Music-Aware Artificial Fireflies to Sensor Relocation by a Robot Team

Rafael Falcon, *Graduate Student Member, IEEE*, Xu Li,
Amiya Nayak, *Senior Member, IEEE* and Ivan Stojmenovic, *Fellow, IEEE*

***Abstract*—Mobile robots can nowadays assist wireless sensor networks (WSNs) in many risky scenarios that unexpectedly arise during their operational lifetime. We focus on an emerging kind of cooperative networking system in which a small team of robotic agents lies at a base station. Their mission is to service an already-deployed WSN by periodically replacing all damaged sensors in the field with passive, spare ones so as to preserve the existing network coverage. This novel application scenario is here baptized as "*multiple-carrier coverage repair*" (MC^{2}R) and modeled as a new generalization of the vehicle routing problem. A hybrid metaheuristic algorithm is put forward to derive nearly-optimal sensor replacement trajectories for the robotic fleet in a short running time. The composite scheme relies on a swarm of artificial fireflies in which each individual follows the exploratory principles featured by Harmony Search. Infeasible candidate solutions are gradually driven into feasibility under the influence of a weak Pareto dominance relationship. A repair heuristic is finally applied to yield a full-blown solution. To the best of our knowledge, our scheme is the first one in literature that tackles MC^{2}R instances. Empirical results indicate that promising solutions can be achieved in a limited time span.**

***Index Terms*—robot-assisted wireless sensor networks; sensor relocation; vehicle routing problem; firefly optimization; harmony search; hybrid metaheuristics**

I. Introduction

A *wireless sensor network* (WSN) [1] is a collection of autonomous sensing nodes that communicate via wireless links. They are deployed in indoor and outdoor scenarios to monitor the region in an unattended fashion and report their measurements to a central location. Yet in spite of their large chain of successful applications in dissimilar domains, WSNs are often unable to surmount many operational challenges that unexpectedly arise during their lifetime such as resource depletion (energy, memory, processing), connectivity disruption, hardware/software faults, malicious attacks or harsh environmental conditions.

Fortunately, the latest advances in multi-robot systems have made possible for a WSN to be assisted by robotic agents. The seamless integration of robotic and sensory devices has been recently coined as a *wireless sensor and robot network* (WSRN) [2]. When the robotic nodes are mainly aimed at optimizing the performance of an autonomous WSN, we refer to the ensemble as a "*robot-assisted wireless sensor network*" (RA-WSN). A RA-WSN [3] usually involves resource-rich, mobile robots tending resource-constrained, stationary sensors. We focus on a kind of robotic actuators that are mobile and able to carry a limited number of sensors as payload.

A novel collaboration framework in which carrier robots contribute to endow the WSN with fault-reactive capabilities has been recently identified in [4]. It is assumed that the network has been deployed somehow and one or more robots are located at a *base station*, where data from every single sensing unit are periodically received. Because of the abundance of cheap sensors scattered, not all of them are actually needed to provide area coverage. Hence, they may follow any distributed scheduling algorithm (e.g. [5]) to decide which sensors will go into sleep mode (*passive units*) and which will monitor the region (*active units*).

Since the quality of the network coverage will be eventually degraded due to failures of the active units, the mission of the robotic team is to collect passive nodes all over the field and drop them at the positions of the faulty sensors. A corporate *route plan* (i.e. set of individual routes departing from and ending at the base station) for robots to follow must be derived. The goal is to minimize the cost of the coverage repair operation (in terms of e.g. distance traveled, replacement latency, etc.), while ensuring that the system quickly reacts to sensor faults, i.e. time to find a corporate solution is usually short, for a replacement action must be taken swiftly.

The previous problem is novel in literature. It was baptized as *carrier-based coverage repair* (CBCR) and cast into the combinatorial optimization world. In [4], the single-robot case was tackled via an ant colony algorithm [6] that yielded good-quality solutions in a limited time span. However, no light was shed on how to solve CBCR when a robot team is available. We address such an important scenario in this report and refer to it as *multiple-carrier coverage repair* (MC^{2}R) problem.

A simple way to tackle MC^{2}R instances is by assigning to each robot a disjoint subset of the damaged nodes (could be empty) that it must replace with a disjoint set of passive sensors. Then, the Nearest Neighbor (NN) heuristic could be applied to shape each individual route, as shown in Fig. 1. NN quickly computes a sub-optimal solution by selecting, at each time step during the route construction process, the nearest feasible node to be added to the route. A new route is created according to a domain-specific criterion, e.g. whenever the total distance walked by the robot exceeds a threshold. Yet because a short delay after periodical data reporting from

R. Falcon, A. Nayak and I. Stojmenovic are with the School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Ave., Ottawa Ontario, K1N 6N5 Canada {rfalc032,anayak,ivan}@site.uottawa.ca

X. Li is with the Institut National de Recherche en Informatique et en Automatique (INRIA), Lille - Nord Europe, France xu.li@inria.fr

sensors and before robotic actuation upon the environment is often tolerated in many real-world RA-WSN applications, more advanced search methods could be applied in order to improve the quality of the robot team trajectories for sensor relocation (see Fig. 2).

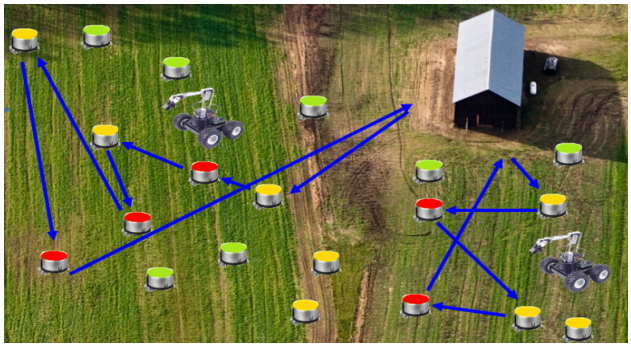

Fig. 1. Robots replacing damaged sensors. Trajectories computed via the Nearest Neighbor heuristic. Active, passive and damaged sensors are represented by green, yellow and red nodes, respectively.

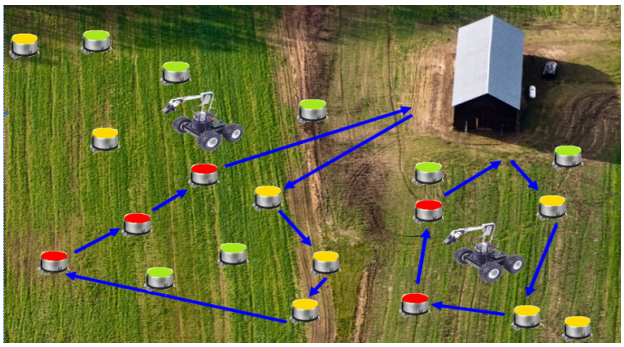

Fig. 2. Robots replacing damaged sensors. Trajectories computed via the proposed hybrid metaheuristic. Active, passive and damaged sensors are represented by green, yellow and red nodes, respectively.

We model MC^{2}R as a special case of a novel generalization of the vehicle routing problem (VRP), here termed as *the one-commodity VRP with selective pickup and delivery* (1-VRP-SELPD). A hybrid metaheuristic algorithm is put forward to solve MC^{2}R instances. The composite scheme relies on a swarm of artificial fireflies in which each individual follows the exploratory principles featured by Harmony Search (HS). Infeasible route plans are gradually driven into feasibility under the influence of a weak Pareto dominance relationship. A repair heuristic is finally applied to yield a full-blown solution. To the best of our knowledge, our scheme is the first one in literature that tackles the MC^{2}R problem. Empirical results indicate that promising solutions can be achieved in a limited time span.

The remainder of the report is structured as follows. Sect. II briefly reviews some relevant studies. Sect. III elaborates on the mathematical formulation of the MC^{2}R problem. The building blocks of the proposed hybrid method are described in Sect. IV whereas Sect. V unfolds the music-aware firefly swarm. The empirical study and conclusions are outlined in Sect. VI and VII, respectively.

II. Relevant Work

A VRP-SELPD instance can be portrayed as a graph $G = (V, E)$. A graph vertex $v \in V$ corresponds to either a delivery or pickup customer. Let V_D and V_P denote the set of delivery and pickup nodes, respectively, with $V_D \cap V_P = \emptyset$. Then $V = V_D \cup V_P \cup \{v_1\}$, where v_1 is the base station. A graph edge $e_{ij} \in E$ stands for the travel time/distance between its two endpoints. Each delivery node demands a certain amount of some commodity, which can be supplied by the pickup nodes. A finite-size fleet of vehicles (each with identical cargo capacity) lies at a central depot, from which individual vehicle routes begin and end. The goal is to satisfy the demand of all delivery nodes by visiting as many pickup nodes as needed while minimizing the overall travel cost. A vertex, if visited, can only welcome one vehicle exactly once. A vehicle's capacity must not be violated along its route.

One may realize that MC^{2}R can be modeled as a special VRP-SELPD instance in which delivery customers are damaged sensors, pickup customers are passive sensors, vehicles are mobile robots, pickups outnumber deliveries and any demand/supply of the commodity (sensors) is exactly one unit.

Close to VRP-SELPD in literature are the pickup-and-delivery VRP (PDVRP) [7], the Team Orienteering Problem (TOP) [8][9], its capacity-aware version (CTOP) [10] and the Capacitated Profitable Tour Problem (CPTP) [10].

Like in PDVRP, delivery and pickup customers are unpaired and the latter may be used to meet the demand of the former. Yet in PDVRP all graph vertices are to be visited whereas a VRP-SELPD solution includes all delivery nodes plus any subset of pickups that meets the corporate demand.

TOP, CTOP and CPTP are similar to VRP-SELPD in that they may leave some graph nodes out of the set of vehicle routes. These problems are grouped under the label "*VRP with profits*" because a reward/profit is collected after visiting each node. TOP and CTOP aim at maximizing the total profit without violating each vehicle's maximum time constraint whereas CPTP strives to optimize the difference between maximum collected profit and total cost. However, these three problems are not concerned with the demand satisfaction of delivery nodes by using the amount of commodity provided by the pickup nodes. Moreover, VRP-SELPD does not restrict the travel time/distance of the vehicles.

III. The MC^{2}R Problem Formulation

In this section, MC^{2}R is formally modeled as a unitary 1-VRP-SELPD instance. Any MC^{2}R scenario can be represented by a tuple $T = \langle G, \vec{q}, R, Q_0, Q \rangle$. A robot team $R = \{r_1, r_2, ...r_m\}$ of exactly m agents, each with initial cargo Q_0 and capacity Q, has to pick up and deliver sensors in a WSN represented by a graph G and a demand vector $\vec{q}$.

The sensor network is described by a complete graph $G = (V, E)$ where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set and E is the edge set, with e_{ij} being the directed edge from

v_i to v_j. Each edge has an associated travel cost c_{ij}^k when traversed by robot $r_k \in R$, $k = 1..m$. Elements of V are either passive sensors (pickup customers) or damaged sensors (delivery customers). Each customer v_i has a unitary demand q_i (-1 for pickup customers and 1 for delivery customers) and $\vec{q} = (q_1, q_2, \ldots, q_n)$. Node v_1 represents the base station with demand $q_1 = 0$, V_D is the set of delivery customers and V_P is the set of pickup customers. The following statements hold.

$$V = \{v_1\} \cup V_D \cup V_P$$
$$V_D = \{v_i \in V : q_i = 1\}$$
$$V_P = \{v_i \in V : q_i = -1\}$$
$$V_D \cap V_P = \emptyset$$

A unique type of commodity (sensors) is to be transported by the robots from one place to another. For simplicity, we will assume that all robots have the same maximum cargo capacity Q and initial cargo Q_0 ($0 \leq Q_0 \leq Q$). Not all graph nodes have to be visited. To ensure the service of all damaged sensors, a delivery node is assigned a profit p_i of 1. The profit p_i of a pickup node v_i and the base station v_1 is set to zero.

The goal is to find a minimal-cost feasible route plan $\Phi_{min} = (\vec{\varphi}_1, \vec{\varphi}_2, \ldots, \vec{\varphi}_z), 1 \leq z \leq m$, where $\vec{\varphi}_i$ is an individual robot route. A route plan Φ is regarded as *feasible* if all its routes are disjoint (i.e. have no nodes in common), each individual route is a Hamiltonian cycle originated at the base station, all damaged nodes are corporately replaced with passive nodes, each robot returns empty to the base station (i.e. no extra passive node is collected) and the robot's capacity constraint is never violated along any route. The *cost* of a route plan is defined here as the total distance traveled by all robots, its *cardinality* stands for the number of routes it contains and its *length* refers to the total number of nodes in it, including the base station. More formally, for a given tuple T, the cost, cardinality and length of any feasible route plan Φ can be calculated as in (1), (2) and (3), respectively.

$$f(\Phi) = \sum_{\vec{\varphi} \in \Phi} \sum_{e_{ij} \in \vec{\varphi}} c_{ij} \tag{1}$$

$$|\Phi| = \text{number of routes in } \Phi \tag{2}$$

$$len(\Phi) = 2 \cdot |V_D| - Q_0 + |\Phi| \tag{3}$$

with c_{ij} being the Euclidean distance between v_i and v_j. Decision variables are defined as follows:

- x_{ij}^k = 1 if edge e_{ij} has been visited by robot k, else 0
- y_i^k = 1 if node v_i has been visited by robot k, else 0
- l_{ij}^k = load of the k-th robot traveling along edge e_{ij}

This leads to the following integer linear programming formulation:

$$Min \sum_{k=1}^{R} \sum_{e_{ij} \in E} c_{ij}^k \cdot x_{ij}^k$$

subject to

$$\sum_{k=1}^{m} \sum_{v_j \neq v_i} x_{ij}^k \leq 1 \qquad \forall v_i \neq v_1 \tag{4}$$
$$\sum_{k=1}^{m} \sum_{v_i \neq v_j} x_{ij}^k \leq 1 \qquad \forall v_j \neq v_1 \tag{5}$$
$$\sum_{v_j \in V} x_{1j}^k = 1 \qquad \forall r_k \in R \tag{6}$$
$$\sum_{v_i \in V} x_{i1}^k = 1 \qquad \forall r_k \in R \tag{7}$$
$$\sum_{v_i \in V} x_{ij}^k - \sum_{v_i \in V} x_{ji}^k = 0 \qquad \forall v_j \neq v_1, \forall r_k \in R \tag{8}$$
$$\sum_{v_j \neq v_1} l_{1j}^k = Q_0 \qquad \forall\ r_k \in R \tag{9}$$
$$\sum_{v_j \neq v_i} l_{ji}^k - \sum_{v_j \neq v_i} l_{ij}^k = q_i y_i^k \qquad \forall v_i \neq v_1, \forall r_k \in R \tag{10}$$
$$0 \leq l_{ij}^k \leq Q x_{ij}^k \qquad \forall e_{ij} \in E,\ \forall r_k \in R \tag{11}$$
$$\sum_{k=1}^{m} \sum_{v_i \in V} p_i y_i^k = p_{min} \tag{12}$$
$$\sum_{v_i \in V} p_i y_i^k > 0 \qquad \forall r_k \in R \tag{13}$$
$$\sum_{v_i \in V} l_{i1}^k = 0 \qquad \forall r_k \in R \tag{14}$$
$$l_{ij}^k \text{ is integer} \qquad \forall e_{ij} \in E, \forall r_k \in R \tag{15}$$
$$x_{ij}^k \in \{0, 1\} \qquad \forall e_{ij} \in E, \forall r_k \in R \tag{16}$$
$$y_i^k \in \{0, 1\} \qquad \forall v_i \in V, \forall r_k \in R \tag{17}$$

The objective function minimizes total costs, expressed in terms of total distance traveled by all robots to fix all damaged nodes. Constraints (4) and (5) ensure that each node except the base station is visited at most once. Equalities (6) and (7) guarantee that each robot departs from and returns to the base station. This does not imply that all robots will be used to construct the route plan, for a robot r_k may remain at the base station by solely choosing the edge $x_{11}^k = 1$. Preservation of the flow per individual route is achieved through (8). Notice that the presence of (8) renders either (4) or (5) redundant, but we have decided to include both for the sake of clarity. Each robot leaves v_1 with an initial load Q_0 as stated in (9). Expressions (10) and (11) represent a robot's loading constraints. The difference in load of a robot entering and leaving a node must equal the demand of that node. A robot can never carry more than its maximum capacity. Constraint (12) ensures that all damaged units are visited by the robotic fleet. The total profit collected in the route plan is set to a minimum value p_{min}, equal to the number of delivery nodes $|V_D|$. Since only delivery customers have a profit of 1, these nodes are automatically included in the solution. Each individual route must repair at least one damaged sensor, as constraint (13) indicates. Every robot must return empty

to the base station according to (14), which means that all passive sensors picked in the route have been dropped at the coordinates of the damaged nodes. Finally, constraints (15), (16) and (17) define the domain of the decision variables.

IV. Firefly Algorithm and Harmony Search

This section explains in short the fundamentals of the Firefly Algorithm (FA) and HS, the building blocks of the hybrid approach unrolled in Sect. V.

A. Firefly Algorithm

This metaheuristic method was enunciated in [11] after drawing inspiration from the phenomenon of bioluminescent communication featured by firefly insects in tropical countries. They produce short and rhythmic flashes to attract potential prey or other fireflies for mating purposes. The brighter a firefly flashes, the higher the chance that other fireflies will be drawn towards it.

In FA, each individual in a swarm S of size $M = |S|$ encodes a candidate solution to the optimization problem in the form of a position vector $\vec{X}_i = (X_{i1}, \ldots, X_{iN})$, $i = 1, \ldots, M$ in the N-dimensional search space. The *brightness* of any firefly $I_0(\vec{X}_i)$ is inversely proportional to its fitness value $f(\vec{X}_i)$ in the minimization case. A firefly $\vec{X}_i$ will feel drawn towards any other individual $\vec{X}_j$ brighter than itself with a strength represented by the function $\beta(\vec{X}_i, \vec{X}_j)$. The best firefly in the swarm $\vec{X}_g$ will perturb its position vector with a controlled amount of randomness so as to favor exploration.

Though initially formulated for numerical optimization problems, discrete FA implementations e.g. [12] have yielded encouraging results in the combinatorial optimization arena.

B. Harmony Search

HS is a simple yet powerful music-inspired metaheuristic scheme [13], since the improvisation process of any musician in his pursuit of harmony is analogous to the search process in the optimization realm.

A candidate solution (harmony) $\vec{X}_i$ is a collection of pitches $(X_{i1}, \ldots, X_{iN})$. To improvise a new pitch X_{ij}, the musician has three options: (1) to draw a good pitch from the Harmony Memory (HM), an elitist data structure containing HMS high-quality harmonies; (2) to slightly adjust the previously chosen pitch or (3) to come up with a brand new pitch.

Good past pitches are drawn from the HM with a probability HMAR (HM acceptance rate) and subsequently twisted with probability PAR (pitch adjustment rate). The third choice is done entirely at random. This pitch improvisation process is repeated until an entire harmony $\vec{X}_i$ is completed. It replaces the worst harmony in HM if $\vec{X}_i$ is of superior quality.

V. Harmony-Seeking Fireflies for MC^{2}R

Given the intrinsic complexity of solving a MC^{2}R instance and the usually short time allotted to come up with a feasible route plan of good quality (due to the tight responsiveness constraints imposed on the WSN), a harmony-seeking firefly algorithm (HSFA) is put forth. It can be envisioned as a firefly swarm in which each member follows the exploratory principles featured by HS. In fact, the HM resides collectively in the swarm (i.e. HMS = M) rather than in a separate data structure in memory. It stores the personal-best position of every firefly and is dynamically updated by the fireflies themselves as their personal bests improve over time. The HSFA pseudocode is outlined in Alg. 1.

TABLE I
MC^{2}R Problem Descriptors and HSFA Parameters

Symbol	Sampling Range	Description
Q	1 - 5	Maximum robot cargo capacity.
R	2 - 5	Number of robots in the team.
NS	30 - 500	Total number of sensor nodes.
DN	5% - 25%	Percentage of delivery nodes.
maxCPUTime	5 - 600 sec	Maximum time for computations.
M	20 - 60	Swarm size (number of fireflies).
HMAR	[0;1]	HM Acceptance Rate. Probability with which a firefly will draw past good solutions from the HM.
PAR	[0;1]	Pitch Adjustment Rate. Probability with which a firefly will perturb the solution drawn from the HM.
pElite	[0;1]	Probability with which a firefly will select its personal-best solution as informer for another firefly. Otherwise, it will select its current position as informer.
p_R	20% - 80%	Percentage of pickup nodes in a firefly's position that will be replaced with unvisited pickups.

Since browsing across the feasible solution space alone is time-consuming (owing to the number of capacity violations and wrongly allocated customer types that could emerge in individual routes), we conduct the search starting from very poor, infeasible solutions and gradually drive them into feasibility under the influence of a weak Pareto dominance relationship, as described in Sect. V-C. This phase consumes around 98% of the available CPU time (which usually varies between 5 $\sim$ 600 sec). The remaining 2% is set aside to construct a feasible route plan out of the best-performing infeasible solution found thus far. This is done by the repair heuristic in Sect. V-D, which can indeed strike to some extent on the overall quality of the best infeasible candidate in order to return a full-blown, feasible solution when the CPU time elapses. In practice, this degradation of the final solution's performance grows with the network size (number of WSN nodes) yet can be conveniently controlled by discarding steep quality-lowering repair steps.

A. Algorithm's Parameters

Table I displays the MC^{2}R problem descriptors and HSFA parameters. While the former (first five table entries) have their values dictated by WSRN operational constraints, proper values for the latter (remaining table entries) were learned after a comprehensive empirical analysis described in Sect. VI.

B. Solution Encoding

A *cyclic path representation* is used to encode the vehicle route plan in the firefly's position, e.g. $\vec{X}_i$ =

Algorithm 1 HSFA for MC^{2}R problem instances

Input: M, Q, R, maxCPUTime, HMAR, PAR, pElite, p_R
Output: Best-ever route plan found $\vec{X}_g$

```
1: initializeDataStructures( );
2: for i = 1 to M do                                   ▷ initialize swarm
3:     X_i ← generateNewSolution( );
4: end for
5: updatePickupUsage( );
6: while tic( ) ≤ 98% maxCPUTime do
7:     X_g ← EvaluateSwarmBrightness( );
8:     for i = 1 to M do
9:         X'_i ← X_i;
10:        if rand( ) ≤ HMAR then   ▷ X_i draws from HM
11:            follower ← false;
12:            for j = 1 to M do
13:                if I_0(X_j) ≻ I_0(X_i) then
14:                    follower ← true;
15:                    X*_j ← chooseInformer(pElite);
16:                    β ← attractiveness(I_0(X_i), I_0(X*_j));
17:                    X'_i ← copyFromInformer(X*_j, X'_i, β);
18:                    if rand( ) ≤ PAR then
19:                        X'_i ← perturbSolution(X'_i);
20:                    end if
21:                    updateHM(X'_i);
22:                end if
23:            end for                                  ▷ j loop
24:            if not follower then
25:                X'_i ← generateNewSolution( );
26:                updateHM(X'_i);
27:            end if
28:        else                          ▷ improvise a new firefly
29:            X'_i ← generateNewSolution( );
30:            updateHM(X'_i);
31:        end if
32:        X_i ← X'_i;
33:    end for                                          ▷ i loop
34:    updatePickupUsage( );
35: end while
36: X_g ← repairBestSolution(X_g);
37: return X_g;
```

$(4, 5, 1, 6, 2, 7, 1, 3)$ means that two routes, viz 1-6-2-7-1 and 1-3-4-5-1, originate and conclude at the base station v_1. The length (number of components) of firefly $\vec{X}_i$'s position will vary from one swarm member to another, being $2 \cdot |V_D| + |\mathcal{R}|_i$, where $|\mathcal{R}|_i$ is the number of routes encoded in $\vec{X}_i$ (i.e. number of 1's in the vector) and it ranges from 1 to R. Hence, we are in presence of a variable-length firefly swarm.

C. Navigation across the Infeasible Search Space

The brightness of a firefly $\vec{X}_i$ is defined as $I_0(\vec{X}_i) = (f_1(\vec{X}_i), f_2(\vec{X}_i), f_3(\vec{X}_i), f_4(\vec{X}_i))$, where $f_1(\cdot)$ is the number of robots in the route plan, $f_2(\cdot)$ the total distance traveled, $f_3(\cdot)$ the number of wrong nodes in the solution (includes missing deliveries and extra pickups) and $f_4(\cdot)$ the number of capacity violations. The first objective aims at minimizing the number of robots engaged in the current sensor relocation task. This is desirable since the "idle" robots can replenish their batteries at the base station and be dispatched in case of emergency to a conflictive region in the deployment field. Objective 2 is improved as shorter route plans are discovered, which means that the network repair latency is shortened too. The last two objectives capture the infeasibility permeating a route plan $\vec{X}_i$. As the algorithm progresses, the minimization of the four objectives is pursued.

It is said that firefly $\vec{X}_j$ is brighter than $\vec{X}_i$ and denoted by $I_0(\vec{X}_j) \succ I_0(\vec{X}_i)$ if $\vec{X}_j$ *weakly dominates* $\vec{X}_i$, i.e. at least one objective is improved, $\exists k \mid f_k(\vec{X}_j) < f_k(\vec{X}_i), k = 1..4$. If that happens, $\vec{X}_j$ becomes the attractor (or *informer*) for $\vec{X}_i$, which is referred to as the *follower* and will try to copy some good components from $\vec{X}_j$.

The procedure in line 1 computes the distance matrix $\mathcal{D}(\cdot, \cdot)$ between any pair of graph vertices and sets the pickup usage vector $\mathcal{P}(\cdot)$ to $\vec{0}$. The former will aid in computing the nearness of a pickup to the set of reported delivery nodes whereas the latter will play a pivotal role in selecting those pickup units that have not been quite represented in previous firefly swarms, thus fostering the exploration of neglected pickups and their inclusion in the group of route plans encoded by the swarm members at the current iteration.

The entire firefly swarm is initialized in lines 2–4 via generatenewSolution( ), which randomly decides on a number of routes $1 \leq r \leq R$, add r 1's and all delivery nodes and then shuffles the vector. The remaining components are filled with pickups by the roulette wheel selection rule shown in (18).

$$p_j = \frac{\varphi_j / (\mathcal{P}_j + 1)}{\sum_{k \in V_P} [\varphi_k / (\mathcal{P}_k + 1)]} \tag{18}$$

where p_j is the probability that the pickup node v_j will be chosen, $\varphi_j = 1 / \sum_{v_k \in V_D} \mathcal{D}(v_j, v_k)$ its "potential" and $\mathcal{P}_j$ its usage counter value. Lines 5 and 34 update the $\mathcal{P}$ vector by adding to the present counter for each v_j the number of times it appears in the encoding of all members of the current swarm. The idea behind the selection rule is to promote the inclusion of those pickups that have been least used in earlier firefly swarms and exhibit good potential, i.e. their geographical location lies fairly close to many delivery units.

Line 7 calculates the brightness of every individual and initializes/updates the best firefly $\vec{X}_g$ with an arbitrary non-dominated solution in the Pareto front. The tic( ) function checks the current clock time and rand( ) generates a number $\sim$ U(0, 1). In line 15, $\vec{X}_j$ will recommend its personal-best position (stored in the HM) as informer for $\vec{X}_i$ with probability pElite or its current encoding otherwise.

The strength of the attractiveness β in line 16 between follower $\vec{X}_i$ and its informer $\vec{X}^*_j$ is given by the average improvement (in %) of the informer over the follower across all improving objectives. The method in line 17 copies from

the informer to $\vec{X_i}'$ exactly $\beta\%$ consecutive vector components, starting at an arbitrary location and removing duplicated entries in the follower. This HM-borrowed subsolution is probabilistically perturbed in line 19 by applying three local search operators in the following order: (1) *Delivery Exchange*, which swaps two random delivery nodes lying in two arbitrarily selected routes; (2) *Pickup Exchange*, which does the same but with pickup nodes and (3) *Pickup Replacement*, which substitutes $\mathrm{p}_R\%$ of the pickups in the firefly encoding with more profitable, unvisited ones as per the rule in (18). Notice that the application of the first two operators may not necessarily improve the candidate solution $\vec{X_i'}$ yet the last operator only modifies it if a better candidate was obtained.

The updateHM( ) method replaces $\vec{X_i}$'s personal best in the HM with $\vec{X_i'}$ if $I_0(\vec{X_i'}) \succ I_0(\vec{X_i})$ and updates $\vec{X_g}$ as well. Finally, $\vec{X_g}$ will be turned into a feasible solution in line 36 should signs of infeasibility still persist as explained next.

D. Building a Feasible Final Route Plan

Our repair heuristic takes $\vec{X_g}$ and builds a feasible final solution if needed by inserting missing deliveries, removing extra pickups and rearranging nodes so as to get rid of any vehicle capacity violations. The first operation may involve multiple routes whereas the two latter may apply to each individual route.

1) *Insert missing deliveries*: This implies expelling some pickups from the route plan that could have been copied from diverse attractors and laying missing deliveries in their positions. For each missing delivery $\overline{v_d} \in V_D$, find the list of pickup locations L_P cross-route-wise where insertion of $\overline{v_d}$ does not increase $f_4(\cdot)$. Try all locations in L_P (or, if $|L_P| = \emptyset$) and retain the one that optimizes $f_2(\cdot)$. Place $\overline{v_d}$ there.
2) *Remove redundant pickups*: It must be ensured that, for each route $R \in \mathcal{R}$, where $\mathcal{R}$ is the set of routes encoded in $\vec{X_g}$, $|V_D|_R = |V_P|_R$, i.e. there are as many pickups in the route as deliveries will be visited. Therefore, each delivery $v_d \in R$ will choose the closest pickup $v_p \in R$ that is yet unselected. All remaining pickups in the route will be removed. This is applied to every route $R \in \mathcal{R}$ with disparate number of pickups and deliveries.
3) *Eliminate capacity violations*: For each route $R \in \mathcal{R}$ and starting from v_1, find the location y where the first capacity violation occurs. Then swap the node at y with a node at $z > y$ that removes the violation and optimizes $f_2(\cdot)$. Proceed from this point on doing the same for each detected capacity violation.

VI. Empirical Analysis

Since MC^{2}R is a brand new problem first solved by HSFA, there is no competing scheme that serves as benchmark. Hence, we will shed light on a few HSFA's algorithmic properties like its exploration ability and the extent of the final solution degradation brought about by the repair heuristic.

TABLE II
WSN Spatial Distributions

Distribution	Depot δ	Delivery Node d_i	Pickup Node p_i
1	$(0,0)$	U	U
2	$(0,0)$	U	$\mathcal{N}(d_i, \sigma^2)$
3	$(0,0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
4	$(0,0)$	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
5	U	U	U
6	U	U	$\mathcal{N}(d_i, \sigma^2)$
7	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(\delta, \sigma^2)$
8	U	$\mathcal{N}(\delta, \sigma^2)$	$\mathcal{N}(d_i, \sigma^2)$
9	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$	$\mathcal{N}(r, \sigma^2)$

A. Simulation Setup

All simulations were conducted in MATLAB R2009b on an Intel Core i7 CPU 860 @ 2.80 GHz with 6 GB of RAM under Windows 7 Home Premium with a 64-bit architecture. They relied on synthetic scenarios, each holding information about the *problem descriptors* (WSRN configuration) and the *algorithm's parameters*. Their values have been uniformly drawn from the sampling ranges portrayed in Table I.

Once the problem descriptors are known, sensors are spatially distributed in a virtual field of $1,000 \times 1,000$ units. First, the depot's location δ (where the robot team lies) is chosen either deterministically at the field center $(0,0)$, or randomly following a bivariate uniform distribution U, or stochastically following a bivariate normal distribution $\mathcal{N}(r, \sigma^2)$ with r being a uniform random 2D point. Second, the coordinates d_i of the delivery nodes are set either randomly following U, or stochastically centered around the depot after $\mathcal{N}(\delta, \sigma^2)$, or stochastically centered around the random point r following $\mathcal{N}(r, \sigma^2)$. Finally, the pickup nodes are positioned either randomly after U, or stochastically following $\mathcal{N}(d_i, \sigma^2)$ and surrounding the coordinates d_i of an arbitrary damaged sensor, or stochastically centered around the depot through $\mathcal{N}(\delta, \sigma^2)$, or stochastically centered around r via $\mathcal{N}(r, \sigma^2)$. The nine different spatial distributions are summarized in Table II.

For the simulations, 50 MC^{2}R data scenarios have been created by randomly drawing values for the problem descriptors and HSFA's parameters from the sampling ranges in Table I and choosing an arbitrary spatial distribution from Table II.

B. Simulation Results

The best route plan found by HSFA in 10 independent runs per data scenario is depicted in Table III as well as the final feasible solution returned by the repair heuristic.

Notice how the number of wrongly allocated nodes and capacity violations remain reasonably bounded within 50% of the total number of nodes in the WSN. This speaks about the ability of the weak Pareto dominance relationship to gradually drive infeasible solutions into the feasible route plan space, even in networks with hundreds of nodes. In nearly all cases, the number of robots engaged in a sensor replacement operation is inferior to the robotic fleet size, meaning that idle robots remain at the base station ready to respond to any emergency in the field.

TABLE III
BEST MC^{2}R SOLUTION COMPUTED BY HSFA

Problem Instance Descriptors						Best Infeasible Route Plan				Final Route Plan		Diff	NN Heuristic	Diff
ID	NS	DN	Q	R	CPU	f_1	f_2	f_3	f_4	f_1	f_2	(%)	f_2	(%)
1	31	2	5	4	548	2	1341.42	0	0	2	1341.42	0	1341.42	0
2	40	4	1	2	184	2	792.65	0	0	2	792.65	0	792.65	0
3	58	13	4	3	241	3	3380.42	10	16	3	3421	1.2	5165.71	51
4	71	7	5	5	328	3	1634.89	9	14	3	2092.66	28	2427.49	16
5	77	12	4	4	493	3	1336.22	16	35	3	1416.39	6	1430.55	1
6	78	8	4	2	349	2	3466.67	13	28	2	4125.34	19	4249.1	3
7	84	4	5	4	209	3	919.61	42	21	3	1351.83	47	1811.45	34
8	87	17	4	3	466	3	3088.89	8	5	3	3922.89	27	4275.95	9
9	97	14	3	3	176	2	4263.11	45	13	2	5073.1	19	6645.76	31
10	118	18	1	2	71	2	3116.68	11	8	2	4269.85	37	6233.98	46
11	120	18	4	3	477	3	4009.37	29	16	3	5933.87	48	4078.82	-45
12	133	29	4	4	111	3	2772.22	27	26	3	3659.33	32	4647.35	27
13	170	9	1	5	255	4	3369.25	23	35	4	4514.8	34	6997.94	55
14	172	40	4	2	198	2	5827.99	51	52	2	6818.75	17	10569.06	55
15	184	26	4	3	371	2	3346.23	56	19	2	4751.65	42	7412.57	56
16	185	44	1	2	103	2	7934.13	6	34	2	8013.47	1	12020.21	50
17	194	37	2	2	364	2	2604.33	14	52	2	3281.46	26	4692.49	43
18	200	20	2	5	508	2	4531.96	12	50	2	6208.79	37	7140.11	15
19	232	14	4	4	522	2	3587.32	36	9	2	5129.87	43	8207.79	60
20	232	42	4	2	330	2	3958.7	32	14	2	4117.05	4	5969.72	45
21	257	51	4	4	193	4	5404.42	54	43	4	7890.45	46	8048.26	2
22	268	16	4	5	305	4	1508.23	71	116	4	1945.62	29	2470.94	27
23	275	58	3	3	276	2	2828.43	38	120	2	3846.66	36	4192.86	9
24	314	35	3	2	119	2	3572.62	78	69	2	3965.61	11	6067.38	53
25	316	60	2	4	323	4	5719.65	2	36	4	8579.48	50	10295.38	20
26	328	43	4	2	506	2	2776.6	1	42	2	3637.35	31	4764.93	31
27	342	34	5	5	183	3	4179.05	61	56	3	5307.39	27	6687.31	26
28	343	41	4	3	302	2	3471.05	13	155	2	4199.97	21	3787.97	-11
29	349	77	4	4	276	3	8724.52	132	0	3	11516.37	32	11861.86	3
30	350	63	5	2	297	2	7718.44	144	30	2	8413.1	9	9422.67	12
31	352	46	1	2	119	2	5215.03	69	67	2	5945.13	14	7847.57	32
32	355	43	5	5	483	4	4277.62	106	0	4	4277.62	0	5090.37	19
33	361	51	1	3	529	3	3271.76	110	140	3	4416.88	35	6669.49	51
34	362	51	5	4	348	3	4538.56	34	82	3	4765.49	5	6338.1	33
35	371	45	4	2	133	2	6602.44	3	127	2	7130.64	8	9055.91	27
36	372	48	1	4	383	2	2949.46	76	179	2	3155.92	7	3503.07	11
37	385	69	4	3	131	3	5155.78	23	55	3	6444.73	25	8313.7	29
38	392	43	5	3	384	3	6730.79	86	48	3	8480.8	26	12042.74	42
39	402	24	5	3	68	3	4614.49	55	16	3	5721.97	24	8525.74	49
40	420	34	5	2	586	2	3006.05	141	36	2	3066.17	2	3188.82	4
41	438	57	1	4	388	3	6988.69	183	25	3	9155.18	31	8521.39	-7
42	438	96	4	3	77	2	5925.91	202	35	2	8237.01	39	9802.04	19
43	440	110	3	5	292	3	11215.45	146	81	3	12224.84	9	17603.77	44
44	441	106	1	4	49	3	11449.52	66	154	3	15685.84	37	17568.14	12
45	446	36	3	3	169	3	2893.97	142	115	3	3646.4	26	3682.86	1
46	458	64	3	2	151	2	7901.82	132	94	2	9482.18	20	14507.74	53
47	460	87	5	5	342	5	7400.6	18	204	5	8288.67	12	9697.74	17
48	470	71	5	2	520	2	3746.04	183	207	2	5019.69	34	7479.34	49
49	481	63	1	4	356	2	7663.47	215	24	2	8506.45	11	9357.1	10
50	482	58	3	3	50	3	3524.6	230	165	3	4370.5	24	5769.06	32

Another encouraging observation is the fact that the repair heuristic only degrades the best infeasible solution by 26.36% on average in terms of the total distance traveled (objective 2). This means that the three-stage procedure to fix the best infeasible candidate solution is able to significantly attenuate the effect of the node insertion/removal operations.

The last two columns of Table III contrast the final route plans yielded by HSFA with those obtained after applying the NN heuristic. To ensure a fair comparison baseline with HSFA, the number of routes in NN equals the number of robots allocated by HSFA in the final route plan. Through statistical validation we can claim that HSFA is able to find significantly better solutions in terms of f_2 in 90% of the data scenarios under consideration, as corroborated via a t-test conducted at 5% significance level. This justifies the time spent in the calculation of the sensor replacement trajectories for the mobile robots, as the network repair latency and robot energy consumption are substantially minimized when HSFA's repair schedule is followed as opposed to the route plans derived by NN.

VII. CONCLUSIONS AND FUTURE WORK

We have targeted MC^{2}R, a novel problem of practical relevance for robot-assisted wireless sensor networks. Through the collective and periodic action of the robotic team, the WSN can self-heal as soon as faulty nodes stem during its operational period, thus preserving the desired coverage.

A new generalization of the vehicle routing problem, here coined as VRP-SELPD, has been formulated and MC^{2}R modeled as a special case of a VRP-SELPD scenario. Its solution was subsequently sought via HSFA, a metaheuristic algorithm leaning upon the hybridization of artificial fireflies and harmony search. This is the first scheme in literature that tackles MC^{2}R scenarios. The conducted empirical analysis indicates that promising solutions can be achieved in a limited time span by the proposed hybrid method.

Our ongoing research efforts are directed towards the estimation of HSFA's parametric sensitivity. In this sense, Reactive Search Optimization [14] techniques are under consideration to shield the algorithm as much as possible against entangled, local-minima-plagued fitness landscapes. A thorough algorithmic characterization per WSN spatial distribution is another short-term goal of practical relevance.

REFERENCES

[1] K. Sohraby, D. Minoli, and T. F. Znati, *Wireless Sensor Networks: Technologies, Protocols and Applications*. Wiley Interscience, 2007.

[2] A. Nayak and I. Stojmenovic, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*. John Wiley & Sons, 2010.

[3] R. Falcon, A. Nayak, and I. Stojmenovic, "Robot-Assisted Wireless Sensor Networks: Recent Applications and Future Challenges," in *Mobile Ad Hoc Networking: the Cutting Edge Directions*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. Wiley, 2012, to appear.

[4] R. Falcon, X. Li, A. Nayak, and I. Stojmenovic, "The One-Commodity Traveling Salesman Problem with Selective Pickup and Delivery: an Ant Colony Approach," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010, pp. 4326–4333.

[5] A. Gallais, J. Carle, D. Simplot-Ryl, and I. Stojmenovic, "Localized Sensor Area Coverage with Low Communication Overhead," *IEEE Transactions on Mobile Computing*, vol. 7, no. 5, pp. 661–672, 2008.

[6] T. Stützle and H. H. Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems*, vol. 16, pp. 889–914, June 2000.

[7] M. Dror, D. Fortin, and C. Roucairol, "Redistribution of Self-Service Electric Cars: a Case of Pickup and Delivery," INRIA-Rocquencourt, Tech. Rep. RR-3543, 1998.

[8] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The Team Orienteering Problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464 – 474, 1996.

[9] C. Archetti, A. Hertz, and M. G. Speranza, "Metaheuristics for the Team Orienteering Problem," *Journal of Heuristics*, vol. 13, pp. 49–76, February 2007.

[10] C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza, "The Capacitated Team Orienteering and Profitable Tour Problems," *Journal of the Operational Research Society*, vol. 60, no. 6, pp. 831–842, 2009.

[11] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization," in *Proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications (SAGA), LNCS 5792*, Sapporo, Japan, 2009, pp. 169–178.

[12] R. Falcon, M. Almeida, and A. Nayak, "Fault Identification with Binary Adaptive Fireflies in Parallel and Distributed Systems," in *2011 IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, Louisiana, June 2011, pp. 1359–1366.

[13] Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, 1st ed. Springer, 2009.

[14] R. Battiti, M. Brunato, and F. Mascia, *Reactive Search and Intelligent Optimization*, ser. Operations Research/Computer Science Interfaces. Springer Verlag, 2008, vol. 45.