Maximizing Number of Satisfiable Routing Requests in Static Ad Hoc Networks

Zane Sumpter[1], Lucas Burson[1], Bin Tang[2], Xiao Chen[3]
[1]Department of Electrical Engineering and Computer Science,
Wichita State University, Wichita, KS, USA
[2]Department of Computer Science, California State University, Dominguez Hills, CA, USA
[3]Department of Computer Science, Texas State University, San Marcos, TX, USA
Email: {zpsumpter, ljburson}@wichita.edu, btang@csudh.edu, xc10@txstate.edu

***Abstract*— We study an energy-efficient routing problem in static ad hoc networks. The problem, referred to as $maxR$, is to maximize the number of routing requests that can be satisfied in the network, under the constraint that each node has finite battery power. The online version of the problem, where the sequence of messages that has to be routed over the network is not known ahead of time, has been studied extensively. In this paper, we study the offline version of the problem where the sequence of requests is pre-known. As far as we know, the offline $maxR$ problem, its hardness and approximability have not been well studied. We show that after appropriate transformation, offline $maxR$ is equivalent to the well-known maximum disjoint path problem, which is NP-hard. We propose a greedy algorithm called GDP that has a constant approximation ratio to the optimal algorithm. GDP can be used as a benchmark to evaluate the performance of online algorithms as it is known that the best offline algorithm performs better than any online algorithm. We then put forward a new online algorithm called MECBE to solve the online $maxR$ problem. Simulation results show that GDP outperforms MECBE, which outperforms the state-of-the-art online algorithm OML, in terms of the number of satisfiable requests, the average energy consumption per request, and the number of energy-depleted nodes.**

Keywords – Static ad hoc networks, message routing, energy efficiency, approximation algorithm, heuristic algorithm.

I. Introduction

Ad hoc networks are multihop communication networks consisting of small computing devices with wireless interfaces. They are mainly used by a group of users for spontaneous communication among themselves without the support of preexisting infrastructure. In this paper, we focus on static ad hoc networks, which find many applications in less mobile and dynamic environments [1], [2]. For example, in an ad hoc meeting, several authors can establish a wireless ad hoc network using their laptops to access or modify the same document (e.g., powerpoint slides, an article, or a book). In this scenario, even though the communication is facilitated by an ad hoc network, the users (i.e., the ad hoc nodes) are mainly static. After the ad hoc nodes form a network, pairs of nodes exchange messages according to the application requirement [12]. For example, in the above ad hoc meeting scenario, to further divide and conquer the editing task, pairwise communication can exist among multiple source-destination pairs.

On the other hand, because of the limited power in ad hoc nodes and the ad hoc nature of such networks, energy efficiency has always been a consideration for routing algorithms in ad hoc networks. Routing is a process to send a message from one node (called *source node*) to another node (called *destination node*) in the network. Every node involved in the routing process uses its battery power. Therefore, to maintain proper network operation, it is critical that every attempt to transmit a message succeeds. Hence, we are interested in designing energy-efficient routing algorithms in static ad hoc networks with the objective of maximizing the total number of messages (requests) that can be successfully satisfied, under the energy constraint of each node. We refer to the problem as $maxR$.

The *online* version of the problem, wherein the sequence of messages is not known ahead of time, has been extensively studied (CMAX [7], MRPC [11], OML [12]). However, the research for offline $maxR$ remains scarce. The offline $maxR$ problem is to maximize the number of satisfiable requests by assuming that the sequence of requests is pre-known. In applications wherein ad hoc nodes work towards the same application goal with well-coordinated communication, it could be the case that the sequence of requests has been decided. Therefore, how to maximize the total number of routing requests that can be satisfied given the known sequence of requests is a relevant and important problem. To the best of our knowledge, the offline algorithm for maxR problem has not been well investigated. It is only mentioned in CMAX [7] to show the performance difference between the offline and online algorithms. The solution to the offline problem itself, its hardness and approximability, remain unclear and has not been well studied.

In this paper, We first show that after appropriate transformation, the offline $maxR$ problem is equivalent to the well-known maximum disjoint path problem [8], [9], which is NP-hard. We then propose an approximation algorithm called *Greedy-Disjoint-Paths* algorithm (GDP). We show that the number of satisfiable requests yielded by GDP is within a constant ratio of that of the optimal algorithm. *It is known that the best offline algorithm performs better than any online algorithm* [12]. So GDP can be served as a benchmark to evaluate the performance of online algorithms that aim to maximize the number of requests in ad hoc networks. In addi-

tion, we propose a new online algorithm called ***MECBE*** that tackles the online version of $maxR$ problem by ***M***inimizing the total ***E***nergy ***C***onsumption and ***B***alancing node ***E***nergy in the network. Simulation results show that GDP outperforms both MECBE and OML, while MECBE outperforms OML, in terms of the number of satisfiable requests, the average energy consumption per request, and the number of energy-depleted nodes.

The rest of the paper is organized as follows: Section II introduces the network model and energy model, formulates the $maxR$ problem, and gives an overview of the related works. Section III presents our proposed algorithms. Section IV compares our algorithms with the existing one and presents the simulation results. And section V concludes the paper and points out the future work.

II. Problem Formulation and Related Works

A. Network and Energy Models

An ad hoc network can be represented by a general undirected graph $G(V, E)$, where $V = \{1, 2, ..., n\}$ is a set of n ad hoc nodes and E is a set of m edges. There is an edge $(u, v) \in E$ connecting node u and node v iff a single-hop transmission between u and v is possible.

We assume that each node u has a finite and unreplenishable initial energy ϵ_u, which is a non-negative integer value. For the energy consumption of sending and receiving a message by a node, we adopt the first order radio model [5] where for k-bit data over distance l, the transmission energy $E_T(k, l) = E_{elec} \times k + \epsilon_{amp} \times k \times l^2$, and the receiving energy $E_R(k) = E_{elec} \times k$, where $E_{elec} = 50nJ/bit$ and $\epsilon_{amp} = 100pJ/bit/m^2$. When the distances among nodes are in the order of one hundred meters, the term with ϵ_{amp} is much larger than the term with E_{elec}. Therefore, we assume that for each node, sending one unit-sized message costs one unit of energy while receiving one message costs zero energy. This assumption is adopted also for the purpose of fair comparison later - Park and Sahni [12] assume no energy consumption during message reception in OML.

B. Problem Formulation

We formulate the $maxR$ problem as follows: There are a set of p routing requests $\mathcal{R} = \{r_1, r_2, \cdots, r_p\}$ in the network where each request $r_i = (s_i, t_i)$ represents that message m_i is sent from source node s_i to destination node t_i, $1 \le i \le p$. We assume that each message is of unit size.

Let $P_i = \{s_i, \cdots, t_i\}$ $(1 \le i \le p)$ be the *routing path* of message m_i, denoting the sequence of distinct nodes along which m_i is routed from s_i to t_i. Let x_{uj} be the energy cost incurred by node u in routing the message m_j, and let $\epsilon_u^{'}$ denote node u's remaining energy level after all of the messages are routed. Then,

$$\epsilon_u^{'} = \epsilon_u - \textstyle\sum_{j=1}^{p} x_{uj}, \qquad \forall u \in V, \tag{1}$$

where $x_{uj} = 1$ if $u \in P_j - \{t_j\}$, and $x_{uj} = 0$ otherwise. That is, if u is a source node or an intermediate relaying node of m_j, its energy cost is one; if it is a destination node or is not involved in the routing of m_j at all, its energy cost is zero.

The objective of the offline $maxR$ is to find a subset $\mathcal{P}_{sat}$ of the set of routing paths $\mathcal{P} = \{P_1, P_2, ..., P_p\}$, such that the number of messages routed in $\mathcal{P}_{sat}$ is maximized, i.e.

$$\max |\mathcal{P}_{sat}|, \tag{2}$$

under the energy constraint that $\epsilon_u^{'} \ge 0, \forall u \in V$, which implies that any node can not spend more energy than its initial energy level.

The online version of $maxR$ assumes that the sequence of messages that has to be routed over the network is not known ahead of the time while offline $maxR$ assumes the message sequence is pre-known. The major difference in algorithm design between the two is that the online algorithms have to satisfy the requests in the given order while the offline ones do not have to.

C. Related Works

The online version of the $maxR$ problem is one of the approaches to realize energy-efficient routing in ad hoc networks. Several researchers have developed energy-efficient algorithms [6], [10], [13] either through maximizing the lifetime (time at which a communication fails first) or through maximizing the capacity (the number of successful communications over some fixed period of time).

Misra and Banerjee [11] propose the MRPC (maximum residual packet capacity) lifetime-maximization heuristic where routing is done along a path with maximum lifetime. A conditional MRPC algorithm CMRPC is also proposed that attempts to balance energy consumption. Kar et al. [7] develop a capacity-competitive (the capacity is the number of messages routed over some time period) algorithm called CMAX (capacity maximization) with logarithmic competitive ratio. To achieve logarithmic competitive ratio, the CMAX algorithm does admission control, that is, it rejects some routes that are possible. Park and Sahni [12] study how to route a sequence of messages for source and destination pairs. They propose a heuristic algorithm, called online maximum lifetime (OML), and show via simulations that OML is superior to both CMAX and MRPC in terms of network lifetime maximization, energy consumption and energy balancing. Therefore, in this paper, we compare our algorithms with OML.

All of the above algorithms are online algorithms. The offline algorithm is only mentioned in CMAX as a theoretical comparison. A most related work that has studied offline problem in terms of energy consumption is done by Shpungin [3]. They give fundamental bounds on the expected total energy consumption and the network lifetime in the optimal offline solution. However, our goal (maximizing the total number of satisfiable requests) is different from that of [3]. To the best of our knowledge, the offline algorithm of maximizing satisfiable routing requests has not been well studied.

OML Online Algorithm [12]. To make the paper self-inclusive, we introduce the OML algorithm in detail as follows.

Algorithm OML: Online Maximum Lifetime heuristic algorithm

1: **for** each request $r_i \in \mathcal{R} = \{r_1, r_2, \cdots, r_p\}$ **do**
2: **Step 1:** [Compute G'']
3: $G' = (V, E')$ where $E' = E - \{(u,v) | ce(u) < w(u,v)\}$.
4: Let P_i' be a shortest s_i to t_i path in G'.
5: If there is no such P_i', the route request fails, stop.
6: Compute the minimum residual energy $minRE$ for sensors other than t_i on P_i'.
7: Let $G'' = (V, E'')$ where $E'' = E' - \{(u,v) | ce(u) - w(u,v) < minRE\}$.
8: **Step 2**: [Find route path]
9: Compute the weight $w''(u,v)$ for each edge of E''.
10: Let P_i'' be a shortest s_i to t_i path in G''.
11: Use P_i'' to route from s_i to t_i.
12: **end for**

Fig. 1. The OML algorithm

The main idea of OML is that to maximize lifetime, delay as much as possible the depletion of a sensor's energy to a level below that needed to transmit to its closest neighbor. OML achieves this by a two-step process to find a path for each routing request $r_i = (s_i, t_i)$. In the first step, all edges (u,v) such that $ce(u) < w(u,v)$ are removed from G because these edges require more energy than available for a transmit. Let the resulting graph be $G' = (V, E')$. Next, determine the minimum energy path, P_i' from s_i to t_i in the pruned graph G'. This may be done using Dijkstra's shortest path algorithm [4]. In case there is no s_i to t_i path in the pruned graph G', the routing request r_i fails. So, assume such a P_i' exists. Using P_i', compute the residual energy, $re(u) = ce(u) - w(u,v)$ for (u,v), an edge on P_i'. Let $minRE = min\{re(u) | u \in P_i' \text{ and } u \neq t_i\}$. Let $G'' = (V, E'')$ be obtained from G' by removing all edges $(u,v) \in E'$ with $ce(u) - w(u,v) < minRE$. That is, all edges whose use would result in a residual energy below $minRE$ are pruned from E'. This pruning is an attempt to prevent the depletion of energy from sensors that are low on energy.

In the second step, find the path to route request r_i. For this, we begin with G'' as above and assign weights to each $(u,v) \in E''$. The weight assignment is done so as to balance the desire to minimize total energy consumption as well as the desire to prevent the depletion of a sensor's energy. Let $eMin(u) = min\{w(u,v) | (u,v) \in E''\}$ be the energy needed by sensor u to transmit a message to its nearest neighbor in G''. Let ρ be defined as below.

$$\rho(u,v) = \begin{cases} 0 & \text{if } ce(u) - w(u,v) > eMin(u) \\ c & \text{otherwise,} \end{cases}$$

where c is a nonnegative constant and is an algorithm parameter. For each $u \in V$, define

$$\alpha(u) = \frac{minRE}{ce(u)}.$$

The weight $w''(u,v)$ assigned to edge $(u,v) \in E''$ is

$$w''(u,v) = (w(u,v) + \rho(u,v))(\lambda^{\alpha(u)} - 1),$$

where λ is another nonnegative constant and an algorithm parameter. As can be seen, this weighting function, through ρ, assigns a high weight to edges whose use on a routing path cause a sensor's residual energy to become low. Also, all edges emanating from a sensor whose current energy is small relative to $minRE$ are assigned a high weight because of the λ term. Thus, the weighting function discourages the use of edges whose use on a routing path is likely to result in the failure of a future route.

Time Complexity of OML. OML performs two shortest path computation per route request. The shortest path algorithm can be implemented based on a Fibonacci heap, and can be computed in $O(m+n\log n)$ [4]. The time complexity of OML is therefore $O(m + n\log n)$ per route request. Therefore, the time complexity of OML is $O(p \times (m + n\log n))$.

III. The Algorithms

In this section, we first propose an offline greedy algorithm GDP to address the $maxR$ problem and then put forward a new online heuristic algorithm MECBE.

A. The Greedy Disjoint Path (GDP) Algorithm

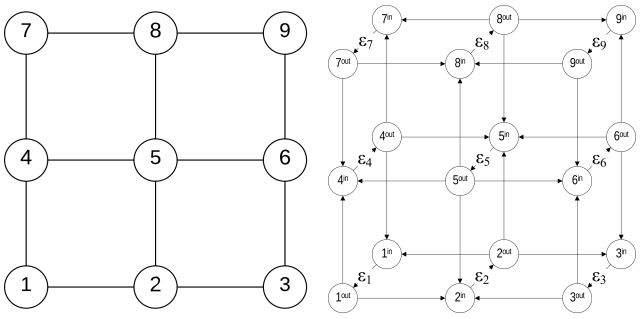
Before presenting the algorithm, we transform the undirected graph $G(V,E)$ into a directed graph $G'(V',E')$, and show that offline $maxR$ on G' is equivalent to the well-known maximum disjoint path problem [8], [9].

1) Graph Transformation: First, replace each undirected edge $(u,v) \in E$ with two directed edges (u,v) and (v,u), and set the capacity of all the directed edges to infinity. Then split each node $u \in V$ into two nodes: in-node u^{in} and out-node u^{out}, and add a directed edge (u^{in}, u^{out}) with capacity ϵ_u, which is the initial energy of node u. All the incoming directed edges of node u are incident on u^{in} and all the outgoing directed edges of node u emanate from u^{out}. Now the initial routing requests $(s_1,t_1), (s_2,t_2), ..., (s_p,t_p)$ in G become the new routing requests $(s_1^{in}, t_1^{in}), (s_2^{in}, t_2^{in}), ..., (s_p^{in}, t_p^{in})$ in $G'(V',E')$. Note that it is the in-node of each destination node that becomes the new destination node, due to the fact that receiving messages does not cost energy in our model. Assume that there are m' and n' number of edges and nodes in $G'(V',E')$, then $n' = 2n$ and $m' = n + 2m$.

Fig. 2 shows an example of an ad hoc network before and after the transformation. It is a 3×3 grid network, with three source-destination pairs: $(3,8)$, $(6,4)$, and $(7,2)$. After the transformation, the three source-destination pairs are: (3^{in}, 8^{in}), (6^{in}, 4^{in}), and (7^{in}, 2^{in}).

Theorem 1: If all the nodes have the same initial energy levels, then offline $maxR$ in $G'(V',E')$ is equivalent to maximum disjoint path problem in $G'(V',E')$.

Proof: The maximum disjoint path problem [8], [9] is as follows: Given a directed graph $G'(V',E')$, an integer capacity c of each edge, and a set of p connection request pairs $\mathcal{R}$: $\{(s_1,t_1), ..., (s_p,t_p)\}$, the goal is to find a realizable subset $\mathcal{I}$ of $\mathcal{R}$ with maximum size. A subset $\mathcal{I}$ of $\mathcal{R}$ is



(a) (b)

Fig. 2. An ad hoc network of 9 nodes before and after transformation. (a) shows the three source-destination pairs before transformation: $(3, 8)$, $(6, 4)$, and $(7, 2)$. (b) shows the three source-destination pairs after transformation: $(3^{in}, 8^{in})$, $(6^{in}, 4^{in})$, and $(7^{in}, 2^{in})$. In (b), the capacity of edge (u^{in}, u^{out}) is ϵ_u, which is the initial energy of node u, while the capacity of other edges is infinity.

Algorithm GDP: a Greedy-Disjoint-Paths algorithm to find the maximum number of completed (satisfiable) requests on $G'(V', E')$.

1: **Notations**: m' is the total number of edges in G'; ϵ is the initial energy of all the nodes; $\beta = m'^{1/(\epsilon+1)}$
2: $\mathcal{I} = \emptyset$, $\mathcal{I}$ is the set of completed requests
3: For all $e \in E'$, set its weight to 1
4: **while** There are still requests that can be satisfied **do**
5: Let P_i be the minimum weighted path so that adding P_i to the selected set of paths does not use any edge more than ϵ times, and P_i connects some (s_i, t_i) pair not yet connected
6: Add i to $\mathcal{I}$ and use path P_i to route the message from s_i to t_i
7: Multiply the length of all edges along P_i by β
8: **end while**

Fig. 3. The GDP algorithm.

realizable in G' if all the requests in $\mathcal{I}$ can be satisfied while each edge in G' is used at most c times.

In $maxR$, assume that the initial energy levels of all the nodes are equaled to ϵ. Then finding the maximum number of satisfiable routing requests from $(s_1^{in}, t_1^{in}), (s_2^{in}, t_2^{in}), ..., (s_p^{in}, t_p^{in})$ in $G'(V', E')$ is exactly the maximum disjoint path problem in $G'(V', E')$ with $\mathcal{R} = \{(s_1^{in}, t_1^{in}), (s_2^{in}, t_2^{in}), ..., (s_p^{in}, t_p^{in})\}$ and $\epsilon = c$. □

The maximum disjoint path problem is NP-hard [8], [9]. Following the idea of Theorem 1 and inspired by the works in [8], [9], we present a greedy algorithm called Greedy-Disjoint-Paths (GDP) algorithm in Fig. 3. GDP starts with each edge of weight 1, and works in iterations. In each iteration it tries to find the minimum weighted path connecting a source and destination pair while satisfying the capacity of each edge. Once the minimum weighted path is selected, the weights of all the edges on that path are multiplied by a constant β ($\beta > 1$). Here, $\beta = m'^{1/(\epsilon+1)}$, where m' is the total number of edges in G'. The intuition behind setting β like this is that an edge will gain more weight if it has been used, which encourages other edges to be selected to route the following messages.

Theorem 2 shows that the GDP algorithm is an approximation algorithm with performance guarantee.

Theorem 2: The GDP algorithm is a $1/(2\epsilon m'^{1/(\epsilon+1)} + 1)$ approximation algorithm. That is, the total number of satisfiable requests by GDP is at least $1/(2\epsilon m'^{1/(\epsilon+1)} + 1)$ times of the maximum number of satisfiable requests in optimal solution.

Proof: Here we give a proof sketch for $\epsilon = 2$ (for more detailed proof, please refer to page 630, [9]). Let I^* be the set of routing requests satisfied in an optimal solution, and I be the set of requests satisfied by GDP algorithm. We consider a path P_i selected by GDP to be *short* if its length is less than β^2 (recall that P_i for $1 \leq i \leq p$ is the routing path of request r_i). Let I_s denote the set of short path selected by GDP. Let $\bar{l}$ be the length function at the first iteration in GDP at which there are no more short paths left to select. For a path P_i^* in the optimal solution I^*, it is short if $\bar{l}(P_i^*) < \beta^2$. We have the first observation (denoted as OB (a)): For a request $i \in I^*$ that is not satisfied by GDP (that is, $i \in I^* - I$), $\bar{l}(P_i^*) \geq \beta^2$.

At the iteration that no short paths is left to choose, the total length of the edges in the graph is $\sum_e \bar{l}_e$. The sum of the edges in the graph starts out as m' (length 1 for each edge, as indicated in GDP). Adding a short path to the solution I_s can increase the length by at most β^3, as the selected path has length at most β^2, and the lengths of the edges are increased by a β factor along the path. We therefore have the second observation (denoted as OB (b)): $\sum_e \bar{l}_e \leq \beta^3 |I_s| + m'$.

Consider OB (a) and all paths in $I^* - I$, we get $\sum_{i \in I^* - I} \bar{l}(P_i^*) \geq \beta^2 |I^* - I|$. On the other hand, each edge is used by at most two paths in the solution I^*, so we have $\sum_{i \in I^* - I} \bar{l}(P_i^*) \leq \sum_e 2\bar{l}_e$. Combining these with OB (b), we get $\beta^2 |I^*| \leq 2(\beta^3 |I| + m') + \beta^2 |I|$. Finally, divide both sides by β^2, and consider that $|I| \geq 1$ and $\beta = m'^{1/3}$, we get $|I^*| \leq (4m'^{1/3} + 1)|I|$. For any energy level ϵ, if we choose $\beta = m'^{1/(\epsilon+1)}$, and consider paths to be short if their length is at most β^ϵ, we get $|I| \geq |I^*|/(2\epsilon m'^{1/(\epsilon+1)} + 1)$. □

Time Complexity of GDP. The time complexity of GDP is determined by the minimum weighted path (i.e., the shortest path) computation, which takes $O(m' + n' \log n')$. There are at most p rounds. In each round it finds among at most p requests one minimum weighted route that can be satisfied. Since $n' = 2n$, $m' = n + 2m$, and $m = O(n^2)$, the time complexity of GDP is $O(p^2 \times (n + 2m + 2n \log(2n))) = O(p^2 \times (m + n \log n))$.

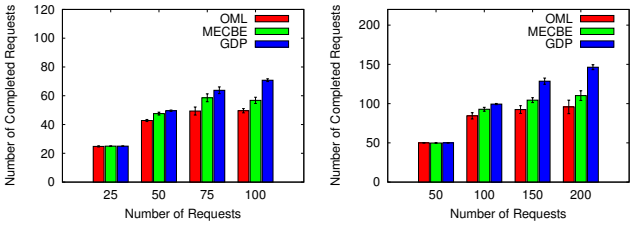
B. The MECBE Algorithm

The new online algorithm MECBE is shown in detail in Fig. 4. It tries to satisfy the maximum number of requests through minimizing total energy consumption and energy balancing in message routing: on one side, it minimizes the total energy consumption by finding the shorter path between source and sink nodes and on the other side, it favors nodes with higher remaining energy. The MECBE algorithm achieves these goals by finding a path between the source

Algorithm MECBE: Minimizing total Energy Consumption and Balancing node Energy to maximize the number of completed requests. Here, $w(u,v)$ is the energy required for a single-hop transmission from node u to v, $ce(u)$ is the current energy of node u.

1: **for** each request $r_i \in \mathcal{R} = \{r_1, r_2, \cdots, r_p\}$ **do**
2: $G' = (V, E')$ where $E' = E - \{(u,v) | ce(u) < w(u,v)\}$
3: In G', find the path $\{s_i, u_1, u_2, \cdots, u_q, t_i\}$ from s_i to t_i that can minimize metric $\sum_{j=1}^{q} \frac{1}{ce(u_j)}$
4: **if** the returned path is NULL **then**
5: Stop the program
6: **end if**
7: **for** each node u_i on the path found except the sink node **do**
8: $ce(u_i) = ce(u_i) - w(u,v)$
9: **end for**
10: **end for**

Fig. 4. The MECBE algorithm.



(a) 10x10 Network. (b) 20x20 Network.

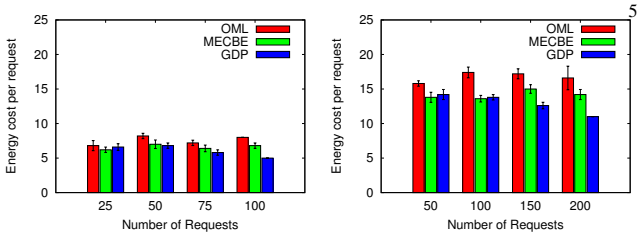
Fig. 5. Number of satisfiable requests in the network.

and destination that can minimize the metric $\sum_{j=1}^{q} \frac{1}{ce(u_j)}$, where $ce(u_j)$ is the current energy of node u_j. Intuitively, a shorter path with higher energy nodes will be selected to route the message. As OML, MECBE first deletes those edges (u,v) where node u's current energy $ce(u)$ is below the required energy $w(u,v)$ for a single-hop transmission from u to v. In this paper, $w(u,v)$ is 1. Then, it selects a path that minimizes the metric between a source and a destination. For example, suppose there are two paths between s_i and t_i: $\{s_i, u_1, u_2, t_i\}$ and $\{s_i, u_3, u_4, t_i\}$. The current energy levels of u_1, u_2, u_3, u_4 are $2, 3, 3, 4$, respectively. According to the metric, the value for the first path is: $\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$ and the value for the second path is: $\frac{1}{3} + \frac{1}{4} = \frac{7}{12}$. The second value is smaller, so the second path will be chosen.

Time Complexity of MECBE. In MECBE, it takes $O(m + n\log n)$ per route request, where m and n are the number of edges and number of nodes in the original graph $G(V,E)$. And there are p requests. Therefore, the time complexity of MECBE is $O(p \times (m + n\log n))$.

IV. Performance Evaluation

In this section, we compare our algorithms with the existing algorithm. Simulations reported in the literature have already established the superiority of OML over CMAX and MRPC [12]. Thus, we compare our algorithms GDP and MECBE with OML.



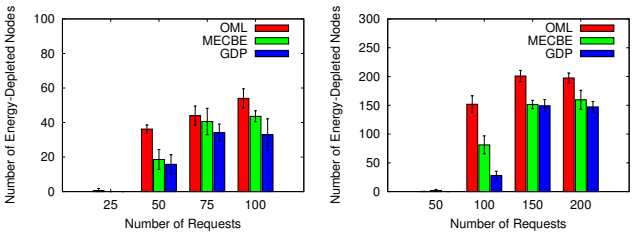
(a) 10x10 Network. (b) 20x20 Network.

Fig. 6. Energy consumption per completed request.

The comparison was conducted using a custom simulator in C. We assumed that ad hoc nodes were connected in a grid, wherein the distance between any two neighbors is one unit. All of our algorithms can be applied to a general network topology.

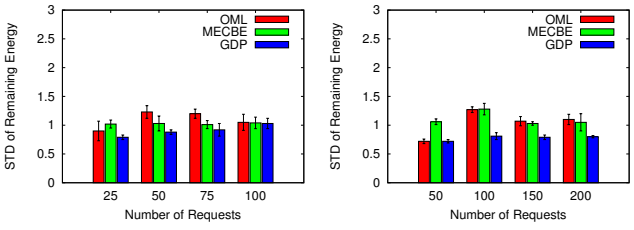
Parameter Setting. In our simulations, we used two settings of grid network topology: 10×10 and 20×20, with 100 and 400 nodes, respectively. We generated a sequence of requests $\mathcal{R} = \{r_1, r_2, \cdots, r_p\}$ with sources and destinations randomly selected from the nodes. In all simulation figures below, each data point is an average of five runs with 95% confidence interval. When comparing all of the three algorithms: GDP, MECBE, and OML, we set the initial energy level of all the nodes to 5 so that not all the request sequences could be finished by all of the algorithms all the time. For 10×10 and 20×20 network, the number of edges are 180 and 760, respectively. So the number of edges m' in the transformed graphs are 460 and 1920, respectively. Therefore, $\beta = m'^{1/(\epsilon+1)} = m'^{1/6}$, which equals 2.78 and 3.52 for 10×10 and 20×20, respectively. When setting the parameters for OML, since we used a grid network, we set the energy consumption $w(u,v)$ of sending a message from u to v to 1. The $eMin(u)$ in OML, which is the energy needed by u to transmit a message to its nearest neighbor, also equals 1. In OML, $\rho(u,v)$ is defined to be zero if node u's current energy is greater than the sum of $w(u,v)$ and $eMin(u)$, which is always the case in our paper. Therefore $\rho(u,v)$ in OML is set to 0. The algorithm parameter λ in OML is set to 10^{11} because it gave a stable performance to OML [12]. Please refer to [12] for more detailed definitions of all of the above parameters.

Number of Completed Requests. Figure 5 shows the number of requests completed (satisfied) for all three algorithms in the 10×10 and 20×20 grid networks, respectively. The results show that when the number of requests is small (e.g. 25), all algorithms can complete all the requests, indicating that the performance difference among the algorithms is small in a less stressful scenario. However, when the number of requests increases, the GDP algorithm satisfies the most number of message requests, with 30 - 40% more requests satisfied in GDP than MECBE and OML. Meanwhile, MECBE outperforms OML with around 20% more requests satisfied. This is because GDP is an approximation



(a) 10x10 Network. (b) 20x20 Network.

Fig. 7. Number of energy-depleted nodes.



(a) 10x10 Network. (b) 20x20 Network.

Fig. 8. Standard deviation of remaining energy of nodes.

algorithm with performance guarantee, while MECBE tries to minimize total energy consumption as well as to balance energy consumption among individual nodes.

Energy Consumption Per Completed Request. Figure 6 compares the energy consumption per completed request, which is equal to total energy consumption divided by the total number of completed requests, for all three algorithms. The results show similar performance comparison in all three algorithms in most cases, with more evident performance difference in larger networks with more message requests. This indicates that even though GDP completes more requests than the other two algorithms, it does not incur much more energy consumption. However, in less challenging scenarios (25 requests in 10×10 networks, and 50 and 100 requests in 20×20 networks), we notice that the MECBE algorithm gives less consumed energy per completed request than the GDP does. This indicates that the GDP is probably not the best offline algorithm for *maxR*, even though it has a constant performance guarantee.

Number of Energy-depleted Nodes. Figure 7 shows the number of energy-depleted nodes resulted from the three algorithms. It shows that in all cases, the number of energy-depleted nodes in OML is much larger than that in the other two algorithms. If the number of energy-depleted nodes is an indicator of the lifetime of ad hoc networks, Figure 7 demonstrates that both MECBE and GDP algorithms result in longer network lifetime than OML does.

Standard Deviation of Remaining Energy. To further investigate how balanced the remaining energy of the nodes is, we calculate the standard deviation (STD) of the remaining energy of all the nodes. Figure 8 shows that GDP balances the energy levels in nodes the best in routing, while OML and MECBE perform similarly in most cases.

V. Conclusion and Future Direction

In this paper, we developed energy-efficient routing algorithms for static ad hoc networks, with the objective of maximizing the number of satisfiable requests under the constraint of limited battery power of each node. We first studied the offline version of the problem and proposed an approximation algorithm. It can be used as a benchmark for evaluating the performance of online algorithms. Then we put forward a new online algorithm MECBE to maximize the number of satisfiable requests through maximizing the total energy consumption and balancing node energy in the network. We showed empirically that MECBE outperforms the OML algorithm in the number of satisfiable requests and is close to the benchmark. MECBE and GDP are also better than OML in terms of energy consumption per completed request, the number of energy-depleted nodes, and the energy balancing among individual nodes. As a first step in our future work, we will consider a more general energy model where different ad hoc nodes could have different initial energy levels, and energy consumption of sending and receiving routing messages depends on the distance between nodes. We will also design distributed versions of the proposed algorithms and explore new energy-efficient routing algorithms in ad hoc networks.

Acknowledgment

This research is partially sponsored by NSF Grants CNS-1116849 and EPS-0903806.

References

[1] X.Y. Li and Y. Wang and H.M. Chen and X. Chu and Y.W. Wu and Y. Qi, "Reliable and Energy-Efficient Routing for Static Wireless Ad Hoc Networks with Unreliable Links", *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, issue 10, Oct. 2009, pp. 1408-1421.

[2] H. Hernndez and C. Blum, "Ant colony optimization for multicasting in static wireless ad-hoc networks", *Swarm Intelligence*, vol. 3, issue 2, Oct. 2009, pp. 125-148.

[3] H. Shpungin, "Energy Efficient Online Routing in Wireless Ad Hoc Networks", *Proc. IEEE SECON 2011*.

[4] M. L. Fredman and R. E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Proc. of the 25th IEEE Annual Symp. on Foundations of Computer Science*, 1984, pp. 338-346.

[5] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks", *Proc. of HICSS*, 2000.

[6] R. Kannan and S. Iyengar, "Game Theoretic Models for Reliable Path-Length and Energy Constrained Routing with Data Aggregation in Wireless Sensor Networks", *IEEE J. Selected Areas in Comm.*, vol. 22, no. 6, 2004, pp. 1141-1150.

[7] K. Kar, M. Kodialam, T. Lakshman, and L. Tassiulas, "Routing for Network Capacity Maximization in Energy-Constrained Ad-Hoc Networks", *Proc. of IEEE INFOCOM*, 2003.

[8] J. Kleinberg, "Approximation Algorithms for Disjoint Paths Problems", *PH.D. Dissertation*, 1996.

[9] J. Kleinberg and E. Tardos, "Algorithm Design", Addison Wesley, 2005.

[10] A. Mohanoor and S. Radhakrishnan and V. Sarangan, "On energy aware routing in wireless networks", *Proc. of BROADNETS 2007*.

[11] A. Misra and S. Banerjee, "MRPC: Maximizing Network Lifetime for Reliable Routing in Wireless Environments", *Proc. of IEEE Wireless Communication and Networking Conference (WCNC)*, 2002.

[12] J. Park and S. Sahni, "An Online Heuristic for Maximum Lifetime Routing in Wireless Sensor Networks", *IEEE Trans. on Computers*, vol. 55, no. 8, Aug. 2006, pp. 1048-1052.

[13] G. Zussman and A. Segall, "Energy Efficient Routing in Ad Hoc Disaster Recovery Networks", *Proc. IEEE INFOCOM*, 2003.