Journal of Artificial Intelligence Research 56 (2016) 573–611 Submitted 11/15; published 8/16

A Study of Proxies for Shapley Allocations of Transport Costs

Haris Aziz HARIS.AZIZ@DATA61.CSIRO.AU
Data61/CSIRO and University of New South Wales (UNSW),
Sydney, Australia

Casey Cahan CCAH002@AUCKLANDUNI.AC.NZ
University of Auckland,
Auckland, New Zealand

Charles Gretton CHARLES@HIVERY.COM
Hivery,
Sydney, Australia; and
Australian National University (ANU),
Canberra, Australia; and
Griffith University,
Gold Coast, Australia

Philip Kilby PHILIP.KILBY@DATA61.CSIRO.AU
Data61/CSIRO and Australian National University (ANU),
Canberra, Australia

Nicholas Mattei NICHOLAS.MATTEI@DATA61.CSIRO.AU
Data61/CSIRO and University of New South Wales (UNSW),
Sydney, Australia

Toby Walsh TOBY.WALSH@DATA61.CSIRO.AU
Data61/CSIRO and University of New South Wales (UNSW),
Sydney, Australia

Abstract

We survey existing rules of thumb, propose novel methods, and comprehensively evaluate a number of solutions to the problem of calculating the cost to serve each location in a single-vehicle transport setting. Cost to serve analysis has applications both strategically and operationally in transportation settings. The problem is formally modeled as the *traveling salesperson game* (TSG), a cooperative transferable utility game in which agents correspond to locations in a traveling salesperson problem (TSP). The total cost to serve all locations in the TSP is the length of an optimal tour. An allocation divides the total cost among individual locations, thus providing the cost to serve each of them. As one of the most important normative division schemes in cooperative games, the Shapley value gives a principled and fair allocation for a broad variety of games including the TSG. We consider a number of direct and sampling-based procedures for calculating the Shapley value, and prove that approximating the Shapley value of the TSG within a constant factor is NP-hard. Treating the Shapley value as an ideal baseline allocation, we survey six proxies for it that are each relatively easy to compute. Some of these proxies are rules of thumb and some are procedures international delivery companies use(d) as cost allocation methods. We perform an experimental evaluation using synthetic Euclidean games as well as games derived from real-world tours calculated for scenarios involving fast-moving goods; where deliveries are made on a road network every day. We explore several computationally tractable allocation techniques that are good proxies for the Shapley value in problem instances of a size and complexity that is commercially relevant.

1. Introduction

We study transport scenarios where deliveries of consumer goods are made from a depot to locations on a road network. At each location there is a customer, e.g., a vending machine or shop, that has requested some goods, e.g. soda, milk, or crisps. The vendor who plans and implements deliveries is faced with two vexing problems. The first difficult hurdle is solving the combinatorial optimization problem of routing and scheduling vehicles to deliver goods in a cost-effective manner. Many varieties of this first problem exist (Golden, Raghavan, & Wasil, 2008), and for our proposes we shall refer to it as the *vehicle routing problem* (VRP). We begin our investigation supposing that VRP has been solved heuristically, and therefore the assignment of locations to routes (and delivery vehicles) has been made.

The second vexing problem is determining how to evaluate the *cost to serve* each location. Specifically, the vendor must decide how to apportion the costs of transportation to each location in an equitable manner. The results of the cost to serve analysis have a variety of important applications. Using the allocation directly the vendor can of course charge locations their allocated portion of the transportation costs. More realistically, vendors use the cost allocations when (re)negotiating contracts with customers; extracting higher per-unit delivery prices from their most expensive customers. Supply chain managers also reference cost allocations when deciding whether or not to include/continue trade with a particular location. Techniques informed by cost allocations in planning for a profitable transport business were recently reviewed by Özener, Ergun, and Savelsbergh (2013). Finally, provided market conditions are favourable, sales managers can be instructed to acquire new customers in territories where existing cost allocations are relatively high in order to share the cost of delivery among more locations.

Addressing the second vexing problem, this paper stems from our work with a fast-moving consumer goods company that operates nationally both in Australia and New Zealand. The company serves nearly 20,000 locations weekly using a fleet of 600 vehicles. Our industry partner is under increasing economic pressure to realise productivity improvements through optimisation of their logistical operations. A key aspect of that endeavour is to understand the contribution of each location to the overall cost of distribution. In this study, we focus at the individual route level for a single truck, where we apportion the costs of the deliveries on that route to the constituent locations. We formalise this setting as a *traveling salesperson game* (TSG) (Potters, Curiel, & Tijs, 1992), where the cost to serve all locations is given by the solution to an underlying *traveling salesperson problem* (TSP). Once formalised as a game, we can use principled solution concepts from cooperative game theory, in particular the Shapley value (Shapley, 1953), in order to allocate costs to locations in a fair and economically efficient manner. The unique axiomatic properties of the Shapley value are enticing to our industry partner, as the allocation is fair in a reasonable and comprehensible sense. Charging customers in a fair manner provides strong justification for delivery prices and encourages trust between the operator and customer.

Calculating the Shapley value of a game is a notoriously hard problem (Chalkiadakis, Elkind, & Wooldridge, 2011). A direct calculation of the Shapley value for a TSG requires the computation of optimal solutions to exponentially many distinct instances of the TSP. Sampling procedures can be used for approximating the value, however these too do not offer a practical solution for larger games. Moreover, we prove that there is no polynomial-time α-approximation of the Shapley value for any constant $\alpha \geq 1$ unless P = NP. In order to be practically applicable, we must be able to calculate a cost allocation for each location on a route, for over 600 unique routes, that may change

daily or weekly as customers change their order volumes. Hence, we need methods that return values within minutes, not hours. Allocations are also used to heuristically evaluate assignments of locations to trucks in a larger VRP. In such a setting we must be able to estimate cost allocations within seconds or fractions of a second, not minutes.

To circumvent the computational difficulties of calculating Shapley values, this work explores six proxies[1] for the Shapley value. We investigates three simple *rules of thumb*, including a simple distance measure that we have seen employed in various industrial engagements. We include these to analyze how good these proxies are relative to the Shapley value, the stated ideal cost allocation rule. Other proxies we develop offer tractable alternatives to the Shapley value, and in some cases appeal to other allocation concepts from cooperative game theory (Peleg & Sudhölter, 2007; Curiel, 2008). Two of our proxies appeal to the well-known *Held-Karp* (Held & Karp, 1962) and *Christofides* (Christofides, 1976) TSP heuristics, respectively.

We report a detailed experimental comparison of proxies using a large corpus of synthetic Euclidean games, and problems derived from real-world tours calculated for fast-moving consumer goods businesses in the cities of Auckland (New Zealand), Canberra, and Sydney (Australia). Our experimentation uncovers a novel computationally cheap proxy that gives good approximations of the Shapley value. Our evaluation also considers the ranking of locations—least to most costly—induced by the Shapley and proxy values. Ranking locations is a common request from our industrial partner and is relevant when, for example, we are interested in identifying the most costly locations to serve. We find that two proxies, one of which is our novel proxy, provide good ranking accuracy with respect to the rank induced by the Shapley value.

2. Preliminaries

We use the framework of cooperative game theory to gain a deeper understanding of our delivery and cost allocation problems (Peleg & Sudhölter, 2007; Chalkiadakis et al., 2011). In cooperative game theory, a game is a pair (N,c) where N is the set of agents of size $|N| = n$ and the second term $c : 2^N \to \mathbb{R}$ is the *characteristic function*. Taking $S \subseteq N$, $c(S)$ is the cost of subset S. A *cost allocation* is a vector $x = (x_1, \ldots, x_n)$ denoting that cost x_i is allocated to agent $i \in n$. We restrict our attention to *economically efficient* cost allocations, which are allocations satisfying $\sum_{i \in n} x_i = c(N)$ – i.e. the sum of allocated costs is equal to the cost of serving the grand coalition.

For any cooperative game (N,c), a *solution concept* ϕ assigns to each agent $i \in N$ the cost $\phi_i(N,c)$. There may be more than one allocation satisfying the properties of a particular solution concept, thus ϕ is not necessarily single-valued, and might give a set of cost allocations (Peleg & Sudhölter, 2007). We sometimes omit (N,c) from our notation of ϕ and other solution concepts when the context is clear. A minimal requirement of a solution concept is *anonymity*, meaning that the cost allocation must not depend on the identities of locations. Prominent solution concepts include the *core*, *least core*, and the *Shapley value*. For $\varepsilon \geq 0$, we say that cost allocation ϕ is in the (multiplicative) ε*-core* if $\sum_{i \in S} \phi_i(N,c) \leq (1+\varepsilon)c(S)$ for all $S \subseteq N$ (Faigle & Kern, 1993). The 0-core is referred to simply as the *core*. Both the core and ε-core can be empty. The ε-core which is

1. We use the word *proxy* instead of *approximation* to ease discussion and, technically, many of these measures are stand-ins for the Shapley value, not approximations of it; i.e., they do not give a guarantee of a quantitatively provable approximation.

non-empty for the smallest possible ε is called the least core. This particular ε is referred to as the *least core value*.[2]

Our work focuses on the single-valued solution concept called the *Shapley value* (Shapley, 1953). Writing $SV_i(N,c)$ for the Shapley value of agent i, formally we have:

$$SV_i(N,c) = \sum_{S \subset N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!}(c(S \cup \{i\}) - c(S)). \quad (1)$$

In other words, the Shapley value divides costs based on the marginal cost contributions of agents.

In the *traveling salesperson problem (TSP)* a salesperson must visit a set of locations $N = \{1,\ldots,n\} \cup \{0\}$ starting and ending at a special *depot* location 0. For $i, j \in N \cup \{0\}$ $i \neq j$, d_{ij} is the strictly positive distance traversed when traveling from location i to j. Here, $d_{ij} = \infty$ if traveling directly from i to j is impossible. Taking distinct $i, j, k \in N \cup \{0\}$, the problem is *symmetric* if and only if $d_{ij} = d_{ji}$ for all $i, j \in N \cup \{0\}$. It satisfies the *triangle inequality* if and only if $d_{ij} + d_{jk} \geq d_{ik}$ (Garey & Johnson, 1979).

A TSP is *Euclidean* when each location is given by coordinates in a (two dimensional) Euclidean space; therefore d_{ij} is the Euclidean distance between i and j. A Euclidean TSP is both symmetric and satisfies the triangle inequality.

A *tour* is given by a finite sequence of locations that starts and ends at the depot 0. The *length* of a tour is the sum of distances between consecutive locations. For example, the length of $[0,1,2,0]$ is $d_{01} + d_{12} + d_{20}$. An optimal solution to a TSP is a minimum length tour that visits every location. It is NP-hard to find an optimal tour, and generally there is no polynomial-time α-approximation for any α unless P = NP (Sahni & Gonzalez, 1976). An α-approximation for a given optimisation problem is an algorithm that runs on an instance x and returns a feasible solution $F(x)$ which has cost $c(F(x))$ related to the optimal solution $OPT(x)$ as follows (Papadimitriou, 1994):

$$\frac{c(F(x))}{c(OPT(x))} \leq \alpha.$$

Informally, α is a bound on the relative error of an approximation function. When $\forall i, j$ d_{ij} are finite, the triangle inequality, and symmetry hold, then polynomial-time approximations exist for the TSP problem (Held & Karp, 1962; Christofides, 1976).

Given a TSP, the corresponding *traveling salesperson game* (TSG) is a pair (N,c). N is the set of agents which corresponds to the set of locations.[3] The second term $c : 2^N \to \mathbb{R}$ is the characteristic function. Taking $S \subseteq N$, $c(S)$ is the length of the shortest tour of all the locations in S. A *cost allocation* is a vector $x = (x_1,\ldots,x_n)$ denoting that cost x_i is allocated to location $i \in N$. For the special depot location, we shall always take $x_0 = 0$ (Potters et al., 1992). Typically, the depot is operated by the agent who is distributing the costs and does not want to incur costs himself. Hence, we refer to n as the *number of locations*, while the corresponding TSP has $n+1$ points.

2. The 0-core of the transport game we focus on in this work can be empty. However, if the game is convex, the Shapley value lies in the core (Tamir, 1989).

3. From here on we focus on a restriction of general games to delivery games (TSGs) and therefore we use *location* instead of *agent* for ease of exposition.

3. Useful Properties of the Shapley Value

When discussing cost allocations with industrial partners, the concept of fairness is often of primary concern. A fair and principled cost allocation scheme would allow them to explain charges to customers in an objective way; making the whole process more transparent. The Shapley value for general games is the unique assignment of costs that satisfies three natural axioms: (1) *anonymity*, the cost allocated to a particular location depends only on the impact visiting the locations has on the total cost; (2) *efficiency*, the entire cost of serving all N locations is allocated; and (3) *strong monotonicity* (Young, 1985), given two games (N,c) and (N,c'), $\forall S : c^i(S) \geq c'^i(S) \implies \phi_i(N,c) \geq \phi_i(N,c')$; where the *marginal contribution* $c^i(S)$ from player i to the total cost of coalition S is:

$$c^i(S) = \begin{cases} c(S) - c(S \setminus \{i\}) & \text{if } i \in S \\ c(S \cup \{i\}) - c(S) & \text{if } i \notin S. \end{cases}$$

Due to these and other derivative axiomatic properties, the Shapley value has been termed "*the most important normative payoff division scheme*" in cooperative game theory (Winter, 2002). These axioms alone make the Shapley value attractive in a cost allocation setting.

The Shapley value has additional attractive properties in terms of existence and computability when used as a cost allocation scheme. For example, whereas the 0-core can be empty, and therefore not yield any allocation at all (Tamir, 1989), the Shapley value always exists in the TSG setting. In logistics, there is often some fixed cost associated with serving a particular location, e.g., special parking or permitting. If we treat a variant of the TSG where some locations have an associated fixed cost in addition to their transportation costs— e.g. parking and loading fees —then the Shapley value will allocate those fixed costs to *only* the associated locations. Formally, given a fixed cost $f(i)$ of serving location i, $f(i)$ does not need to be removed before computing the Shapley value, as follows. Suppose c is the characteristic function of the TSG defined above, and c' satisfies the identity $c'(S) = c(S) + \sum_{i \in S} f(i)$. Then, by the additivity propertity of the Shapley value (Shapley, 1953) we have $SV_i(N,c') = SV_i(N,c) + f(i)$.

For delivery settings, an additional observation is that charging locations according to the Shapley value may incentivise them to *recruit* new customers in their vicinity. Locations that recruit nearby locations for a vendor can reasonably expect to lower the transportation costs they are allocated. In detail, consider a vendor serving locations $N = \{1, \ldots, n\}$. From the vendor's perspective, adding a new location, $n+1$, to an existing delivery route is clearly a good idea if the revenue generated by delivering to that location is greater than the marginal cost $c(N \cup \{n+1\}) - c(N)$ of the new delivery. Because existing locations in the vicinity of $n+1$ are already paying for deliveries, charging the additional customer the marginal quantity $c(N \cup \{n+1\}) - c(N)$ will typically be unfair. In this case, existing customers would likely be subsidizing new customers, and therefore disincentivised from finding new business for the vendor. The Shapley value mitigates this, and can be expected to provide recruitment incentives. Making this discussion more concrete, suppose the game is Euclidean with $N = \{x\}$ a single agent at distance 100 from the depot and the new agent y is at distance 5 from x. The transportation cost of serving $\{x,y\}$ can be as high as 210. Clearly, charging the new agent at most $c(\{x,y\}) - c(\{x\}) = 10$ while x continues to pay around 200 is unfair. On the other hand, if the vendor allocates costs according to the Shapley value, the existing customer's costs *decrease* when the new agent joins.

Another possible benefit in delivery settings is that, if the characteristic function is concave then the Shapley value lies in the non-empty 0-core. Formally, concavity is satisfied if for all $i \in N, S \subset$

$T \subseteq N \setminus \{i\} : c(S \cup \{i\}) - c(S) \geq c(T \cup \{i\}) - c(T)$. Charging customers according to Shapley/core values actually guarantees that they are incentivised to recruit. Specifically, for all $i \in N : SV_i(N \cup \{n+1\}, c) < SV_i(N, c)$. In other words, the Shapley allocation of costs to existing locations decreases when a new customer $n+1$ is added. Unfortunately general TSGs do not necessarily have concave characteristic functions. However, as long as the cost function is "roughly" concave is all that is necessary for existing locations to realise savings. In practice there are synergies, and incentives for further recruitment on routes where we charge according to the Shapley value. In our empirical data, even when the game is not concave we frequently observe such incentives given a Shapley allocation. And compared to charging customers according to their marginal contribution to costs, we do not explicitly disincentivise recruitment. Summarizing, if an agent knows that all locations are charged according to the Shapley value, they can typically expect incentives to recruit new locations in their vicinity.

4. Computing the Shapley Value

Our focus now shifts to computing the Shapley value. Considering games in general, it should be noted that a direct evaluation of Equation 1 requires that we sum over exponentially many quantities. Such a direct approach to the calculation of the Shapley value is therefore not practical for any game of a reasonable size. Indeed, starting from Mann and Shapley (1962), authors motivate auxiliary restrictions and constraints, for example on the size and importance of coalitions, in order to describe games where the Shapley value can be calculated. More recent literature proposes a variety of approaches to directly calculate the Shapley value for certain games (Conitzer & Sandholm, 2006; Ieong & Shoham, 2005), however efficient calculation of the value for TSGs has remained elusive. We require an accurate baseline in order to experimentally evaluate the proxies we later develop for the Shapley value of the TSG. To this purpose we investigate exact and general sampling-based approximations of the Shapley value. We treat our transport setting specifically, describing a novel procedure for an exact evaluation of the Shapley value of a TSG by following Bellman's dynamic programming solution to the underlying TSP. We also discuss how in general the Shapley value can be evaluated approximately using a sampling procedure. We study this sampling approach in TSGs using two distinct characterisations of the Shapley value which are amenable to sampling-based evaluation. We perform a detailed empirical study of sampling-based evaluations using Synthetic TSG instances where the underlying TSP model is Euclidean. In closing we give a hardness proof relating to the computation of the Shapley value of TSGs, showing that approximation of the Shapley value in TSGs is intractable.

4.1 Dynamic Programming

We found that the steps performed by a *dynamic programming* (DP) algorithm for the underlying TSP expose the margins—i.e. terms of the form $c(S \cup \{i\}) - c(S)$—that are summed over in a direct evaluation of Equation 1. The Shapley value of a TSG can therefore be computed as a side effect while a DP procedure computes the optimal solution to the underlying TSP. This procedure is formally captured in Algorithm 1: DP-TSP-Shapley. The algorithm as written assumes that distance costs are symmetric and that location 0 is a special depot location, both of these assumptions can be relaxed for the more general case of simply computing Shapley values leveraging dynamic programming.

Algorithm 1 DP-TSP-Shapley

Input: $N = \{1,\ldots,n\} \cup \{0\}$ locations with d_{ij} the cost to travel from i to j.
Output: List SV with SV_i for all $i \in N$.

```
// c(S, j) is the length of the shortest path starting at 0, through all locations in S, and ending at j.
c ← []
// T(S) is the length of the shortest tour of all locations in S starting and ending at location 0.
T ← []
SV ← []
for i ∈ {1,...,|N|} do
    c({0,i},i) ← d_{0,i}
    T({i}) ← 2 · d_{0i}
    SV_i ← ((|N|−1)!/|N|!) · T({i})
end for
for s ∈ {2,...,|N|} do
    for each S which is a subset of N of size s do
        S_DEPOT ← S ∪ {0}
        for j ∈ S do
            c(S_DEPOT, j) ← min_{∀i∈S,i≠j} c(S_DEPOT \ {j}, i) + d_ij
        end for
        T(S) ← min_{j∈S} c(S_DEPOT, j) + d_{j0}
        for i ∈ S do
            SV_i ← (|S−1|!(|N|−|S−1|−1)!/|N|!) · (T(S) − T(S \ {i}))
        end for
    end for
end for
```

These ideas can be made concrete by following the procedure outlined by Bellman (1962). The equations at the heart of that TSP solution procedure recursively define a cost function, $c(S, j)$, which is the shortest path through all locations in S starting at the depot 0 and ending at j.[4]

$$\begin{aligned} c(\{j\}, j) &= d_{0j}. \\ c(S, j) &= \min_{k \in S, k \neq j} (c(S \setminus \{j\}, k) + d_{kj}). \end{aligned}$$

Following the above recursive definition, a DP process iteratively tabulates $c(S, j)$ for successively larger coalitions S. At each iteration of subset size $|S| < |N|$ the procedure tabulates all quantities $c(S, j)$ taking $|S| = n$. By computing the values $c(S,0)$ for $|S| < |N|$, we have access to the characteristic function evaluation $c(S)$ of subtours of locations in S, as follows:

$$c(S) = c(S,0) = \min_{j \in S}(c(S, j) + d_{j0}).$$

Therefore, one can incrementally evaluate the sum in Equation 1 for a TSG, while calculating optimal subtours for progressively larger coalitions (supersets) within a classical DP procedure. Intuitively, as we compute a tour using Bellman's algorithm, by additionally evaluating $c(S,0)$ for each

4. Our notations depart slightly from Bellman's seminal work. Whereas we take $c(S, j)$ to be the cost of each optimal tour-prefix path (i.e. starting at the depot 0 and ending at j), Bellman originally took $c(S, j)$ to be the cost of optimal tour-suffix paths starting from j, traversing the locations in S and ending at the depot 0.

encountered subset S we obtain all quantities required to calculate the marginal costs of locations. We have therefore highlighted a concrete relationship between a classical procedure for the TSP and the Shapley value of the corresponding TSG. The dynamic programming algorithm is very fast up to about 18 locations, where the size of the table and the number of subsets become unmanageable.

4.2 Computational Complexity

We now consider, for the most general setting of the TSG, the complexity of calculating the Shapley value. Below we prove that the Shapley value of a location in the TSG cannot be approximated within a constant factor in polynomial-time unless P = NP.

Theorem 1 *There is no polynomial-time α-approximation of the Shapley value of the location in a TSG for constant $\alpha \geq 1$ unless* P = NP.

Proof. Let $G(N,E)$ be a graph with nodes N and edges E. If an α-approximation exists we can use it to solve the NP-complete Hamiltonian cycle problem on G. First, from G, construct a complete weighted and undirected graph $G'(N,E')$, where (i,j) has weight 1 if (i,j) is in the transitive closure of E, and otherwise has weight $n!\alpha$. If there is a Hamiltonian cycle in G, then the Shapley value of any $i \in N$ in the TSG posed by G' is at most 1. Suppose there is no Hamiltonian cycle in G. We show there exists a permutation π of N that induces a large Shapley value for any node j as follows: repeatedly add a node from $N \setminus j$ to π so that there remains a Hamiltonian cycle amongst elements in π; when there is no such node then add j. The marginal cost of adding j to π is at least $n!\alpha$. The Shapley value of j is the average cost of adding it to a coalition $S \subseteq N \setminus j$, therefore its Shapley value is at least α. Even though edge weights in G' are large, we can represent G' compactly in $O(\log(n) + n^2 \log(\alpha))$ space. An α-approximation on G' for j therefore decides the existence of the Hamiltonian cycle in G. ❑

4.3 Sampling-Based Evaluation

Using either the dynamic programming solution, or indeed the state-of-the-art TSP solver Concorde (Applegate, Bixby, Chvatal, & Cook, 2007) in a direct calculation of the Shapley value, we find it impractical to compute the exact Shapley value for instances of the TSG larger than about 10 locations (recall that this does not include the depot, hence the corresponding TSPs have 11 points). A direct method requires an exponential number of characteristic function computations, each of which requires solving an NP-hard problem. Figure 4.3 shows the exponential increase in runtime computing the Shapley value on our experimental setup (described in detail in Section 4.4) via a direct enumeration method. To obtain an accurate baseline for games of a commercially interesting size our investigation now turns to sampling procedures. Indeed, because the Shapley value is a population average it is reasonable to estimate the value using a sampling procedure.

The first use of sampling to approximate the Shapley value of games was proposed and studied by Mann and Shapley (1960). Perhaps the most elegant and general method proposed by Mann and Shapley is called *Type-0* sampling. This method repeatedly draws a permutation of the locations uniformly at random. The marginal cost of each agent i is then calculated by taking the difference in the cost of serving agents up to and including i in the permutation and the cost of serving the agents proceeding i. By repeatedly sampling permutations and computing the marginal costs of including each agent i in this way, we arrive at an unbiased estimate of the Shapley value.

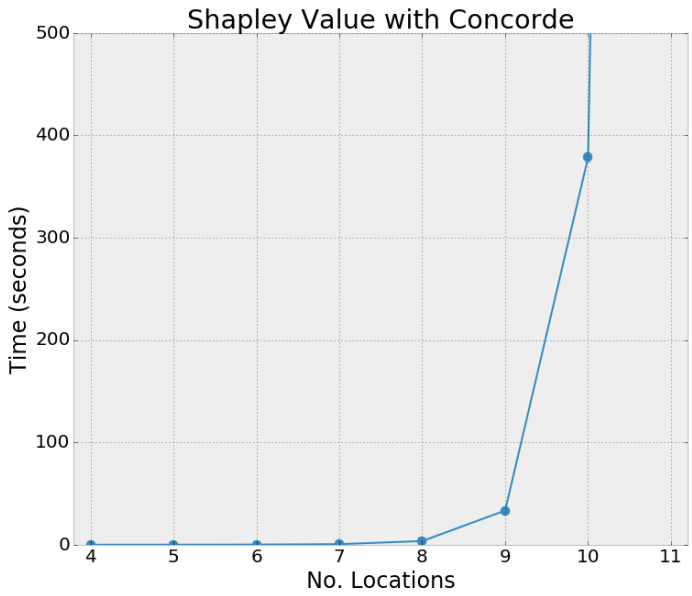


Figure 1: Runtime of computing the Shapley value via brute force enumeration with calls to Concorde for instances with between 4 and 10 locations. The graphs show the mean and standard deviation of the running time over 1,070 games per number of locations. Comparing with Figure 3 we observe the time increasing exponentially, with our practical limit hit around 10 locations.

Type-0 sampling has appeared over the years in various guises, and is reported under different names in the literature on approximating *power indices*—of which the Shapley value is but one—in coalitional games. A recent rediscovery of Type-0 sampling is the *ApproShapley* algorithm by Castro, Gómez, and Tejada (2009); who also provide asymptotic bounds on the sampling error of *ApproShapley*. *ApproShapley* shall be the focus of our sampling work, however prior to giving its details, it is worth briefly reviewing other classes of game where sampling-based evaluations have been explored. Bachrach, Markakis, Resnick, Procaccia, Rosenschein, and Saberi (2010) have previously examined Type-0 sampling in *simple games*—i.e. where the value of a coalition is either 0 or 1—deriving bounds that are *probably approximately correct*. In other words, the actual Shapley value lies within a given error range with high probability. Continuing in this line of work, Maleki, Tran-Thanh, Hines, Rahwan, and Rogers (2013) show that if the range or variance of the marginal contribution of the players is known ahead of time, then more focused (termed *stratified*) sampling techniques may be able to decrease the number of samples required to achieve a given error bound. Other methods of approximating the Shapley value, specifically for weighted voting games, have appeared in the literature including those based on multi-linear extensions (Leech, 2003; Owen, 1972) and focused random sampling (Fatima, Wooldridge, & Jennings, 2008, 2007). Most recently, Type-0 sampling for computing the Shapley value has been applied to a planning setting where a set of delivery companies attempt to pool resources in order to more effectively service a probabilistic set of orders that appear within a territory over a rolling horizon (Kimms & Kozeletskyi, 2015).

To calculate the Shapley value of a TSG via sampling we employ the Type-0 method suggested by Mann and Shapley (1960) and Castro et al. (2009) called *ApproShapley*; pseudocode is given in Algorithm 2. Writing $\pi(N)$ for the set of $|N|!$ permutation orders of locations N, taking $\Pi \in \pi(N)$ we write Π_i for the subset of N which precedes location i in Π. An alternative formulation of the Shapley value can be written in terms of $\pi(N)$, by noting that value equals the marginal cost of each location when we construct coalitions in all possible ways, as follows.

$$SV_i(N,c) = \frac{1}{|N|!} \sum_{\Pi \in \pi(N)} (c(\Pi_i \cup \{i\}) - c(\Pi_i)). \tag{2}$$

For each sampled permutation, *ApproShapley* evaluates the characteristic function for each $i \leq |N|$ computing the length of an optimal tour for the set of locations in the i-sized prefix. By construction, the cost allocation produced by *ApproShapley* is economically efficient. As a small but important optimisation, in our work we cache the result of each evaluation of the characteristic function to avoid solving the same TSP twice. Note that lines 15 and 17 of Algorithm 2, which normalize the values to sum to 1.0, are not strictly necessary since the given algorithm is efficient. However, we include the code here so that all proxies and algorithms surveyed return a cost vector that sums to 1.0.

Algorithm 2 ApproShapley

Input: $N = \{1,\ldots,n\}$ locations with cost $c(S)$ to serve a subset $S \subseteq N$ and m samples.
Output: List SV with SV_i for all $i \in N$.

1 $SV \leftarrow []$
2 **for** $i \leftarrow 1$ **to** $|N|$ **do**
3 $\quad SV_i \leftarrow 0$
4 **end for**
5 **for** $SampleNumber \leftarrow 1$ **to** m **do**
6 $\quad$ // $RAND(X)$ returns a random element of X.
7 $\quad Perm \leftarrow RAND(\pi(N))$
8 $\quad S \leftarrow \emptyset$
9 $\quad$ **for** $i \leftarrow 1$ **to** $|N|$ **do**
10 $\quad\quad S \leftarrow S \cup \{Perm_i\}$
11 $\quad\quad SV_{Perm_i} \leftarrow SV_{Perm_i} + (c(S) - c(S \setminus \{Perm_i\}))$
12 $\quad$ **end for**
13 **end for**
14 $TotalValue \leftarrow \sum_{i \in N} SV_i$
15 **for** $i \leftarrow 1$ **to** $|N|$ **do**
16 $\quad SV_i \leftarrow SV_i \times (c(N)/TotalValue)$
17 **end for**
18 **return** SV

Algorithm 3 SubsetShapley

Input: $N = \{1,\ldots,n\}$ locations with cost $c(S)$ to serve a subset $S \subseteq N$ and m samples.
Output: List SV with SV_i for all $i \in N$.

1 $SV \leftarrow []$
2 **for** $i \leftarrow 1$ **to** $|N|$ **do**
3 $\quad SV_i \leftarrow 0$
4 **end for**
5 **for** $SampleNumber \leftarrow 1$ **to** m **do**
6 $\quad$ **for** $i \leftarrow 1$ **to** $|N|$ **do**
7 $\quad\quad S \leftarrow \emptyset$
8 $\quad\quad$ **for** $j \leftarrow 1$ **to** n **do**
9 $\quad\quad\quad$ // $RAND(X)$ returns a random element of X.t
10 $\quad\quad\quad$ **if** $i \neq j$ **and** $RAND(\{0,1\}) = 1$ **then**
11 $\quad\quad\quad\quad S \leftarrow S \cup \{j\}$
12 $\quad\quad\quad$ **end if**
13 $\quad\quad$ **end for**
14 $\quad\quad SV_i \leftarrow SV_i + |S|!(n - |S| - 1)! \cdot (c(S \cup \{i\}) - c(S))$
15 $\quad$ **end for**
16 **end for**
17 $TotalValue \leftarrow \sum_{i \in N} SV_i$
18 **for** $i \leftarrow 1$ **to** $|N|$ **do**
19 $\quad SV_i \leftarrow SV_i \times (c(N)/TotalValue)$
20 **end for**
21 **return** SV

In our work, we also considered an alternative sampling method, which samples not over permutations, but rather over subsets of locations as implied by the formulation in Equation 1 of Section 2. There are fewer subsets than there are permutations, seemingly an advantage in a sampling-based evaluation of the Shapley value. Using a limited number of subsets to estimate the Shapley value was explored, and shown to be an effective measure, by Papapetrou, Gionis, and Mannila (2011). We name this method *SubsetShapley* and provide pseudocode in Algorithm 3. Like *ApproShapley*, this method produces an economically efficient allocation. For *ApproShapley*, the estimate of SV_i is updated once per drawn permutation while for *SubsetShapley* if we draw only a single random subset, we only update our estimate for one location. Thus, for *SubsetShapley*, at every iteration of the sampling loop at Line 6 we draw a different set $S \subseteq N \setminus \{i\}$ uniformly at random *for each location i*, making the two methods comparable based on the total number of updates per location per iteration. However, we only use one sample in *ApproShapley* for all locations and one sample per location in *SubsetShapley*. For each i, the update to SV_i is then the weighted marginal contribution, formally $SV_i \leftarrow SV_i + |S|!(n-|S|-1)!(c(S \cup i) - c(S))$. The coefficient $|S|!(n-|S|-1)!$ ensures that for each subset S of sampled locations, we account for the number of permutations where locations S are ordered before location i. Note that without this term, this algorithm does not converge to the Shapley value in the limit.

4.4 Experimental Setup and Evaluation of Sampling Methods

It is important both here and in our later experimental evaluation to be confident that we have sampled a sufficient number of times over a sufficient number of games to establish confidence in our sampling scheme and to ensure the statistical significance of our results. We must ensure that, for every game, we have taken enough samples to have a high probability of a low error on any individual Shapley value. For our overall evaluation we must ensure that we have sampled enough games from the representative population of all possible games. In this section we described our experimental setup and derive precise statistical bounds for our results.

Not all proxies and estimators of the Shapley value that we consider yield economically efficient allocations of the cost of the optimal tour. For this reason, we will discuss the Shapley value and all proxies for it in terms of the induced *fractional* (also called *normalized*) allocation of the cost of the optimal tour. Formally, $\phi_i^{\mathrm{SV}} = {}^{SV_i}/_{\sum_{j \in n} SV_j}$. Fractional allocations allow us to compare efficient and non-efficient cost allocations on equal footing, in a way that would be used in operational contexts such as transport settings. This formulation also enables us to allocate the cost of the optimal route only having to solve the NP-hard TSP once.

We generated a collection of Euclidian games we call the *Synthetic dataset*. In the Synthetic dataset we generate locations $|N| \in \{4, \ldots, 20\}$ on a 100×100 unit square. The coordinates of these locations are generated in an independent and identically distributed (i.i.d.) manner represented by 32-bit floating point values.[5] To these n players we add a depot location, also chosen uniformly at random over the square. Hence, for all reported results there are a total of $n+1$ locations for the underlying routing problem and n locations that must have costs allocated to them. All timing experiments reported were performed on a computer with an Intel Xeon E5405 CPU running at 2.0 GHz with 4 GB of RAM running Debian 6.0 (build 2.6.32-5-amd64 Squeeze10). Additional computing power for non-timing experiments was provided by Data61/NICTA's heterogenous compute cluster.

5. Corpus available online at `https://github.com/nmattei/ShapleyTSG`

We use statistical measures to report all our empirical results. We provide a brief overview of key concepts here and refer the reader to the textbook by Corder and Foreman (2009) for details. Denote $|x|$ as the absolute value of the quantity x. Writing $\bar{x}$ denotes the average of a set $\{x_1,\ldots,x_n\}$ and let $\hat{x_i}$ denote an estimate of value x_i from the set X. The standard deviation (σ) of x is: $\sigma = \sqrt{1/n\sum_{i=1}^{n}(x_i-\bar{x})^2}$. To measure accuracy we use *root mean squared errors (RMSE)*, a common metric to quantify the error over a number of predictions. For a set of k paired observations $X = \{x_1,\ldots,x_k\}$ and estimates $\hat{X} = \{\hat{x_1},\ldots,\hat{x_k}\}$, the *RMSE* between X and $\hat{X}$ (the *RMSE* of X with respect to $\hat{X}$) is: $RMSE = \sqrt{1/k\sum_{i=1}^{k}(x_i-\hat{x_i})^2}$.

We perform a similar analysis to that of Castro et al. (2009) to determine the number of samples required to have high confidence in the values obtained via sampling methods in our setting. To establish that the error of our sampling procedure is below ε with probability greater than $1-\alpha$, we use the central limit theorem and the assumption that our errors are normally distributed, giving:

$$\text{No. Samples} \geq Z^2_{\alpha/2}\frac{\sigma^2}{\varepsilon^2} \implies P(|SV_i - \widehat{SV_i}| \leq \varepsilon) \geq 1-\alpha,$$

where $Z \sim N(0,1)$ is a normal random variable. Given a game, we do not know the variance of all the locations for all permutations, and it is infeasible to compute this value. We can estimate the variance given the maximum (x^i_{max}) and minimum (x^i_{min}) change in the cost function of an individual location $i \in N$. If i is co-located with the depot the minimum impact on cost is $x^i_{min} = 0$. As we have a 100×100 unit square the maximum possible distance between any two points is $100 \times \sqrt{2} \leq 142$. The greatest impact this can have on cost is if i is the only location and is added opposite the depot along the diagonal, causing an increase in cost equal to $x^i_{max} = 2 \cdot 142 = 284$.

The maximum variance of a random variable is reached when the variable takes the two extreme values with probability $1/2$. We can then use the following inequality to estimate the variance:

$$\sigma^2 \leq \frac{1}{2}(x^i_{max} - \frac{x^i_{max}+x^i_{min}}{2})^2 + \frac{1}{2}(x^i_{min} - \frac{x^i_{max}+x^i_{min}}{2})^2 \leq \frac{(x^i_{max}-x^i_{min})^2}{4}.$$

Applying this to our previous equation yields a formula for determining the error in our setting:

$$\text{No. Samples} \geq Z^2_{\alpha/2} \cdot \frac{(x^i_{max})^2}{4 \cdot \varepsilon^2} \implies P(|\widehat{SV_i} - SV_i| \leq \varepsilon) \geq 1-\alpha.$$

What we tolerate as an error bound has a significant effect on the size of the games we can actually use for testing (as sampling is very time consuming). Selecting $\varepsilon = 0.75$ means that each location's Shapley value will not be off by more than 0.75 distance units (kilometers), which is very low for a fast moving consumer goods setting. As this derivation of error only gives us an absolute bound on the pairwise error between a location's actual Shapley value SV_i and its estimated Shapley value $\widehat{SV_i}$, we must derive the maximum possible error for all points in order to bound the error of ϕ^{SV}. The total error for a game is given by:

$$\frac{SV_i+\varepsilon}{\sum_j^n(SV_j+\varepsilon)} = \frac{SV_i+\varepsilon}{\sum_j^n(SV_j)+n\varepsilon} = \frac{SV_i+0.75}{\sum_j^n(SV_j)+0.75n}.$$

Observe that $\sum_j^n(SV_j)$ is the exact cost of the grand tour of all the points and our overestimate is $0.75n$. Hence, we are overestimating the grand tour by at most 15 distance units (kilometers) for our $n = 20$ instances. Thus the error in any location's ϕ^{SV} is negligible. We want α to be very small, giving us high confidence that we have converged; we set $\alpha = 0.005$, giving us a 99.5% probability

that our error will be less than ε. Substituting this into our error equation ($Z_{0.05/2} = Z_{0.0025} = -2.81$) we get:

$$\text{No. Samples} = 300,000 \geq -2.81^2 \cdot \frac{(2 \cdot 142)^2}{4 \cdot 0.75^2} \approx 283,052$$

In order to draw conclusions from our calculated *RMSE* values we must have statistical confidence in the mean ($\overline{RMSE}$) over a set of games. The *RMSE* itself is an average over normalized values, each of which take on values in $[0,1]$. Assuming the errors in our problem are normally distributed we can bound the variance using the same techniques described above, arriving at a variance of $\sigma^2 = 1/4$. From this we can use standard techniques from statistics and engineering (Natrella, Croarkin, & Guthrie, 2012) to determine that the number of games we need to use in order to have a 95% confidence interval that the absolute error in the measurement of the $\overline{RMSE}$, which is an aggregate measure of error for *all* locations in *all* games, is within 0.03 (or roughly $\pm 3\%$):

$$\text{No. Games} = 1,070 \geq Z^2_{\alpha=0.05}\left(\frac{\sigma^2}{\delta^2}\right) = (-1.96)^2\left(\frac{1/4}{(0.03)^2}\right) \approx 1,067.$$

Intuitively, this means we are 95% sure that if we re-ran our entire experiment with new values the mean error for a particular proxy would fall within 3%. Hence we can say that the mean error value, measured over the set of 1,070 games, is accurate.

We compare the performance of *ApproShapley* and *SubsetShapley* using our Synthetic dataset. We use *Concorde* (Applegate et al., 2007) to evaluate the characteristic function of the TSG by solving the underlying TSP.

All optimal tour lengths calculated by Concorde are cached to speed up running time. Therefore, we never re-evaluate a TSP with the same set of points. TSPs with less than four locations are evaluated by brute force. For each game in the Synthetic dataset we calculated the exact Shapley value of every location, so that we could compare the sampled allocation with their exact counterparts. We find the *ApproShapley* method of sampling over permutations provides a faster convergence, as seen in Figure 2. After just 1000 iterations *ApproShapley* achieves a $\overline{RMSE}$ below 0.01, with a significantly smaller standard deviation than *SubsetShapley*. Moving to $100,000$ samples in the bottom row of Figure 2, we see that the mean *RMSE* for *ApproShapley* is still significantly lower than for *SubsetShapley*.

Figures 3 and 4 depict the mean running time and the number of calls to Concorde for the two algorithms, respectively. We see that *ApproShapley* runs faster than *SubsetShapley* in all instances that we tested. This difference in runtime grows as the number of locations increases. *ApproShapley*'s faster running time is likely due to the need of only randomly generating one permutation instead of n sets. Figure 4 provides an insight into the behavior of the two algorithms. *SubsetShapley* fills the cache much quicker than *ApproShapley*, which explains the later flattening of the runtime curve for *ApproShapley* seen in Figure 3. Both methods eventually evaluate all 2^{17} possible points, saturating the cache. However, this early filling of the cache by *SubsetShapley* does not translate to faster overall runtime. In practice, *ApproShapley* achieves a lower error, earlier, and continues to converge towards an error of 0 faster than *SubsetShapley*.

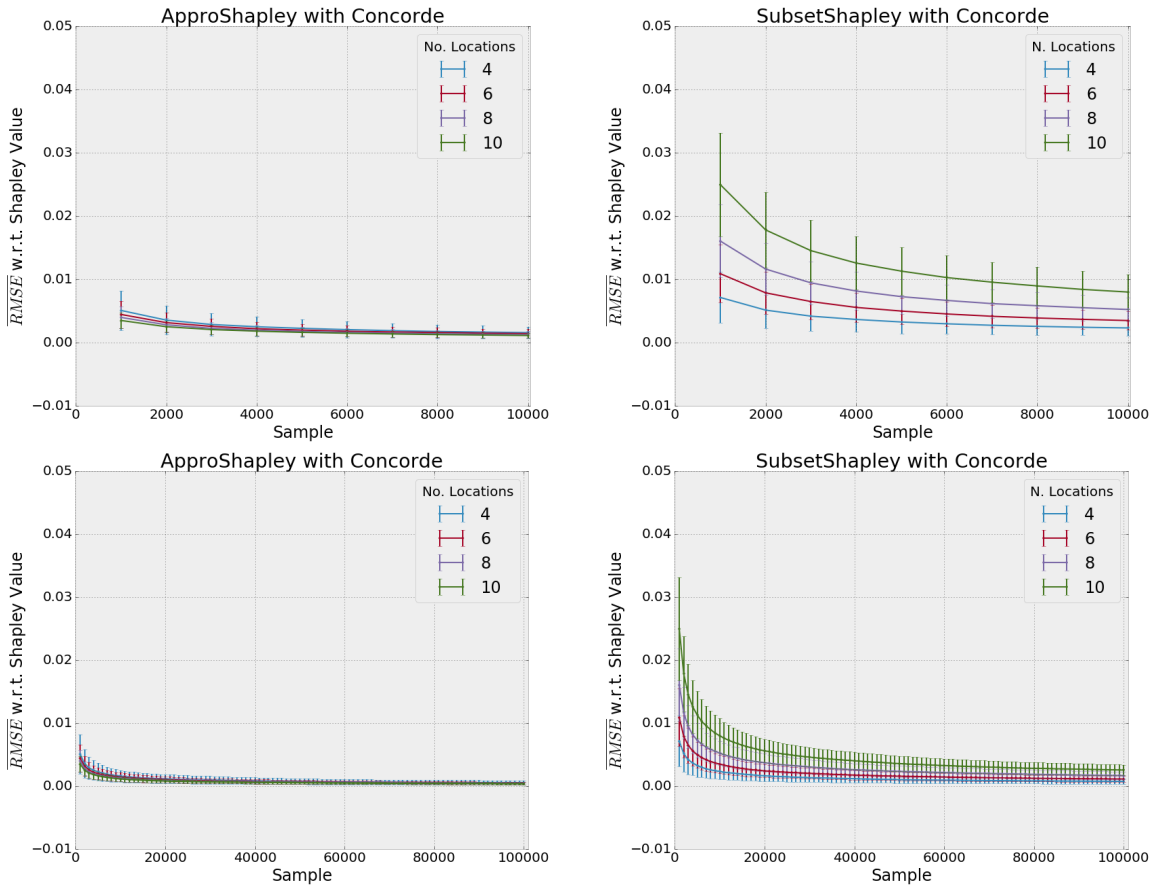


Figure 2: Comparison of the accuracy of ApproShapley (left) and SubsetShapley(right) for 10,000 iterations (top) and 100,000 iterations (bottom) for TSGs with 10 locations. The graphs show $\overline{RMSE}$ and its standard deviation over 1,070 instances between the sampled and actual Shapley values. ApproShapley converges in fewer samples and is more stable between samples than SubsetShapley.

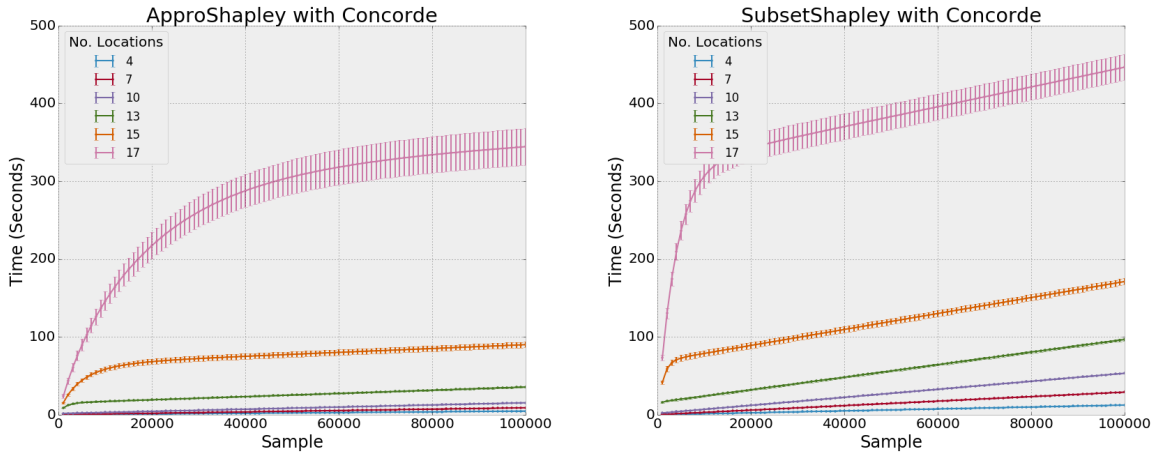


Figure 3: Comparison of the runtime performance of *ApproShapley* (left) and *SubsetShapley* (right) for TSGs with 4 to 17 locations (not including the depot). The graphs show the mean and standard deviation over 1,070 instances of the running time of the respective algorithm. Because *ApproShapley* only needs to generate one permutation, compared to Sample-Shapley's n sets, it generally runs more quickly.

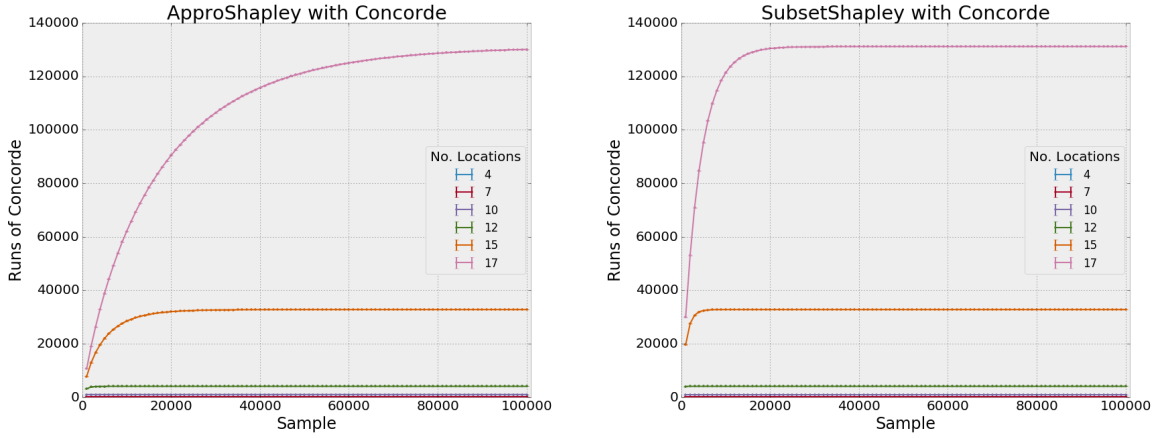


Figure 4: Comparison of the number of calls to Concorde as a function of sample number made by *ApproShapley* (left) and *SubsetShapley* (right) for TSGs with 4 to 17 locations (not including the depot). The graphs show the mean and standard deviation over 1,070 instances of the number of calls to Concorde. *SubsetShapley* fills the cache much quicker than *ApproShapley*, this explains the later flattening of the runtime curve for *ApproShapley* seen in Figure 3. However, the earlier cache filling does not lead to a decrease in running time.

5. Proxies for the Shapley Value

The use of *ApproShapley* requires that we solve an NP-hard problem each time we evaluate the characteristic function. This is feasible for small TSG instances with less than a dozen locations, however it does create an unacceptable computational burden in larger, realistically sized games. We now describe a variety of proxies for the Shapley value that require much less computation in practice. We have seen some of these proxies in use in the real-world to allocate costs, hence their inclusion and analysis. We define and discuss these proxies in terms of their induced *fractional* allocation, $\phi_i^{\text{SV}} = {}^{SV_i}/_{\sum_{j\in n} SV_j}$, as discussed in Section 4.4. An overview of the worst case and practical running times of all these algorithms is presented in Table 1.

Method or Proxy	Worst Case Runtime	Practical Running Time 10 Loc.	20 Loc.	30 Loc.
ApproShapley (Concorde)	Exponential	$\approx 30sec.$	$\approx$ 4,500 sec.	$>$ 90,000 sec.
Shortcut Distance (ϕ^{SHORT})	Exponential	$\approx$ 5 sec.	$\approx$ 10 sec.	$\approx$ 15 sec.
Re-routed Margin (ϕ^{REROUTE})	Exponential	$\approx$ 20 sec.	$\approx$ 25 sec.	$\approx$ 30 sec.
Depot Distance (ϕ^{DEPOT})	$O(n)$	$\leq$ 1 sec.	$\leq$ 1 sec.	$\leq$ 1 sec.
Moat Packing (ϕ^{MOAT})	Exponential	$\leq$ 5 sec.	$\leq$ 5 sec.	$\leq$ 5 sec.
Christofides (ϕ^{CHRIS})	$O(n^3)$	$\approx$ 30 sec.	$\approx$ 2,500 sec.	$\approx$ 40,000 sec.
Blend (ϕ^{BLEND})	Exponential	$\leq$ 5 sec.	$\leq$ 5 sec.	$\leq$ 5 sec.

Table 1: Summary of the proxies for the Shapley value surveyed in this paper.

5.1 Depot Distance (ϕ^{DEPOT})

The distance from the depot — i.e. d_{i0} for location i — is our most straightforward proxy. We allocate cost to location i proportional to d_{i0}. The fraction allocation to location i is

$$\phi_i^{\text{DEPOT}} = \frac{d_{i0}}{\sum_{i=1}^{n} d_{i0}}.$$

For this proxy, a location that is twice as distant from the depot as another has to pay twice the cost. We can evaluate this proxy in time linear in the number of locations. In practice, computing this value is instantaneous.

5.2 Shortcut Distance (ϕ^{SHORT})

Another proxy that is straightforward to calculate and which has been used in commercial routing software is the *shortcut distance*. This is the change in cost realized by skipping a location when traversing a given optimal tour. Without loss of generality, suppose the optimal tour visits the locations according to the sequence $[0,1,2,\ldots]$. Formally, $\text{SHORT}_i = d_{i-1,i} + d_{i,i+1} - d_{i-1,i+1}$, where locations 0 and $n+1$ are the depot, and d_{ij} is the cost of travel from location i to j. The fractional

allocation given by the shortcut distance is then

$$\phi_i^{\text{SHORT}} = \frac{\text{SHORT}_i}{\sum_{j \in N} \text{SHORT}_j}.$$

We can evaluate this proxy by solving one TSP instance and one operation per location. In practice we can compute this metric in less than 30 seconds.

5.3 Re-routed Margin (ϕ^{REROUTE})

For a location $i \in N$, REROUTE_i is defined as $c(N) - c(N \backslash i))$. The allocation to a player can be computed with at most two calls to an optimal TSP solver. The fractional allocation is

$$\phi_i^{\text{REROUTE}} = \frac{(c(N) - c(N \backslash i))}{\sum_{j=N}(c(N) - c(N \backslash j))}.$$

We can evaluate this proxy by solving $n+1$ TSPs: one for the grand tour and one for leaving out each location. In practice we can compute this metric nearly instantaneously.

5.4 Christofides Approximation (ϕ^{CHRIS})

A more sophisticated proxy is obtained if we use a heuristic when performing characteristic function evaluations in *ApproShapley*, rather than solving the individual induced TSPs optimally. For this proxy we use sampling to estimate the Shapley value and we use an approximation algorithm to estimate the underlying TSP cost. To approximate the underlying TSP characteristic function, the Christofides (1976) heuristic, an $O(|N|^3)$ time procedure is used. To obtain a fractional quantity ϕ_i^{CHRIS}, we divide the allocation to location i by the total allocated costs. Assuming a symmetric distance matrix satisfying the triangle inequality, the Christofides heuristic is guaranteed to yield a tour that is within $3/2$ the length of the optimal tour.

We briefly describe how the heuristic works. The TSP instance is represented as complete undirected graph $G = (V, E)$, with one vertex in V for each location, and an edge E between every distinct pair of vertices. For $i, j \in V$ the edge $(i, j) \in E$ has weight d_{ij}. A tour is then obtained as follows: (1) compute the minimum spanning tree (MST) for G, (2) find the minimum weight perfect matching for the complete graph over vertices with odd degree in that MST (typically performed using the *Hungarian Algorithm*), (3) calculate a Eulerian tour for the graph obtained by combining the MST from Step 1 and the matched edges from Step 2 (this is guaranteed to yield a Eulerian multigraph, i.e., a graph where every vertex has even degree), and (4) obtain a final tour for the TSP by removing duplicate locations from the Eulerian tour.

In the best case, the call to the Christofides heuristic will return a solution that is exactly the solution to the TSP. Hence, this method requires as many calls per number of locations as we derived in Section 4.4. Figure 5 shows the runtime of *ApproShapley* when we replace calls to Concorde with calls to a program that solves a TSP using the Christofides heuristic. Comparing these results to those in Figure 3, we see that for small numbers of locations (≤ 10) the runtimes for Concorde and the Christofides heuristic are almost the same. However, as the number of locations grows, the Christofides heuristic shows a significant speed improvement. For commercially interesting sizes, 20 locations and 300,000 samples, computing ϕ^{CHRIS} takes on the order of 2500 seconds (about 30 minutes). It is not practically computable ($\approx$ 12 hours) for problems with 30+ locations, as shown in Table 1.

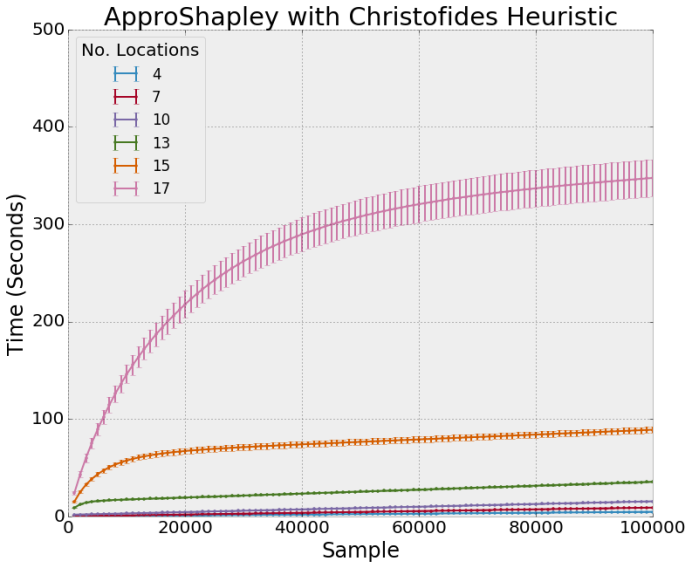


Figure 5: Runtime performance of ApproShapley with calls to the Christofides heuristic for TSGs with 4 to 17 locations (not including the depot). The graphs show the mean and standard deviation over 1,070 instances of the running time of the respective algorithm. Comparing with Figure 3 we see that the Christofides heuristic decreases runtime and that this decrease grows larger as the number of locations increases.

5.5 Nested Moat-Packing (ϕ^{MOAT})

Another way to allocate costs in TSGs is based on dividing the locations into regions using a concept called a *moat* (Cook, Cunningham, Pulleylank, & Schrijver, 1998). Intuitively, given a Euclidian TSP and a set of locations N with the depot location 0, a moat is a closed strip of constant width that separates a set of locations $S \subseteq N$ from its compliment $\overline{S}$. We assume without loss of generality that 0 is always in $\overline{S}$. In order to deliver to any location in S, one would need to traverse the moat in order to reach all points in the set S, and then cross the moat again to return to any point in $\overline{S}$. Hence, a reasonable cost allocation is to charge the locations in $\overline{S}$ *twice* the cost of traversing the moat surrounding S. If the locations in S had their own delivery truck, there would be no reason to cross the territory of the moat surrounding S. In the following we use techniques described by Faigle, Fekete, Hochstättler, and Kern (1998) and additionally refined by Özener et al. (2013) with some extensions for our setting.

Formally, given a set of locations $N \cup \{0\}$, let $S \subseteq N$ and $\overline{S}$ be the compliment of S, and let M be the set of all bipartitions of the locations $\{S, \overline{S}\}$ with the assumption that $0 \in \overline{S}$. Let $w_{S,\overline{S}}$ be the width (distance) of the moat between the set S and $\overline{S}$. We refer to a vector of moats $\vec{w}$ and the width of an individual moat as w_S. Locations themselves cannot occur in the moat, as the moat itself is a strip of "unoccupied" area on the map. Additionally, one only needs to consider circular moats as this gives the minimal straight-line distance to a location inside the moat and is less than or equal to the moat width. In order to have a well-formed cost allocation we want the moats to be as large as

possible, i.e., have maximum width. Hence, we can formulate a linear program to find a maximum moat packing:

$$\begin{aligned} MPV = \max & \sum_{S,\overline{S} \in M} w_{S,\overline{S}} \\ w_{S,\overline{S}} \geq 0 & \qquad \forall \{S,\overline{S}\} \in M \\ s.t. & \sum_{i \in S, j \in \overline{S}} w_{S,\overline{S}} \leq d_{ij} \quad \forall i,j \in N \cup \{0\}. \end{aligned} \tag{3}$$

Though Equation 3 has exponentially many constraints, it can be solved in time polynomial in the number of locations using dual techniques, returning at-most a polynomial number of moats with width ≥ 0. We shall use the notation $\vec{w}$ to refer to the small set of moats with non-zero width (Faigle et al., 1998). The vector of moat widths $\vec{w}$ given by the solution to Equation 3 may have many moats that overlap, leading to ambiguities over which widths to allocate to which locations. Thus, an arbitrary solution to Equation 3 does not yield a cost allocation. For this reason we must refine our maximum moat packing to a maximum *nested* moat packing. In a nested moat packing two distinct intersecting subsets cannot be encapsulated by the same non-empty moat unless one of those coalitions is a subset of the other. Formally, a packing is *nested* if and only if $\forall S', S''$ s.t. $w_{S'} > 0$ and $w_{S''} > 0$, if $S' \cap S'' \not\equiv \emptyset$ then either $S' \subseteq S''$ or $S'' \subseteq S'$. For any optimal solution $\vec{w}$ to Equation 3 yielding objective value MPV over the set of partitions M there is a corresponding nested packing with the same MPV (Cornuéjols, Naddef, & Pulleyblank, 1985; Faigle et al., 1998). Once we have a nested moat packing as it is clear which set of moats must be crossed to reach any location from any other location, at which point we can derive a cost allocation for each location. Figure 6 is a concrete example of a nested moat packing for 6 locations. In Figure 6 each of the 6 locations has its own moat (light colors). Additionally, the moats for locations 5 and 6 are then surrounded by an outer darker moat for the set $\{5,6\}$.

Given a non-nested vector of moats $\vec{w}$ we follow the post-processing procedure described by Özener et al. (2013). For the nesting criteria defined above to be violated there must be three distinct non-empty sets of locations S, S' and S'', such that $w_{S \cup S'} > 0$ and $w_{S' \cup S''} > 0$. Given $\vec{w}$ we update the values as follows: let $\tau \leftarrow \min\{w_{S \cup S'}, w_{S' \cup S''}\}$, make the following assignment updates to the moat widths: $w_S \leftarrow w_S + \tau$, $w_{S''} \leftarrow w_{S''} + \tau$, $w_{S \cup S'} \leftarrow w_{S \cup S'} - \tau$, and $w_{S' \cup S''} \leftarrow w_{S' \cup S''} - \tau$. This iterative procedure terminates yielding a nested packing, taking exponential time in the worst case. However, in all our experiments we found that nesting takes only a fraction of a second. This leaves us with the allocation:

$$\phi_i^{\text{MOAT}} = \frac{1}{MPV} \times \sum_{w_S > 0, i \in S} \frac{w_S}{|S|} = \frac{1}{\sum_{w_S > 0} w_S} \times \sum_{w_S > 0, i \in S} \frac{w_S}{|S|}.$$

There are two key observations about the allocation derived from a (nested) moat packing. First, $2 \times MPV$, i.e., the sum of crossing all the moats twice, is exactly the value of the Held-Karp relaxation for the underlying TSP if the TSP is symmetric and satisfies the triangle inequality (Held & Karp, 1962). Thus, $2 \times MPV$ is a lower bound for the optimal tour for the underlying TSP and $3 \times MPV$ is an upper bound.[6] Secondly, we observe that the allocation derived from a nested moat packing where x_i is the cost of location i satisfies $\sum_{i \in N} x_i \geq c(N)$ and $\forall S \subseteq N : \sum_{i \in S} x_i \leq (1+\varepsilon)c(S)$. These constraints are exactly those for the multiplicative core we defined in the preliminaries for

6. The tightness of the bounds of the Held-Karp relaxation, i.e., the integrality gap, is a longstanding open question in combinatorial optimisation (Cook et al., 1998).

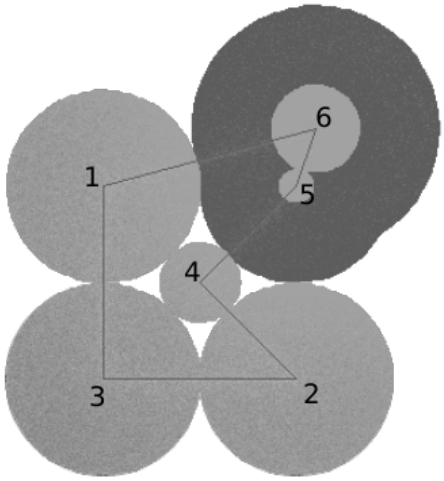


Figure 6: An optimal nested moat-packing (colored regions) and optimal tour (line) for a TSP with 6 locations. The locations are indicated by the labels $\{1,2,\ldots,6\}$, and occur at the center of the light moats. Each moat around locations 1 to 4 (light colored regions) is associated with only one location. The moat around the set of locations $\{5,6\}$ (dark colored region) is *nested*. This larger moat encloses two smaller, independent moats (light colored regions) around locations 5 and 6, respectively. There are 7 moats in total, and the optimal tour must cross each moat twice.

the ε-core. Although the allocation achieved using nested moat packing to distribute $3 \times MPV$ is not economically efficient, it is a core allocation of an approximate cost. Faigle et al. (1998) present a proof that the nested moat packing provides a $\frac{1}{2}$-core allocation with respect to the actual cost function for each location by distributing $3 \times MPV$; they conjecture $\varepsilon \leq \frac{1}{3}$.

5.6 Blended Proxy (ϕ^{BLEND})

An interesting question is whether or not blending a set of proxies that were practically computable could provide a good estimate of the actual Shapley value. Framing this as a prediction or machine learning problem, we want to learn a model to predict our output ϕ^{SV} given an input set consisting of the easily computable proxies, $\{\phi^{\text{DEPOT}}, \phi^{\text{SHORT}}, \phi^{\text{REROUTE}}, \phi^{\text{MOAT}}\}$. All analysis in this section is carried out using SciKitLearn (Pedregosa et al., 2011), a machine learning library for Python.

First, we need to decide what sort of model is best for this setting. As each of these proxies is attempting to estimate the same value, they are correlated. Consequently, one place to start is to use *principal component analysis (PCA)* (Bishop, 2006) to understand how much variance can be captured by a low dimensional model given our input set. We use SciKitLearn to run a PCA decomposition over the set of all Synthetic data. SciKitLearn uses the linear algebra package of SciPy to perform a *singular value decomposition (SVD)* of the data matrix; keeping only the most significant singular vectors to project the data into lower dimensional spaces. This decomposition shows that 98% of the variance can be explained by one component (vector), as depicted in Figure 7. Hence, a simple linear blend of a subset of the proxies should provide good predictive power.

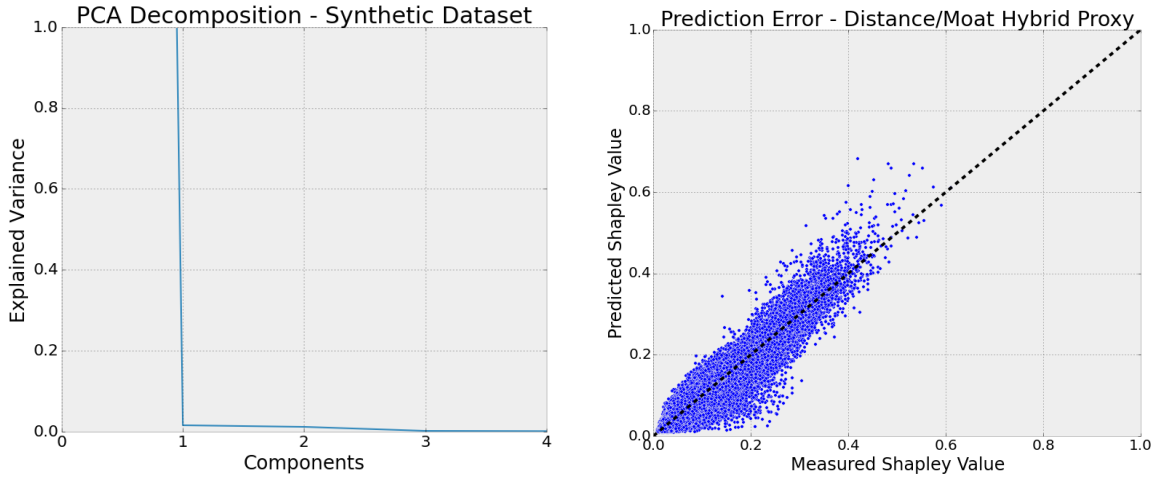


Figure 7: The explained variance ratio of ϕ^{SV} given by a PCA decomposition of the input set of $\{\phi^{\mathrm{DEPOT}}, \phi^{\mathrm{SHORT}}, \phi^{\mathrm{REROUTE}}, \phi^{\mathrm{MOAT}}\}$ (left). Using a 1 dimensional model over the input set we can explain 98% of the observed variance in ϕ^{SV}. Prediction error graph for 10 fold cross-validation over the Synthetic dataset for a linear model blending ϕ^{DEPOT} and ϕ^{MOAT} (right). Each point represents the prediction error for a location in the Synthetic dataset and the dotted line ($y = x$) would be an ideal predictor. The correlation of actual and predicted values for the 10 fold cross-validation is $\overline{R^2} = 0.8825$ and $\sigma = 0.0025$.

As we have selected a simple linear model, we must now decide which elements of the input set of proxies, $\{\phi^{\mathrm{DEPOT}}, \phi^{\mathrm{SHORT}}, \phi^{\mathrm{REROUTE}}, \phi^{\mathrm{MOAT}}\}$, we should use. We want to use the minimal set of features, as using too many features can cause overfitting (Bishop, 2006). For selection, we use SciKit (Pedregosa et al., 2011) to perform a k-best feature selection which takes each of the input set in turn and computes a cross correlation between this element and the others, this is converted using ANOVA to a score and a significance (p) value for each feature. We can compare these scores to find the individual elements of the input set which are the most significant. Doing this we find the scores for all of the elements of the input set to be statistically significant, hence useable. Looking at the normalized scores themselves, $\{\phi^{\mathrm{DEPOT}} = 1.0, \phi^{\mathrm{SHORT}} = 0.0260, \phi^{\mathrm{REROUTE}} = 0.4946, \phi^{\mathrm{MOAT}} = 0.6305\}$, we see that $\phi^{\mathrm{DEPOT}}$and $\phi^{\mathrm{MOAT}}$are the two highest scoring indicators. We choose to limit our linear model to two highest scoring elements $\phi^{\mathrm{DEPOT}}$and ϕ^{MOAT}as these are both significantly higher scoring than the others and adding more elements may cause overfitting.

Now that we have a model and the input variables to train on, we need to learn the model and perform cross-validation. For this tests we take the full Synthetic dataset and perform a 10-fold cross-validation (Bishop, 2006). To perform *k-fold cross-validation*, we take the dataset and break it into k equal sized folds $F = \{f_1, \ldots, f_k\}$. We "hold out" each one of these pieces from the training set in turn (i.e., train on $F \setminus \{f_i\}$) and use it as the test set (i.e., predict f_i). To select the 10 folds to be used for cross-validation we use a stratified k-fold sampling, which ensures that every k fold has the same statistical distribution as the whole training set (Pedregosa et al., 2011). Since we are using a linear model, we use the coefficient of determination, R^2, as our fitness measure. For the 10 fold cross-validation, we get a mean $\hat{R^2} = 0.8825$ with a standard deviation of $\sigma = 0.0025$. The graph

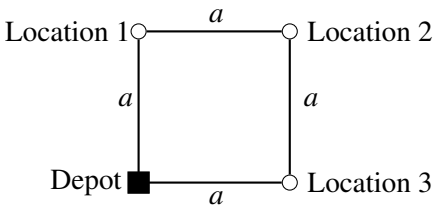
of predicted ϕ^{SV}as a function of the actual value is shown in Figure 7. The low σ shows that our model is robust and the high value of $\hat{R}^2$ indicates that our model is a good predictor. By training over the entire Synthetic dataset we get our final model:

$$\phi^{\text{BLEND}} = 0.579 \cdot \phi^{\text{DEPOT}} + 0.318 \cdot \phi^{\text{MOAT}} + 0.009.$$

6. Analysis of Naïve Proxies

We refer to the three proxies ϕ^{DEPOT}, ϕ^{SHORT} and ϕ^{REROUTE} as being *naïve*. Contrastingly, we call ϕ^{CHRIS}, ϕ^{MOAT} and ϕ^{BLEND} the *sophisticated* proxies. The formulation of the naïve proxies ϕ^{DEPOT} and ϕ^{SHORT} make them amenable to direct analysis of their worst case performance. We consider settings where the naïve proxies ϕ^{DEPOT} and ϕ^{SHORT} can perform quite badly.

In order to illustrate this, consider a TSG where the depot is at one corner of a square of dimension a with one location at each of the other 3 corners. Locations nearest the depot are indexed 1 and 3, and the third location indexed 2.



Our naïve proxies yield the following allocations:

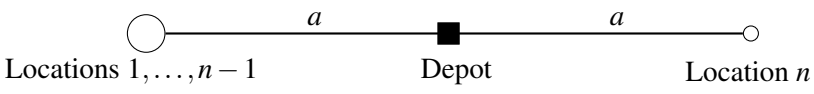
i	ϕ^{SV}	ϕ^{DEPOT}	ϕ^{SHORT}
1, 3	$0.299a$	$0.293a$	$0.333a$
2	$0.402a$	$0.415a$	$0.333a$

Observe ϕ^{DEPOT} performs well in this case (maximum of $\approx 11\%$ error) while ϕ^{SHORT} does not (minimum of $\approx 16\%$ error).

We now identify some pathological cases on which the ϕ^{SHORT} and ϕ^{DEPOT} proxies perform poorly. Our first result demonstrates that ϕ^{DEPOT} and ϕ^{SHORT} may under-estimate the true Shapley value badly.

Theorem 2 *There exists an n location TSP problem on which, for some location i, the ratio* $\phi_i^{\text{DEPOT}}/\phi_i^{\text{SV}}$ *goes to 0 as n goes to* ∞*. In the same instance, the ratio* $\phi_i^{\text{SHORT}}/\phi_i^{\text{SV}}$ *goes to* 0 *as n goes to* ∞ *for* $\Theta(n)$ *of the locations.*

Proof. Suppose the first $n-1$ locations are at distance a from the depot, whilst the nth location is located at a distance a in the opposite direction from the depot.



Note that the normalization constant for ϕ^{SV}, $\sum_{j\in n} SV_j = 4a$. Now $\phi_n^{\text{SV}} = {}^{2a}/_{4a} = {}^{1}/_{2}$ since the cost of adding the nth location to any coalition is $2a$. Leaving, for $i < n$,

$$\phi_i^{\text{SV}} = \frac{{}^{2a}/_{(n-1)}}{4a} = \frac{1}{2(n-1)}.$$

On the other hand, the normalization constant for ϕ^{DEPOT}, $\sum_{i=1}^{n} d_{i0} = na$ since all locations are equidistant from the depot. Giving, for all $i \leq n$, $\phi_i^{\text{DEPOT}} = \frac{1}{n}$.

Thus for $i < n$,

$$\frac{\phi_i^{\text{DEPOT}}}{\phi_i^{\text{SV}}} = \frac{{}^{1}/_{n}}{{}^{1}/_{2(n-1)}} = \frac{2n-1}{n}$$

which goes to 2 as $n \to \infty$. On the other hand,

$$\frac{\phi_n^{\text{DEPOT}}}{\phi_n^{\text{SV}}} = \frac{{}^{1}/_{n}}{{}^{1}/_{2}} = \frac{1}{2n}$$

which goes to 0 as $n \to \infty$.

Note that the shortcut proxy, ϕ^{SHORT} performs poorly on this example. For $i < n$, $\phi_i^{\text{SHORT}} = 0$ since all the locations are co-located, leaving $\phi_n^{\text{SHORT}} = 1$. For $i < n$ we have $\phi_i^{\text{SV}} = {}^{1}/_{2}(n-1)$. Thus, for $i < n$,

$$\frac{\phi_i^{\text{SHORT}}}{\phi_i^{\text{SV}}} = \frac{0}{{}^{1}/_{2(n-1)}} = 0$$

and

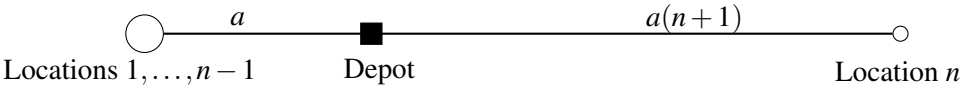
$$\frac{\phi_n^{\text{SHORT}}}{\phi_n^{\text{SV}}} = \frac{1}{{}^{1}/_{2}} = 2$$

❑

Our second result demonstrates that ϕ^{SHORT} can also grossly over-estimate the true Shapley value.

Theorem 3 *There exists an n location TSG where the ratio* ${}^{\phi_i^{\text{SV}}}/_{\phi_i^{\text{DEPOT}}}$ *goes to 0 as n goes to* ∞ *for* $\Theta(n)$ *of the locations.*

Proof. Suppose the first $n-1$ locations are at distance a from the depot, whilst the nth location is located at a distance $(n+1)a$ from the depot in the opposite direction.



Note that the normalization constant for ϕ^{SV}, $\sum_{j\in n} SV_j = 2a + 2a(n+1) = 2a(n+2)$. The Shapley value SV_i for any $i < n$ is $\frac{2a}{n-1}$, thus

$$\phi_i^{\text{SV}} = \frac{{}^{2a}/_{n-1}}{2a(n+2)} = \frac{1}{(n-1)(n+2)}.$$

While the fractional Shapley allocation for location n is

$$\phi_n^{\mathrm{SV}} = \frac{2a(n+1)}{2a(n+2)} = \frac{1}{2}.$$

The normalization constant for ϕ^{DEPOT} is $\sum_{i=1}^{n} d_{i0} = a(n-1) + a(n+1) = 2an$. For location n the assignment from the distance based proxy is

$$\phi_n^{\mathrm{DEPOT}} = \frac{a(n+1)}{2an} = \frac{n+1}{2n}.$$

For $i < n$,

$$\phi_i^{\mathrm{DEPOT}} = \frac{a}{2an} = \frac{1}{2n}.$$

Thus, for location n we have

$$\frac{\phi_n^{\mathrm{SV}}}{\phi_n^{\mathrm{DEPOT}}} = \frac{1/2}{n+1/2n} = \frac{2n}{2n+1}$$

which goes to 1 as n goes to ∞.

For $i < n$ we have

$$\frac{\phi_i^{\mathrm{SV}}}{\phi_i^{\mathrm{DEPOT}}} = \frac{1/(n-1)(n+2)}{\frac{1}{2n}} = \frac{2n}{(n-1)(n+2)}$$

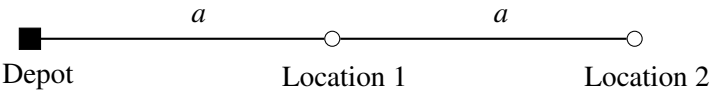
which goes to 0 as n goes to ∞.

For the ϕ^{SHORT} we again have $i < n$, $\phi_i^{\mathrm{SHORT}} = 0$ leaving $\phi_n^{\mathrm{SHORT}} = 1$. Thus, $\phi_n^{\mathrm{SV}}/\phi_n^{\mathrm{SHORT}} = 1/2$ while for $i < n$, $\phi_i^{\mathrm{SV}}/\phi_i^{\mathrm{SHORT}}$ is undefined. ❑

Our third result demonstrates that ϕ^{SHORT} may under-estimate the Shapley value badly even on very simple examples which may be embedded in larger problems.

Theorem 4 *There exists a 2 location TSG instance for which $\phi^{\mathrm{SHORT}}/\phi^{\mathrm{SV}} = 0$ for one of the two locations.*

Proof. Suppose the first location is located a distance a from the depot with the second location located a distance of a farther down the road.



For the first location $\phi_1^{\mathrm{SHORT}} = 0$, since removing it has no effect on the distance to the second location. This leaves $\phi_2^{\mathrm{SHORT}} = 1$. The Shapley value for the first location is

$$SV = \frac{2a}{2} + \frac{0}{2} = a.$$

Which gives $\phi^{\mathrm{SV}} = a/4$ and thus

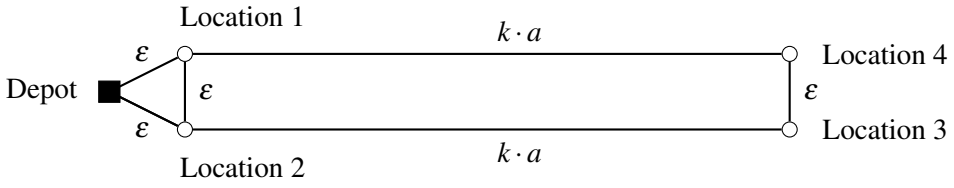
$$\frac{\phi^{\mathrm{SHORT}}}{\phi^{\mathrm{SV}}} = \frac{0}{a/4} = 0.$$

❑

Our fourth and final result demonstrates that ϕ^{SHORT} may over-estimate the Shapley value badly.

Theorem 5 *There exists a four location TSG for which* $\phi^{\text{SV}}/\phi^{\text{SHORT}} = 0$ *for two of the four cities.*

Proof. Consider a four location TSG where locations 1 and 2 are ε from each other and the depot while cities 3 and 4 are at a distance a from the depot and ε from each other.



We note that here $\varepsilon \ll k \cdot a$, as such we will hide ε terms in $O(\varepsilon)$. The marginal cost saved by skipping any location is ε, this means that all locations have the same allocation according to ϕ^{SHORT}, namely for all $i \in \{1,\ldots,4\}$, $\phi_i^{\text{SHORT}} = 1/4$.

Note that the normalization constant for ϕ^{SV}, $\sum_{j\in n} SV_j = 2ka + O(\varepsilon)$. To compute the Shapley values for locations 1 and 2 we observe that, in any given permutation, each location adds a multiple of ε, thus by symmetry, for $i \in \{3,4\}$,

$$\phi_i^{\text{SV}} = \frac{O(\varepsilon)}{2ka + O(\varepsilon)}$$

To compute the Shapley value for locations 3 and 4 we observe that, no matter where in the permutation they appear, the first contributes $2ka$ while the other contributes only ε. Consequently, by symmetry, for locations $i \in \{3,4\}$,

$$\phi_i^{\text{SV}} = \frac{\frac{2ka+O(\varepsilon)}{2}}{2ka + O(\varepsilon)} = \frac{1}{2}.$$

Thus, locations $i \in \{1,2\}$, we have

$$\frac{\phi^{\text{SV}}}{\phi^{\text{SHORT}}} = \frac{\frac{O(\varepsilon)}{2ka+O(\varepsilon)}}{1/4} = \frac{4O(\varepsilon)}{2ka + O(\varepsilon)}.$$

The term goes to 0 as k goes to ∞.

❑

All of the games illustrated above to illustrate poor performance by these proxies are relatively simple and extremely degenerate. In real-world settings we would not expect the locations for delivery to be setup along a straight line or in a symmetrical box. Hence we are motivated to compare the proxies using data that more accurately reflects the domain in which we hope to deploy our proxies.

7. Empirical Study

We implemented each of the six proxies discussed, along with a version of *ApproShapley* that uses *Concorde* (Applegate et al., 2007) to evaluate the characteristic function of the TSG. All code and data used in this project is available in a public Git repository at: `https://github.com/nmattei/ShapleyTSG`. Rather than calculating ϕ^{SV} by direct enumeration as a baseline to compare proxies,

we estimate that value using *ApproShapley* with *Concorde*. As described in Section 4.4, this method achieves an extremely good approximation of the true Shapley value and is computable in a reasonable time for testing on games with up to 20 locations.

We use our corpus of 1070 Synthetic games, constructed as described in Section 4.4, for games with $n \in \{4,\ldots,20\}$ locations. We also test against a corpus of 119 *Real-World games* generated from large VRPs in the cities of Auckland (New Zealand), Canberra, and Sydney (Australia). Heuristic solutions to those VRPs are calculated using the *Indigo* solver (Kilby & Verden, 2011). Indigo is a flexible heuristic solver implementing an Adaptive Large Neighbourhood Search, the basic structure of which is described in detail by Ropke and Pisinger (2006).[7] To give an indication of the scale and difficulty of these VRPs, the Auckland model comprises $1,166$ locations to be served using a fleet of at most 25 vehicles over a 7 day period. From these heuristic solutions we collect we collect tours of length 10 and 20 to create TSGs for testing. Because real-world distance matrices are asymmetric (in all cases asymmetry is negligible), and we induce symmetric problems by resolving for the greater of d_{ij} and d_{ji}, i.e., setting $d_{ij} = d_{ji} = \max\{d_{ij}, d_{ji}\}$. In total, we obtain 71 Real-World games of size 10 (14 in Auckland, 5 in Canberra, and 52 in Sydney) and 48 games of size 20 (10 in Auckland, 7 in Canberra, 31 in Sydney).[8]

To evaluate how well proxies perform in approximating ϕ^{SV} we use several different test statistics, which we briefly review here (Corder & Foreman, 2009). Already discussed in Section 4.4 is the root-mean-squared-error (*RMSE*) for each game. Additionally, we may want to know the *maximum absolute point-wise error (MAPE)*, i.e., the maximum of the absolute error values for each point-wise estimate:

$$MAPE = \arg\max_{x,\hat{x}\in[X,\hat{X}]} |x - \hat{x}|.$$

We can use this measure for a particular game or compare the *average maximum absolute point-wise error over a set of games ($\overline{MAPE}$)*. This value lets us know, on average, the most we are overcharging a particular customer (unlike *RMSE* which only tells us the aggregate error). Note that using the same arguments from Section 4.4 we have the same guarantees on the acuracey of $\overline{MAPE}$ as we do for $\overline{RMSE}$, i.e., $\pm 3\%$.

One question often repeated in our consultation with logistics companies is *who is my most expensive customer?* In order to know where to focus efforts on contract negotiations or sales functions, companies desire an understanding of the rank ordering of the cost of servicing locations. We use Kendall's τ, written as KT and first introduced by Kendall (1938), to compare the ranking, i.e., least expensive to most expensive, of locations induced by the Shapley allocation and our proxies. The value τ measures the amount of disagreement between two rankings. It is customary to report τ as a normalized value (correlation coefficient) between 1 and -1, where $\tau = 1$ means that two lists are perfectly correlated (equal) and $\tau = -1$ means that two lists are perfectly anti-correlated (they are equal if one list is reversed). An intuitive interpretation of τ between two lists is that $\tau\%$ of the orderings in the two lists are the same.

In detail, let X and Y be two partial orders over a set of items. If $a \gtrless b \in X \cap Y$ then we say X and Y are *concordant* on (a,b). If $a = b \in X \cup Y$ then we say there is a *tie*, and otherwise (a,b) is

7. *Indigo* is a strong vehicle routing solution platform, recently computing 5 new best solutions for $1,000$ customer problems from the VRPTW benchmark library. The solutions computed using Indigo were certified by Dr. Geir Hasle, Chief Research Scientist at SINTEF and maintainer of the VRPTW benchmark library, as the best currently known on September 24th of 2013. `http://www.sintef.no/Projectweb/TOP/VRPTW/Homberger-benchmark/1000-customers`.
8. Due to commercial agreements with our industrial partners we cannot release these Real-World games.

discordant. Where M is the number of concordant pairs, N discordant pairs, T ties exclusively in X, U ties exclusively in Y, the normalised KT distance τ between X and Y is:

$$\tau = \frac{M-N}{\sqrt{(M+N+T)\times(M+N+U)}}.$$

Our analysis makes use of the significance, or p-value, of a computed τ. The p-value is computed using a two-tailed t-test where the null hypothesis is that there is no correlation between orderings ($\tau = 0$). Taking our significance threshold to be the customary 0.05, we can reject the null hypothesis when $p \leq 0.05$. When $p \geq 0.05$ we fail to reject the null hypothesis. Hence, a p-value ≤ 0.05 is a statistically significant result. Intuitively this means that it is so unlikely that two random lists would show such a high degree of correlation we can say the two lists are significantly correlated.

7.1 Synthetic Data

Figure 8 gives an overview of our data, showing the $\overline{RMSE}$ and $\overline{\tau}$ for each proxy from ϕ^{SV} for all game sizes of the Synthetic data. Tables 2 to 5 give a more in-depth look at the performance of the proxies on a variety of interesting measures including their $\overline{RMSE}$, $\overline{MAPE}$, $\overline{\tau}$, number of statistically significant τ's, and number of games with correctly identified top elements. In general, ϕ^{SHORT} and ϕ^{REROUTE} proxies are by far the worst, particularly in terms of approximating Shapley value, but also in terms of the ranking induced by the corresponding allocations. The computationally more expensive proxy ϕ^{REROUTE} always dominates ϕ^{SHORT}; though both of these proxies are dominated by ϕ^{DEPOT}, ϕ^{MOAT}, ϕ^{BLEND}, and ϕ^{CHRIS} in all tests save one.

	10 Locations		15 Locations		20 Locations		All Data	
	$\overline{RMSE}$	σ	$\overline{RMSE}$	σ	$\overline{RMSE}$	σ	$\overline{RMSE}$	σ
Shortcut Distance	0.3850	0.0968	0.3342	0.0764	0.2992	0.0606	0.3727	0.0564
Re-routed Margin	0.2565	0.0699	0.2168	0.0533	0.1915	0.0424	0.2493	0.0488
Depot Distance	0.0994	0.0275	0.0950	0.0235	0.0893	0.0195	0.0978	0.0059
Moat-Packing	0.1617	0.0502	0.1437	0.037	0.1302	0.0293	0.1564	0.0197
Christofides	0.0495	0.0216	0.0526	0.0177	0.0523	0.0142	0.0520	0.0046
Blend	0.0710	0.0191	0.0742	0.0168	0.0733	0.0154	0.0745	0.0075

Table 2: Average root mean squared error ($\overline{RMSE}$) and standard deviation (σ) for the Synthetic data for games with 10, 15, and 20 locations. Lower is better.

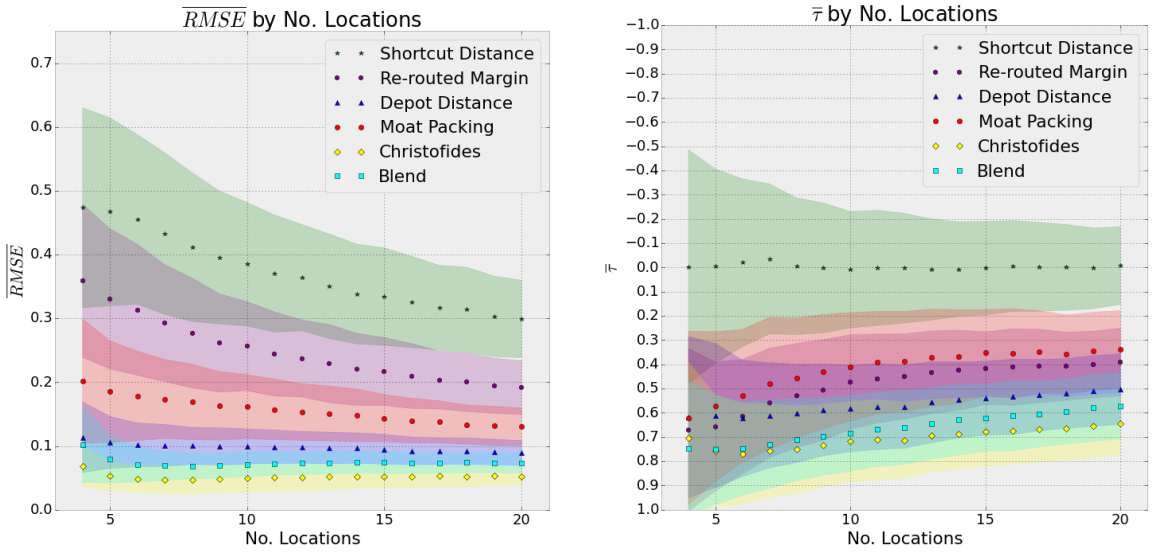


Figure 8: Performance of the proxies according to: (left) $\overline{RMSE}$ over the 1070 games generated for each number of locations, and (right) $\overline{\tau}$ over the 1070 games generated for each number of locations. The error bands correspond to plus or minus one standard deviation. The vertical axis of our $\overline{\tau}$ plot has been inverted for ease of comparison, i.e., more correlated lists are towards the bottom of the graph (1.0).

	10 Locations		15 Locations		20 Locations		All Data	
	$\overline{MAPE}$	σ	$\overline{MAPE}$	σ	$\overline{MAPE}$	σ	$\overline{MAPE}$	σ
Shortcut Distance	0.2802	0.1088	0.2278	0.0843	0.1944	0.0700	0.2605	0.1155
Re-routed Margin	0.1866	0.0805	0.1460	0.0596	0.1203	0.0461	0.1741	0.089
Depot Distance	0.0637	0.0238	0.0589	0.0226	0.0523	0.0193	0.0620	0.0261
Moat-Packing	0.1078	0.0452	0.0888	0.035	0.0722	0.0252	0.1003	0.0508
Christofides	0.0311	0.0147	0.0318	0.0137	0.0299	0.0108	0.0329	0.0158
Blend	0.0441	0.0154	0.0443	0.0155	0.0417	0.0145	0.0472	0.0224

Table 3: Average maximum absolute error ($\overline{MAPE}$) and standard deviation (σ) for the Synthetic data for games with 10, 15, and 20 locations. Lower is better.

Looking first at the error in the estimation of ϕ^{SV}, the top of Figure 8 depicts the $\overline{RMSE}$ and σ over 1070 games for each proxy as we increase the number of locations per game. The overall trend is positive with each proxy becoming more accurate (lower $\overline{RMSE}$) as we increase the number of locations. In the figure ϕ^{CHRIS} strictly dominates all of the other proxies in $\overline{RMSE}$ performance. However, we also see that ϕ^{BLEND} and ϕ^{DEPOT} are competitive with ϕ^{CHRIS} both in terms of $\overline{RMSE}$ and σ. ϕ^{BLEND} is the winner in this category for practical purposes as it offers performance extremely close to ϕ^{CHRIS}, with a tighter distribution on error than ϕ^{DEPOT}, for a fraction of the computation

time. Table 2 shows a more detailed breakdown of Figure 8 for particular numbers of locations. This table allows us to see that the $\overline{RMSE}$ for ϕ^{BLEND} never goes above 0.1 with $\sigma \leq 0.01$ for the larger instances that are commercially interesting.

Table 3 sheds more light on the types of error that the proxies are likely to make. Again we see that ϕ^{BLEND} achieves a better $\overline{MAPE}$ than any of the other proxies save ϕ^{CHRIS}, on average only overcharging in the worst case by $\approx 4.7\%$ with $\sigma = 2.2\%$ of the true Shapley value, a mere 1% more than ϕ^{CHRIS}. Again we see that ϕ^{DEPOT} is a fairly accurate proxy for ϕ^{SV}, only overcharging by $\approx 6\%$ with $\sigma = 2.6\%$ for the largest instanes tested. However, ϕ^{DEPOT} is strictly dominated by ϕ^{BLEND} in all error measures we considered and can be computed in similar time. Given that ϕ^{BLEND} is computable in a fraction of the time for ϕ^{CHRIS}, has competitive overall error, and scales up to and beyond commercially interesting problem sizes. It is the clear winner for this measure.

	10 Locations		15 Locations		20 Locations		All Data	
	$\overline{\tau}$	σ	$\overline{\tau}$	σ	$\overline{\tau}$	σ	$\overline{\tau}$	σ
Shortcut Distance	0.0098	0.2403	0.0031	0.1931	-0.0076	0.1604	-0.0027	0.0106
Re-routed Margin	0.4732	0.1947	0.4160	0.1505	0.3908	0.1397	0.4828	0.0892
Depot Distance	0.5815	0.1791	0.5400	0.1524	0.5018	0.1454	0.5659	0.0385
Moat-Packing	0.4098	0.2235	0.3526	0.1787	0.3392	0.1610	0.4190	0.0829
Christofides	0.7186	0.1663	0.6791	0.1430	0.6463	0.1286	0.7048	0.0374
Blend	0.6834	0.1567	0.6206	0.1385	0.5706	0.1369	0.6616	0.0593

Table 4: Average Kendall's tau rank correlation coefficient ($\overline{\tau}$) and Standard Deviation (σ) for the Synthetic data for games with 10, 15, and 20 locations. Higher is better; $+1$ means the two lists are perfectly correlated and -1 means the two lists are perfectly anti-correlated.

	10 Locations		15 Locations		20 Locations		All Data	
	% Sig.	% Top	% Sig.	% Top	% Sig.	% Top	% Sig.	% Top
Shortcut Distance	1.49%	19.81%	3.73%	9.62%	5.32%	6.91%	4.72%	10.45%
Re-routed Margin	18.13%	77.75%	49.90%	73.08%	60.74%	67.00%	53.15%	69.85%
Depot Distance	16.44%	68.59%	70.18%	51.40%	84.85%	46.26%	69.67%	52.04%
Moat-Packing	12.42%	71.68%	39.53%	62.42%	45.60%	56.82%	41.76%	61.13%
Christofides	29.90%	82.52%	90.18%	78.31%	96.54%	74.39%	85.28%	76.08%
Blend	29.90%	80.84%	89.15%	68.41%	94.11%	59.71%	83.12%	65.38%

Table 5: (Left columns) The percentage of games out of 1070 where τ is statistically significant ($p < 0.05$) between the ranking induced by ϕ^{SV} and the ranking induced by ϕ^{PROXY}. (Right columns) The percentage out of 1070 games where the most expensive element according to the raking induced by ϕ^{SV} matched the most expensive element in the ranking induced by ϕ^{PROXY}. Higher is better for both statistics.

Turning to the proxies' performance on ranking, the bottom of Figure 8 depicts the average Kendall's tau rank correlation coefficient ($\overline{\tau}$) and the standard deviation (σ) for 1070 games for each proxy as we increase the number of locations per game. The overall trend in this graph, as opposed to the top one, is slightly negative. As we increase the number of locations, the ranking computed by each proxy is increasingly uncorrelated with the ranking induced by ϕ^{SV}. On the positive side, ϕ^{CHRIS}, ϕ^{BLEND}, and ϕ^{DEPOT}, all return lists which have over a 0.6 correlation, i.e., 60% of the pairs of elements are ordered correctly. Table 4 gives a closer look at the results for particular numbers of locations. We see that ϕ^{CHRIS} and ϕ^{BLEND} both maintain τ near 0.6 across the range of problems, hence they correctly order most of the pairs of elements. These two proxies again strictly dominate all other proxies; even ϕ^{DEPOT} performs poorly when measured against $\overline{\tau}$.

Table 5 gives us a more nuanced look at the ranking results. We see again that for larger games the percentages of τ's that are statistically significant increase for all proxies for all locations, even as the τ's themselves decrease. This is because the lists are recovering a significant portion of the pairwise relations compared to the total number of pairwise relations. We see again that in terms of statistically significant τ's, ϕ^{CHRIS} and ϕ^{BLEND} strictly dominate all other proxies by almost 15% for all of the data considered. As an answer to the common customer question of *whose costing me the most*, the results are a bit more mixed. Comparing just the highest ranked elements we see that the performance of ϕ^{BLEND} drops below that of the performance of ϕ^{REROUTE}, a surpassingly strong proxy for this measure. However, if we want the top element according to ϕ^{SV} to be in the *top 3* elements according to ϕ^{PROXY}, ϕ^{BLEND} achieves this feat over 90% of the time. Though all other proxies see an increase in performance in this relaxed measure as well, only ϕ^{CHRIS} and ϕ^{BLEND} are above 90% for all numbers of locations studied. Hence, we again see that ϕ^{BLEND} provides strong performance at a practically computable running time across a range of game sizes.

7.2 Real-World Data

Measuring the performance of proxies on the Real-World corpus from Auckland, Canberra, and Sydney, we find that the overall quality of allocation is slightly lower compared to the measurements on the Synthetic corpus. We identified no significant performance differences between cities, and therefore report all data here as aggregate statistics over the Real-World corpus of 71 games with 10 locations and 48 games with 20 locations. Tables 6 to 9 provide an in-depth perspective on the performance of the proxies on the Real-World dataset with the same measures as the Synthetic dataset. Again we see that the performance of ϕ^{SHORT} and ϕ^{REROUTE} is strictly dominated according to all statistical measures by all the other proxies; except ϕ^{REROUTE}'s ability to select the most costly location with surprisingly high accuracy.

	10 Locations		20 Locations		All Games	
	$\overline{RMSE}$	σ	$\overline{RMSE}$	σ	$\overline{RMSE}$	σ
Shortcut Distance	0.4511	0.1477	0.3245	0.0929	0.3878	0.0633
Re-routed Margin	0.4380	0.1472	0.3030	0.0934	0.3705	0.0675
Depot Distance	0.1380	0.065	0.0838	0.0313	0.1109	0.0271
Moat-Packing	0.2692	0.1486	0.2088	0.0937	0.2390	0.0302
Christofides	0.1519	0.0823	0.1104	0.0529	0.1311	0.0207
Blend	0.1442	0.0687	0.0826	0.0292	0.1134	0.0308

Table 6: Average root mean squared error ($\overline{RMSE}$) and standard deviation (σ) for the Real-world data for the 71 games with 10 locations and 48 games with 20 locations. Lower is better.

	10 Locations		20 Locations		All Games	
	$\overline{MAPE}$	σ	$\overline{MAPE}$	σ	$\overline{MAPE}$	σ
Shortcut Distance	0.3678	0.1716	0.2462	0.1050	0.3187	0.1599
Re-routed Margin	0.3568	0.1695	0.2286	0.1012	0.3051	0.1588
Depot Distance	0.0835	0.0388	0.0390	0.0118	0.0655	0.0378
Moat-Packing	0.2196	0.1463	0.1498	0.0959	0.1914	0.1328
Christofides	0.1178	0.0780	0.0739	0.0536	0.1001	0.0725
Blend	0.0961	0.0523	0.0412	0.0134	0.0740	0.0493

Table 7: Average maximum absolute error ($\overline{MAPE}$) and standard deviation (σ) for the Real-world data for the 71 games with 10 locations and 48 games with 20 locations. Lower is better.

Turning first to the error in the estimation of ϕ^{SV}, we see that the results reported in Table 6 are strictly higher on every measure for every proxy compared to the results in Table 2, the corresponding test for the Synthetic dataset. We see again that the error decreases as we increase the number of locations for all proxies. The difference between Real-World and Synthetic does not render the proxies unuseable. Observe that the $\overline{RMSE}$ for ϕ^{BLEND} only increases by 0.01 between Synthetic and Real-world while the $\overline{RMSE}$ for ϕ^{DEPOT} only increases by about 0.003. This is a solid indicator for the usefulness of ϕ^{BLEND}, as none of the Real-World instances were included in the training set for the model. In an interesting twist, the more computationally expensive ϕ^{CHRIS} fares worse on the Real-World data, doubling its error (an increase of 0.05) with respect to the Synthetic dataset.

The increase in $\overline{RMSE}$ is not followed when looking at $\overline{MAPE}$. Comparing Table 7 to its Synthetic dataset partner Table 3, we see that both ϕ^{BLEND} and ϕ^{DEPOT} actually have a lower $\overline{MAPE}$ and a lower σ for the Real-World datasets with 20 locations. Observe that when comparing performance according to $\overline{MAPE}$ we see that ϕ^{BLEND} and ϕ^{DEPOT} are only separated by 1% of performance, and both strictly outperform all other metrics, even ϕ^{CHRIS}. We can see that ϕ^{BLEND} and ϕ^{DEPOT} are both reasonable proxies for ϕ^{SV} in the Real-world corpus, achieving an overall $\overline{RMSE}$ less than 0.09 and an absolute worst error per location of less than 0.05 (5% of true cost).

A possible explanation for the extremely good performance of ϕ^{DEPOT} requires a closer look at the distribution of costs in the Real-World dataset. As the locations along a route are heuristically allocated from a larger VRP, the allocations tend to cluster around a uniform allocation of around 0.05–0.08 per location, and many locations are equidistant from the depot. Consequently, the Real-World data seems to be drawn from a different distribution than the Synthetic data (i.e., the locations are not selected uniformly at random). Thus, the performance of ϕ^{BLEND} in both the ideal, uniformly random case and a strongly degenerate real-world case is a strong argument for the portability of ϕ^{BLEND} across domains.

	10 Locations		20 Locations		All Games	
	$\overline{\tau}$	σ	$\overline{\tau}$	σ	$\overline{\tau}$	σ
Shortcut Distance	0.0756	0.3015	0.0061	0.2148	0.0408	0.0348
Re-routed Margin	0.3651	0.2793	0.4734	0.1602	0.4193	0.0542
Depot Distance	0.1055	0.3416	0.3322	0.1932	0.2188	0.1134
Moat-Packing	0.3480	0.2504	0.3814	0.1721	0.3647	0.0167
Christofides	0.2457	0.3408	0.5403	0.1589	0.3930	0.1473
Blend	0.1498	0.3287	0.4093	0.1809	0.2796	0.1297

Table 8: Average Kendall's τ rank correlation coefficient ($\overline{\tau}$) and Standard Deviation (σ) for the Real-World data for the 71 games with 10 locations and 48 games with 20 locations. Higher is better; $+1$ means the two lists are perfectly correlated and -1 means the two lists are perfectly anti-correlated.

	10 Locations		20 Locations		All Data	
	% Sig.	% Top	% Sig.	% Top	% Sig.	% Top
Shortcut Distance	4.22%	12.67%	8.33%	18.75%	5.88%	15.12%
Re-routed Margin	28.16%	57.74%	72.91%	70.83%	46.21%	63.02%
Depot Distance	12.67%	53.52%	43.75%	60.41%	25.21%	56.30%
Moat-Packing	25.35%	60.56%	62.50%	56.25%	40.33%	58.82%
Christofides	22.53%	57.74%	89.58%	62.50%	49.57%	59.66%
Blend	14.08%	56.33%	68.75%	64.58%	36.13%	59.66%

Table 9: (Left columns) The percentage of Real-World data for the 71 games with 10 locations and 48 games with 20 locations where τ is statistically significant ($p < 0.05$) between the ranking induced by ϕ^{SV} and the ranking induced by ϕ^{PROXY}. (Right columns) The percentage of Real-World data for the 71 games with 10 locations and 48 games with 20 locations where the most expensive element according to the raking induced by ϕ^{SV} matched the most expensive element in the ranking induced by ϕ^{PROXY}. Higher is better for both statistics.

Tables 8 and 9 give an indication of how the proxies perform in terms of ranking. A closer look at Table 8 reveals the difference between ϕ^{BLEND} and ϕ^{DEPOT}. Again, comparing against the results the Synthetic dataset shows that all proxies perform strictly worse on the Real-world data, except ϕ^{SHORT} which manages to go from a *negative* list correlation on the Synthetic dataset to a (barely) positive correlation on the Real-world dataset. Judging the performance of $\overline{\tau}$ we see that most of the proxies are still recovering about 50% of the pairwise comparisons on the Real-World data. Again, we also see the good performance of ϕ^{REROUTE} on the ranking metric. Additionally, for games with 20 locations, ϕ^{REROUTE}, ϕ^{CHRIS}, ϕ^{MOAT}, and ϕ^{BLEND} have about the same $\overline{\tau}$, with ϕ^{CHRIS} the best.

A review of Table 9 reveals that while the measure of $\overline{\tau}$ is lower overall, the majority of the ranking correlations are still statistically significant for 20 location games. At first glance the proxies appears to not hold up when looking at only the top element. Every proxy sees decreased performance to $\approx 60\%$ accuracy when selecting the top element, and ϕ^{REROUTE} has the best performance, followed by ϕ^{BLEND} and ϕ^{CHRIS}. Relaxing the notion of top (most costly) element as we did in the Synthetic data, i.e., that the top element according to ϕ^{SV} is in the top 3 elements according to ϕ^{PROXY}, ϕ^{BLEND} outperforms all other proxies (including ϕ^{REROUTE}) on the 20 location games with 93% accuracy, and comes within 3% of outperforming ϕ^{REROUTE} on the entire corpus or Real-world data with 79% accuracy.

In summary we see that the proxies perform worse in terms of both $\overline{RMSE}$ and $\overline{\tau}$ in the Real-World dataset than they do on the Synthetic dataset. In all of our testing we see that ϕ^{BLEND}, ϕ^{DEPOT}, and ϕ^{CHRIS} out perform the other proxies on the majority of measures. When comparing the proxies against a variety of decision criteria including practical running time, overall numerical error, and ranking performance, ϕ^{BLEND} emerges as the clear winner and overall most consistent performer on both the Synthetic and Real-world data.

8. Related Work

The theory of cooperative games has a rich history in which various solution concepts for allocating costs and other quantities have been proposed (Peleg & Sudhölter, 2007; Young, 1994). In addition to the Shapley value, other solution concepts include the core, the nucleolus and the bargaining set. Of these, the Shapley value is considered the "most important" allocation scheme in cooperative game theory (Winter, 2002).

Application of the Shapley value spans well beyond transportation setting. For example, the Shapley value has been applied in allocating the cost of network infrastructure (Koster, 2009; Marinakis, Migdalas, & Pardalos, 2008), promoting collaboration between agents (Zlotkin & Rosenschein, 1994) by prescribing an allocation that incentivises agents to collaborate in the completion of tasks, and as an incentive compatible way to share departmental costs in corporations (Young, 1985). Considering applications in networks more broadly, use of the Shapley value follows a general framework, where agents correspond to the nodes (or edges) of a graph (Curiel, 2008; Koster, 2009; Marinakis et al., 2008; Tijs & Driessen, 1986; Aziz & de Keijzer, 2014). Here the definition of the characteristic function depends on the application domain, with proposed evaluations based on: (i) the size of maximum matching, (ii) network flow, (iii) the weight of a minimum spanning tree, and (iv) the weight of a Hamiltonian cycle (Curiel, 2008; Deng & Fang, 2008). Allocation concepts are not solely devised and employed for allocating costs. For example, the Shapley value has been used to measure the *importance* of agents in social networks (Moretti & Patrone, 2008),

and to measure the *centrality* of nodes in networks (Michalak, Aadithya, Szczepanski, Ravindran, & Jennings, 2013).

Another solution concept that has been used to gauge the importance of agents is the *Banzhaf value* (Banzhaf III, 1964). The Banzhaf value is defined for simple voting games – i.e. cooperative games in which the value of the coalition is either zero or one – but the Banzhaf value of an agent can suitably be extended to general cooperative games. However, even within the context simple voting games, the Banzhaf value is more suitable for measuring the influence of an agent and less suitable for *allocate* power between agents (Felsenthal & Machover, 1998). Since our focus is to allocate costs, we focus on the Shapley value.

While solution concepts from the theory of transferable utility (TU) cooperative games (Peleg & Sudhölter, 2007; Chalkiadakis et al., 2011) have been used for allocations of costs, the Shapley value has rarely received serious attention in the transportation science literature. The associated computational cost is prohibitively high for the general case, and consequently strong notions of fairness are often taken to be a secondary consideration. Though *ApproShapley* is an FPRAS (fully polynomial-time randomized approximation scheme) for computing the Shapley value if the game is convex (Liben-Nowell, Sharp, Wexler, & Woods, 2012), this does not apply for the domain considered in this work. The website *Spliddit* uses the Shapley value to split cab fares between up to 6 people (Goldman & Procaccia, 2014).

Other prominent TU game solution concepts are *nucleolus* and *core*. TSGs are introduced in Potters (1992), where in addition to describing that game, the authors describe a variety of game known as the *routing game*.[9] For the latter an auxiliary constraint forces locations to be visited, in any coalition, in the order they are traversed by a specific tour. Assuming that the tour corresponds to the optimal for the underlying TSP, then the game has a non-empty core. Derks and Kuipers (1997) presented a quadratic-time procedure for computing a core allocation of the routing game. They also characterize suboptimal tours that specify routing games with non-empty cores. It should be noted that there are no known tractable procedures to compute a tour which guarantees the core is non-empty for the routing game. Conditions for the non-emptiness of the core in TSGs were further developed by Tamir (1989). We have already noted that Faigle et al. (1998) developed a procedure to calculate a multiplicative ε-core allocation for Euclidean TSGs. Yengin (2012) develop a notion of a *fixed route* game with *appointments* which admits a tractable procedure for computing Shapley values. That model is not suitable for typical scenarios that involve the delivery of goods to locations from a depot. TU concepts in TSGs and routing games are developed for a practical gas delivery application by Engevall et al. (1998).

Turning our attention to vehicle routing problems and transportation settings more generally, Göthe-Lundgren, Jörnsten, and Värbrand (1996) develop a column generation procedure to calculate the *nucleolus* of a homogeneous vehicle routing problem, i.e., all vehicles are equivalent. In doing so they develop a procedure to determine if the core of that vehicle routing game is empty. Engevall et al. (2004) extend that work for a very practical setting of distributing gas using a *heterogeneous* fleet of vehicles. More recently Özener et al. (2013) examine a number of solution concepts—including allocations derived according to the nested moat-packing of Faigle et al. (1998), and a highly specialized approximation of the Shapley allocation—in deriving cost allocations for real-world *inventory routing* problems. They show that TU game allocations, espe-

9. Note that the journal publication of Potters et al. (1992) extends a technical report introducing the game as early as 1987.

cially core/duality-based allocations, have significant advantages over the existing cost allocations which their industrial client was using.

9. Conclusions and Future Work

We studied the problem of fairly apportioning costs in transportation scenarios, specifically TSGs. The Shapley value is an appealing division concept for this task as its axiomatic fairness properties are ones most appreciated by our commercial partners. Since the Shapley value cannot be evaluated in reasonable time, we considered a number of proxies for the Shapley value. We examined proxy performance both in terms of their approximation quality with respect to the Shapley value and the induced ranking of locations by Shapley value, a key question for operational and business concerns. The stand-out proxies with respect to both measures as tested on Synthetic and Real-world data are ϕ^{CHRIS} and ϕ^{BLEND}, a mixture of ϕ^{DEPOT} and ϕ^{MOAT}. However, when taking computation time into account and the ability to scale to problems of commercial interest: around 30 locations per route and over 600 total routes for a delivery day, only ϕ^{BLEND} remains feasible.

A key extensions of our work is the more general setting of vehicle routing games (VRPs). The Shapley value would be useful to quantify the importance of location synergies that are unique to the multi-vehicle model. The transport companies we interact with desire to understand the impact of time windows (both the duration and position of allowable service times), and the effect of delivery frequency on allocated costs. Thus, a highly motivated and rich variety of problems is available for future work. Additionally, future research should consider weighted Shapley values for situations where some coalitions (and therefore margins) are more likely to occur than others. Formal approximation ratios, to complement the strong empirical evidence we obtained, is an important subject for future research. There also remains the need for formal studies which employ proxy allocations to inform solutions to hard optimisation problems in transportation domains. Finally, scaling to larger transportation scenarios may require abstracting locations in a meaningful way. An approximation approach that may be fruitful here was proposed by Soufiani, Charles, Chickering, and Parkes (2014), where agents are partitioned into groups and assigned weights within those groups in a novel and effective way.

Acknowledgments

Data61/CSIRO (formerly known as NICTA) is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. Casey Cahan was supported by an Summer Research Scholarship at The Australian National University. Toby Walsh also receives support from the Asian Office of Aerospace Research and Development (AOARD 124056) and the German Federal Ministry for Education and Research through the Alexander von Humboldt Foundation.

We would like to thank Stefano Moretti and Patrice Perny from LIP6; Hossein Azari Soufiani from Harvard University; David Rey and Vinayak Dixit from the rCiti Project at the University of New South Wales School of Civil and Environmental Engineering, Tommaso Urli from Data61 and ANU; and the reviewers and attendees of the 5th Workshop on Cooperative Games in MultiAgent Systems (CoopMAS-2014) for their helpful feedback and comments on early version of this work.

References

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2007). *The traveling salesman problem: a computational study*. Princeton University Press.

Aziz, H., & de Keijzer, B. (2014). Shapley meets Shapley. In *Proceeding of the 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, pp. 99–111.

Bachrach, Y., Markakis, E., Resnick, E., Procaccia, A. D., Rosenschein, J. S., & Saberi, A. (2010). Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, *20*(2), 105–122.

Banzhaf III, J. F. (1964). Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, *19*, 317–343.

Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, *9*(1), 61–63.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the shapley value based on sampling. *Comput. Oper. Res.*, *36*(5), 1726–1730.

Chalkiadakis, G., Elkind, E., & Wooldridge, M. (2011). Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *5*(6), 1–168.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem.. Tech. rep., DTIC Document.

Conitzer, V., & Sandholm, T. (2006). Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, *170*(6), 607–619.

Cook, W. J., Cunningham, W. H., Pulleylank, W. R., & Schrijver, A. (1998). *Combinatorial Optimization*. John Wiley & Sons, Inc.

Corder, G. W., & Foreman, D. I. (2009). *Nonparametric statistics for non-statisticians: a step-by-step approach*. Wiley.

Cornuéjols, G., Naddef, D., & Pulleyblank, W. (1985). The traveling salesman problem in graphs with 3-edge cutsets. *Journal of the ACM*, *32*(2), 383–410.

Curiel, I. (2008). Cooperative combinatorial games. In Chinchuluun, A., Pardalos, P., Migdalas, A., & Pitsoulis, L. (Eds.), *Pareto Optimality, Game Theory And Equilibria*, Vol. 17 of *Springer Optimization and Its Applications*, pp. 131–157. Springer New York.

Deng, X., & Fang, Z. (2008). Algorithmic cooperative game theory. In Chinchuluun, A., Pardalos, P. M., Migdalas, A., & Pitsoulis, L. (Eds.), *Pareto Optimality, Game Theory And Equilibria*, Vol. 17 of *Springer Optimization and Its Applications*. Springer-Verlag.

Derks, J., & Kuipers, J. (1997). On the core of routing games. *International Journal of Game Theory*, *26*(2), 193–205.

Engevall, S., Göthe-Lundgren, M., & Värbrand, P. (1998). The traveling salesman game: An application of cost allocation in a gas and oil company. *Annals of Operations Research*, *82*, 203–218.

Engevall, S., Göthe-Lundgren, M., & Värbrand, P. (2004). The heterogeneous vehicle-routing game. *Transportation Science*, *38*(1), 71–85.

Faigle, U., & Kern, W. (1993). On some approximately balanced combinatorial cooperative games. *ZOR Methods and Models of Operations Research*, *38*(2), 141–152.

Faigle, U., Fekete, S., Hochstättler, W., & Kern, W. (1998). On approximately fair cost allocation in euclidean tsp games. *Operations-Research-Spektrum*, *20*(1), 29–37.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2007). A randomized method for the Shapley value for the voting game. In *Proceedings of the 6th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 07)*, pp. 157–165, New York, New York, USA.

Fatima, S. S., Wooldridge, M., & Jennings, N. R. (2008). A linear approximation method for the Shapley value. *Artificial Intelligence*, *172*(14), 1673–1699.

Felsenthal, D. S., & Machover, M. (1998). *The Measurement of Voting Power: Theory and Practice, Problems and Paradoxes*. Edward Elgar Cheltenham.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, Vol. 43. Springer.

Goldman, J., & Procaccia, A. D. (2014). Spliddit: Unleashing fair division algorithms. *Journal of the ACM*, *13*(2), 41–46.

Göthe-Lundgren, M., Jörnsten, K., & Värbrand, P. (1996). On the nucleolus of the basic vehicle routing game. *Mathematical Programming*, *72*(1), 83–100.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial & Applied Mathematics*, *10*(1), 196–210.

Ieong, S., & Shoham, Y. (2005). Marginal contribution nets: a compact representation scheme for coalitional games. In *Proceedings of the 6th ACM conference on Electronic Commerce (EC 06)*, pp. 193–202.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93.

Kilby, P., & Verden, A. (2011). Flexible routing combing constraint programming, large neighbourhood search, and feature-based insertion. In *2nd Workshop on Artificial Intelligence and Logistics*. Barcelona, Spain.

Kimms, A., & Kozeletskyi, I. (2015). Shapley value-based cost allocation in the cooperative traveling salesman problem under rolling horizon planning. *EURO Journal on Transportation and Logistics*, 1–22.

Koster, M. (2009). *Cost Sharing*. Springer-Verlag New York.

Leech, D. (2003). Computing power indices for large voting games. *Management Science*, *49*(6), 831–837.

Liben-Nowell, D., Sharp, A., Wexler, T., & Woods, K. (2012). Computing shapley value in supermodular coalitional games. In *18th International Conference on Computing and Combinatorics (COCOON 2012)*, pp. 568–579.

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., & Rogers, A. (2013). Bounding the estimation error of sampling-based shapley value approximation with/without stratifying. *CoRR*, *abs/1306.4265*.

Mann, I., & Shapley, L. S. (1960). Values for large games IV: Evaluating the electoral college by monte carlo. Technical report, The RAND Corporation, Santa Monica, CA, USA.

Mann, I., & Shapley, L. S. (1962). Values for large games IV: Evaluating the electoral college exactly. Technical report, The RAND Corporation, Santa Monica, CA, USA.

Marinakis, Y., Migdalas, A., & Pardalos, P. M. (2008). Cost allocation in combinatorial optimization games. In Chinchuluun, A., Pardalos, P., Migdalas, A., & Pitsoulis, L. (Eds.), *Pareto Optimality, Game Theory And Equilibria*, Vol. 17 of *Springer Optimization and Its Applications*, pp. 217–244. Springer New York.

Michalak, T. P., Aadithya, K. V., Szczepanski, P. L., Ravindran, B., & Jennings, N. R. (2013). Efficient computation of the Shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research*, *46*, 607–650.

Moretti, S., & Patrone, F. (2008). Transversality of the Shapley value. *TOP*, *16*(1), 1–41.

Natrella, M., Croarkin, C., & Guthrie, W. (2012). *NIST/SEMATECH e-Handbook of Statistical Methods*. U.S. Department of Commerce. URL: http://www.itl.nist.gov/div898/handbook/.

Owen, G. (1972). Multilinear extensions of games. *Management Science*, *18*(5-part-2), 64–79.

Özener, O. O., Ergun, O., & Savelsbergh, M. (2013). Allocating cost of service to customers in inventory routing. *Oper. Res.*, *61*(1), 112–125.

Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley Publishing Company, Inc.

Papapetrou, P., Gionis, A., & Mannila, H. (2011). A Shapley value approach for influence attribution. In *Proceedings of the 2011 European Conference on Machine Learning and Principles of Knowledge Discovery in Databases (ECML PKDD 2011)*, pp. 549–564. Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peleg, B., & Sudhölter, P. (2007). *Introduction to the Theory of Cooperative Games*. Springer.

Potters, J. A., Curiel, I. J., & Tijs, S. H. (1992). Traveling salesman games. *Mathematical Programming*, *53*(1-3), 199–211.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the ACM*, *23*(3), 555–565.

Shapley, L. S. (1953). A value for *n*-person games. In Kuhn, H., & Tucker, W. W. (Eds.), *Contributions to the Theory of Games*, Vol. 2 of *Annals of Mathematical Studies*. Princeton University Press.

Soufiani, H. A., Charles, D. J., Chickering, D. M., & Parkes, D. C. (2014). Approximating the shapley value via multi-issue decomposition. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 14)*, pp. 1209–1216.

Tamir, A. (1989). On the core of a traveling salesman cost allocation game. *Operations Research Letters*, *8*(1), 31–34.

Tijs, S. H., & Driessen, T. S. H. (1986). Game theory and cost allocation problems. *Management Science*, *32*(8), 1015–1028.

Winter, E. (2002). The Shapley value. In *Handbook of Game Theory with Economic Applications*, chap. 53, pp. 2025–2054. Elsevier.

Yengin, D. (2012). Appointment games in fixed-route traveling salesman problems and the Shapley value. *International Journal of Game Theory*, *41*(2), 271–299.

Young, H. P. (1985). Producer incentives in cost allocation. *Econometrica*, *53*(4), 757–765.

Young, H. P. (1994). Cost allocation. In *Handbook of Game Theory with Economic Applications*, Vol. 2, pp. 1193–1235. Elsevier B.V.

Young, H. P. (1985). Monotonic solutions of cooperative games. *International Journal of Game Theory*, *14*(2), 65–72.

Zlotkin, G., & Rosenschein, J. S. (1994). Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994)*, pp. 432–437.