1. Introduction

The set-partitioning problem is a well known NP-hard combinatorial problem. A real-world problem of this class for the vehicle routing problem in Florida area was formulated by Juricek [5]. He requires partitioning of customers to clusters. The clusters should be constructed as “optimal” areas for 1-vehicle routing. An intuitive criterion for the well-formed area of the cluster for one vehicle is the smallest convex polygon that encloses all points of cluster – convex hull.

We present two problems:

- **Convex set partitioning problem:** A set of points in a plane have to be partitioned to a given number of clusters so that every two convex hulls of cluster members have empty intersection and the total area of cluster’s convex hulls is minimal.

- **Convex location problem:** A set of q seeds of clusters are searched with a bounded capacity of cluster and convex constraint for a pair of assigned points.

2. Convex hull problem

Finding the convex hull of a set of points in plane is one of most interesting elementary problems in computational geometry [1].

Let \( P = \{p_1, p_2, ..., p_n\}, n \geq 3 \) be a set of distinct points in the Euclidean plane. The **convex hull** of \( P \) is the minimal convex area (bordered by points in \( P \)) that contains every point of \( P \). We will denote this area by \( \text{convh}(P) \). For convenience we say that a point \( p \in P \) is a boundary point of the convex hull of \( P \) if \( p \) is a boundary vertex of the convex hull of \( P \). The convex hull problem is the problem of computing the convex hull of \( P \) and reporting the boundary points of the convex hull in the order in which they appear on the hull – we will note this set \( \text{convh}(P) \).

As early as 1972, Graham gave a convex hull algorithm with \( O(n \log(n)) \) worst-case running time. In our application we used the Python program [3] implementing Graham’s scan algorithm.

3. Convex set partitioning problem

The problem of partitioning customers (represented by the points on the plane) to the convex hulls of vehicle areas can be formulated as the following optimization problem (CSP):

Let us have the set of \( n \) points \( P = \{p_1, p_2, ..., p_n\} \) in the plane and the integer \( q \) where \( n \geq 6 \) and \( 2 \leq q \leq \lfloor n/3 \rfloor \). The goal is to
find a set partitioning of the set $P$ to $q$ partitions – clusters of points – $P(q) = (P_1, P_2, \ldots, P_q)$ such that every two clusters have empty intersection and the total area of the convex hulls of partitioning members is minimal.

Fig. 2 shows an example of the convex set partitioning of set $P$ to three clusters $P_1, P_2, P_3$ of points. Every two convex hulls of partitioning have empty intersection i.e. $\text{convh}(P_i) \cap \text{convh}(P_j) = \emptyset$ for $1 < i < j < 3$ and the total area of the convex hulls $|\text{convh}(P_1)| + |\text{convh}(P_2)| + |\text{convh}(P_3)|$ is minimal, where $|p|$ denotes the area of the polygon $p$.

Let $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$ be the set of all subsets of the boundary points from $P$ which lie on the convex hulls. We will use the abbreviate notations $M = \{1, 2, \ldots, m\}$ and $N = \{1, 2, \ldots, n\}$. We can now define the binary variables

$$x_i = \begin{cases} 1 & \text{if solution has clusters with convex hull of points } H_i \in \mathcal{H} \\ 0 & \text{otherwise} \end{cases}.$$

We define the point-to-cluster incidence matrix $A = (a_{ij})$ where

$$a_{ij} = \begin{cases} 1 & \text{if } p_i \in \text{convh}(H_j). \\ 0 & \text{otherwise} \end{cases}$$

and the cluster-to-cluster incidence matrix $B = (b_{ij})$ where

$$b_{ij} = \begin{cases} 1 & \text{if } \text{convh}(H_i) \cap \text{convh}(H_j) \neq \emptyset. \\ 0 & \text{otherwise} \end{cases}$$

Let us remind that $|A|$ denotes the area of object $A$. Now we can formulate and solve the CSP problem as the following binary programming problem (BCSP):

$$\sum_{j=1}^{m} \text{convh}(H_j) x_j \rightarrow \min,$$  \hspace{1cm} (1)

$$\sum_{j=1}^{m} x_j = q$$  \hspace{1cm} (2)

The objective function (1) minimizes the total area of the convex hulls in the solution. The cardinality constraint (2) states that the partitioning contains exactly $q$ clusters. The set partitioning constraints (3) ensure that every point of sets $P$ lies in exactly one cluster. Constraint (4) tells us that every two convex hulls are disjoint. The logical constrain (5) chooses a set of clusters in the solution $x_{n+k} = 1, k = 1, 2, \ldots, q$. The optimal solution of the CSP problem has the following form

$$P(q) = (P_1, P_2, \ldots, P_q),$$

$$P_i = \{ p \in P : p \in \text{convh}(H_k), x_k = 1 \}.$$

For the exact solution of the BCSP we have the known set $H$ which has an exponential growth. Its size is too large for the solving of practical problems via accessible bivalent programming solvers.

One approach how to try to find an approximate solution is the reduction of the set $H$. The natural way how to do it is based on the capacities $c$ of the points $p$. Let $C$ be the capacity of vehicles. We can define the capacity of a cluster as the sum of capacities of its points. The cluster of the points $P$ is then a feasible cluster if

$$\sum_{j=1}^{m} c_j \leq C.$$

(6)

Now we can formulate the problem which helps to reduce the set $H$.

4. Convex location problem

When we approximate $q$ convex hulls of the CSP problem as $q$-median of seeds of clusters we have the following optimization problem (CL):

Let us have the given set of $n$ points $P = \{p_1, p_2, \ldots, p_n\}$ in the plane, integer $q$ where $n \geq 6$ and $2 \leq q \leq \lfloor n/2 \rfloor$ and positive integer numbers $c_1, c_2, \ldots, c_q, C$. The goal is to find a $q$ seeds of feasible clusters of points – $S(q) = \{s_1, s_2, \ldots, s_q\}$ such every two convex hulls of cluster have empty intersection and the sum of Euclidean distances of points to seeds of clusters is minimal.

This model measures the cluster by a sum of distances from the seed of cluster to the points of cluster as we can see in Fig. 3. The convexity constraint assigns the point $p$ to the cluster $P_i$ because the distance from the point $s_1$ to point $p$ is shorter than the distance from the point $s_2$ to point $p$.

Let $D = (d_{ij})$ be the matrix of Euclidean distances between the points $P$ and let us denote the maximum distance by $d_{\text{max}}$. We can define the bivalent variables $y_i = 1$ if the point $p_i$ is a seed of
the same cluster and \(y_i = 0\) otherwise. We also define the bivalent variables \(x_{ij} = 1\) if the point \(p_j\) is assigned to the seed \(p_i\). It is easy to construct the set \(S = \{p_i \in P : y_i = 1\}\).

Now we can formulate and solve the CL problem as the bivalent-programming problem (BCL):

\[
\begin{align*}
\sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} & \rightarrow \min, \\
\sum_{i \in N} y_i & = q, \\
\sum_{i \in N} x_{ij} & = 1, \quad j \in N, \\
\sum_{j \in N} c_{ij} x_{ij} & \leq C_k, \quad i \in N, \\
d_{ij} x_{ij} & = (d_{ij} - d_{min}) y_i \leq d_{min}, \quad i, j \in N, i \neq j \neq k, \\
x_{ij}, y_i & \in \{0, 1\}. 
\end{align*}
\]

The objective function (7) minimizes the total distance of the points to the nearest seed. The cardinality constraint (8) states that the solution has \(q\) clusters. The assignment constraint (9) ensures that every point of the set \(P\) is assigned exactly to one computed seed. The capacity constraint (10) is valid for feasible clusters only. The convexity constraint (11) ensures that the constructed convex hulls are really of shape of a convex polygon. Finding such polygon is similar to the same phase in Voronoi algorithm. The constraint describes a situation when we need to decide to which cluster a point belongs. When there is more possibilities it is always assigned to a cluster with nearest seed. The last logical constraint (12) chooses seeds and defines clusters with its seed.

It can be proved that the BCL problem is NP-hard. Janacek and Gabrisova studied the possible approach to an approximate solution of the capacitated location problem based on lagrangean relaxation in [4]. Computational experiments with program glpsol for solution of mixed linear programming problems show that using Open Source library GLPK (GNU Linear Programming Kit) [7] we can find a good approximate solution of a real-world problem.

5. Instances of Florida area

We verified the models on the two instances of Florida area with \(n = 234\) points and the number of clusters \(q = 10\) and \(q = 13\). The capacities of points are daily weights of customer’s demands. For every \(q\) we solved 7 instances of BCSP problems with \(C \in \{C_k : k \in \{0, \ldots, 6\}, C_0 > C_1 > \ldots > C_6\}\)

\[
\text{where } C_i = \frac{\sum_{k=1}^i c_k}{q + k}
\]

and \(C_i + 1\) is the maximum of cluster capacities in the solution of instance with the vehicle capacity \(C_{i-1}\) for \(k = 1, \ldots, 6\). The points on convex hulls define the reduced set \(H_q = \{H_1, H_2, \ldots, H_q\}\). By solving the BCSP problem, we got the clusters for ten vehicles as we can see in Fig. 4. The convex hulls were constructed by Graham’s scan algorithm [3] implemented in Python [2], [6], [8]. An interesting property of the obtained solution is that the clusters have a relatively uniform distribution of the cluster capacity defined by the left side of inequality (6). Note that the total time of computing via [7] program glpsol was between 15 – 16 hours for one \(q\).

6. Open questions

The experiments with Florida instances show that real world BCSP problems with several hundreds of points can require long computational times. Reduction of the problem by solving some associated BCL problems is possible. Open questions are

- the approximate ratio for presented heuristic,
- the applicability of estimation of vehicle service with the aid of the area and weight of cluster,
- the possibility of weakening the formulation of the convexity constraint (11) in the BCL problem.
the development of special methods for solving the BCSP and BCL problems.
possibility of formulation and solution of BCSP via Voronoi diagrams

Acknowledgement: The research of the author is supported by the Slovak Scientific Grant Agency under grant VEGA 1/0135/08.

References