Data collection planning - TSP(N), PC-TSP(N), and OP(N))

Jan Faigl

Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague

Lecture 08

B4M36UIR – Artificial Intelligence in Robotics

Overview of the Lecture

- Part 1 – Data Collection Planning
  - Data Collection Planning – Motivational Problem
  - Traveling Salesman Problem (TSP)
  - Traveling Salesman Problem with Neighborhoods (TSPN)
  - Generalized Traveling Salesman Problem (GTSP)
  - Noon-Bean Transformation
  - Orienteering Problem (OP)
  - Orienteering Problem with Neighborhoods (OPN)

Part I

Part 1 – Data Collection Planning

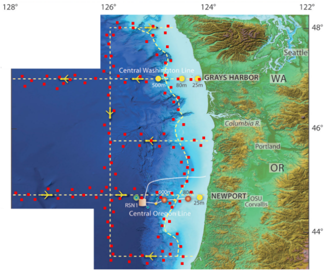
Outline

Autonomous Data Collection

- Having a set of sensors (sampling stations), we aim to determine a cost-efficient path to retrieve data by autonomous underwater vehicle (AUV) from the individual sensors

  E.g., Sampling stations on the ocean floor

- The planning problem is a variant of the **Traveling Salesman Problem**



Two practical aspects of the data collection can be identified

1. Data from particular sensors may be of different importance
2. Data from the sensor can be retrieved using wireless communication

These two aspects can be considered in Prize-Collecting Traveling Salesman Problem (PC-TSP) and Orienteering Problem (OP) and their extensions with neighborhoods.

Prize-Collecting Traveling Salesman Problem with Neighborhoods (PC-TSPN)

- Let n sensors be located in $\mathbb{R}^2$ at the locations $S = \{s_1, \ldots, s_n\}$
- Each sensor has associated penalty $\zeta(s_i) \geq 0$ characterizing additional cost if the data are not retrieved from s_i
- Let the data collecting vehicle operates in $\mathbb{R}^2$ with the motion cost $c(p_1, p_2)$ for all pairs of points $p_1, p_2 \in \mathbb{R}^2$
- The data from s_i can be retrieved within δ distance from s_i

PC-TSPN – Optimization Criterion

The **PC-TSPN** is a problem to

- **Determine a set of unique locations** $G = \{g_1, \ldots, g_k\}$, $k \leq n$, $g_i \in \mathbb{R}^2$, at which data readings are performed
- **Find a cost efficient tour** T visiting G such that the total cost $C(T)$ of $T$is minimal

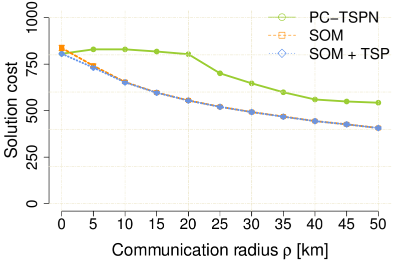
$$\mathcal{C}(T) = \sum_{(g_{l_i}, g_{l_{i+1}}) \in T} c(g_{l_i}, g_{l_{i+1}}) + \sum_{s \in S \setminus S_T} \zeta(s), \qquad (1)$$

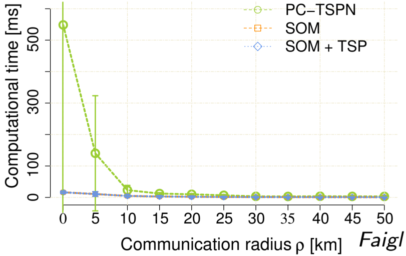
where $S_T \subseteq S$ are sensors such that for each $s_i \in S_T$ there is g_{l_j} on $T = (g_{l_1}, \ldots, g_{l_{k-1}}, g_{l_k})$ and $g_{l_j} \in G$ for which $|(s_i, g_{l_j})| \leq \delta$.

- PC-TSPN includes other variants of the TSP
  - for $\delta = 0$ it is the PC-TSP
  - for $\zeta(s_i) = 0$ and $\delta \geq 0$ it is the TSPN
  - for $\zeta(s_i) = 0$ and $\delta = 0$ it is the ordinary TSP

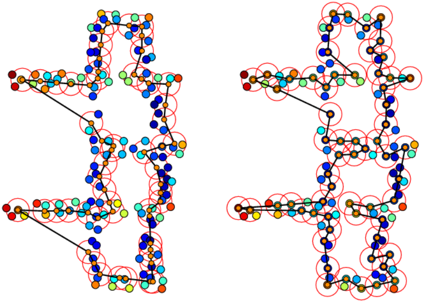
PC-TSPN – Example of Solution

Ocean Observatories Initiative (OOI) scenario





SOM PC-TSPN PC-TSPN



Faigl and Hollinger – (IROS 2014, TNNLS 2017)

Outline

- Data Collection Planning – Motivational Problem
- Traveling Salesman Problem (TSP)
- Traveling Salesman Problem with Neighborhoods (TSPN)
- Generalized Traveling Salesman Problem (GTSP)
- Noon-Bean Transformation
- Orienteering Problem (OP)
- Orienteering Problem with Neighborhoods (OPN)

Traveling Salesman Problem (TSP)

- Let S be a set of n sensor locations $S = \{s_1, \ldots, s_n\}$, $s_i \in \mathbb{R}^2$ and $c(s_i, s_j)$ is a cost of travel from s_i to s_j
- **Traveling Salesman Problem** (**TSP**) is a problem to determine a closed tour visiting each $s \in S$ such that the total tour length is minimal, i.e.,
  - determine **a sequence of visits** $\Sigma = (\sigma_1, \ldots, \sigma_n)$ such that

$$
\begin{array}{ll}
\textit{minimize}_{\ \Sigma} & L = \left(\sum_{i=1}^{n-1} c(s_{\sigma_i}, s_{\sigma_{i+1}}) \right) + c(s_{\sigma_n}, s_{\sigma_1}) \\
\textit{subject to} & \Sigma = (\sigma_1, \ldots, \sigma_n), 1 \leq \sigma_i \leq n, \sigma_i \neq \sigma_j \text{ for } i \neq j
\end{array}
\tag{2}
$$

- The TSP can be considered on a graph $G(V, E)$ where the set of vertices V represents sensor locations S and E are edges connecting the nodes with the cost $c(s_i, s_j)$
- For simplicity we can consider $c(s_i, s_j)$ to be Euclidean distance; otherwise, it is a solution of the path planning problem

Euclidean TSP

- If $c(s_i, s_j) \neq C(s_j, s_i)$ it is the **Asymmetric TSP**
- The TSP is known to be NP-hard unless P=NP

Existing solvers to the TSP

- Exact solutions
  - Branch and Bound, Integer Linear Programming (ILP)

E.g., Concorde solver – http://www.tsp.gatech.edu/concorde.html

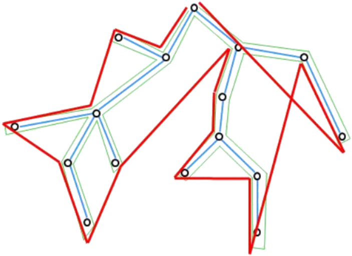
- Approximation algorithms
  - Minimum Spanning Tree (MST) heuristic – $L \leq 2L_{opt}$
  - Christofides's algorithm – $L \leq \frac{3/2}{L}_{opt}$
- Heuristic algorithms
  - Constructive heuristic – Nearest Neighborhood Algorithm
  - 2-Opt – local search algorithm proposed by Croes 1958
  - Lin-Kernighan (LK) heuristic

E.g., Helsgaun's implementation of the LK heuristic
http://www.akira.ruc.dk/~keld/research/LKH

- Soft-Computing techniques, e.g.,
  - Variable Neighborhood Search (VNS)
  - Evolutionary approaches
  - Unsupervised Learning

MST-based Approximation Algorithm to the TSP

- Minimum Spanning Tree Heuristic
  1. Compute the MST T of the input graph G
  2. Construct a graph H by doubling every edge of T
  3. Shortcut repeated occurrences of a vertex in the Tour



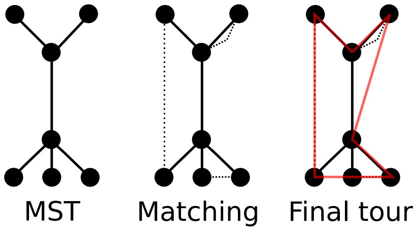
- For the triangle inequality, the length of such a tour L is

$$L \leq 2L_{optimal},$$

where $L_{optimal}$ is the cost of the optimal solution of the TSP

Christofides's Algorithm to the TSP

- Christofides's algorithm
  1. Compute the MST of the input graph G
  2. Compute minimal matching on the odd-degree vertices
  3. Shortcut a traversal of the resulting Eulerian graph



- For the triangle inequality, the length of such a tour L is

$$L \leq \frac{3}{2} L_{optimal},$$

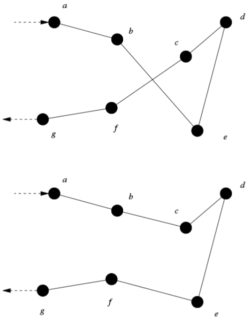
where $L_{optimal}$ is the cost of the optimal solution of the TSP

Length of MST is $\leq L_{optimal}$

Sum of lengths of the edges in the matching $\leq \frac{1}{2} L_{optimal}$

2-Opt Heuristic

1. Use a construction heuristic to create an initial route
   - NN algorithm, cheapest insertion, farther insertion
2. Repeat until no improvement is made
   2.1 Determine swapping that can shorten the tour (i, j) for $\leq 1i \leq n$ and $i+1 \leq j \leq n$
      - route[0] to route[i-1]
      - route[i] to route[j] in reverse order
      - route[j] to route[end]
      - Determine length of the route
      - Update the current route if length is shorter than the existing solution



Unsupervised Learning based Solution of the TSP

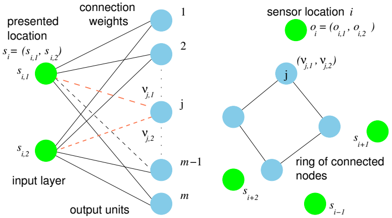
- Sensor locations $S = \{s_1, \ldots, s_n\}$, $s_1 \in \mathbb{R}^2$; Neurons $\mathcal{N} = (\nu_1, \ldots, \nu_m)$, $\nu_i \in \mathbb{R}^2$, $m = 2.5n$
- Learning gain G; epoch counter i; gain decreasing rate $\alpha = 0.1$; learning rate $\mu = 0.6$

1. $\mathcal{N} \leftarrow$ init ring of neurons as a small ring around some $s_i \in S$, e.g., a circle with radius 0.5
2. $i \leftarrow 0$; $\sigma \leftarrow 12.41n + 0.06$;
3. $I \leftarrow \emptyset$ *//clear inhibited neurons*
4. **foreach** $s \in \Pi(S)$ (a permutation of S)
   - 4.1 $\nu^* \leftarrow \mathrm{argmin}_{\nu \in \mathcal{N} \setminus I} ||(\nu, s)||$
   - 4.2 **foreach** ν in d neighborhood of ν^*

     $$\nu \leftarrow nu + \mu f(\sigma, d)(s - \nu)$$

     $$f(\sigma, d) = \begin{cases} e^{-\frac{d^2}{\sigma^2}} & \text{for } d < 0.2m, \\ 0 & \text{otherwise,} \end{cases}$$

   - 4.3 $I \leftarrow I \bigcup \{\nu^*\}$ // inhibit the winner
5. $\sigma \leftarrow (1 - \alpha)\sigma$; $i \leftarrow i + 1$;
6. `If` (**termination condition** is not satisfied) `Goto` Step 4; `Otherwise` retrieve solution

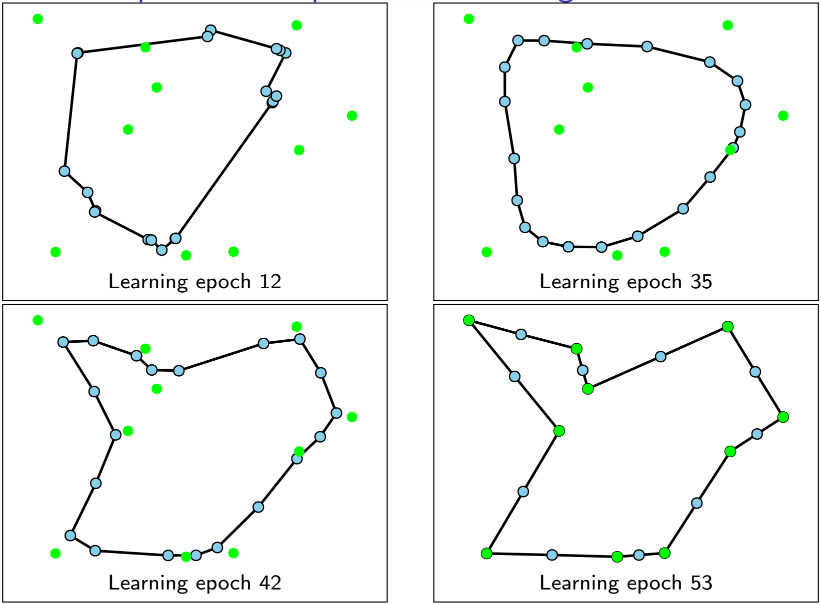


Termination condition can be

- Maximal number of learning epochs $i \le i_{max}$, e.g., $i_{max} = 120$
- Winner neurons are negligibly close to sensor locations, e.g., ≤ 0.001

Somhom, S., Modares, A., Enkawa, T. (1999): Competition-based neural network for the multiple travelling salesmen problem with minmax objective. Computers & Operations Research.
Faigl, J. et al. (2011): An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem. Neurocomputing.

Example of Unsupervised Learning for the TSP

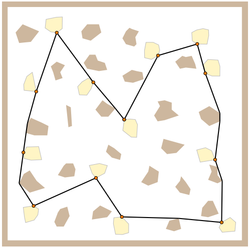


Outline

Traveling Salesman Problem with Neighborhoods (TSPN)

- Instead visiting a particular location $s \in \mathbb{R}^2$ we can request to visit, e.g., a region $r \subset \mathbb{R}^2$ to save travel cost, i.e., visit regions $R = \{r_1, \ldots, r_n\}$
- The TSP becomes the **TSP with Neighborhoods** (**TSPN**) where it is necessary, in addition to the determination of the order of visits Σ, determine suitable locations $P = \{p_1, \ldots, p_n\}$, $p_i \in r_i$, of visits to R
- The problem is a combination of combinatorial optimization to determine Σ with continuous optimization to determine P

$$\begin{array}{ll} \textit{minimize}_{\ \Sigma,P,R} & L = \left(\sum_{i=1}^{n-1} c(p_{\sigma_i}, p_{\sigma_{i+1}})\right) + c(p_{\sigma_n}, p_{\sigma_1}) \\ \textit{subject to} & R = \{r_1, \ldots, r_n\}, r_i \subset \mathbb{R}^2 \\ & P = \{p_1, \ldots, p_n\}, p_i \in r_i \\ & \Sigma = (\sigma_1, \ldots, \sigma_n), 1 \le \sigma_i \le n, \\ & \sigma_i \ne \sigma_j \text{ for } i \ne j \\ & \text{Foreach } r_i \in R \text{ there is } p_i \in r_i \end{array} \qquad (3)$$

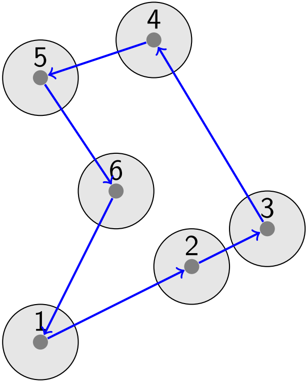


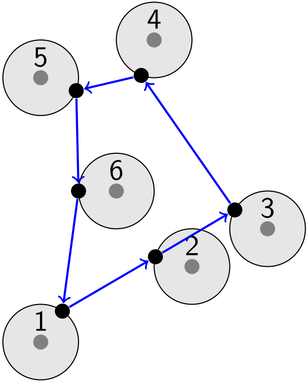
In general, TSPN is APX-hard, and cannot be approximated to within a factor $2 - \epsilon$, $\epsilon > 0$, unless P=NP.

Safra, S., Schwartz, O. (2006)

Traveling Salesman Problem with Neighborhoods (TSPN)

- Euclidean TSPN with disk shaped δ-neighborhoods
- Sequence of visits to the regions with particular locations of the visit





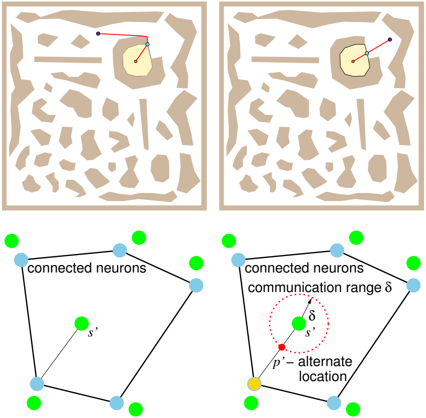
Approaches to the TSPN

- Direct solution of the TSPN – approximation algorithm and heuristics
  E.g., using evolutionary techniques or unsupervised learning
- **Decoupled approach**
  1. Determine sequence of visits Σ independently on the locations P
     E.g., as the TSP for centroids of the regions R
  2. For the sequence Σ determine the locations P to minimize the total tour length, e.g.,
     - Touring polygon problem (TPP)
     - Sampling possible locations and forward search for best locations
     - Continuous optimization such as hill-climbing
       E.g., Local Iterative Optimization (LIO), Váňa, Faigl (IROS 2015)
- Sampling-based approaches
  - For each region, sample possible locations of visits into a discrete set of locations for each region
  - The problem can be then formulated as the **Generalized Traveling Salesman Problem** (**GTSP**)
- Euclidean TSPN with, e.g., disk-shaped δ neighborhoods
  - Simplified variant with regions as disks with radius δ – remote sensing with the δ communication range

Unsupervised Learning for the TSPN

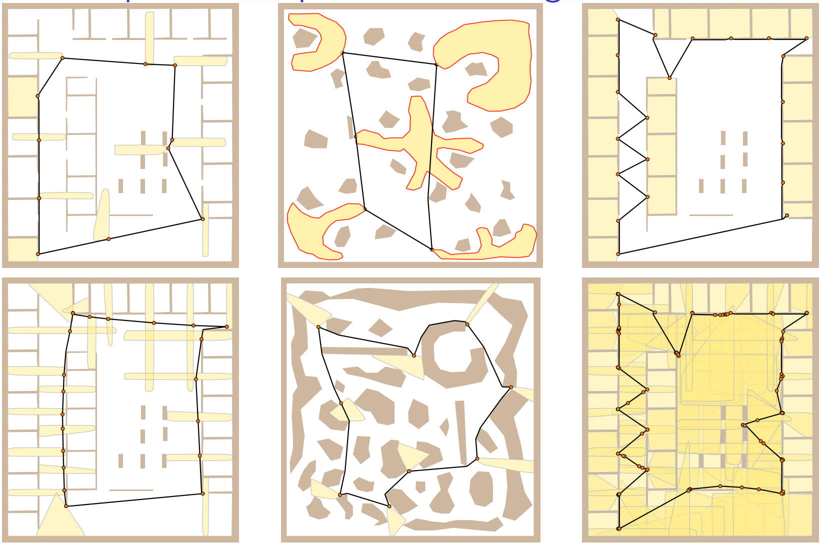
- In the unsupervised learning for the TSP, we can sample suitable sensing locations during winner selection

- We can use the centroid of the region for the shortest path computation from ν to the region r presented to the network
- Then, an intersection point of the path with the region can be used as an alternate location
- For the Euclidean TSPN with disk-shaped δ neighborhoods, we can compute the alternate location directly from the Euclidean distance



Faigl, J. et al. (2013): Visiting convex regions in a polygonal map. Robotics and Autonomous Systems.

Example of Unsupervised Learning for the TSPN



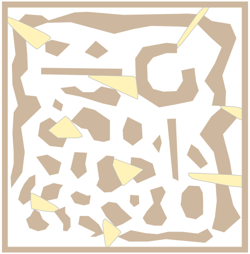
It also provides solutions for non-convex regions, overlapping regions, and coverage problems.

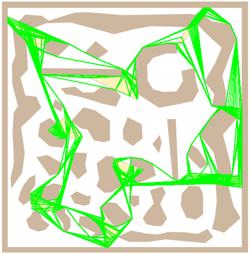
The TSPN as the TPP – Iterative Refinement

- Let the sequence of n polygon regions be $R = (r_1, \ldots, r_n)$

Li, F., Klette, R.: Approximate algorithms for touring a sequence of polygons. 2008

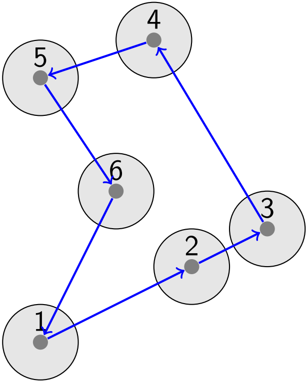
1. Sampling the polygons into a discrete set of points and determine all shortest paths between each sampled points in the sequence of the regions visits. *E.g., using visibility graph*
2. *Initialization:* Construct an initial touring polygons path using a sampled point of each region
   Let the path be defined by $P = (p_1, p_2, \ldots, p_n)$, where $p_i \in r_i$ and $L(P)$ be the length of the shortest path induced by P
3. *Refinement:* **For** $i = 1, 2, \ldots, n$
   - Find $p_i^* \in r_i$ minimizing the length of the path $d(p_{i-1}, p_i^*) + d(p_i^*, p_{i+1})$, where $d(p_k, p_l)$ is the length path from p_k to p_l, $p_0 = p_n$, and $p_{n+1} = p_1$
   - If the total length of the current path over point p_i^* is shorter than over p_i, replace the point p_i by p_i^*.
4. Compute path length L_{new} using the refined points
5. *Termination condition:* If $L_{new} - L < \epsilon$ Stop the refinement. Otherwise $L \leftarrow L_{new}$ and go to Step 3.
6. *Final path construction:* use the last points and construct the path using the shortest paths among obstacles between two consecutive points.

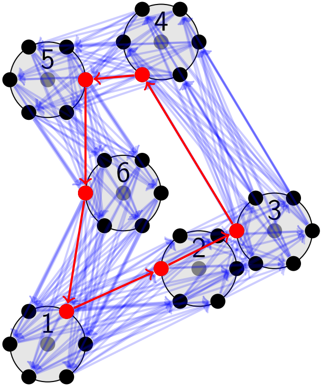




Sampling-based Decoupled Solution of the TSPN

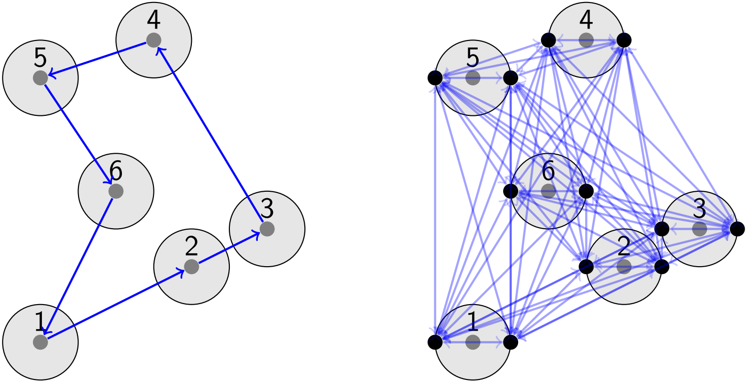
- Sample each neighborhood with, e.g., $k = 6$ samples
- Determine sequence of visits, e.g., by a solution of the ETSP for the centroids of the regions
- Finding the shortest tour takes in a forward search graph in $\mathcal{O}(nk^3)$ for nk^2 edges in the sequence





Sampling-based Solution of the TSPN

- For an unknown sequence of the visits to the regions, there are $\mathcal{O}(n^2k^2)$ possible edges
- Finding the shortest path is NP-hard, we need to determine the sequence of visits, which is the solution of the TSP



The descrite variant of the TSPN can be formulated as the GTSP

Outline

- Data Collection Planning – Motivational Problem
- Traveling Salesman Problem (TSP)
- Traveling Salesman Problem with Neighborhoods (TSPN)
- Generalized Traveling Salesman Problem (GTSP)
- Noon-Bean Transformation
- Orienteering Problem (OP)
- Orienteering Problem with Neighborhoods (OPN)

Generalized Traveling Salesman Problem (GTSP)

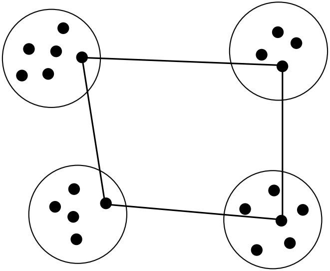
- For sampled neighborhoods into a discrete set of locations, we can formulate the problem as the **Generalized Traveling Salesman Problem** (GTSP)

Also known as the Set TSP or Covering Salesman Problem, etc.

- For a set of n sets $S = \{S_1, \ldots, S_n\}$, each with particular set of locations (nodes) $S_i = \{s^i_1, \ldots, s^i_{n_i}\}$
- The problem is to determine the shortest tour visiting each set S_i, i.e., determining the order Σ of visits to S and a particular locations $s^i \in S_i$ for each $S_i \in S$

$$\textit{minimize}_{\Sigma} \quad L = \left(\sum_{i=1}^{n-1} c(s^{\sigma_i}, s^{\sigma_{i+1}}) \right) + c(s^{\sigma_n}, s^{\sigma_1})$$

$$\textit{subject to} \quad \Sigma = (\sigma_1, \ldots, \sigma_n), 1 \le \sigma_i \le n, \sigma_i \neq \sigma_j \text{ for } i \neq j$$
$$s^{\sigma_i} \in S_{\sigma_i}, S_{\sigma_i} = \{s^{\sigma_i}_1, \ldots, s^{\sigma_i}_{n_{\sigma_i}}\}, S_{\sigma_i} \in S$$



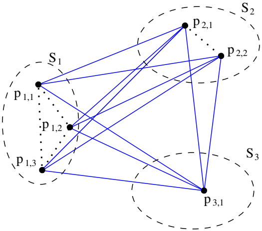
- In addition to exact, e.g., ILP-based, solution, a heuristic algorithm **GLNS** is available (besides other heuristics)

Smith, S. L., Imeson, F. (2017), GLNS: An effective large neighborhood search heuristic for the Generalized Traveling Salesman Problem. Computers and Operations Research.

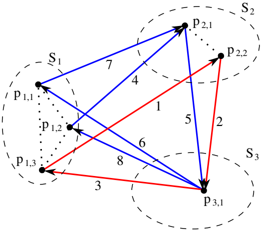
Implementation in Julia – https://ece.uwaterloo.ca/~sl2smith/GLNS

Transformation of the GTSP to the Asymmetric TSP

- The Generalized TSP can be transformed into Asymmetric TSP that can be then solved, e.g., by LKH or exactly by Concorde (by further transformation to the TSP)



GTSP



GATSP

- The transformation of the GTSP to ATSP has been proposed by Noon and Bean in 1993, and it is called as the **Noon-Bean Transformation**

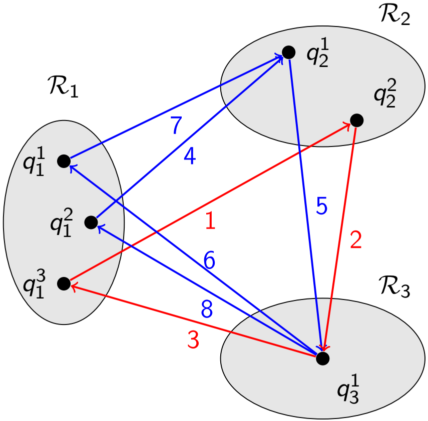
Noon, C.E., Bean, J.C. (1993), An efficient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research.
Ben-Arieg, et al. (2003), Transformations of generalized ATSP into ATSP. Operations Research Letters.

Outline

Noon-Bean Transformation

- Noon-Bean transformation to transfer GTSP to ATSP

- Modify weight of the edges (arcs) such that the optimal ATSP tour visits all vertices of the same cluster before moving to the next cluster
- Adding large constant M to the weights, e.g., a sum of the n heaviest edges
- Ensure visiting all vertices of the cluster in a prescribed order, i.e., creating zero-length cycles within each cluster
- The transformed ATSP can be further transformed to the TSP
- For each vertex of the ATSP created 3 vertices in the TSP, i.e., it increases the size of the problem three times

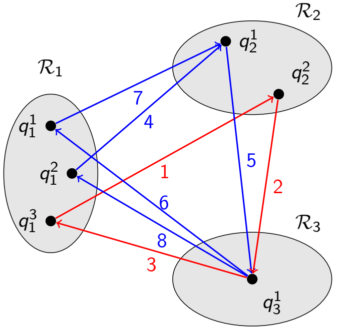


Noon, C.E., Bean, J.C. (1993), An efficient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research.

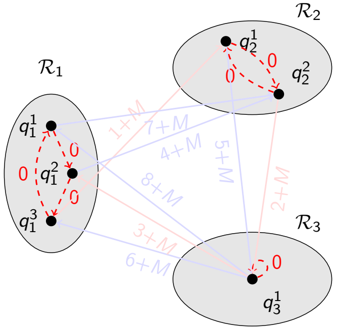
Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M

   To ensure visit of all vertices in a cluster before the next cluster

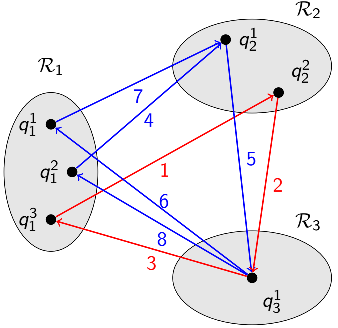


$\Rightarrow$

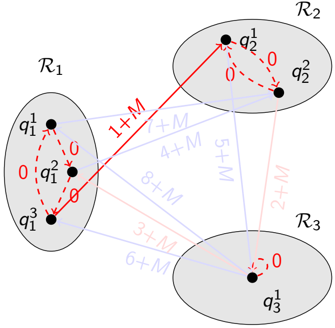


Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M
   To ensure visit of all vertices in a cluster before the next cluster

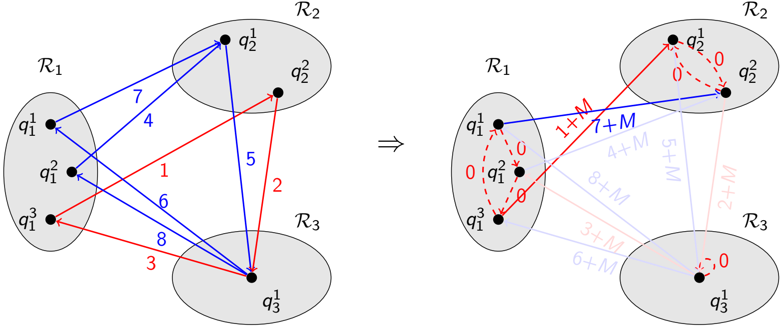


$\Rightarrow$



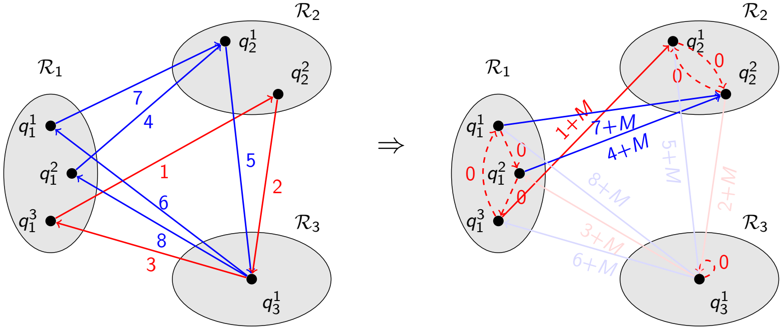
Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M)
   To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M
   To ensure visit of all vertices in a cluster before the next cluster



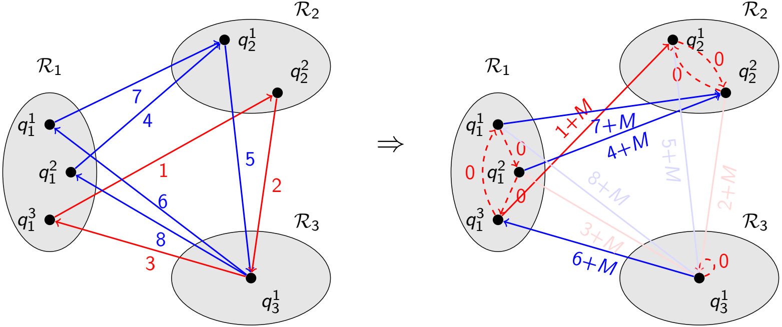
Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M
To ensure visit of all vertices in a cluster before the next cluster



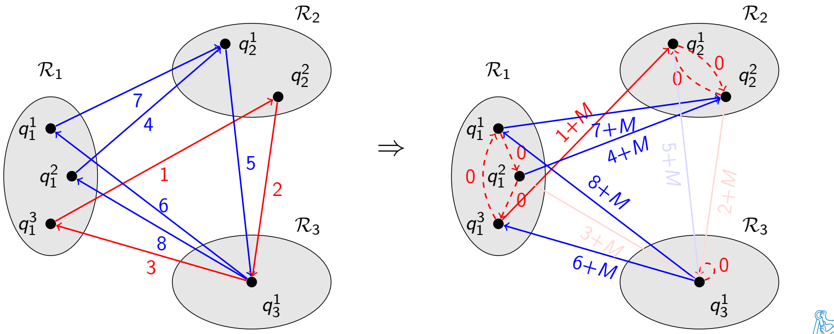
Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M
   To ensure visit of all vertices in a cluster before the next cluster



Example – Noon-Bean transformation (GATSP to ATSP)

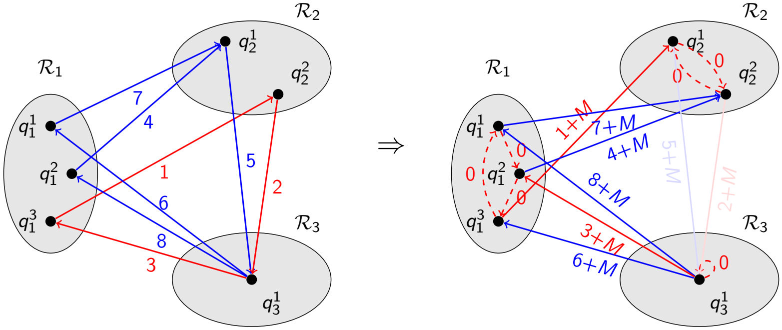
1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M
   To ensure visit of all vertices in a cluster before the next cluster



Example – Noon-Bean transformation (GATSP to ATSP)

1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M

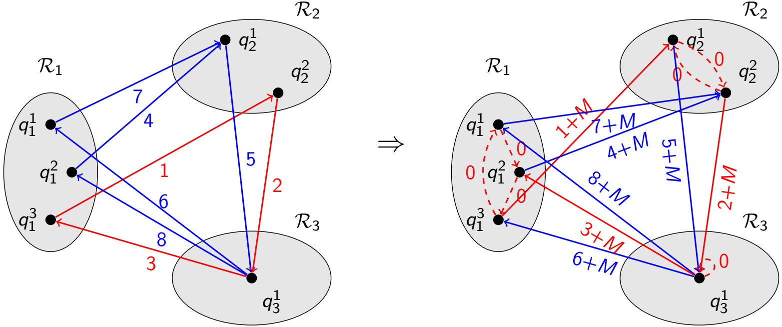
   To ensure visit of all vertices in a cluster before the next cluster



Example – Noon-Bean transformation (GATSP to ATSP)

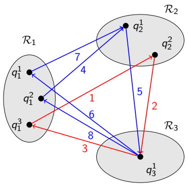
1. Create a zero-length cycle in each set and set all other inter-cluster arc to ∞ (or 2M) To ensure all vertices of the cluster are visited before leaving the cluster
2. For each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M

To ensure visit of all vertices in a cluster before the next cluster



Noon-Bean transformation – Matrix Notation

- 1. Create a zero-length cycle in each set; and 2. for each edge (q_i^m, q_j^n) create an edge (q_i^m, q_j^{n-1}) with a value increased by sufficiently large M



$\Rightarrow$

	q_1^1	q_1^2	q_1^3	q_2^1	q_2^2	q_3^1
q_1^1	∞	∞	∞	7	–	–
q_1^2	∞	∞	∞	4	–	–
q_1^3	∞	∞	∞	–	1	–
q_2^1	–	–	–	∞	∞	5
q_2^2	–	–	–	∞	∞	2
q_3^1	6	8	3	–	–	∞

∞ represents there are not edges inside the same set; and – denotes unused edge

Original GATSP

	q_1^1	q_1^2	q_1^3	q_2^1	q_2^2	q_3^1
q_1^1	∞	∞	∞	7	–	–
q_1^2	∞	∞	∞	4	–	–
q_1^3	∞	∞	∞	–	1	–
q_2^1	–	–	–	∞	∞	5
q_2^2	–	–	–	∞	∞	2
q_3^1	6	8	3	–	–	∞

Transformed ATSP

	q_1^1	q_1^2	q_1^3	q_2^1	q_2^2	q_3^1
q_1^1	∞	0	∞	–	$7{+}M$	–
q_1^2	∞	∞	0	–	$4{+}M$	–
q_1^3	0	∞	∞	$1{+}M$	–	–
q_2^1	–	–	–	∞	0	$5{+}M$
q_2^2	–	–	–	0	∞	$2{+}M$
q_3^1	$8{+}M$	$3{+}M$	$6{+}M$	–	–	0

Noon-Bean Transformation – Summary

- It transforms the GATSP into the ATSP which can be further
  - Solved by existing solvers, e.g., the Lin-Kernighan heuristic algorithm (LKH) http://www.akira.ruc.dk/~keld/research/LKH
  - the ATSP can be further transformed into the TSP and solved optimality by, e.g., Concorde solver http://www.tsp.gatech.edu/concorde.html
- It runs in $\mathcal{O}(k^2 n^2)$ time and uses $\mathcal{O}(k^2 n^2)$ memory, where n is the number of sets (regions) each with up to k samples
- The transformed ATSP problem contains kn vertices

- The main issue of the transformation is related to the suitable selection of the constant M that is need to forbid the repetitive visitation of the same set
  - I.e., the problem is how to set sufficiently large M but do not cause numeric troubles

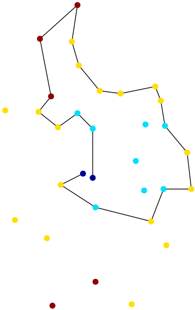
Noon, C.E., Bean, J.C. (1993), An efficient transformation of the generalized traveling salesman problem. INFOR: Information Systems and Operational Research.

Outline

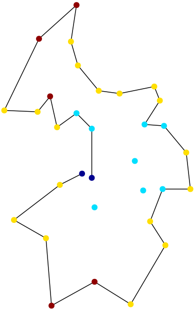
The Orienteering Problem (OP)

- The problem is to collect as many rewards as possible within the given **travel budget** (T_{max}), which is especially suitable for robotic vehicles such as multi-rotor Unmanned Aerial Vehicles (UAVs)
- The starting and termination locations are prescribed and can be different

The solution may not be a closed tour as in the TSP



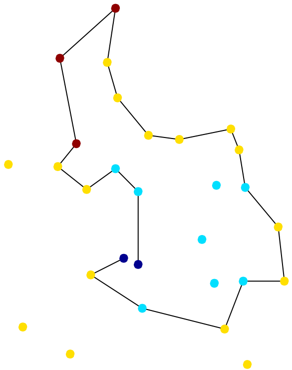
Travel budget $T_{max} = 50$,
Collected rewards $R = 190$



Travel budget $T_{max} = 75$,
Collected rewards $R = 270$

Orienteering Problem – Specification

- Let the given set of n sensors be located in $\mathbb{R}^2$ with the locations $S = \{s_1, \ldots, s_n\}$, $s_i \in \mathbb{R}^2$
- Each sensor s_i has an associated score ς_i characterizing the reward if data from s_i are collected
- The vehicle is operating in $\mathbb{R}^2$, and the travel cost is the Euclidean distance
- Starting and final locations are prescribed
- We aim to determine a subset of k locations $S_k \subseteq S$ that maximizes the sum of the collected rewards while the travel cost to visit them is below T_{max}



The Orienteering Problem (**OP**) combines two NP-hard problems:

- Knapsack problem in determining the most valuable locations $S_k \subseteq S$
- Travel Salesman Problem (**TSP**) in determining the shortest tour

Orienteering Problem – Optimization Criterion

- Let $\Sigma = (\sigma_1, \ldots, \sigma_k)$ be a permutation of k sensor labels, $1 \leq \sigma_i \leq n$ and $\sigma_i \neq \sigma_j$ for $i \neq j$
- Σ defines a tour $T = (s_{\sigma_1}, \ldots, s_{\sigma_k})$ visiting the selected sensors S_k
- Let the start and end points of the tour be $\sigma_1 = 1$ and $\sigma_k = n$
- The **Orienteering problem** (**OP**) is to determine the number of sensors k, the subset of sensors S_k, and their sequence Σ such that

$$\begin{aligned}
&\mathit{maximize}_{k,S_k,\Sigma} && R = \sum_{i=1}^{k} \varsigma_{\sigma_i} \\
&\mathit{subject\ to} && \sum_{i=2}^{k} |(s_{\sigma_{i-1}}, s_{\sigma_i})| \leq T_{max} \ \mathit{and} \\
& && s_{\sigma_1} = s_1, s_{\sigma_k} = s_n.
\end{aligned} \tag{4}$$

The OP combines the problem of determining the most valuable locations S_k with finding the shortest tour T visiting the locations S_k. It is NP-hard, since for $s_1 = s_n$ and particular S_k it becomes the TSP.

Existing Heuristic Approaches for the OP

- The Orienteering Problem has been addressed by several approaches, e.g.,

RB	4-phase heuristic algorithm proposed in [3]
PL	Results for the method proposed by Pillai in [2]
CGW	Heuristic algorithm proposed in [1]
GLS	Guided local search algorithm proposed in [4]

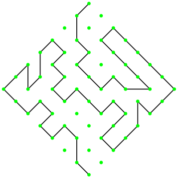
[1] I.-M. Chao, B. L. Golden, and E. A. Wasil.
A fast and effective heuristic for the orienteering problem.
European Journal of Operational Research, 88(3):475–489, 1996.

[2] R. S. Pillai.
The traveling salesman subset-tour problem with one additional constraint (TSSP+ 1).
Ph.D. thesis, The University of Tennessee, Knoxville, TN, 1992.

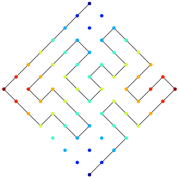
[3] R. Ramesh and K. M. Brown.
An efficient four-phase heuristic for the generalized orienteering problem.
Computers & Operations Research, 18(2):151–165, 1991.

[4] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden.
A guided local search metaheuristic for the team orienteering problem.
European Journal of Operational Research, 196(1):118–127, 2009.

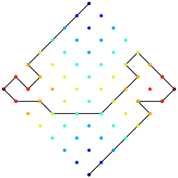
OP Benchmarks – Example of Solutions



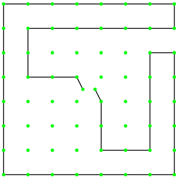
T_{max}=80, R=1248



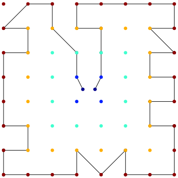
T_{max}=80, R =1278



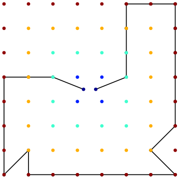
T_{max}=45, R=756



T_{max}=95, R=1395



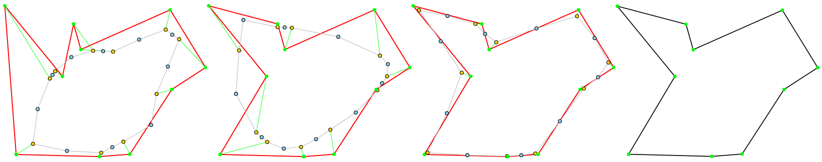
T_{max}=95, R=1335



T_{max}=60, R=845

Unsupervised Learning for the OP 1/2

- A solution of the OP is similar to the solution of the PC-TSP and TSP
- We need to satisfy the limited travel budget T_{max}, which needs the final tour over the sensing locations
- During the unsupervised learning, the winners are associated with the particular sensing locations, which can be utilized to determine the tour as a solution of the OP represented by the network:



Learning epoch 7 Learning epoch 55 Learning epoch 87 Final solution

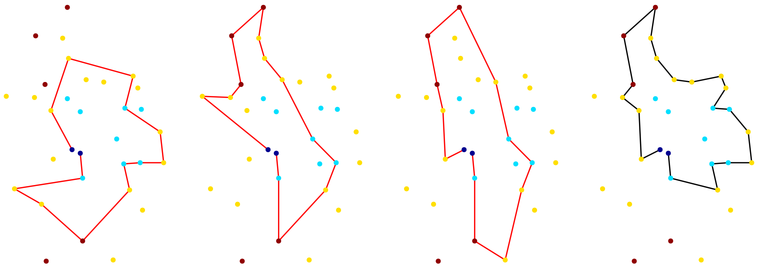
- This is utilized in the **conditional adaptation** of the network towards the sensing location only if the tour represented by the network after the adaptation would satisfy T_{max}

Unsupervised Learning for the OP 2/2

- The winner selection for $s' \in S$ is conditioned according to T_{max}
  - The network is adapted only if the tour T_{win} represented by the current winners would shorter or equal than T_{max}

$$\mathcal{L}(T_{win}) - |(s_{\nu_p}, s_{\nu_n})| + |(s_{\nu_p}, s')| + |(s', s_{\nu_n})| \leq T_{max}$$

- The unsupervised learning performs a *stochastic search* steered by the rewards and the length of the tour to be below T_{max}



Epoch 155, R=150 Epoch 201, R=135 Epoch 273, R=125 Final solution, R=190

Comparison with Existing Algorithms for the OP

- Standard benchmark problems for the Orienteering Problem various scenarios with several values of T_{max}
- The results are presented as the average ratios (and standard deviations) to the best-known solution

Instances of the Tsiligirides problems

Problem Set	**RB**	**PL**	**CGW**	**Unsupervised Learning**
Set 1, $5 \leq T_{max} \leq 85$	0.99/0.01	1.00/0.01	1.00/0.01	1.00/0.01
Set 2, $15 \leq T_{max} \leq 45$	1.00/0.02	0.99/0.02	0.99/0.02	0.99/0.02
Set 3, $15 \leq T_{max} \leq 110$	1.00/0.00	1.00/0.00	1.00/0.00	1.00/0.00

Diamond-shaped (Set 64) and Square-shaped (Set 66) test problems

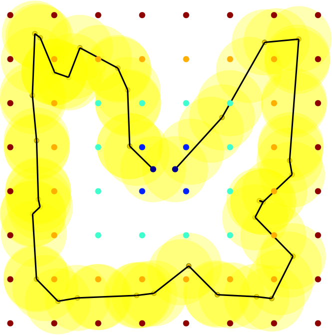
Problem Set	**RB†**	**PL**	**CGW**	**Unsupervised Learning**
Set 64, $5 \leq T_{max} \leq 80$	0.97/0.02	1.00/0.01	0.99/0.01	0.97/0.03
Set 66, $15 \leq T_{max} \leq 130$	0.97/0.02	1.00/0.01	0.99/0.04	0.97/0.02

Required computational time is up to units of seconds, but for small problems tens or hundreds of milliseconds.

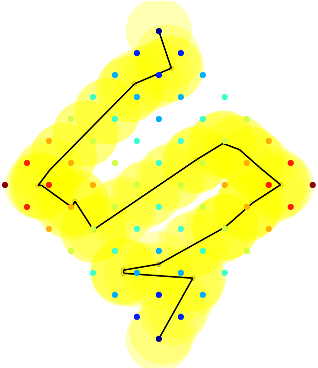
Outline

Orienteering Problem with Neighborhoods

- Similarly to the TSP with Neighborhoods and PC-TSPN we can formulate the **Orienteering Problem with Neighborhoods**.



T_{max}=60, δ=1.5, R=1600



T_{max}=45, δ=1.5, R=1344

Orienteering Problem with Neighborhoods

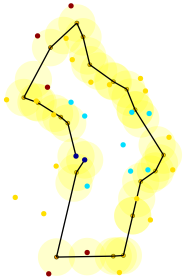
- Data collection using wireless data transfer allows to reliably retrieve data within some communication radius δ
  - Disk-shaped δ-neighborhood
- We need to determine the most suitable locations P_k such that

$$\text{maximize}_{k,P_k,\Sigma} \quad R = \sum_{i=1}^{k} \varsigma_{\sigma_i}$$

$$\text{subject to} \quad \sum_{i=2}^{k} |(p_{\sigma_{i-1}}, p_{\sigma_i})| \le T_{max},$$

$$|(p_{\sigma_i}, s_{\sigma_i})| \le \delta, \quad p_{\sigma_i} \in \mathbb{R}^2,$$

$$p_{\sigma_1} = s_1, p_{\sigma_k} = s_n.$$



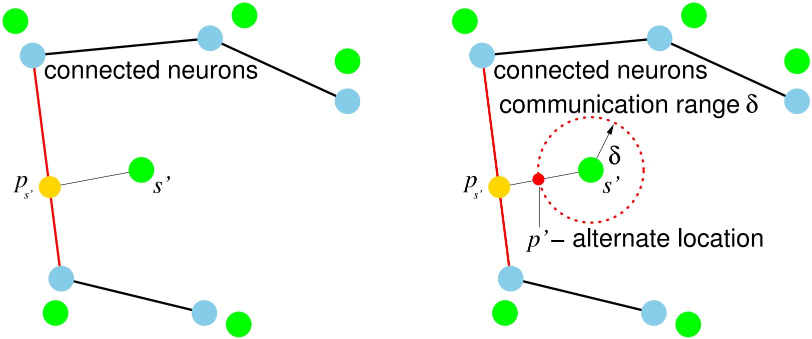
$T_{max} = 50, \; R = 270$

***Introduced by** Best, Faigl, Fitch (IROS 2016, SMC 2016, IJCNN 2017)*

- More rewards can be collected than for the OP formulation with the same travel budget T_{max}

Generalization of the Unsupervised Learning to the Orienteering Problem with Neighborhoods

- The same idea of the alternate location as in TSPN



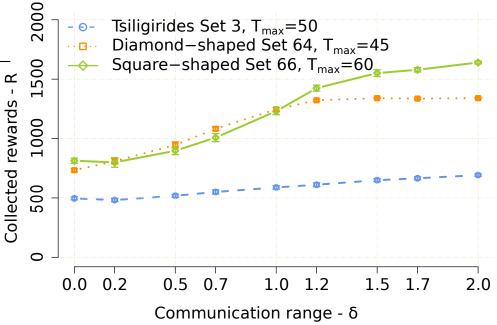
- The location p' for retrieving data from s' is determined as the alternate goal location during the conditioned winner selection

Influence of the δ-Sensing Distance

- Influence of increasing communication range to collected rewards

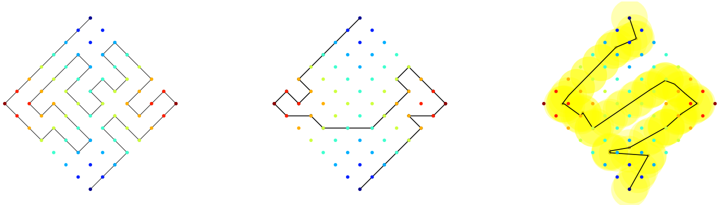
Problem	Solution of the OP R_{best}	R_{SOM}
Set 3, T_{max}=50	520	510
Set 64, T_{max}=45	860	750
Set 66, T_{max}=60	915	845

- *Allowing to data reading within the communication range δ may significantly increases the collected rewards, while keeping the budget under T_{max}.*



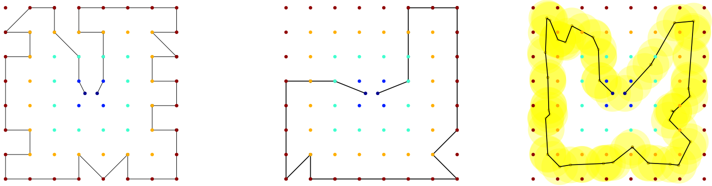
OP with Neighborhoods (OPN) – Example of Solutions

- Diamond-shaped problem Set 64 – SOM solutions for T_{max} and δ



T_{max}=80, δ=0.0, R=1278 T_{max}=45, δ=0.0, R=756 T_{max}=45, δ=1.5, R=1344

- Square-shaped problem Set 66 – SOM solutions for T_{max} and δ



T_{max}=95, δ=0.0, R=1335 T_{max}=60, δ=0.0, R=845 T_{max}=60, δ=1.5, R=1600

In addition to unsupervised learning, **Variable Neighborhood Search** (**VNS**) for the OP has been generalized to the OPN.

Summary of the Lecture

Topics Discussed

- Data Collection Planning – motivational problem and solution
  - Prize-Collecting Traveling Salesman Problem with Neighborhoods (PC-TSPN)
- Traveling Salesman Problem (TSP)
  - Approximation and heuristic approaches
- Traveling Salesman Problem with Neighborhoods (TSPN)
  - Sampling-based and decoupled approaches
  - Unsupervised learning
- Generalized Traveling Salesman Problem (GTSP)
  - Heuristic and transformation (GTSP$\rightarrow$ATSP) approaches
- Orienteering problem (OP)
  - Heuristic and unsupervised learning based approaches
- Orienteering problem with Neighborhoods (OPN)
  - Unsupervised learning based approach

- Next: Data-collection planning with curvature-constrained vehicles