds to $\text{rows}$ and $y$ corresponds to $\text{columns}$. Given this location a simple movement program of the termite is the depicted in Algorithm 3 and Fig 1.

One termite is set to get $\text{data}$ in the world of $\text{Props}^{50 \times 50}$ (size of the world equal to 2500). The termite is located in

```
1 //process can fail with a probability of 0.1
2 double probFailure = 0.1;
3 Percept p;
4 Action actions;
5
6 while (status != Action.DIE) {
7   if (Math.random() < probFailure) {
8       status = Action.DIE;
9       }
10   p = environment.sense(this);
11   actions = compute(p);
12   environment.act(this, actions);
13 }
```

Algorithm 2: Crash model for Agent Program of termite $i$
while (status != Action.DIE) {
  ...
  if (x != width-1) {
    action = right;
  } else {
    action = downright;
  }
  ...
}

**Algorithm 3:** Sequential exploration for termite \(i\)

![Sequential Exploration Algorithm](image)

Figure 1: Sequential Exploration Algorithm.

a random location and the world is explored in a sequential way. Given this size of the world, a maximum of 2500 movements is enough for completing information. Sequential exploration experiments were performed 30 times for several values of \(p_f\) and results were averaged with the aim of determining how much information a termite can collect in each case.

Table 1 presents the summary of experiments for sequential exploration. Column *Data Collected* displays the average and standard deviation of data collected over the 30 experiments performed. As expected, a greater value in the \(p_f\) parameter implies a lesser probability of exploring the entire world. For instance, for a \(p_f \geq 10^{-3}\) exploration of the world was never achieved. For the experiments performed with a \(p_f = 4 \times 10^{-4}\) a full exploration of the world was achieved only in 12 of the 30 executions. For \(p_f = 10^{-5}\) this was achieved in 28 of 30 experiments. The single termite could explore the entire world in a reliable way only if \(p_f \leq 10^{-6}\).

**Random Exploration with Communication**

In these experiments, more than one termite is added to explore the world in a random fashion starting in random locations in the same environment (\(Prop50^{50x50}\)). Termites move in a random way in the environment but have the same \(p_f\). For the same world, experiments were performed with populations of 10, 30 and 50 termites. A maximum of 3000 iterations were defined to get the complete information and each experiment was performed 30 times. Termites can communicate and exchange information with neighbors as specified in the last section (*Termites Definition*).

**Levy Walk Exploration With Communication**

A Levy walk is a movement process in which a particle makes a sequence of movements in the same random direction during a time length that depends of another uniform random variable. Foraging mechanisms present in some animals appears to obey Levy walks (Benhamou, 2007). Levy Walks have been used for solving the networking coverage problem in robots by moving them until find a location with an acceptable number of neighbors and make connections (Beal, 2013). In this paper, we adapt the motion mechanism of (Beal, 2013) in the following way (Alg. 4): \(randomDir()\) returns a random direction \(dir \in \{down, left, right, up, upleft, upright, downright, downleft\}\), \(\alpha\) is a uniform random number that represents the increment rate of \(accumulator\), \(T\) defines a threshold of \(accumulator\) for generate a new direction \(dir\) and \(\Delta t\) defines an increase of \(\alpha\) in terms of time. Several experiments using Levy walks were performed using \(T = 1\) and \(\Delta t = 1\) and varying the other parameters in the same way it was done with random exploration.

<table>
<thead>
<tr>
<th>(p_f)</th>
<th>Data Collected</th>
<th>Successful</th>
<th>Experiments</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10^{-1})</td>
<td>6.8 ± 6.17</td>
<td>0/30</td>
<td></td>
</tr>
<tr>
<td>(10^{-2})</td>
<td>113.27 ± 90.36</td>
<td>0/30</td>
<td></td>
</tr>
<tr>
<td>(10^{-3})</td>
<td>792.2 ± 560.57</td>
<td>0/30</td>
<td></td>
</tr>
<tr>
<td>(4 \times 10^{-4})</td>
<td>1532.93 ± 906.9</td>
<td>12/30</td>
<td></td>
</tr>
<tr>
<td>(10^{-4})</td>
<td>2047 ± 825</td>
<td>22/30</td>
<td></td>
</tr>
<tr>
<td>(10^{-5})</td>
<td>2496.2 ± 14.85</td>
<td>28/30</td>
<td></td>
</tr>
<tr>
<td>(10^{-6})</td>
<td>2500 ± 0</td>
<td>30/30</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2500 ± 0</td>
<td>30/30</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Summary of experiments for crash sequential exploration (*Population = 1, width = 50, height = 50*. A \(p_f = 4 \times 10^{-4}\) is added because \(1/(width \times height) = 4 \times 10^{-4}\).

**Pheromone-based Exploration**

In this approach, each termite determines its movements by using stigmergy inspired by swarms and the Ant Colony
System algorithm (Dorigo and Gambardella, 1997). As a main difference, in this proposal termites are looking for new information (present in other nestmates) instead of looking for food. In this way, termites will have a status determined by the amount of local information that each one has: SEEKERS which are termites that look for others for getting new information and to explore locations with more pheromone, and CARRIERS that are termites believing they have more information than others and explore world locations with less pheromone. Pheromone value \( \tau_w \) defined in an environment is used. At the beginning all world locations have a pheromone value of 0.5.

A termite \( i \) chooses a direction \( \text{dir} \) that corresponds to the application of a biased exploration or an exploitation rule depending of a random variable \( q \in [0, 1] \) by applying Eq. 1 (Dorigo and Gambardella, 1997):

\[
\text{dir} = \begin{cases} 
\text{exploitation rule} & \text{if } q \leq 0.9 \\
\text{biased exploration} & \text{otherwise}
\end{cases}
\] (1)

Exploitation rule generates a direction depending of a termite status:

- **SEEKERS**: If a termite is SEEKER, this termite will choose the direction with the maximum amount of pheromone in its vicinity looking for CARRIERS. If all values in the vicinity are the same, a random direction is chosen.

- **CARRIERS**: If a termite is CARRIER, this termite will choose the direction with the minimum amount of pheromone in its vicinity. If all values in the vicinity are the same, a random direction is chosen.

Biased exploration is a random-proportional rule (Dorigo and Gambardella, 1997) which gives to a termite \( i \) a probability of choosing a direction \( p_d(x, y) \) depending the amount of pheromone \( \tau_w \) in its vicinity \( \text{neighborhood}(i) \) (Eq. 2). \( \text{neighborhood}(i) \) represent the locations in the Moore neighborhood of \( i \) with \( r = 1 \):

\[
p_d(x, y) = \frac{\tau_w(x, y)}{\sum_{(k, l) \in \text{neighborhood}(i)} \tau_w(k, l)}
\] (2)

Each time a termite \( i \) performs a movement, it updates its local amount of pheromone \( \tau_l(i) \) (local update rule of Eq 3) and updates the pheromone in this world location \( \tau_w(x, y) \) (global update rule of Eq 4):

\[
\tau_l(i) = (\tau_l(i) + 0.01 \times (0.5 - \tau_l(i)))
\] (3)

\[
\tau_w(x, y) = \tau_w(x, y) + 0.01 \times (\tau_l(i) - \tau_w(x, y))
\] (4)

If a termite \( i \) turns into a SEEKER or finds new information, its pheromone value is updated to 0 (\( \tau_l(i) = 0 \)). Equation 3 is based on the local update rule of ACS (Dorigo and Gambardella, 1997) and represents a local update rule that makes possible that termite pheromone value increases with the time until a certain point reducing the amount of pheromone of the world (global update rule of equation 4). This influence is reduced with the time making pheromone in the world converge to its default value of 0.5. A termite in SEEKER state can reach a maximum value of pheromone of 0.5.

If a termite \( i \) becomes a CARRIER or finds new information, the pheromone of the termite gets a value of 1 (\( \tau_l(i) = 1 \)). Local update rule of Equation 3, produces a decrease of the local amount of pheromone until reaching a minimum value of 0.5. At the same time, a CARRIER increases the amount of pheromone in the world (global update rule Eq. 4). This influence is reduced with the time making pheromone in the world converge to its default value (0.5). A termite in a CARRIER state can reach a minimum value of pheromone of 0.5.

If one termite has more information than the other termite then it turns into a CARRIER; or a SEEKER in the other case (Fig 2). In this way and just like in nature, termites are donors to other neighbors in a cascade scheme called Trophallaxis cascade. Each time that a termite \( s \) senses some another neighbor \( r \), \( s \) sends its information to \( r \). \( r \) merge its information \( I_r = I_s \cup I_r \) and additionally \( r \) calculates the difference between information \( \text{dif} = I_s \setminus I_r \). In this way two following scenarios are possible (Fig 2):

- \( \text{dif} = \emptyset \) means (in a local way) that \( r \) had at least the same information as \( s \), so termite \( r \) turns into a CARRIER and sets its pheromone value in one \( \tau_l(r) = 1 \).

- Otherwise (if \( \text{dif} \neq \emptyset \)), \( r \) turns into a SEEKER and sets its pheromone in zero \( \tau_l(r) = 0 \).

Figure 2: Communication and Status determination of termite \( r \).

Experiments were performed with 10, 30 and 50 termites. All agents at the beginning are SEEKERS, so they start exploring the world and when they make contact with another nestmate information interchange starts. Figure 3 shows how termites start as SEEKERS (white points) explore the world and turn into CARRIERS (blue points) over time. Red locations in the world represent the variation in the amount of pheromone with the time and how agents explore the world. Squares 3, 4, 5 and 6 of Figure 3 depict cases where communication occurs (green circles).

Fig 4 shows how exploration influences the states of the termites between SEEKERS and CARRIERS. The Termites axis represents the individuals in the simulation and the Iteration axis represents the average round number. After some iterations all the population becomes CARRIERS for all the experiments performed. Bigger populations makes termites turn into CARRIERS in a fast way.

Figure 3: Pheromone exploration with 10 termites, white points are SEEKERS and blue points are CARRIERS.

Results Analysis
Tables 2 and 3 present the results for the experiments of Random Exploration (column Random Expl.). The column Inf. Col. shows the average and standard deviation of data collected for agents in the 30 executions. Column Ag. Compl. is the average and standard deviation of agents with complete data for the 30 executions of each experiment, Table 3 presents the average round number of the agents that completed information at first place in each experiment.

In the experiments performed, a smaller value of $p_f$ means more data obtained and more termites with complete information. However with 10 termites ($Pop = 10$) it was impossible to get the complete data for random exploration. For the given size, random movements and small populations do not warrant exploration of the whole data in the 3000 iterations specified because is difficult for termites to meet and communicate and termites tend to repeat paths. For 30 and 50 termites, for a $p_f \geq 4 \times 10^{-4}$ it is observed more than one agent with all the data information (Agents Complete).

Columns Levy Walk Expl. and Ph. Expl. of Table 2 presents the results for Levy walks and pheromone exploration respectively. Results show that Levy walks and pheromone exploration work even with 10 termites exploring the world with a $p_f \leq 10^{-4}$ for Levy walks and a $p_f \leq 4 \times 10^{-4}$ in some experiments of pheromone exploration. For 30 and 50 termites the entire information is obtained even with a $p_f = 10^{-3}$. More data is collected as $p_f$ decreases. Bigger populations produce more exploration and a fast information dissemination.

Levy walks are a good technique for exploring new information because the average of information collected is higher compared to pheromone exploration (Table 2). However, the number of agents with complete information before 3000 iterations is bigger for pheromone exploration in 10 and 30 agents and a $p_f = 10^{-3}$. In table 3, it is observed
that pheromone exploration allows some individuals collect information in a faster way than the other two methods, because the best agents require a small number of rounds in collect all the information.

In the experiments performed, communication is important to reduce the time necessary for getting and disseminating information. Even communication in random exploration tends to reduce the number of rounds necessary for a termite to get all the information from 2500 of sequential exploration with one termite to 823.57 for 50 termites (Table 3). Results with pheromone exploration are even better getting an average number of 451.13 rounds needed for the best termite to collect the complete data without failures ($p_f = 0$).

As expected, bigger populations provide a better performance for exploration and for sharing information in a decentralized, scalable and simple way even with unreliable termites. Additionally, Table 2 shows how Levy walks and pheromone-based exploration imply more resistance to failures compared to random exploration and how a fast data synchronization implies more resistance to failures.

### Conclusions and Future Work

In this paper we proposed a solution for obtaining global information via a set of unreliable termites (agents), which explore, get local data and share collected information. The communication mechanism allows neighbors to interchange information, enabling agents to acquire global data as the result of local interactions and cooperation.

It is possible to see how pheromone exploration and Levy walks improve world exploration compared to random exploration. Figure 5 shows how a termite explores the world (a trace is added on visited locations). As expected, pheromone exploration tends to avoid path repetition during exploration vs random exploration where a termite can explore a determined location more than once. Additionally, Levy walks cover an area in a better way than random by maintaining the same direction for some time.

Changes in the status of a termite between SEEKER and CARRIER, restarts the amount of local pheromone that a termite has, reinforcing the trace and rewarding communication. Future work is intended to test other methods to relate local information with the local update rule of pheromone exploration, tend to avoid path repetition during exploration vs random exploration where a termite can explore a determined location more than once. Additionally, Levy walks cover an area in a better way than random by maintaining the same direction for some time.

![Table 2: Summary of average of information collected and number of agents with complete information (pop = 10, 30, 50, width = 50, height = 50, maxiter = 3000).](attachment:image2.png)

![Table 3: Summary of averages of the number of rounds required for the best agents to collect all information (pop = 10, 30, 50, width = 50, height = 50, maxiter = 3000).](attachment:image3.png)

If the same world size is maintained and the population size is increased, a greater amount of information is obtained and a reduction in the average rounds necessary for gathering the whole world information is achieved. Possible next steps include studying the influence of each of these algorithms in terms of number of messages and local information obtained by each termite maintaining the same density of agents and performing experiments with different world sizes.

Acknowledgements

This work was supported by "Programa de becas para estudiantes sobresalientes de posgrado", the Alife research group - Universidad Nacional de Colombia, and "Convocatoria del Programa Nacional de Proyectos para el Fortalecimiento de la Investigación, la Creación y la Innovación en Posgrados de La Universidad Nacional De Colombia 2013-2015" code 23418.

References


