This regular paper was presented as part of the main technical program at IFIP WMNC'2013

A Bird's Eye View on Reinforcement Learning Approaches for Power Management in WSNs

Luigi Rucco, Andrea Bonarini, Carlo Brandolese and William Fornaciari
Politecnico di Milano - DEI
Piazza L. da Vinci, 32 - 20133 Milano, Italy
Email: {rucco, bonarini, brandole, fornacia}@elet.polimi.it

***Abstract*—This paper presents a survey on the adoption of Reinforcement Learning (RL) approaches for power management in Wireless Sensor Networks (WSNs). The survey has been carried out after a review expressly focused on the most relevant and the most recent contributions for the topic. Moreover, the analysis encompassed proposals at every methodological level, from dynamic power management to adaptive autonomous middleware, from self learning scheduling to energy efficient routing protocols.**

I. Introduction

Wireless Sensor Networks are a well-known class of distributed embedded systems, composed by small autonomous and resource-constrained devices, called sensor nodes, used to monitor a wide range of phenomena and to report sensed data to a base station. Power consumption is a critical issue for these systems, since nodes are usually battery-powered with a limited amount of available energy and their lifetime maximization is a foremost requirement. Node can also be possibly equipped with energy harvesting devices (e.g. a micro solar panel), in such cases energy neutrality is the main goal, meaning that the consumed and harvested energy should be balanced as much as possible by opportunely adapting the system operation. Moreover, since data flow from leaf nodes to the base station at the top of the routing hierarchy, the synchronization and coordination among the nodes is a crucial prerequisite for the energy efficiency of the whole network.

Many solutions have been devised to tackle the power management issue in WSNs and we refer to [1] for an overview on the various approaches.

Although an extensive and accurate review of the literature on computational intelligence and RL models for WSNs may be found in previous works [11][25], this paper focuses on power management, deepening the aspects related to energy consumption minimization of the most relevant proposals.

In special regards to power management, which represents a broad vein of research in the WSN field, the introduction of RL-based approaches appears particularly promising to dynamically adapt the operation of the network to run-time variations in the topology and energy distribution of the networks. A structural non-stationarity, in fact, is an intrinsic characteristic of this class of networks, because of the occurrence of disturbing events like node losses (or new nodes introduction), radio interferences, natural obstacles in the communication channel, non-uniform energy depletion, malicious attacks and many others. The possibility to react to such events by dynamically reconfiguring the power management operation is a key success factor enabled by reinforcement learning models. Possible drawbacks, on the other hand, can be represented by the computational and memory burdens, so that each proposal tries to tackle this issue through opportune optimizations.

The paper is organized as follows. Section II provides a classification of the contributions, which are then detailed in the following sections. In particular, Section III presents approaches related to dynamic power management, Section IV describes RL-based adaptive autonomous middleware, Section V shows the proposals regarding energy-efficient scheduling and Section VI deals with power-aware routing protocols. In the end, Section VII reports some final considerations.

II. Classification

Reinforcement learning has a strong and solid background in the field of artificial intelligence and machine learning in particular [22]. The main idea underlying RL models is that an agent can learn the most profitable behavior through trial-and-error interactions with the external environment. In particular, the autonomous agent senses the environment or the phenomenon of interest through dedicated interfaces and forms a representation of the current state. Then it takes an action that makes it to pass to a new state. In consequence of each state transition, an opportune reward –in general a scalar value– is assigned to the agent according to a policy specifically defined to encourage the right behavior in reference to the considered environment. The agent learns to take the right actions run-after-run, trying to maximize the sum of the obtained rewards over time. Many RL techniques have been proposed to tackle general problems and we refer to [9] for an overview on the most common and widely-used models. As said, here we will focus on reinforcement learning for power management in wireless sensor networks, a field characterized by very peculiar issues at application, hardware and network level. In this section we propose a classification of the most relevant approaches to the power management, while the specific characteristics of each class of approaches will be discussed in the following sections.

978-1-4673-5616-9/13/$31.00

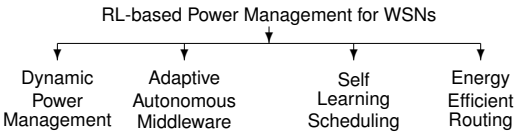

Fig. 1. Classication of RL-based Power Management Approaches

After an accurate literature analysis, four main classes of Reinforcement Learning based approaches to the power management in WSNs have been identified, as reported in Figure 1.

Dynamic Power Management models aim at optimizing the duty cycle of the node to minimize energy consumption. Reinforcement learning has been used in WSNs with energy harvesting capability, to enforce energy neutrality between the harvested and the consumed energy, incorporating also Quality of Service metrics.

The Adaptive Autonomous Middleware tries to extend the adoption of reinforcement learning for a comprehensive and energy-efficient resource management both at local and network-level. A full-fledged middleware has been defined to provide support for the adaptive resource management.

Self-learning scheduling can act at two different levels: at a distributed level to optimize sleep/wake cycles of the nodes and at local-level for scheduling tasks execution. Distributed RL-based scheduling are used to determine the sleep/wake duty cycle of the nodes such that communication and application specific operations are globally guaranteed, while reducing as much as possible energy consumption on the single nodes and enforcing a graceful degradation of the remaining energy throughout the network. At local level, reinforcement learning is used to schedule the tasks execution in the most energy-efficient way and to preserve, at the same time, Quality of Service requirements.

Routing is by far the most studied layer in wireless sensor networks, with an amount of publications that represents a research niche on its own. In regards to power-management, some RL-based protocols have been defined aiming at the discovery of routing paths with minimum energy cost, while preserving –at a varying extent– functional requirements like delivery delay reduction, throughput maximization and transmissions success rate, among the others.

The growing interest for the adoption of RL models to dynamically tackle the complexity of power management in WSNs can also be observed in the trend of the literature, which, from initial contributions by one or at most few research groups on a specific issue, now covers the topic of power management at almost every architectural level of a WSN. The following sections provide a detailed description of each of the above presented classes of models and the related proposals.

III. Dynamic Power Management

Reinforcement learning has been used as a model for dynamic power management in energy harvesting wireless sensor networks [2] (then generalized for diverse application in embedded systems [12]). In this kind of networks, sensor nodes are provided with energy harvesting devices (e.g. micro solar panels, piezoelectric mechanisms, etc.) capable of retrieving renewable energy from environment. The nodes are also provided with a battery that stores the harvested energy and powers the system when needed. The goal is to optimize the duty cycle of the nodes in such a way that the balance between the consumed and harvested energy is preserved, obtaining energy neutrality in system operation. The state of the system is composed of three states $(S_D, S_H, S_B) \in \mathbf{S}$, where S_D is defined as a distance function obtained by subtracting the consumed energy from harvested energy in the i_{th} sensing period, S_H is the energy harvested at the i_{th} period, normalized on the total harvested energy, and S_B is the percentage of energy remained in the battery. An action is defined as a level of the system duty cycle, bounded by the minimum and the maximum value that the system can bear $A = a(i) \in [D_{min}, D_{max}]$. The reward is modeled as distance from energy neutrality, the higher the distance, the smaller is the reward $r_D = -|S_D(i)|/(e_{maxharvest} - e_{minharvest})$. Similarly it is possible to express reward as a function of the remaining battery, the higher remaining energy, the higher is the reward $r_D = -r_D(1 - 2e_b(i)/E_B)$, where $e_b(i)$ is the energy of the battery at the i_{th} period and E_B the battery capacity. The algorithm deployed on the nodes proceeds as usual in reinforcement learning: starts by initializing the state, then for each sensing period i chooses a duty cycle, adjusts the duty cycle, determines the reward and updates the Q-value according to the classical Q-learning formula:

$$\begin{aligned} Q(s_i, a_i) =& (1-\eta)Q(s_i, a_i) + \eta[r_{i+1} + \\ &+ \gamma\, max_{a_{i+1}} Q(s_{i+1}, a_{i+1})] \end{aligned} \tag{1}$$

where η and γ, both between 0 and 1, are respectively the *learning* and the *discount* rate. The algorithm restarts each day, since solar energy exposure can vary with weather conditions and, at a slower rate, with the seasons. The learning rate is uniformly decreased as the time passes during the day. Results show very good performance of the algorithm, especially in longer sustainable operations.

In [8], the authors propose an extended version of the work which is aware of the provided Quality of Service. In this variant, the state array is enriched with a component S_Q which accounts for the quality of service and is defined as $S_Q \in \{QoS_0, QoS_1, ..., QoS_n\}$, where QoS_i represents the transmission quality in the sensing period i. It should be noted that the authors entail that data with higher priority need to be delivered in a more reliable way and use the transmission quality as a proxy for this requirement. To obtain higher levels of QoS, higher levels of duty cycle are needed, so the actions must be oriented consequently. The rewards are positive if the chosen action satisfies the QoS level, i.e. $a(i) \geq QoS_i$, while,

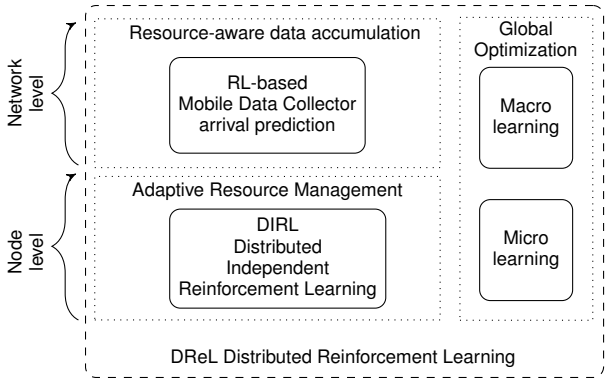

Fig. 2. Adaptive autonomous middleware layers

if no QoS are specified, the energy neutrality criterion applies. The overall algorithm has the same structure seen above and the same is also the Q-learning formula.

IV. Adaptive Autonomous Middleware

In [20] and [21] an entire RL-middleware for WSN management is presented, providing four main contributions: (1) adaptive resource management, (2) global optimization through multi-tier RL, (3) resource-aware data accumulation, (4) a complete middleware to support the previous features. The logical architecture of the whole framework is reported in Figure 2.

A. Adaptive resource management

The adaptive resource management of the WSN is based on a Distributed Independent Reinforcement Learning Approach (DIRL). Given a certain set of constraints specified at application level, this approach allows each sensor node to independently schedule its task to achieve the maximum possible reward. The reward is determined through a reward function that maps some optimization parameters relevant for a task (e.g. consumed energy, bandwidth, etc.) onto a real number. In this way, a utility value is associated to each task, on the basis of the resources usage. The RL system has been defined such that each sensor node corresponds to an agent of a Multi Agent Reinforcement Learning model, while an action is an application task to be scheduled. A state is a set of both application and system variables, describing relevant properties in reference to the specific application. System variables can be, for example, the number of neighboring nodes, residual energy, mobility, etc. Application variables can be the number of readings, the signal strength and so on. A weighted hamming distance method is used to classify group of similar states in order to reduce the complexity of the problem. Both exploration and exploitation strategy are adopted and the goal of the policy is to take the best choice in terms of the task to be scheduled at the current state. A classical Q-learning model is used for the system, under some limiting hypothesis: each node is independent and does not affect (nor it is affected by) neighbors, the system is single-threaded and the task allocation is done a-priori. Since each node acts selfishly, no communications are needed to determine the reward of an action. The results provided for the adaptive resource management framework, even though under the limiting hypothesis above reported, show a good degree of efficiency.

B. Global Optimization

The global optimization accounts for cooperative scenarios, in which the selfishness of the nodes may lead to an overall degradation of the system. A two-tier RL is proposed to this purpose: a micro-learning tier where individual nodes self-schedule their tasks and a macro-learning tier where a set of closely connected nodes determine the reward to assign to the nodes of the set in order to align them toward the system-wide application goal. The bottom-up nature of the approach also allows for a more pro-active and real-time adaptive behavior of the single nodes, reducing the communication overhead and avoiding the need for centralized processing. The global optimization middleware structure has been inspired by COllective INtelligence (COIN) [24] principles for designing both the global utility function and the the individual nodes, so that they can align to the global goal.

C. Resource-aware data accumulation

A resource-aware data accumulation framework is also proposed for sparse WSNs with a Mobile Data Collector (MDC), being the last a system (autonomous or human-driven) that periodically passes through the various zones covered by the network to gather data from local nodes. In particular, RL is used by each node to learn the pattern of arrival of the MDC, consequently adapting the duty cycle.

D. Integration within a common middleware

The three components, above described, are supported by a common middleware referred to as Distributed Reinforcement Learning (DReL). This middleware provides also an interface to design optimized applications and data, to support their dissemination on target nodes, as well as the calculation and the distribution of rewards.

V. Self-learning scheduling

The optimization of the duty cycle to reduce energy consumption is the main objective of the self-learning scheduling policies, which can act at various levels of a wireless sensor network architecture. A first distinction can be made between local and global policies: in the first case the goal is to determine the best sleep-wake cycle on the single node, while global approaches aim at deciding which nodes should stay active and which can sleep at any given time, without loss of quality in data transmission. Other proposals focus on the MAC layer, for orchestrating the duty cycle according to the traffic load. Finally, at application level, cooperative models can be implemented for the optimal scheduling of the tasks on the nodes.

A. Local self-learning scheduling

The approach presented in [18] integrates both a packet-transmission scheduling and a sleep scheduling, the first determined through a Q-learning model and the second calculated from the transmission parameter. Data received and transmitted by nodes are stored in a FIFO queue and two scheduling parameters are set: a sleep parameter p_s and a transmission parameter p_t. Each node decides whether to send a packet from the queue by generating a random number in the interval [0,1] and comparing it with p_t: if the generated number is smaller, then the packet is sent. On the other hand, if the generated number is greater than p_t and smaller than $p_t + p_s$ the node goes to sleep, otherwise it remains idle. A Q-learning module is adopted and the overall algorithm proceeds as follows: the node determines the current state and takes an action according to an $\epsilon - greedy$ policy that selects with probability $(1 - \epsilon)$ the highest-valued action and with probability ϵ a new action, so enabling both exploitation and exploration to a certain extent; then a data delivery (DD) phase takes place, during which the node sends or receives packets and then determines if to sleep or to transmit according to the transmission and sleep parameters. After that, there is a scheduling update phase, where the node checks the length of the packets queue, computes the reward and updates the Q-value. The whole algorithm is then repeated for a given number of episodes. Provided that the reduction of energy consumption is the objective of the model, a state is defined in terms of length of the queue of packets, since in the proposed model energy consumption is mostly affected by packet transmissions. In particular, there are three states: $S = \{0, 1, 2\}$, being 0 the state in which the queue is empty, 1 representing the queue when decreasing, while 2 when increasing. The transmission parameter p_t is augmented passing from the state 0 to the state 2. To generate a continuous action the authors partitioned the interval of p_t into N subinterval, obtaining N+1 endpoints each characterized by a discrete action and a Q-value. An action for a sub-interval i, referred to as a_s^i, is a function of the discrete action of the previous endpoint d_i and the next endpoint d_{i+1} weighted by their Q-values, such that:

$$a_s^i = \frac{d_i Q_s^i + d_{i+1} Q_s^{i+1}}{Q_s^i + Q_s^{i+1}}. \tag{2}$$

The Q-value q_s^i for the subinterval related to the action a_s^i is then determined as:

$$q_s^i = Q_s^i + \frac{Q_s^{i+1} - Q_s^i}{d_{i+1} - d_i}(a_s^i - d_i). \tag{3}$$

The reward is computed as $r_t = \lambda e_t + \eta(n_{t-1} - n_t) + \theta$ after the DD phase and takes into account three main contributions: energy consumption, queue length and the baseline constant value. Note that e_t is the energy consumed in the DD phase, n_{t-1} and n_t are the queue length at $t - 1$ and t, while λ, η and θ are weight factors. If $s = s_t$ and $a = a_t$ the Q-learning function has the classical formulation:

$$\begin{aligned} q_t(s,a) =& (1-\alpha) q_{t-1}(s_t, a_t) + \\ &+ \alpha(r_{i,t} + \gamma \, max_{a_t \in A} q_{t-1}(s'_t, a'_t)) \end{aligned} \tag{4}$$

while it is equal to $q_{t-1}(s, a)$ otherwise.

After having computed the discrete Q-value for the subinterval, the Q-value of neighboring endpoints can be computed as

$$Q(s, d_i) = \frac{d_{i+1} - a_t}{d_{i+1} - d_i} q_t(s, a), \; left \tag{5}$$

$$Q(s, d_{i+1}) = \frac{a_t - d_i}{d_{i+1} - d_i} q_t(s, a), \; right. \tag{6}$$

The other subinterval actions and Q-values can be computed in cascade as described for a_s^i and q_s^i. These values are used by the $\epsilon-greedy$ policy to select the action in the next DD period. Results obtained by the authors on simulation at MAC level, also in comparison with the SMAC protocol, demonstrated good performance.

B. Global self-learning scheduling

The previous scheduling acts at local level on the single node to determine its duty cycle. In WSNs the scheduling problem often assumes distributed nature with the goal of determining which nodes must be active and which other can sleep [19], without losses (or at least minimizing losses) in data forwarding quality. A self-organizing wakeup distributed scheduling of that kind is presented in [17], [16], [15]. The proposed RL model is mounted on top of a simple MAC protocol, in which time is divided in frames, i.e. discrete time units. A standard duty cycle is defined such that each node is awake for a fixed number of slots per each period: the duration of the awake-period is application dependent and is specified by the user, while the position of this period in a frame can be opportunely chosen. The objective of the RL model is, in the end, to schedule the awake period in a way that maximizes both the throughput and the energy efficiency: this entails for a node to be active when parents and children nodes are awake, thus synchronizing along the data forwarding path, while remaining asleep when neighboring nodes –at the same routing level– are active, in order to avoid interference in data transmission. The assignment of the reward depends on the correct transmission of the packets to the destination nodes: when a destination node receives a packet it sends back an ACK message to the source node and a reward is hence assigned to the source node. The approach claims to work on any multi-hop routing protocol, provided that through the RL algorithm coalitions are formed across different hops in the routing path. A coalition vertically extends over multiple hop levels and only one node per each hop is active, while other nodes at the same level are asleep to avoid interferences. The formation of coalitions for different topologies is demonstrated by the authors. A stateless Q-learning model with implicit exploration strategy is used. The model is stateless since the size of a frame is fixed and unchanged throughout the lifetime of the network and the length of the active period in a frame is fixed as well, been it decided by the user and lasting for a certain number of time slots. The model components are defined as follows.

Actions. Since both the frame and the awake-period lengths are fixed, an action just consists of choosing where to place the active period in a frame. The action space complexity depends on the number of time slots in frame: the higher it is, the higher is the action space complexity with the advantage, however, of a greater optimization flexibility. Dividing a frame in the right number of time slots, so, represents itself an optimization problem which is not explicitly faced by the authors. To each slot a Q-value is assigned: this Q-value says how much it is convenient to stay awake in the related time slot, given the past observations. If a node decides to stay awake in a given time slot, then the related Q-value is updated at the end of the frame depending on what happened during the active period, e.g. a communication event occurred or no events occurred.

Rewards. In the specific implementation the reward is assigned for each successful transmission (both toward the parent or from the children), so a 1 is assigned if an ACK is received from a parent (or sent to a child), 0 otherwise. Moreover also packet overhearing (unwilling/unuseful radio listening of neighbors transmissions) is modeled, by assigning a reward of 0. Note that, however, the framework is theoretically more general and is prone to be computed with metrics that can also be different from the ACK messages, provided that these metrics correctly account for the quality of the communication.

Q-learning update. The Q-value of each slot is firstly randomly initialized with a value in the [0,1] interval. Then the Q-value of each time slot s for the agent i is updated according to the classical formula:

$$Q_s^i = (1-\alpha) \cdot \overline{Q_s^i} + \alpha \cdot r_{s,e}^i \tag{7}$$

where $\overline{Q_s^i}$ is the previous Q-value for the slot s and $r_{s,e}^i$ the reward associated to the occurrence of the event e. Note that the Q-value of every slot in which an event occurred is updated: this differs from classical Q-learning methods where only the Q-value of the selected action is updated. Note that since the model is "'stateless"' a discount value γ has not been defined. The Q-value of a slot, as said, indicates how much it is convenient for a node to stay awake in that slot. Since the duty cycle of the system is defined by the user, who specifies the active period of a node as a series of D consecutive time slots, a node will choose the action $a_{s'}$ (s' is the slot where the active period begins) for which the sum of the Q-values for the D consecutive time slots is the highest possible, i.e. $s' = ArgMax_{s \in S} \sum_{j=0}^{D} Q_{s+j}^i$. As said, the Q-value of each slot for which an event occurs is updated.

Policy. By updating each slot singularly, and not the duty cycle active period D as a whole (nor just the initial active slot), an intrinsic exploration is enforced. In this way it is possible to dynamically tune the duty cycle, by choosing as active period the D consecutive slots that present the highest sum of the Q-values as the time runs out. The random initialization of the Q-values, moreover, entails that at the beginning the exploration is potentially extended to all the slots of the framework, then converging on the D consecutive time slots for which the sum of the Q-values becomes larger. The convergence speed depends on the duty cycle (i.e. the length of the slots sequence D) and on the learning rate.

Results obtained in simulations show how the model proposed by the authors is effective in choosing the active period within a frame, obtaining also synchronization across multiple hops and de-synchronization at the same hop level. This leads to efficient data forwarding and coalitions formation across the various levels of the routing path.

C. MAC-level scheduling

At MAC level another interesting approach is presented in [13] and [14], where both the frame active time and the duty cycle are dynamically tuned on the basis of the traffic load. Even in this case time is divided into frames, in turn divided in time slots[1]. The RL engine selects the active time slots in a frame, trying to maximize throughput ($\frac{payload\ bit}{sec}$) and energy efficiency ($\frac{effective\ transmission}{total\ active\ time}$). Given these two goals, the authors try to allocate active time slots as a function of the traffic. They model states as the number of packets (n_b) queued for transmission at the beginning of a frame, while the reserved active time t_r is the action. In modeling the reward function, the authors consider the number of packets successfully sent (n_s) and received (n_r). They also use the number of packets queued for transmission at the beginning of the next frame (n_b') as an indicator of the effectiveness of the chosen active slots (since if the active slots are chosen properly, the node is able to effectively transmit an higher number of packets to its neighbors). To prevent the early *sleeping problem*[2], a negative reward is assigned as the sum of failed transmissions from the neighbors to the node when it was sleeping. The number of failed transmissions is communicated by the neighbors to the node in a reserved field of the packet header. To account for the Quality of Service, the data contention window of the collision avoidance protocol is dimensioned according to three different level of priority (high, medium, low) assigned to the traffic. A classical Q-learning update formula is used (with an $\epsilon - greedy$ formula for both exploitation and exploration), where the learning factor is fixed ($\alpha = 0.1$) to account for the frequent variation of the traffic load. At the beginning of each frame the action, i.e. the active time slots, with the highest Q-value is chosen. Results show very good performance at the cost of a limited computational complexity.

D. Cooperative task-level scheduling

Another energy-efficient scheduling approach is presented in [10], where a cooperative Q-learning model is used to determine the best task to be schedule at each time step. A WSN is modeled as a multi-agent system, where the states are represented by system variables (e.g. object in the sensing range of a node, data to transmit, energy consumed for an action), while the actions coincide with tasks (e.g. sense,

[1]The dimension of a time slot in this model depends on the bandwidth and packet size.

[2]A node goes to sleep when some of its neighbors still have to transmit some packets to it.

transmit, receive). The initial Q-value is set to zero for each node and is updated according to the following function:

$$Q^i_{t+1}(s^i_t, a^i_t) = (1-\alpha)Q^i_t(s^i_t, a^i_t) + \\ + \alpha[r^i_{t+1}(s^i_{t+1}) + \gamma\Psi_{neigh}] \quad (8)$$

where,

$$\Psi_{Neigh} = \sum_{j \in Neigh(i)} f^i(j) V^j_t(s^j_{t+1}). \quad (9)$$

Note that V^j_t is the value function of the node j at time t, while the value for node i at time $t+1$ is expressed by:

$$V^i_{t+1}(S^i_t) = max_{a \in A^i} Q^i_{t+1}(S^i_t, a) \quad (10)$$

which is the maximum Q-value obtainable by executing the best action among the various possible. The term r^i_{t+1} is the reward[3] after having executed the action a at time t. Finally, γ is the discount factor and α the learning rate, while the weight for the values of the neighboring nodes $f^i(j)$ is set to 1 if no neighbors exist, otherwise it is determined as:

$$f^i(j) = \frac{1}{|Neigh(i)|}. \quad (11)$$

After having initialized to zero the Q-values, the algorithm enters in a endless loop performing the following actions: determines the current status by evaluating the system variables that have been chosen to describe a state, calculates the Q-value for all the actions in the action set, chooses the action with the maximum Q-value, sends this value to the neighbors, moves to the next state according to the transition caused by the chosen action. Results obtained in simulations and evaluations show good performance in terms of energy efficiency for the model, confirming the validity of RL-based approaches to tackle this class of problems.

VI. Energy efficient routing

The routing protocol represent the backbone tier of a WSN, since this is the architectural level that most affects the quality of service and the energy consumption of the entire network. Following we present noticeable models which pursue the goal of optimizing energy consumption at a decreasing level of generality, in particular: the discovery of the shortest routing path at minimum energy consumption in multi-sink multi-source WSNs, the restructuring of the RL model itself to reduce memory burdens in resource constrained scenarios and, finally, the adaptation of RL-based models to specific hardware and radio characteristics of the network, such as Ultra Wide Band communication.

[3]Note that the reward depends on the specific application. In the presented work, for example, the authors targeted an object tracking application, so a positive reward is given to a node if it performs the action "'sense"' when the object enters in the sensing range of the node.

A. General multi-source multi-sink routing models

In [6], [7], [5], [4] a Q-learning based routing approach is presented for multi-source multi-sink WSNs, which self-adapts to node failures and mobility. The goal is that of identifying the shortest routing path, with best energy efficiency. The data flow in the proposed protocol, named FROMS, has two directions: requests (also called sink announcements) flow from sink nodes to sensor nodes, while data flow from sensor nodes to the requiring sinks. Each node retains in its routing tables all the possible routes to a sink and not just the best path. Each node must choose the next hop toward the sink and this choice is made according to energy consumption for data transmission: a Q-value is associated to each next hop, representing the convenience of forwarding packets through that hop. This Q-value is updated several times during the node lifetime. Since multiple sinks are present, multiple paths are possible so the model has a non negligible complexity, especially in terms of memory [4]. Both exploitation (select the best known route) and exploration (i.e. try to find out other energy efficient routes) are enforced through a classical $\epsilon - greedy$ policy. The key feature of FROMS is a feedback mechanism from neighboring nodes: each node, in fact, extracts cost information from feedback packets overheard from neighbors. The cost of a root is so derived from the three steps seen above, namely sink announcements (which permits initial costs evaluation), route selection through the exploit/explore policy and cost estimation from feedbacks, which also represent the basis to calculate the reward. The Q-values are updated through a classical Q-learning formula, with a learning parameter γ taken near 1 in order to have fast estimation of the route cost. Being D the set of all the sinks in the network and $D_i \in D$ a subset of sinks, the cost function to reach the D_i destinations, choosing as next hop the neighbor n_i, is:

$$E_{hops}(n_i) = (\sum_{d \in D_i} hops^{n_i}_d) - 2(|D_i| - 1) \quad (12)$$

where $hops_d$ are the number of hops from the neighbor n_i to each sink $d \in D_i$. To calculate the cost to reach all sinks from all the possible neighbors k, the cost function changes as:

$$E_{hops}(route) = (\sum_{i=1}^{k} E_{hops}(n_i)) - (k-1). \quad (13)$$

The Q-value coincides with the hop-based cost function $Q_{hops}(route) = E_{hops}(route)$, but this does not take into account energy consumption. So an extension is proposed where a term $E_{battery}$ is used to consider the node with minimum remaining battery $E_{battery} = min_{n_i \in route}(battery)$.

Consequently, the Q-value function becomes:

$$Q_{comb}(route) = hcm(E_{battery}) \cdot E_{hops} \quad (14)$$

[4]The authors claim to have developed an heuristic to store in memory just the most promising routes. Moreover, the approach is also expensive from an energetic point of view at the moment of the routes initialization, but this operation happens only at the start or when a new sink joins the network.

where hcm is a commonly-used function[5] that estimates the hop count according to the remaining energy on the nodes. To account for this extension, the feedback packets from neighbors have been adapted to contain both the number of hops and the remaining battery.

B. RL routing model at minimum memory burden

In [23] a least squares reinforcement learning routing model, named AdaR, is presented. AdaR strongly reduces memory burdens and does not depend on initial settings, also reducing the energy consumption for initialization. Each node represents a state s, while forwarding a packet is an action a that causes a state transition from the node s to a neighbor s'. In such a model, the routing table of a node s simply coincides with the Q-values table. A more efficient Least Squares Policy Iteration (LSPI) is used, instead of classical Q-learning function, trying to reduce the convergence time and to avoid the issue of how to fine-tune the learning, the discount and eventually the exploration parameters. LSPI approximates the Q-value Q^π, for a given policy π, as a linear weighted combination of k basis functions, s.t.:

$$\widehat{Q^\pi}(s,a,w) = \sum_{i=1}^{k} \phi_i(s,a) w_i = \phi(s,a)^T w \qquad (15)$$

being $\phi_i(s,a)$ the ith basis function, expressing information about a state-action pair, and w_i its weight in the linear equation. The learning process is then expressed as a function of the matrix Φ of the basis functions, for each state-action pair and the rewards vector R. The weights are determined by resolving a linear system –the description of which we omit for the sake of conciseness– after some parameters have been learned by sampling from the environment. Though this approach can appear –and indeed is– quite cumbersome, it makes sense in the proposed work since authors suppose to have a multi-objectives optimization policy for which a linear combination of basis functions is expressive to represent the trade-off between these goals. Moreover, considering an origin state s and an action a that entails a transition to s', they use as components of the basis functions the following criteria: the difference $d(s,a)$ of the distances of s and s' from the base station (in terms of number of hops), the energy $e(s,a)$ remaining on s', the number of paths $c(s,a)$ to which s' belongs, the link reliability $l(s,a)$ between s and s'. Thus, the basis function for a node s taking a certain action a becomes

$$\phi(s,a) = \{d(s,a), e(s,a), c(s,a), l(s,a)\}. \qquad (16)$$

The algorithm has the following flow: each time a packet is forwarded, a tuple $< s, a, s', \phi(s,a) >$ is appended to the packet, so that when the packet arrives to the base station it is possible to determine the quality of a whole routing path and to assign a reward to each tuple (s,a,s') according to the estimated quality. After that, the weights w are calculated by the sink and broadcasted to the nodes, which can now choose the most convenient action by computing the Q-value with the received weights. The process goes on until a fixed point is reached, so that the policy is stable. Results show good performance if compared to basic Q-learning approaches, provided that the conditions exist for modeling the basis functions in the way described by the authors [6].

C. RL routing adaptation to network-specific characteristics

A power efficient routing protocol, specific for Ultra Wide Band (UWB) WSNs, is presented in [3]. A RL model is used to optimize both functional requirements, as delay and routing failure, and non-functional objectives as energy consumption and network lifetime, which are relevant in regards to the present survey. The reinforcement learning based geographic routing protocol (RLGR) targets more powerful nodes, often used as cluster heads, since UWB transceivers are not typically mounted on smaller nodes. Cluster heads are supposed to remain fixed in their position, otherwise the information of the location should be communicated to the neighbors by including it in periodic HELLO packets. Since geographic routing is enforced, a fundamental hypothesis is that the location of the sink, as well as of the neighbors, is known to each cluster head, thanks to the high radio range of the UWB. Cluster heads also store information about the remaining energy of the neighbors, being this information communicated through periodic HELLO packets.

Also this proposal uses a classical Q-learning model:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma\, max_{a'} Q(s',a') - Q(s,a)] \quad (17)$$

where, as usual, α is the learning rate, γ the discount factor, the current node is the state s and an action consists of selecting the neighbor a for which the Q-value is maximized[7]. The reward function considers the multi-objective nature of the problem, in particular:

$$r = \beta \cdot \frac{adv}{adv_{avg}} + (1-\beta)\frac{E_r}{E_l} \qquad (18)$$

where E_r and E_I are, respectively, the remaining energy and the initial energy on the neighbors. The term $adv = d(i,k) - d(n,k)$ represents the advance from node i to the sink k by choosing n as next hop, while $adv_{avg} = \sum_{j=1}^{m} \frac{|adv_j|}{m}$ is the average advance through all the neighbors. In such a way the node with more residual energy and higher relative advance is chosen, balancing the need for graceful energy degradation throughout the network and the reduction of delivery delay. The reward is set to a constant value R_C if the sink is the next hop[8]. To prevent the *void problem* of geographic routing, if the forwarding node cannot reach the sink, it drops the

[5]The acronym hcm stands for *hop-count multiplier*: this function increases the cost of routes in which the nodes have depleted their battery the most, so these routes become less appealing.

[6]It should be noted that the complexity entailed by such a model poses a serious trade-off between the modeling effort, required to formalize the problem, and the actual gain in terms of energy efficiency, considering also the tolerance in terms of flexibility as the considered parameters vary during the lifetime of the network.

[7]In this proposal the Q-value represents the total reward to send a packet from the current node s to the sink, through the neighbor a.

[8]It is supposed that the sink has unlimited energy.

packet and sends back a negative constant reward ($-R_D$) to the origin node, which in that way can choose another neighbor as next hop. Finally, another negative reward value ($-R_S$) is foreseen for those cases in which the node has a remaining energy under a given minimum threshold: by sending to its neighbors this negative reward, the node signals that its energy is going to be depleted so neighbors can update their Q-value consequently. The overall algorithm proceeds as follows. Each node initializes its Q-values table, which is indeed the routing table, with the initial localization and energy information of its neighbors collected through the periodic HELLO packets. When a node has to forward a packet, it chooses the neighbor closer to the sink with the highest Q-value[9] and with remaining energy above the lower threshold. If none of the neighbors closer to the sink has remaining energy exceeding the lower threshold, the algorithm resorts to the neighbor with the highest Q-value among those which are placed farther from the sink. If also at this step no neighbors are available, the packet is dropped, otherwise the packet is forwarded, the reward is determined as described above and the Q-values are updated. Then the algorithm starts again from the beginning. Simulation results are compared with classical GPRS protocol and show a noticeable improvement in overall performance.

VII. Conclusions

This paper presented a survey on RL-based approaches for power management in WSNs. Four classes of approaches have been identified and an extensive analysis of the literature per each class has been carried out. A major consideration can be made on the advantage of using RL in WSNs: these models turn out to be very handy and easy to design, as well as extremely promising to dynamically adapt and optmize both selfish and distributed operation of sensor nodes. Through opportune optimizations, moreover, the computational and memory burdens can be considerably contained, making these models very useful for resource-constrained WSN nodes.

References

[1] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Netw.*, 7(3):537–568, May 2009.

[2] R. Chaoming Hsu, C.-T. Liu, and W.-M. Lee. Reinforcement learning-based dynamic power management for energy harvesting wireless sensor network. In B.-C. Chien, T.-P. Hong, S.-M. Chen, and M. Ali, editors, *Next-Generation Applied Intelligence*, volume 5579 of *Lecture Notes in Computer Science*, pages 399–408. Springer Berlin Heidelberg, 2009.

[3] S. Dong, P. Agrawal, and K. Sivalingam. Reinforcement learning based geographic routing protocol for uwb wireless sensor network. In *Proceedings of IEEE GLOBECOM'07*, pages 652–656, 2007.

[4] A. Forster, A. Murphy, J. Schiller, and K. Terfloth. An efficient implementation of reinforcement learning based routing on real wsn hardware. In *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,*, pages 247 –252, oct. 2008.

[5] A. Förster and A. L. Murphy. Froms: Feedback routing for optimizing multiple sinks. In *Proc. of the 3rd Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[9]Note that an $\epsilon - greedy$ policy is implemented such that it is possible to choose with probability ϵ also a neighbor with a Q-value which is not the maximum possible, enforcing exploration along with exploitation.

[6] A. Förster and A. L. Murphy. Balancing Energy Expenditure in WSNs through Reinforcement Learning: A Study. In *Proceedings of the 1st International Workshop on Energy in Wireless Sensor Networks (WEWSN)*, Santorini Island, Greece, 2008.

[7] A. Förster and A. L. Murphy. Froms: A failure tolerant and mobility enabled multicast routing paradigm with reinforcement learning for wsns. *Ad Hoc Netw.*, 9(5):940–965, July 2011.

[8] R. Hsu, C.-T. Liu, K.-C. Wang, and W.-M. Lee. Qos-aware power management for energy harvesting wireless sensor network utilizing reinforcement learning. In *Computational Science and Engineering, 2009. CSE '09. International Conference on*, volume 2, pages 537 – 542, aug. 2009.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[10] M. Khan and B. Rinner. Resource coordination in wireless sensor networks by cooperative reinforcement learning. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 895 –900, march 2012.

[11] R. Kulkarni, A. Forster, and G. Venayagamoorthy. Computational intelligence in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 13(1):68 –96, quarter 2011.

[12] C.-T. Liu and R. C. Hsu. Adaptive power management based on reinforcement learning for embedded system. In *Proceedings of the 21st international conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence*, IEA/AIE '08, pages 513–522, Berlin, Heidelberg, 2008. Springer-Verlag.

[13] Z. Liu and I. Elhanany. Rl-mac: a qos-aware reinforcement learning based mac protocol for wireless sensor networks. In *Networking, Sensing and Control, 2006. ICNSC '06. Proceedings of the 2006 IEEE International Conference on*, pages 768 –773, 0-0 2006.

[14] Z. Liu and I. Elhanany. Rl-mac: a reinforcement learning based mac protocol for wireless sensor networks. *Int. J. Sen. Netw.*, 1(3/4):117–124, Jan. 2006.

[15] M. Mihaylov, Y.-A. L. Borgne, A. Now, and K. Tuyls. Self-organizing synchronicity and desynchronicity using reinforcement learning. In J. Filipe and A. L. N. Fred, editors, *ICAART (2)*, pages 94–103. SciTePress, 2011.

[16] M. Mihaylov, Y.-A. L. Borgne, K. Tuyls, and A. Nowé. Decentralised reinforcement learning for energy-efficient scheduling in wireless sensor networks. *Int. J. Commun. Netw. Distrib. Syst.*, 9(3/4):207–224, Aug. 2012.

[17] M. Mihaylov, Y.-A. Le Borgne, K. Tuyls, and A. Nowé. Reinforcement learning for self-organizing wake-up scheduling in wireless sensor networks. *Communications in Computer and Information Science*, 271:382–397, 2012.

[18] J. Niu. Self-learning scheduling approach for wireless sensor network. In *Future Computer and Communication (ICFCC), 2010 2nd International Conference on*, volume 3, pages V3–253 –V3–257, may 2010.

[19] C. Schurgers. Wakeup strategies in wireless sensor networks. In Y. Li, M. Thai, and W. Wu, editors, *Wireless Sensor Networks and Applications*, Signals and Communication Technology, pages 195–217. Springer US, 2008.

[20] K. Shah. *Reinforcement Learning Based Strategies for Adaptive Wireless Sensor Network Management*. ProQuest, UMI Dissertation Publishing, 2011.

[21] K. Shah and M. Kumar. Distributed independent reinforcement learning (dirl) approach to resource management in wireless sensor networks. In *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE Internatonal Conference on*, pages 1 –9, oct. 2007.

[22] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. A Bradford Book, Mar. 1998.

[23] P. Wang and T. Wang. Adaptive routing for sensor networks using reinforcement learning. In *Computer and Information Technology, 2006. CIT '06. The Sixth IEEE International Conference on*, page 219, sept. 2006.

[24] D. H. Wolpert and K. Tumer. Collective intelligence, data routing and braess' paradox. *J. Artif. Int. Res.*, 16(1):359–387, June 2002.

[25] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal. Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues. *J. Netw. Comput. Appl.*, 35(1):253–267, Jan. 2012.