Efficient computation of cost allocations for the Vehicle Routing Problem

Phil Kilby and Dan Popescu

CSIRO Data61 and ANU, Canberra ACT 2601, Australia

Outline of the presentation

- Outline of the problem: how to “fairly” allocate route costs for VRP
- Briefly explain why (relatively) simple solutions do not work
- Briefly describe ideal solution, and why we may not be able to achieve it
- In depth description of our proposed approximate solutions
- Comparative performance against state-of-the-art
- Conclusions

Problem statement

- Travelling Salesman Problem (TSP): to optimally serve a distributed set of clients using the shortest possible route: famously non-trivial.
- Once TSP is solved, still non-trivial how to distribute overall cost to all clients served. **This is the problem we address.**
- Some simple and intuitive ideas of cost sharing:
  - assign costs in proportion to (straight line or shortest path) distance to the depot
  - by computing marginal cost of each customer to grand route
  - unfortunately ALL of the simple solutions can fail quite badly!

Simple solutions illustrated

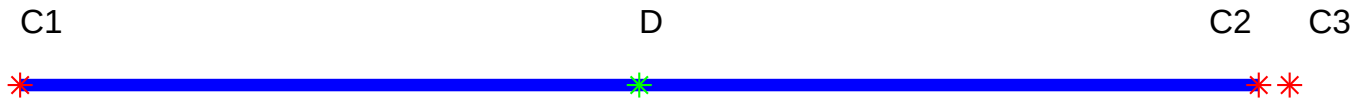


Diagram illustrating why “simple” solutions fail:

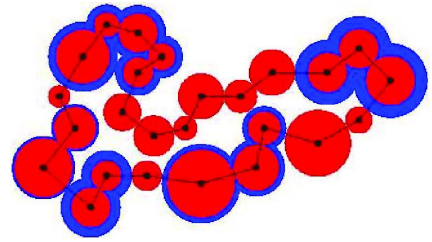
1. Fair cost structure: $(\frac{1}{2},\frac{1}{4},\frac{1}{4})$
2. Distance from depot: $(\frac{1}{3},\frac{1}{3},\frac{1}{3})$
3. Marginal cost to grand tour: $(1,0,0)$

The optimal solution

- Based on the Shapley value *[Shapley (1953)]*:
- $\mathcal{N} = \{1, 2, ..., n\}$ players, cost $c(\mathcal{S})$ for any $\mathcal{S} \subset \mathcal{N}$.
- $\Phi_i = \sum_{\mathcal{S} \subset \mathcal{N} \setminus \{i\}} \frac{|\mathcal{S}|!(n-|\mathcal{S}|-1)!}{n!} (c(\mathcal{S} \cup \{i\}) - c(\mathcal{S}))$
- $\Phi_i = \frac{1}{n} \sum_{s=0}^{n-1} \frac{1}{\binom{n-1}{s}} \sum_{\substack{\mathcal{S} \subset \mathcal{N} \setminus \{i\} \\ |\mathcal{S}|=s}} (c(\mathcal{S} \cup \{i\}) - c(\mathcal{S}))$
- **Good news and bad news:**
  - perfect solution!
  - high computational cost, unpractical for $n > 20$.

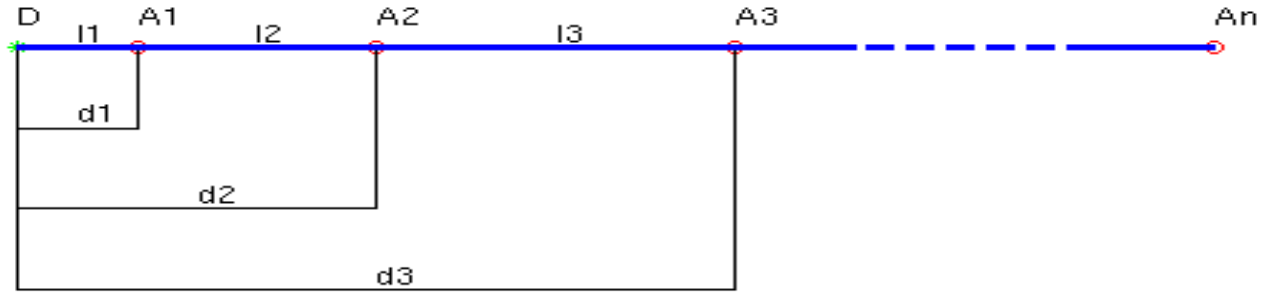
Approximate solutions

- To better balance the good and bad news:
  - practical computational times
  - some errors, but not too bad
- *Nested moat* packing:



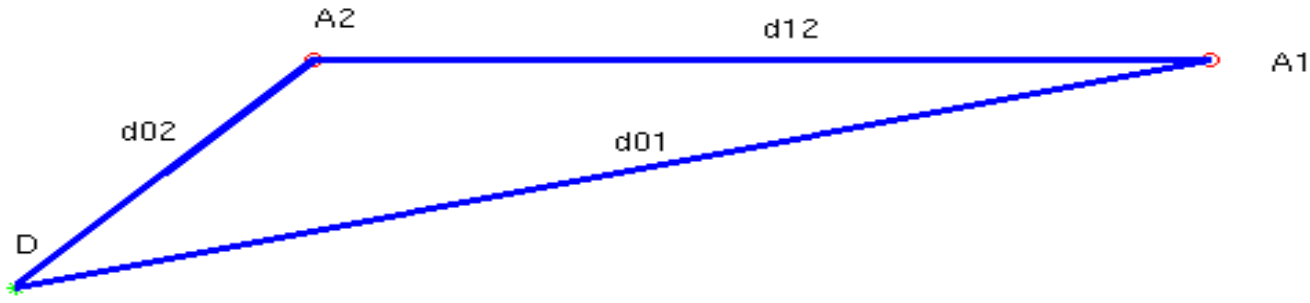
  - draw *moats* of constant width around locations
  - assign costs based on grand tour crossings with moats
- *Christofides* approximation of TSP
- *ApproShapley*: random permutation subsampling
  - $\Phi_i = \frac{1}{n!} \sum_{\pi \in \mathcal{S}_n} (c(\pi_i \cup \{i\}) - c(\pi_i))$
  - π_i is the set of elements preceding i in π

Our approach: from 1D case



- *Airport problem* [Littlechild (1973)]:
  - $\Phi_k^* = \sum_{j=1}^{k} \frac{2l_j}{n-j+1}$
  - exact value for 1D case
- Can be reformulated as:
  - $\Phi_n^* = 2d_n - \sum_{i=1}^{n-1} \frac{2d_{n-i}}{i(i+1)}$
  - or $\Phi_n^* = 2d_n - Sc(2d_1, 2d_2, \cdots, 2d_{n-1})$
  - *Shared Contribution* function: $Sc(x_1, x_2, \cdots, x_k) = \sum_{i=1}^{k} \frac{y_i}{i(i+1)}$ and $[Y]$ = sort($[X]$).
  - in general: $\Phi_k^* = 2d_k - Sc(2d_1, ..., 2d_{k-1}, 2d_k, 2d_k, \cdots, 2d_k)$
- If both $\{A_i\}$ (right of D) and $\{B_j\}$ (left of D):
  - A's and B's Shapley values as above; sets A, B act independently of each other.

The multidimensional case

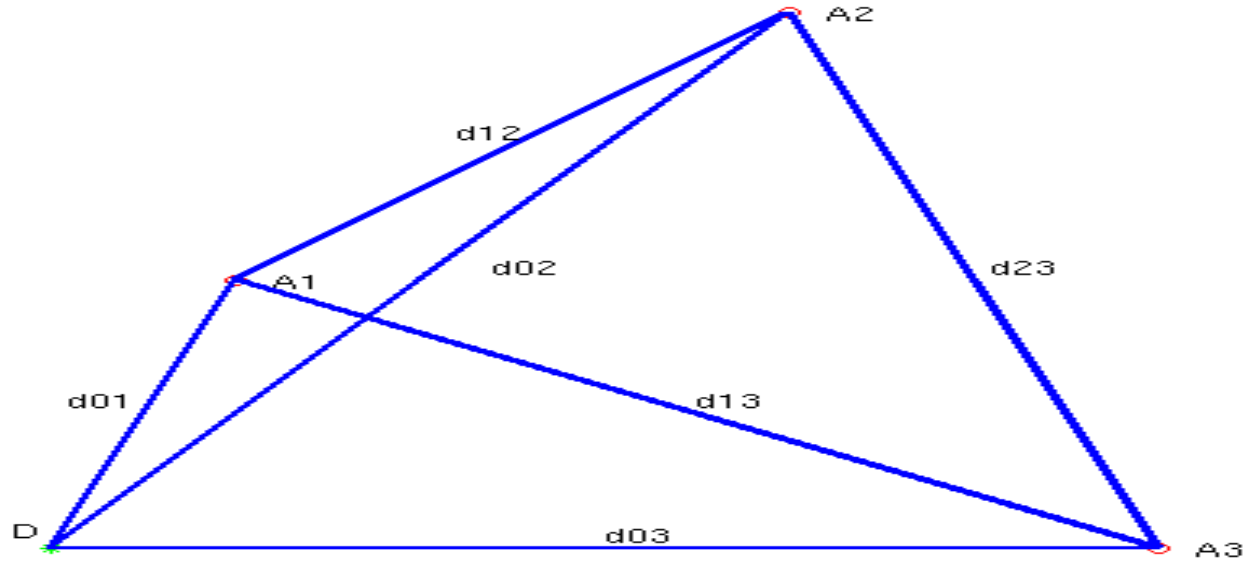


- Case with $n = 2$ locations and depot:
  - $\Phi_1 = \frac{1}{2}(2d_{01} + (d_{01} + d_{12} + d_{02} - 2d_{02})) = 2d_{01} - \frac{1}{2}(d_{01} + d_{02} - d_{12})$
  - $\Phi_2 = 2d_{02} - \frac{1}{2}(d_{01} + d_{02} - d_{12})$
- We define $n \times n$ *shared distance matrix* $\hat{s}$:
  - $s_{ij} = d_{0i} + d_{0j} - d_{ij}$
- Then we can reformulate:
  - $\Phi_1 = s_{11} - \frac{1}{2}s_{12} = s_{11} - Sc(s_{12})$
  - $\Phi_k^* = s_{kk} - Sc(s_{1k}, ..., s_{k-1,k}, s_{k+1,k}, \cdots, s_{nk})$
- **Generalize as** *Appro-O(1):*
  - $\Phi_k^{(1)} = s_{kk} - Sc(s_{1k}, ..., s_{k-1,k}, s_{k+1,k}, \cdots, s_{nk})$

Appro-O(1) merits

- Uniformly covers everything so far:
  - all of the 1D case
  - $n = 2$ multidimensional
- Good approximation for $n > 2$ multidimensional
  - as we shall see from comparative analysis ...
- Low computational complexity n^2
- What if we want to refine more?

Beyond Appr-O(1)

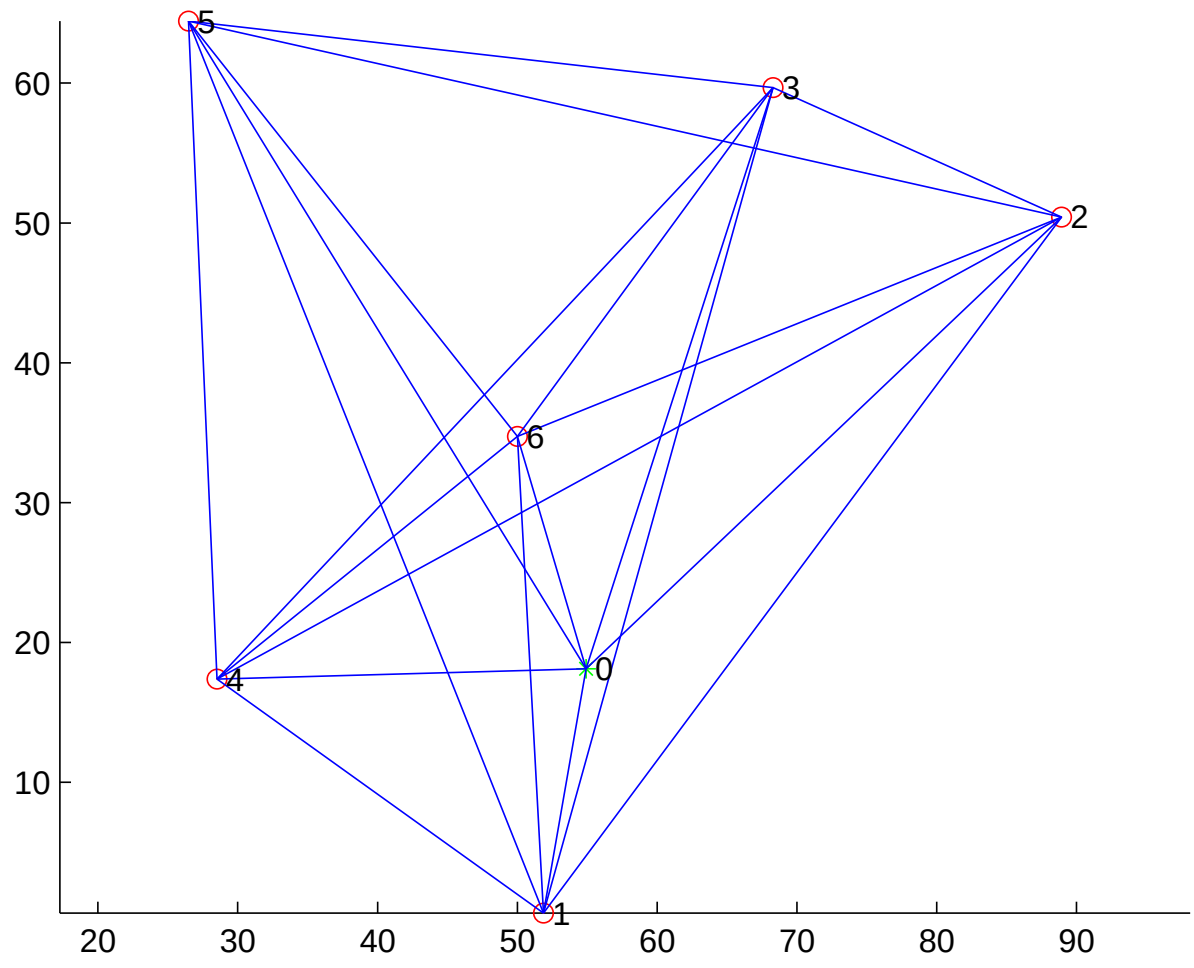


- Case with $n = 3$ locations and depot:
  - $\Phi_1 = s_{11} - \frac{1}{2}(s_{12} + s_{13}) + \frac{1}{3}\min\{s_{12}, s_{13}, s_{23}\}$
- If $\min\{s_{12}, s_{13}, s_{23}\}$ is either s_{12} or s_{13}:
  - $\Phi_1 = s_{11} - Sc(s_{12}, s_{13})$
- Reformulation with *formal intersection* as:
  - $\Phi_1^{(1)} = s_{11} - \frac{1}{2}(s_{12}+s_{13}) + \frac{1}{3}\min\{s_{12}, s_{13}\} = s_{11} - \frac{1}{2}(s_{12}+s_{13}) + \frac{1}{3} \cap \{s_{12}, s_{13}\}$
  - $\Phi_1 = s_{11} - \frac{1}{2}(s_{12}+s_{13}) + \frac{1}{3}\min\{s_{12}, s_{13}, s_{23}\} = s_{11} - \frac{1}{2}(s_{12}+s_{13}) + \frac{1}{3} \cap \{s_{12}, s_{13}\}$

Appro-O(2)

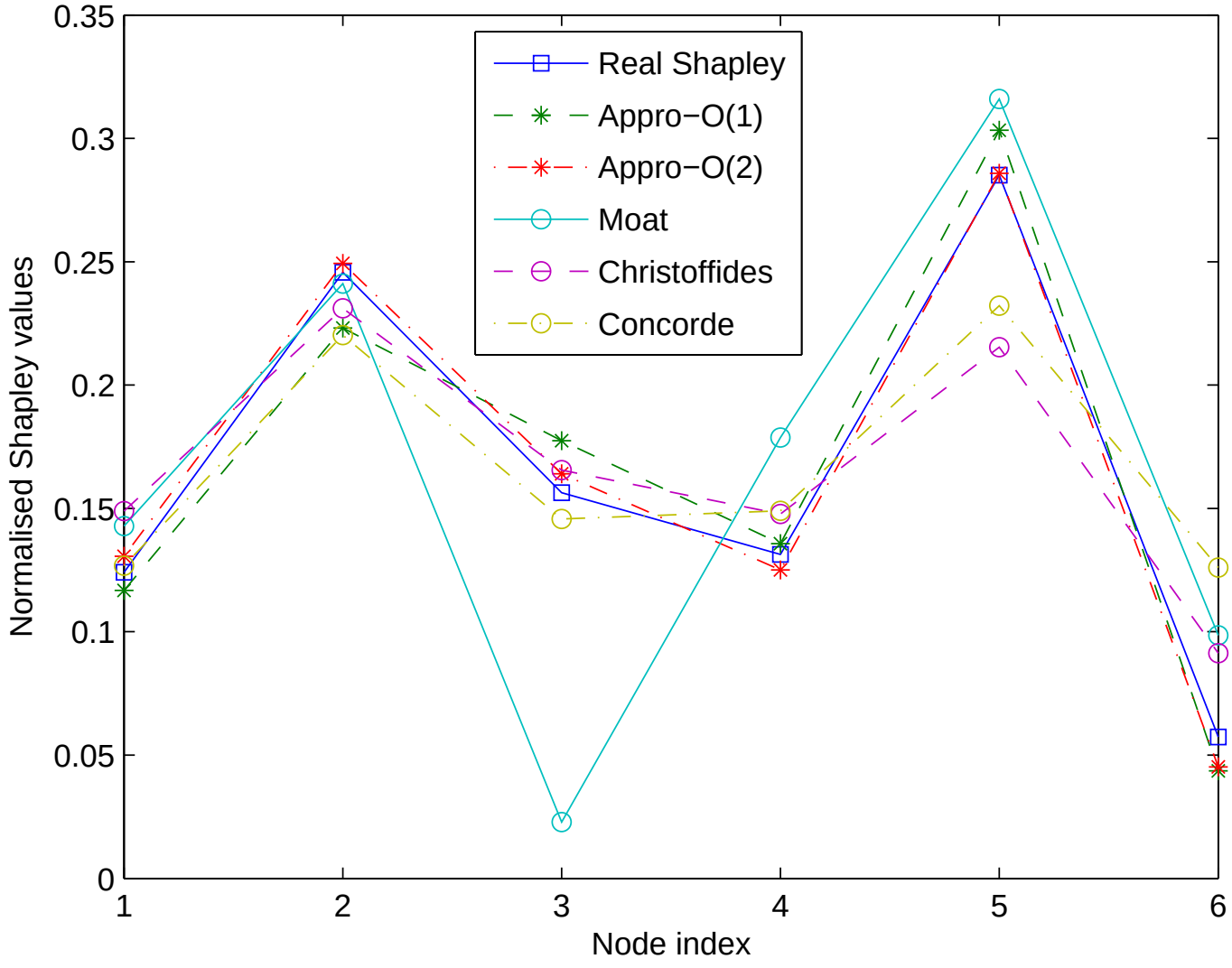
- 1D case fits perfectly to pattern:
  - $\Phi_k^* = s_{kk} - (\frac{1}{2}\sum_{i\neq k} s_{ki} - \frac{1}{3}\sum_{i\neq j\neq k} \cap\{s_{ki}, s_{kj}\} + \frac{1}{4}\sum_{i\neq j\neq l\neq k} \cap\{s_{ki}, s_{kj}, s_{kl}\} - \frac{1}{5}\cdots)$
  - where each $\cap$ is the min of its constituents
- Next refinement:
  - replace each $\cap$ of order ≥ 2 by min 2-intersection
  - $\cap\{s_{ki}, s_{kj}\} = \min\{s_{ki}, s_{kj}, s_{ij}\}$
  - still has exponential complexity
- Appro-O(2):
  - sorts $\hat{s}$ matrix in *standard form*
  - assigns rank r_{ij} to each intersection $\cap\{s_{ki}, s_{kj}\}$
  - $\Phi_k^{(2)} = s_{kk} - \frac{1}{2}\sum_{i\neq k} s_{ki} + \frac{2}{(r_{ij}+1)(r_{ij}+2)(r_{ij}+3)}\sum_{i\neq j\neq k} \cap\{s_{ki}, s_{kj}\}$
  - $O(n^3)$ computational complexity

Comparative results (1)



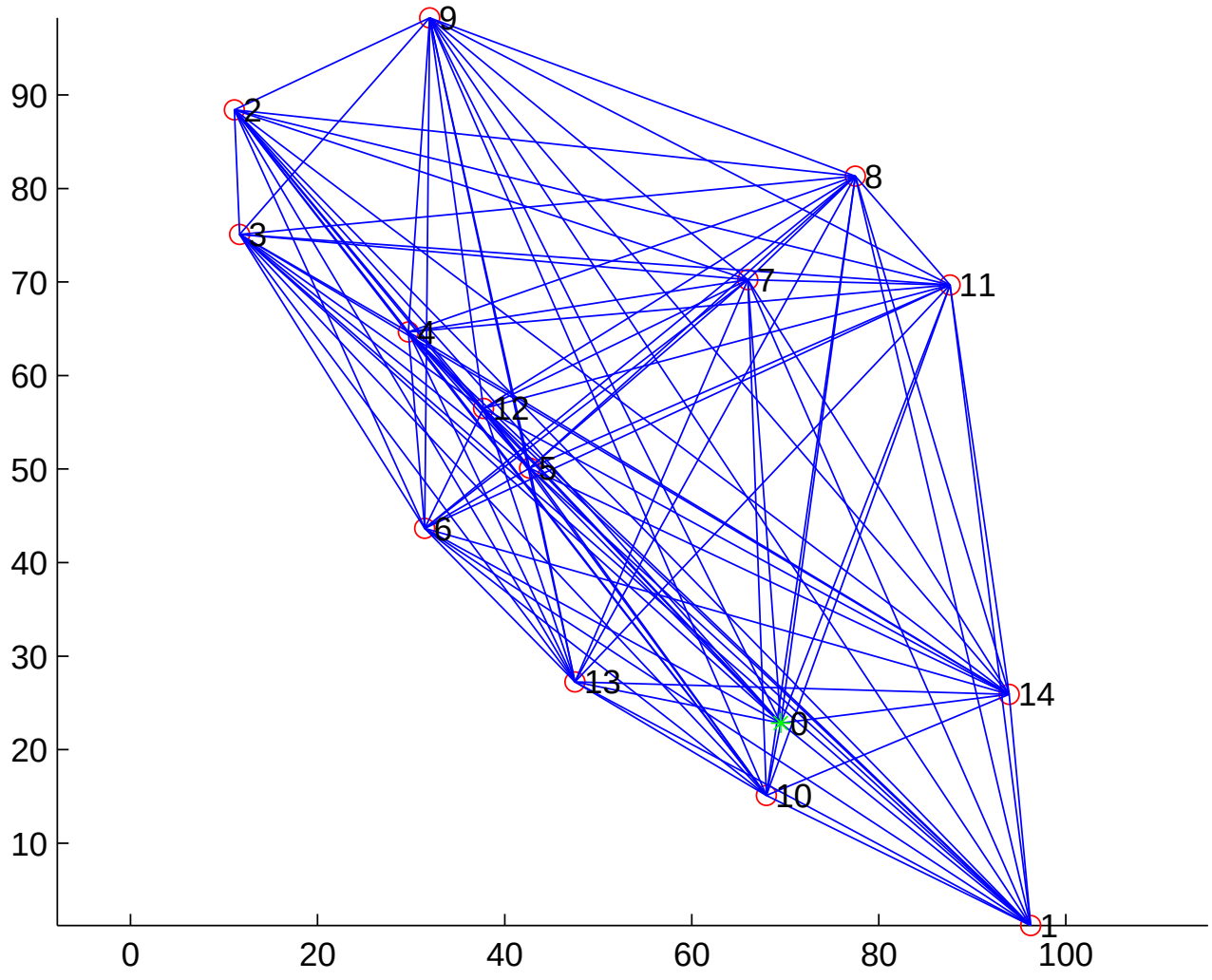
Example of Euclidean network with 6 customers (red dots) and depot (green dot).

Comparative results (1)*



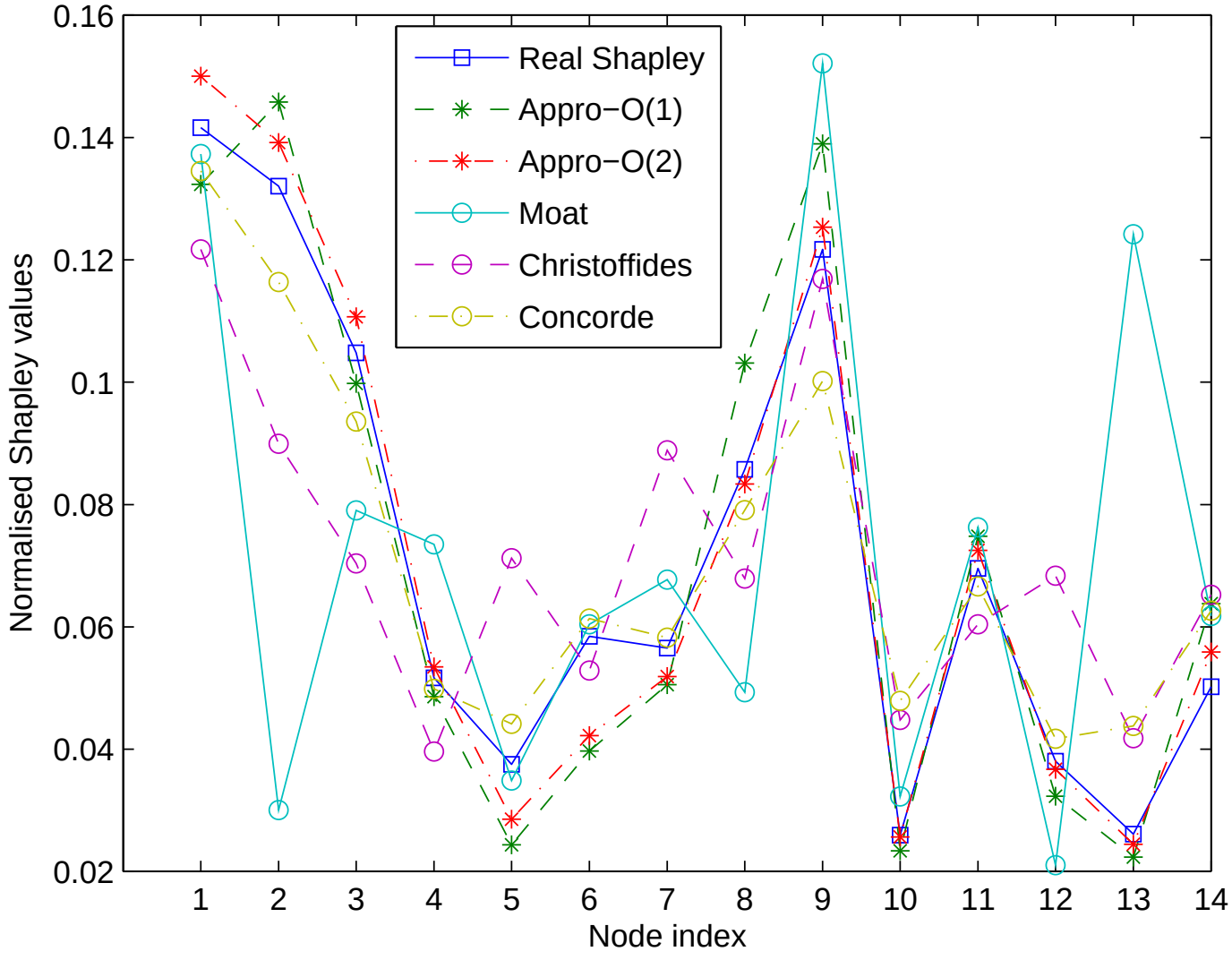
Results on example of Euclidean network with 6 customers.

Comparative results (2)



Example of Euclidean network with 14 customers (red dots) and depot (green dot).

Comparative results (2)*



Results on example of Euclidean network with 14 customers.

Comparative results (3)

Size	Appro-O(1)	Appro-O(2)	Moat	Christofides	Concorde
5	0.0107	0.0003	0.0754	0.0640	0.0715
7	0.0125	0.0024	0.0725	0.0418	0.0391
10	0.0129	0.0053	0.0582	0.0291	0.0212
12	0.0128	0.0067	0.0513	0.0249	0.0162
15	0.0126	0.0078	0.0426	0.0205	0.0114
20	0.0117	0.0082	0.0325	0.0162	0.0074

Average RMS error against normalised true Shapley values for several approximation methods, includes tests of [Aziz et. al (2016)]. For each network size, the average is computed over 1000 randomly generated networks samples.

Comparative results (4)

Size	Appro-O(1)	Appro-O(2)	Moat	Christofides	Concorde
5	0.0164	0.0006	0.1096	0.2111	0.2075
7	0.0221	0.0041	0.1196	0.0704	0.0678
10	0.0254	0.0106	0.1065	0.0542	0.0406
12	0.0263	0.0143	0.1013	0.0495	0.0320
15	0.0275	0.0179	0.0906	0.0438	0.0235
20	0.0274	0.0207	0.0740	0.0375	0.0160

Average MAX error against normalised true Shapley values for several approximation methods, includes tests of [Aziz et. al (2016)]. For each network size, the average is computed over 1000 randomly generated networks samples.

Conclusions

- We have defined two new approximation for the Shapley value of the Euclidean TSG
  - based on generalising a 1D idea to the multidimensional case
  - deeper insight into the process of distance sharing
  - no assumption about the dimension of the Euclidean space
  - possibly usable for non-Euclidean scenarios

- Extensively tested on 2D data
  - both approximations have polynomial time complexity
  - outperform existing methods in the $literature^*$ in accuracy
  - Appro-O(2) can handle network sizes of a few hundreds
  - Appro-O(1) can handle newtork sizes of a few thousands
  - no real competitor for $n > 50$