# A Cooperative Game-Theoretic Approach to the Social Ridesharing Problem

<table>
<thead>
<tr>
<th>Títol</th>
<th>A Cooperative Game-Theoretic Approach to the Social Ridesharing Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publication Type</td>
<td>Journal Article</td>
</tr>
<tr>
<td>Year of Publication</td>
<td>2017</td>
</tr>
<tr>
<td>Authors</td>
<td>Bistaffa F [1], Farinelli A [2], Chalkiadakis G [3], Ramchurn SD [4]</td>
</tr>
<tr>
<td>Journal</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>Volume</td>
<td>246</td>
</tr>
<tr>
<td>Paginació</td>
<td>86-117</td>
</tr>
<tr>
<td>Date Published</td>
<td>05/2017</td>
</tr>
<tr>
<td>Editor</td>
<td>Elsevier</td>
</tr>
</tbody>
</table>
In this work, we adopt a cooperative game theoretic approach in order to tackle the social ridesharing (SR) problem, where a set of commuters, connected through a social network, form coalitions and arrange one-time rides at short notice. In particular, we address two fundamental aspects of this problem. First, we focus on the optimisation problem of forming the travellers' coalitions that minimise the travel cost of the overall system. To this end, we model the formation problem as a Graph-Constrained Coalition Formation (GCCF) one, where the set of feasible coalitions is restricted by a graph (i.e., the social network). Our approach allows users to specify both spatial and temporal preferences for the trips. Second, we tackle the payment allocation aspect of SR, by proposing the first approach that computes kernel-stable payments for systems with thousands of agents. We conduct a systematic empirical evaluation that uses real-world datasets (i.e., GeoLife and Twitter). We are able to compute optimal solutions for medium-sized systems (i.e., with 100 agents), and high quality solutions for very large systems (i.e., up to 2000 agents). Our results show that our approach improves the social welfare (i.e., reduces travel costs) by up to 36.22% with respect to the scenario with no ridesharing. Finally, our payment allocation method computes kernel-stable payments for 2000 agents in less than an hour while the state of the art is able to compute payments only for up to 100 agents, and does so 84 times slower than our approach.