PERFORMANCE ANALYSIS OF A COOPERATIVE FLOW GAME ALGORITHM IN AD HOC NETWORKS AND A COMPARISON TO DIJKSTRA'S ALGORITHM

Serap Ergün
Süleyman Demirel University, Faculty of Technology
Department of Software Engineering
Isparta, Turkey

Sirma Zeynep Alparslan Gök
Süleyman Demirel University, Arts and Sciences Faculty
Department of Mathematics
Isparta, Turkey

Tuncay Aydoğan
Süleyman Demirel University, Faculty of Technology
Department of Software Engineering
Isparta, Turkey

Gerhard Wilhelm Weber
Poznan University of Technology
Faculty of Engineering Management
Poznan, Poland

(Communicated by Leen Stougie)

Abstract. The aim of this study is to provide a mathematical framework for studying node cooperation, and to define strategies leading to optimal node behaviour in ad hoc networks. In this study we show time performances of three different methods, namely, Dijkstra's algorithm, Dijkstra's algorithm with battery times and cooperative flow game algorithm constructed from a flow network model. There are two main outcomes of this study regarding the shortest path problem which is that of finding a path of minimum length between two distinct vertices in a network. The first one finds out which method gives better results in terms of time while finding the shortest path, the second one considers the battery life of wireless devices on the network to determine the remaining nodes on the network. Further, optimization performances of the methods are examined in finding the shortest path problem. The study shows that the battery times play an important role in network routing and more devices provided to keep the network. To view the time performance analysis of the methods MATLAB is used. Also, considering the cooperation between the nodes, it is envisaged that using cooperative game theory brings a new approach to network traffic engineering and routing methods.

2010 Mathematics Subject Classification. Primary: 91A80, 90B18; Secondary: 68W40.
Key words and phrases. Ad hoc networks, Dijkstra's algorithm, Dijkstra's algorithm with battery times, cooperative flow game algorithm, simulation, optimization.
* Corresponding author: Serap Ergün.
1. Introduction. Wireless networks are considered necessary for the support of emerging services with their increasing requirements and are characterized by a distributed, dynamic, self-organizing architecture. These wireless networks are categorized according to their specific characteristics. Typical examples are ad hoc, mesh, sensor, cognitive radio networks, etc.

An ad hoc network is a wireless network without any fixed infrastructure or centralized control. The operation of the network is based on cooperation. The network consists of terminals, which act as routers in the network. The terminals in the network cooperate and relay the traffic of each other with inter-terminals [27]. The terminal is not only responsible for sending and receiving its own data, but it also has to forward the traffic of the other terminals. An ad hoc network is illustrated in Figure 1.

![Figure 1. An ad hoc network](image)

The problem of energy in ad hoc networks has been addressed before in [9], [10], [27], [38], [43]. The approach in those works is to minimize the total consumed energy to reach the destination, which minimizes the energy consumed per unit flow or packet. The potential problem in current protocols in ad hoc networks is that they find the lowest energy route and use that for every communication. However, that is not the best thing to do for network lifetime. Using a low energy path frequently leads to energy depletion of the nodes along that path and in the worst case may lead to network losses.

Ad hoc devices generally operate on batteries. The remaining battery capacity limits the lifetime of nodes and the possibility of sending, receiving or relaying data. There is also an important influence on the overall communication performance and the lifetime of the ad hoc mobile wireless net, because the lack of nodes can result in partitioning of the network. So power consumption becomes an important issue in network routing. To counteract this problem, we propose algorithms to select the routes and the corresponding power levels such that the time until the batteries of the nodes drain-out is maximized. In order to maximize the lifetime of the nodes (or can be thought as network), the traffic should be routed such that the energy consumption is balanced among the nodes in proportion to their energy reserves, instead of routing to minimize the absolute consumed power.

In the early to mid-1990, game theory was applied to networking problems including flow control, congestion control, routing and pricing of Internet services. More recently, there has been growing interest in adopting game-theoretic methods to model today’s leading communications and networking issues, including power control and resource sharing in wireless and peer-to-peer networks [26]. In the past three to four years there has been renewed interest in developing networking games, this time to analyze the performance of ad hoc networks. Since the game theoretic models developed for ad hoc networks focus on distributed systems, results and conclusions generalize well as the number of players (nodes) are increased [25], [30], [43].
Functionalities of the network layer which is the third layer of OSI (Open Systems Interconnection) include the establishment of routes and the forwarding of packets along those routes [26], [40]. In most cases, game theory may be applied to aid a node in determining which the optimal route is deciding whether it should forward a received packet or not.

Today, the availability of wireless devices has increased immensely. However, the availability is limited because of the devices have to be charged in certain periods. In providing communication in ad hoc networks, it also experienced a slight decrease in wireless devices’ battery. On account of battery energy at network nodes is a very limited resource that needs to be utilized efficiently [9], [38], [46]. Therefore, we use the shortest path problem with the objective of maximizing the network lifetime given the sets of source and destination nodes and propose a cooperative flow game algorithm constructed from a flow network model.

A suitable game theoretic model to support decision making of coalition values is that of cooperative games. Cooperative games have been proved useful for solving sharing problems in situations in cooperative environments. The model of cooperative games has had a broad applicability in Operational Research, climate negotiations and policy, in environmental management and pollution control, etc.

In this game, the players are associated with arcs of the network. The value of a coalition is the maximum flow value from the ‘source’ to the ‘sink’ in the sub-network consisting of the original vertex set and those arcs corresponding with the players in the coalition. For certain classes of cooperative games, such as flow games and minimum cost spanning tree games, there is a natural way to formulate them as (generalized) linear production games [28].

The recent interest in ad hoc networks has led to a number of routing algorithms that use the limited resources available at nodes more efficiently. These algorithms typically try to find the minimum energy path to optimize energy usage at a node. In this paper we take the view that always using lowest energy paths may not be optimal from the point of view of network (node) lifetime and long-term connectivity. To optimize these measures, we propose a new algorithm called Dijkstra’s algorithm with battery times that uses the battery times of the nodes as a second parameter in the general Dijkstra’s algorithm. It is believed that Dijkstra’s algorithm with battery times, which is modified from general Dijkstra’s algorithm and cooperative flow game algorithm constructed from flow network model, contribute to the literature. We present simulation results which show the improved performance obtained by our method and the increase in network lifetime. While the primary metric of interest is network length, we also show the performance results with regard to other metric as battery times.

Cooperative game theory provides analytical tools to study the behaviour of rational players when they cooperate and consider the utility of all the players. An important application of cooperative games (also called coalitional games) is that they provide a mathematical formulation for collective decision-making and optimization problems. Under such circumstances, very often, the characteristic function value of a coalition can be represented succinctly as the optimum value of a combinatorial optimization problem. Such cooperative games are called combinatorial optimization games [35].

The aim of this study is to show the performances of three different methods in ad hoc networks. These methods are Dijkstra’s algorithm, Dijkstra’s algorithm with battery times and cooperative flow game algorithm constructed from a flow...
network model. There are two main outcomes of this study by taking the shortest problem. The first one finds out which method gives better results in terms of time while finding the shortest path. The shortest path represents the shortest connection between two vertices (or nodes) in a network. The second one, considers the battery life of wireless devices on the network while finding the shortest path, to determine the remaining nodes on the network. Simulations in MATLAB (for details see Tables 1, 2 and 3) is done for the performance analysis. So, system will be implemented with the optimal performance.

The five main contributions of this paper are:

(i) The new Dijkstra’s algorithm with battery times is generated from general Dijkstra’s algorithm. With this algorithm, the battery times of ad hoc devices are used as a second parameter in network routing. Eventually, simulation results show that the battery times also play an important role in network routing and more devices provided to keep the network.

(ii) Finding the shortest problem, cooperative flow game algorithm constructed from a flow network model has found the shortest way using the cost of cooperation between the nodes and gave even better results than other algorithms.

(iii) We present two new algorithmic techniques to shortest path problem.

(iv) The simulation models with 10, 15 and 20 nodes in ad hoc network is supported by simulation results and related tables.

(v) Due to the interdisciplinary character of our work, we intend this study to be accessible for researchers from different areas.

Now, we give related research about the study in the framework of optimization algorithms for networks and game theory. The shortest path problem which is one of the most studied network optimization problems has been used frequently in the literature [1], [8], [13], [17], [23], [31], [34], [45]. Game theoretic approaches of the shortest path problem including both noncooperative games [6], [29], [45], [48] and cooperative games [7], [14] have been studied, too. Also, some applications in ad hoc networks with game theoretic approaches can be seen [3], [15], [26], [28], [30], [31], [41], [43]. Different from the studies in literature we make cooperation between nodes in network routing provided by cooperative game theory and using battery time parameter as a second parameter in Dijkstra’s algorithm.

Given the results from the literature, we want to see how to bring together flow games, shortest path games, ad hoc networks and some algorithms. We hope that this might give us some insight with a comparison of the three different methods for ad hoc networks.

The organization of the paper is as follows. In Section 2, some basic concepts from cooperative game theory, cooperative flow game, shortest path problem, and Dijkstra’s algorithm that will be used in later sections are given. We give Dijkstra’s algorithm with battery times which is modified from general Dijkstra’s algorithm and cooperative flow game algorithm constructed from flow network model in section 3. Further, we compare Dijkstra’s algorithm, Dijkstra’s algorithm with battery times and cooperative flow game algorithm constructed from flow network model in an ad hoc network in Section 4, and we devote this section to give simulation results. Finally, we conclude in Section 5, with an offer on future research and application.

2. Preliminaries. In this section, we give some basics from cooperative game theory and cooperative flow game, shortest path problems and Dijkstra’s algorithm.
2.1. Some basics from cooperative game theory and cooperative flow game. We give in the following some basics from classical games in coalitional form and some basics from cooperative flow games. For an extensive description see [13], [18], [24], [35].

A cooperative *n*-person game in coalitional form is an ordered pair $< N, c >$, where $N = \{1, 2, \ldots, n\}$ (the set of players) and $c : 2^N \to \mathbb{R}$ is a map, assigning to each coalition $S \subseteq 2^N$ a real number, such that $c(\emptyset) = 0$. This function $v$ is called the characteristic function of the game, $c(S)$ is called the *worth* (or *value*) of coalition $S$. Often, we identify a game $< N, c >$ with its characteristic function.

Now, we define cooperative flow games constructed from a flow network model, which we use in our model.

A cooperative flow network can be described by a graph with node set $V$ and arc set $E$ a cooperative flow network derives games are called cooperative flow games. There are two distinguished nodes: the source ($S_o$) and the sink ($S_i$). It is allowed that several arcs have the same end points [18], [24].

Consider a directed network $D = (V, E; t)$ and $t : E \to \mathbb{R}$ is the data transmission time.

Then the cooperative flow game $\Gamma_f = (E, v)$ associated with the network $D$ is defined as follows: (i) The player set is $N = \{1, 2, \ldots, n\}$; (ii) The arc set is $E = \{(i, j) \mid i, j \in N, i \neq j\}$; (iii) For each coalition $S \subseteq E$, $c(S)$ is the value of the flow from $S_o$ to $S_i$ in the network of $D$ consisting only of arcs belonging to $S$.

The Shapley value [39] is one of the most interesting one-point solution concepts in classical cooperative game theory. The Shapley value associates to each $n$-person game one (payoff) vector in $\mathbb{R}^n$.

Let $\pi(N)$ be the set of all permutations $\sigma : N \to N$. The set $P^\sigma(i) = \{r \in N \mid \sigma^{-1}(r) < \sigma^{-1}(i)\}$ consists of all predecessors of $i$ with respect to the permutation $\sigma$. Let $c \in G^N$ and $\sigma \in \pi(N)$. The marginal vector $m^\sigma(c) \in \mathbb{R}^n$ with respect to $\sigma$ and $v$ has as $i$-th coordinate $m^\sigma_i(c) = c(P^\sigma(i) \cup \{i\}) - c(P^\sigma(i))$ for each $i \in N$. The Shapley value $\Phi(v)$ of a game $v \in G^N$ is the average of the marginal vectors of the game, i.e., $\Phi(v) = \frac{1}{n!} \sum_{\sigma \in \pi(N)} m^\sigma(v)$.

$$ \Phi(v) = \frac{1}{n!} \sum_{\sigma \in \pi(N)} m^\sigma(v). \quad (1) $$

2.2. The shortest-path problem. The shortest-path problem is one of the most studied network optimization problems. The shortest-path problem is a particular network model that has received a great deal of attention for both practical and theoretical reasons. The essence of the problem can be stated as follows: Given a network with distance $d_{ij}$ (or travel time, or cost, etc.) associated with each arc, find a path through the network from a particular origin (source) to a particular destination (sink) that has the shortest total distance. The simplicity of the statement of the problem is somewhat misleading, because a number of important applications can be formulated as shortest- (or longest-) path problems where this formulation is not obvious at the outset. These include problems of equipment replacement, capital investment, project scheduling, and inventory planning [11], [37], [47]. Further, the shortest-path problem often occurs as a sub-problem in more complex situations, such as the sub-problems in applying decomposition to traffic-assignment problems [1].
Table 1. The pseudo code of Dijkstra’s Algorithm

| Line | Code |
|------|------|
| 1    | function Dijkstra(Graph, source): |
| 2    | dist[source] := 0 // Distance from source to source |
| 3    | for each vertex v in Graph: // Initializations |
| 4    | if v ≠ source |
| 5    | dist[v] := infinity // Unknown distance function from source to v |
| 6    | previous[v] := undefined // Previous node in optimal path from source |
| 7    | end if |
| 8    | add v to Q // All nodes initially in Q (unvisited nodes) |
| 9    | end for |
| 10   | while Q is not empty: // The main loop |
| 11   | u := vertex in Q with min dist[u] // Source node in first case |
| 12   | remove u from Q |
| 13   | for each neighbor v of u: // where v has not yet been removed from Q. |
| 14   | alt := dist[u] + length(u, v) |
| 15   | if alt < dist[v]: // A shorter path to v has been found |
| 16   | dist[v] := alt |
| 17   | previous[v] := u |
| 18   | end if |
| 19   | end for |
| 20   | end while |
| 21   | return dist[], previous[] |
| 22   | end function |

Algorithms for this problem have been studied for a long time [2], [5], [9], [13], [16], [18], [19], [20], [21]. Until recently, all theoretical developments in the single source shortest paths problem for general graphs (networks) were based in Dijkstra’s algorithm. We note that the problem may also be solved directly (and more efficiently) using a variant of Dijkstra’s algorithm.

In fact, we can interpret the shortest-path problem as a network-flow problem very easily. We simply want to send one unit of flow from the source to the sink at minimum cost. At the source, there is a net supply of one unit; at the sink, there is a net demand of one unit; and at all other nodes there is no net inflow or outflow.

2.3. Dijkstra’s algorithm. Dijkstra’s algorithm, conceived by computer scientist Edsger Dijkstra in 1956 and published in 1959 [13], is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in routing and as a subroutine in other graph algorithms.

Table 1. The pseudo code of Dijkstra’s Algorithm.

The algorithm which can be seen in Table 1 as a pseudo code, the code \( u := \text{vertex in } Q \text{ with min dist}[u] \), searches for the vertex \( u \) in the vertex set \( Q \) that
has the least \( \text{dist}[u] \) value. The command \( \text{length}(u, v) \) returns the length of the edge joining (i.e., the distance between) the two neighbour-nodes \( u \) and \( v \). The variable \( \text{alt} \) is the length of the path from the root node to the neighbour node \( v \) if it were to go through \( u \). If this path is shorter than the current shortest path recorded for \( v \), that current path is replaced with this alt path. The previous array is populated with a pointer to the “next-hop” node on the source graph to get the shortest route to the source.

For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e., the shortest path) between that vertex and every other vertex. It can also be used for finding costs of shortest paths from a single vertex to a single destination vertex by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra’s algorithm can be used to find the shortest route between one city and all other cities. The connections can be between cities, power sources, batteries, access times, base stations, etc. As a result, the shortest path algorithm is widely used in network routing protocols, most notably IS-IS (Intermediate System to Intermediate System to) and OSPF (Open Shortest Path First) [18].

The functionality of Dijkstra’s algorithm can be extended with a variety of modifications. For example, sometimes it is desirable to present solutions which are less than mathematically optimal. To obtain a ranked list of less-than-optimal solutions, the optimal solution is first calculated. A single edge appearing in the optimal solution is removed from the graph, and the optimum solution to this new graph is calculated. Each edge of the original solution is suppressed in turn and a new shortest-path in calculated. The secondary solutions are then ranked and presented after the first optimal solution [36], [46].

As Dijkstra’s algorithm is used to solve shortest path problems, it is possible to use this algorithm also in other network situation such as genetic networks [21], railroad transport on wide-area networks [36], sparse network [46]. To be precise in [4], a method for the study of cluster stability is purposed. Pairs of samples from the data, according to two sampling distributions are used. The first distribution corresponds to the high density zones of data-elements distribution. The second one, associated with the cluster margins, is related to the low density zones. The resemblance is measured by the total number of edges, in the clusters minimal spanning trees, connecting points from different samples.

3. The Methods. Our main contributions and their algorithms are given in this section.

3.1. Dijkstra’s algorithm with battery times. In this section, we extend the Dijkstra’s algorithm which has been defined by Dijkstra (1956). We name the new algorithm as Dijkstra’s algorithm with battery times.

Most ad hoc mobile devices operate on batteries. The remaining battery capacity limits the lifetime of hosts and with that the possibility to send, receive or relay data. There is also an important influence on the overall communication performance and the lifetime of the ad hoc mobile wireless net, because the lack of hosts can result in partitioning of the network. Herewith, power consumption becomes an important issue [41]. It is possible to give priority to battery life when using Dijkstra’s algorithm for routing with some of the changes made in pseudo code in
Table 2. The pseudo code of Dijkstra’s Algorithm with Battery Times

```
1 function DijkstraBatteryTime(Graph, source):
2    dist[source] := 0 // Distance from source to source
3    for each vertex v in Graph: // Initializations
4        if v ≠ source
5           dist[v] := infinity // Unknown distance function from source to v
6        previous[v] := undefined // Previous node in optimal path from source
7    end if
8    add v to Q // All nodes initially in Q (unvisited nodes)
9    end for
10   while Q is not empty: // The main loop
11      u := vertex in Q with min dist[u] // Source node in first case
12      remove u from Q
13      for each neighbor v of u: // where v has not yet been removed from Q.
14         alt := dist[u] + length(u, v)
15         battime:=dist[u]+length(u,v)
16         if alt < dist[v]: // A shorter path to v has been found
17            if battime<dist[v]: // A shorter path to v has been found
18               dist[v]:=battime
19               previous[v]:=u
20            end if
21         end if
22      end for
23    end while
24   return dist[], previous[]
25 end function
```

Table 2. The changes have been made mainly after line 10 and can be seen in Table 2.

3.2. Cooperative flow game algorithm. Now, we define cooperative flow game algorithm constructed from a flow network model. The pseudo code of cooperative flow game constructed from flow network model is given with by Table 3.

3.2.1. Example. Now, we give a simple example for better understanding.

Example: Consider a network with 4 nodes and these nodes act as routers. Node 1 is the source node and node 4 is the destination node. The others are the intermedi nodes. The aim of each node is to transmit the IP (Internet Protocol) packets to next and nearest node. Factors such as suitable bandwidth, the intensity of data traffic on the line, the transmission rate, the number of routers on the path, may constitute reasons for the selected nodes. In this example, the transmission time between nodes is chosen as the criterion.
Table 3. The pseudo code of cooperative flow game algorithm

```
1 function CooperativeFlowGame(Graph, source):
2    dist[source] := 0 // Distance from source to source
3  for each vertex v in Graph: // Initializations
4      if v ≠ source
5          dist[v] := infinity // Unknown distance function from source
to v
6      previous[v] := undefined // Previous node in optimal path
7          end if
8      add v to Q // All nodes initially in Q (unvisited nodes)
9    end for
10   subset[v]; // Calculate all the subset’s (coalitions) values
11   while Q is not empty: // The main loop
12      u := vertex in Q with min dist[u] // Source node in first case
13         remove u from Q
14     for each coalition vs of u: // where vs has not yet been
15        alt := dist[u] + length(u, vs)
16            if alt < dist[vs]: // A shorter path to vs has been found
17                dist[vs] := alt // Choose this coalition
18                previous[vs] := u
19            end if
20     end for
21   end while
22   return dist[], previous[]
23 end function
```

Let $\Gamma_f = (E, v)$ be a cooperative flow game with $N = \{1, 2, 3, 4\}$. The arc set is $E = \{(1, 2), (2, 4), (1, 4), (1, 3), (3, 4)\}$. When we model the game according to Figure 2, the value of the coalitions can be calculated as $c(\{1\}) = 6$, $c(\{1, 2\}) = 5$, $c(\{1, 3\}) = 7$, $c(\{1, 2, 3, 4\}) = 12$, and $c(\{\emptyset\}) = 0$ otherwise.

Player 1 and player 4 must be in the coalition because one of them is the source node and the other one is the destination node. Here, as a measure of the shortest path between two nodes, the data transmission time between nodes is given in Figure 2. The optimum path in this network with minimum cost is the $(1 \rightarrow 2 \rightarrow 4)$ way. For the first step, node 2 is chosen because its value is less than the one of the node 3. Going to the destination node with the value of node 2 and node 4 can be seen as an optimal path. The reason of not selecting the path $(1 \rightarrow 4)$ is that $3+2$ is less than $6$. Furthermore, path $(1 \rightarrow 3 \rightarrow 4)$ is not chosen because of $3+2 < 5+2$.

Then the marginal vectors are given in the following table, where $\sigma : N \rightarrow N$ is identified with $(\sigma(1), \sigma(2), \sigma(3), \sigma(4))$. Firstly, for $\sigma_0 = (1, 2, 3, 4)$ we calculate the marginal vectors as follows:

$m^{\sigma_0}_{1^1}(c) = c(\{1\}) = 0$,
$m^{\sigma_0}_{2^1}(c) = c(\{1, 2\}) - c(\{1\}) = 0$,
$m^{\sigma_0}_{3^1}(c) = c(\{1, 2, 3\}) - c(\{1, 2\}) = 0$, 

...
Table 4. The marginal vectors of the cooperative flow game

| \( \sigma \) | \( m_1^\sigma \) | \( m_2^\sigma \) | \( m_3^\sigma \) | \( m_4^\sigma \) | \( \sigma \) | \( m_1^\sigma \) | \( m_2^\sigma \) | \( m_3^\sigma \) | \( m_4^\sigma \) |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| \( \sigma_1 = (1, 2, 3, 4) \) | 0 | 0 | 0 | 12 | \( \sigma_{13} = (3, 1, 2, 4) \) | 0 | 0 | 0 | 12 |
| \( \sigma_2 = (1, 2, 4, 3) \) | 0 | 0 | 12 | 0 | \( \sigma_{14} = (3, 1, 4, 2) \) | 0 | 7 | 0 | 5 |
| \( \sigma_3 = (1, 3, 2, 4) \) | 0 | 0 | 0 | 12 | \( \sigma_{15} = (3, 2, 1, 4) \) | 0 | 0 | 0 | 12 |
| \( \sigma_4 = (1, 3, 4, 2) \) | 0 | 7 | 0 | 5 | \( \sigma_{16} = (3, 2, 4, 1) \) | 5 | 0 | 0 | 7 |
| \( \sigma_5 = (1, 4, 2, 3) \) | 0 | 0 | 12 | 0 | \( \sigma_{17} = (3, 4, 1, 2) \) | -1 | 7 | 0 | 6 |
| \( \sigma_6 = (1, 4, 3, 2) \) | 0 | 7 | 5 | 0 | \( \sigma_{18} = (3, 4, 2, 1) \) | 5 | 1 | 0 | 6 |
| \( \sigma_7 = (2, 1, 3, 4) \) | 0 | 0 | 0 | 12 | \( \sigma_{19} = (4, 1, 2, 3) \) | 0 | 0 | 12 | 0 |
| \( \sigma_8 = (2, 1, 4, 3) \) | 0 | 0 | 12 | 0 | \( \sigma_{20} = (4, 1, 3, 2) \) | 0 | 7 | 5 | 0 |
| \( \sigma_9 = (2, 3, 1, 4) \) | 0 | 0 | 0 | 12 | \( \sigma_{21} = (4, 2, 1, 3) \) | 0 | 0 | 12 | 0 |
| \( \sigma_{10} = (2, 3, 4, 1) \) | 5 | 0 | 0 | 7 | \( \sigma_{22} = (4, 2, 3, 1) \) | 5 | 0 | 7 | 0 |
| \( \sigma_{11} = (2, 4, 1, 3) \) | 0 | 0 | 12 | 0 | \( \sigma_{23} = (4, 3, 1, 2) \) | -1 | 7 | 6 | 0 |
| \( \sigma_{12} = (2, 4, 3, 1) \) | 5 | 0 | 7 | 0 | \( \sigma_{24} = (4, 3, 2, 1) \) | 5 | 1 | 6 | 0 |

\[ m_4^{\sigma_1}(c) = c(\{1, 2, 3, 4\}) - c(\{1, 2, 3\}) = 12. \]

The other marginal vectors can be calculated similarly, which is shown in Table 4.

Table 4. The marginal vectors of the cooperative flow game.

\[ \Phi(c) = (1.1667, 1.8333, 4.5, 4.5). \]

The Shapley value shows that the selection of path \((1 \rightarrow 2 \rightarrow 4)\) is reasonable for collaborative selection.

To find this path, we use Dijkstra’s algorithm and cooperative flow game algorithm constructed from flow network model. The path \((1 \rightarrow 2 \rightarrow 4)\) is seen as the optimal choice.

4. Simulation. Let us consider three ad hoc networks consisting of 10, 15 and 20 nodes, respectively. In each ad hoc network, there is one source node, one destination node and the other nodes are inter-nodes. The aim of these ad hoc networks is to find the shortest path. To determine the shortest path between source node and destination node, Dijkstra’s algorithm (DA), Dijkstra’s algorithm
with battery times (DWBT) and cooperative flow game algorithm (CFGA) methods are used. The time performances of the methods are compared. Additionally, three networks which consist of 10, 15 and 20 nodes are created and the transfer time of transmission between nodes is determined randomly.

In this section, some simulation results including different transmissions areas and made with different number of nodes are assessed. The number of nodes can be varied as 10, 15, 20, and transmissions gap of nodes is selected as 3, 4, 5 in the analysis. The packet size was fixed at 64 byte and the simulation time is fixed at 180 seconds. Moreover, the bandwidth of network is fixed at 1 Mbps and the network is assumed to be unobstructed. With the successful conclusion of the application, to identify an optimum routing behaviour model, DA, DWBT, CFGA are designed and modeled in the network. In the simulation results, using all three algorithms, the same optimal behaviour is obtained, but in different times. In Figure 3, the time performances of DA, DWBT and CFGA methods in the network with 10, 15, 20 nodes can be seen.

![Figure 3. Time Performances of three algorithms](image)

In the 10-node network, the shortest path is found in 2.8 seconds with DA, in 3.3 seconds with DWBT and in 1.5 seconds with CFGA. In the 15-node network, the shortest path is found in 3.1 seconds with DA, in 3.7 seconds with DWBT and in 1.7 seconds with CFGA. In the 20-node network, the shortest path is found in 3.7 seconds with DA, in 4.5 seconds with DWBT and in 2.1 seconds with CFGA. When the number of nodes increases, the time to find the shortest path also increases. As
can be seen from the values, in three different networks, CFGA is the most efficient one. Further, DWBT is slower than the others in finding the shortest way.

In the other simulation, we try to show the necessity of the consideration of an ad hoc device battery life while finding the shortest path. The networks with 10, 15 and 20 nodes are used again. Values between nodes are taken as the battery times. During 180 seconds after the simulation has started, the lifetimes of nodes in the network are shown in Figure 4.

Figure 4. The lifetime of the nodes

Whenever the nodes die (when its battery power gets completely drained) during the course of a simulation, our simulation code would check whether the graph become partitioned. To avoid the termination of a simulation due to the battery power exhaustion at source or destination nodes, all source and sink nodes are configured to have “infinite” power resources. All the other ‘intermediate’ nodes are configured with identical initial battery power levels.

In the 10-node network after 180 seconds in simulation, the shortest path is found by DA with 4 nodes, by DWBT with 3 nodes and by CFGA with 2 nodes which fall from the network. In the 15-node network, the shortest path is found by DA with 5 nodes, by DWBT with 4 nodes, by CFGA with 2 nodes which fall from the network. In the 20-node network, while the shortest path is found by DA with 7 nodes, by DWBT with 6 nodes, by CFGA with 4 nodes which fall from the network.

When we include the battery times to DA, the time to find the optimal path takes longer. However, the remaining life of DWBC is longer than the one of DA owing to the number of nodes. As in the time performances, CFGA gives optimal results on the remaining nodes in the networks. When the nodes cooperate with each other in order to work for longer, less nodes fall from the network by using DA and DWBT.

In Table 5, a brief comparision of three methods in each ad hoc network is provided.

For example, \( CFGA < DA \) means that \( CFGA \) is faster than \( DA \). And \( CFGA > DWBT \) means that \( CFGA \) nodes is more than \( DWBT \) nodes.
Table 5. The comparison of three methods

|                      | Time performances | Lifetime of nodes |
|----------------------|-------------------|-------------------|
|                      | $CFG A < DA < DWBT$ | $CFG A > DWBT > DA$ |

5. **Conclusion and outlook.** In this study, we try to show the performances of three different methods in an ad hoc network within the shortest path problem. To this end, Dijkstra’s algorithm, Dijkstra’s algorithm with battery times and cooperative game algorithm constructed from cooperative flow network model is generated and tested for time and network performance. Finally, as a result of this study, cooperative game constructed from cooperative flow network model is found as the best optimal solution.

One of the contributions of this study occurs by modifying the Dijkstra’s algorithm in order to use this algorithm in ad hoc networks. The battery time of the devices in ad hoc networks is important for the continuation of the network. Because, transmitting the IP packets to the node which locates in the routing table is required to complete the routing. If the packet cannot find the node, the packet is lost and the routing is failed. This is likely to occur because of using wireless devices in ad hoc networks.

But when such a scenario considered, Dijkstra’s algorithm which uses one single metric as a cost between nodes is insufficient, although Dijkstra’s algorithm is helpful for inferring the underlying interaction of nodes in order to gain an insight about the pathway structure in a network based on the maximum interaction of nodes. Thanks to Dijkstra’s algorithm with battery times which we introduce by extending the general Dijkstra’s algorithm and cooperative flow game algorithm constructed from flow network model, the nodes can stay longer in the networks. Furthermore, the algorithms can find the shortest path in a short time compared to the Dijkstra’s algorithm.

The interrelation between Operations Research and cooperative game theory is a widely studied research area and can be summarised under the heading of Operational Research games. In practical situations, a number of important cooperative games arise from Operational Research situations. One can say that an important part of the interplay between cooperative games and Operations Research situations occurs by using the basic structure of a graph, network or a system that underlies various types of combinatorial optimization problems. Minimum spanning tree games, fixed tree games, flow games, assignment games, transportation games, etc., are under the area of Operational Research games. If one assumes that at least two players are located in the control parts (e.g., vertices, edges, resource bundles, jobs) of the underlying system, then a cooperative game can be constructed. Operations Research situation analyse cases in which one decision maker, guided by an objective function, faces an optimization problem. Cooperative game theory concentrates on the question of how to act in an optimal way and, in particular, on the issues of computational complexity and the design of efficient algorithms.

The readers’ attention is drawn to the fact that economic situations and Operations Research applications provide the analytical tools of optimization, systems theory and decision making through which significant present challenges are investigated and solutions are offered in cooperative game theory.
As a future work, this study can be applied to the other optimization situations such as minimum-cost spanning trees, transportation problems, maximum-flow augmenting paths, postman and related arc routing problems, the traveling salesman and related vertex routing problems, location problems, etc. in the framework of cooperative game theory. Furthermore, the simulation results this study shows that the routing algorithms can be compared with each other and the other cooperative games. These routing algorithms can be Bellman-Ford algorithm, Flooding algorithm as static routing and distance vector algorithm, link state algorithm as dynamic routing in future studies. Also, fuzzy routing, hierarchical routing, heuristic routing and geographic routing situations can be modelled with cooperative game theory, too. It is envisaged that using cooperative game theory will bring a new approach to network traffic engineering.

REFERENCES

[1] R. Ahuja, T. Magnanti and J. Orlin, Network Flows: Theory, Algorithms, and Applications, PrenticeHall, Upper Saddle River, NJ, 1993.
[2] R. K. Ahuja, K. Mehlhorn, J. Orlin and R. E. Tarjan, Faster algorithms for the shortest path problem, Journal of the ACM (JACM), 37 (1990), 213–223.
[3] Y. Bachrach and E. Porat, Path disruption games, In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, volume 1-Volume 1 (pp. 1123–1130). International Foundation for Autonomous Agents and Multiagent Systems, (2010, May).
[4] Z. Barzily, Z. Volkovich, B. Akteke-Öztürk and G. W. Weber, On a minimal spanning tree approach in the cluster validation problem, Informatica, 20 (2009), 187–202.
[5] R. Bellman, On a Routing Problem (No. RAND-P-1000), Rand Corp Santa Monica Ca, 1956.
[6] P. Borm, H. Hamers and R. Hendrickx, Operations research games: A survey, TOP, 9 (2001), 139–199.
[7] R. Branzei, D. Dimitrov and S. Tijs, Models in Cooperative Game Theory: Crisp, Fuzzy And Multi-Choice Games, In: Lecture notes in economics and mathematical systems, Springer, Berlin, vol 556, 2005.
[8] T. S. Chandrashekar and Y. Narahari, Economic mechanisms for shortest path cooperative games with incomplete information, In Internet and Network Economics, Springer Berlin Heidelberg, (2005), 70–79.
[9] J. H. Chang and L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, In INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings, IEEE, 1 (2000), 22–31.
[10] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks, MobiCom ’01 Proceedings of the 7th Annual International Conference on Mobile Computing and Networking, (2001), 85–96.
[11] S. M. Choi, X. Huang and W. K. Ching, Minimizing equilibrium expected sojourn time via performance-based mixed threshold demand allocation in a multiple-server queueing environment, Journal of Industrial and Management Optimization, 8 (2012), 299–323.
[12] R. B. Dial, Algorithm 360: Shortest-path forest with topological ordering [H], Communications of the ACM, 12 (1969), 632–633.
[13] E. W. Dijkstra, A note on two problems in connection with graph, Numer. Math., 1 (1959), 269–271.
[14] T. S. Driessen, A survey of consistency properties in cooperative game theory, SIAM review, 33 (1991), 43–59.
[15] J. Feigenbaum, C. Papadimitriou, R. Sami and S. Shenker, A BGP-based mechanism for lowest-cost routing, PODC ’02 Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing, (2002), 173–182.
[16] L. R. Ford and D. R. Fulkerson, Flows in Networks, Princeton University Press, Princeton, N.J., 1962.
[17] V. Fragnelli, I. García-Jurado and L. Méndez-Naya, On shortest path games, Mathematical Methods of Operations Research, 52 (2000), 251–264.
[18] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *Journal of the ACM (JACM)*, 34 (1987), 596–615.

[19] H. N. Gabow and R. E. Tarjan, Faster scaling algorithms for network problems, *SIAM Journal on Computing*, 18 (1989), 1013–1036.

[20] G. Gallo and S. Pallottino, Shortest path algorithms, *Annals of Operations Research*, 13 (1988), 1–79.

[21] J. Gebert, M. Lätsch, E. M. P. Quek and G. W. Weber, Analyzing and optimizing genetic network structure via path-finding *Journal of Computational Technologies*, 9 (2004).

[22] A. V. Goldberg, Scaling algorithms for the shortest path problem, *SIAM Journal on Computing*, 24 (1995), 494–504.

[23] J. Hershberger, S. Suri and V. Prices, Shortest Paths: What is an edge worth?, *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, IEEE Computer Soc., Los Alamitos, CA, (2001), 252–259.

[24] E. Kalai and E. Zemel, Generalized network problems yielding totally balanced games, *Operations Research*, 30 (1982), 998–1008.

[25] E. Kalai and E. Zemel, Totally balanced games and games of flow, *Mathematics of Operations Research*, 7 (1982), 476–478.

[26] J. Leino, *Applications of Game Theory in Ad Hoc Networks*, Master’s Thesis, Department of Engineering Physics and Mathematics, Helsinki University of Technology, (2003).

[27] W. Liang, Constructing minimum-energy broadcast trees in wireless ad hoc networks, In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, (2002), 112–122.

[28] A. B. MacKenzie and L. A. DaSilva, Game theory for wireless engineers, *Synthesis Lectures on Communications*, 1 (2006), 1–86.

[29] N. Nisan and A. Ronen, Algorithmic mechanism design, *Annual ACM Symposium on Theory of Computing (Atlanta, GA, 1999)*, ACM, New York, (1999), 129–140.

[30] R. C. Prim, Shortest connection networks and some generalizations, *Bell System Technical Journal*, 36 (1957), 1389–1401.

[31] H. Reijnierse, M. Maschler, J. Potters and S. Tijs, Simple flow games, *Games and Economic Behavior*, 16 (1996), 238–260.

[32] S. Srivastava, J. Neel, A. B. MacKenzie, R. Menon, L. A. DaSilva, E. H. Hick, J. H. Reed and R. P. Gilles, Using game theory to analyze wireless ad hoc networks, *IEEE Communications Surveys and Tutorials*, 7 (2005), 46–56.
[45] M. Voorneveld and S. Grahn, Cost allocation in shortest path games, Mathematical methods of operations research, 56 (2002), 323–340.

[46] M. H. Xu, Y. Q. Liu, Q. L. Huang, Y. X. Zhang and G. F. Luan, An improved Dijkstra’s shortest path algorithm for sparse network, Applied Mathematics and Computation, 185 (2007), 247–254.

[47] Y. Zhao, S. Jin and W. Yue, Adjustable admission control with threshold in centralized CR networks: Analysis and optimization, Journal of Industrial & Management Optimization, 11 (2015), 1393–1408.

[48] L. Zhou, A new bargaining set of an n-person game and endogenous coalition formation, Games and Economic Behavior, 6 (1994), 512–526.

Received May 2016; revised March 2018.

E-mail address: serapbakioglu@sdu.edu.tr
E-mail address: sirmagok@sdu.edu.tr
E-mail address: tuncayaydogan@sdu.edu.tr
E-mail address: gerhard.weber@put.poznan.pl