Clustered Spanning Tree-Conditions for Feasibility

Nili Guttmann-Beck, Zeev Sorek, Michal Stern

Clustered Spanning Tree - Conditions for Feasibility

Nili Guttmann-Beck [1] Zeev Sorek[1] Michal Stern[1,2]

[1] *Academic College of Tel-Aviv Yaffo, Yaffo, Israel*
[2] *Caesarea Rothchild Institute, university of Haifa, Haifa, Israel*

received October 2018, *revised March 2019*, *accepted* .

Let $H =< V, \mathcal{S} >$ be a hypergraph, V is a set of vertices and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of not necessarily disjoint clusters $S_i \subseteq V$ such that $\cup_{i=1}^{m} S_i = V$. The Clustered Spanning Tree problem is to find a tree spanning all the vertices in V which satisfies that each cluster induces a subtree, when it exists. We provide an efficient and unique algorithm which finds a feasible solution tree for H when it exists, or states that no feasible solution exists. The paper also uses special structures of the intersection graph of H to construct a feasible solution more efficiently. For cases when the hypergraph does not have a feasible solution tree, we consider adding vertices to exactly one cluster in order to gain feasibility. We characterize when such addition can gain feasibility, find the appropriate cluster and a possible set of vertices to be added.

Keywords: Clustered spanning tree, Feasibility

1 Introduction

Let $H =< V, \mathcal{S} >$ be a hypergraph, where V is a set of vertices and $\mathcal{S}$ is a set of clusters $S_1, \ldots, S_m$, $S_i \subseteq V$ for $i \in \{1, \ldots, m\}$, $\cup_{i=1}^{m} S_i = V$, and the clusters are not necessarily disjoint. The Clustered Spanning Tree problem, denoted by *CST*, is to find a tree spanning all the vertices of V, such that each cluster induces a subtree, if one exists.

One of the main results of this paper (Algorithm ES, Figure 1) is a novel algorithm for the essential question of whether a feasible solution tree exists for a given instance of the *CST* problem. This algorithm requires $O(|V|^2 m)$ time complexity and can handle every instance hypergraph. In the first stage of the algorithm a weighted graph is constructed, V is the set of vertices of this graph, an edge (v, u) exists in the graph if there is a cluster which contains both vertices, the weight of each edge in the graph is equal

to the number of clusters containing both endpoints of this edge. Next, we find a maximum spanning tree for this graph. A feasible solution for the *CST* problem exists if and only if the weight of this tree is $\sum_{i=1}^{m} |S_i| - m$. Furthermore, the maximum spanning tree offers a feasible solution, when it exists.

This paper also introduces how a solution for the *CST* problem can be derived using information from the hypergraph and the corresponding intersection graph. First, we prove that when H has a feasible solution tree T^H, then subtrees of T^H are feasible solution trees for the corresponding induced subproblems. This also proves that when an induced hypergraph does not have a feasible solution tree, neither does H. When the intersection graph contains a cut-edge, we prove that deciding whether a feasible solution exists can be based on the decision made independently for each component of the intersection graph. The feasible solution tree for the given hypergraph is constructed using the feasible solution subtrees created from the corresponding subproblems and thus may significantly reduce the required complexity. For the special case where every vertex in V is contained in at most 2 clusters from $\mathcal{S}$, the CST problem has a feasible solution if and only if the corresponding intersection graph is a tree. In all cases, a feasible solution tree is offered when it exists.

Another main result of the paper (Theorems 4.12 and 4.15) considers hypergraphs which do not have a feasible solution tree. In these cases we want to characterize when adding vertices to exactly one cluster can gain feasibility. Assuming the clusters satisfy the Helly Property, we prove that when all the chordless cycles of the intersection graph share a joint node, adding an appropriate set of vertices to the corresponding cluster creates a hypergraph with a feasible solution tree. We classify all the sets of vertices whose addition creates feasibility.

Throughout this paper, we assume that the intersection graph of H is connected. Otherwise, a feasible solution tree for H can be constructed by properly adding edges between the feasible solutions of each connected component, if they exist.

The following theorem, summarized in McKee and McMorris (1999), which will be used throughout the paper, gives sufficient and necessary conditions for the feasibility decision problem for a given instance of the CST problem.

Theorem 1.1 *(Duchet (1976), Flament (1978), Slater (1978)) A hypergraph $H =< V, \mathcal{S} >$ has a feasible solution tree if and only if it satisfies the Helly property and its intersection graph is chordal.*

Verifying whether a hypergraph satisfies the Helly Property requires $O(|V|^4 m)$ time complexity, according to Dourado et al. (2009). This time complexity dominates the time required to verify that the intersection graph is chordal. Thus using Theorem 1.1 to check whether a hypergraph has a feasible solution tree requires $O(|V|^4 m)$ time complexity.

Related problems consider different structures of the solution tree and the clusters' induced subtrees.

In Swaminathan and Wagner (1994) a polynomial algorithm is presented, which constructs a tree where each cluster spans a path. The most restricted problem where both the tree and subtrees are required to be paths, is in fact the Consecutive Ones Problem, which Booth and Lueker (1976) solve in linear time using PQ-trees.

Considering the optimization *CST* problem, the edges of E have weights and the objective is to find a feasible solution tree with minimum weight. This problem was solved by Korach and Stern in Korach and Stern (2003) where an optimum solution is found in $O(|V|^4 m^2)$ time complexity, when a feasible solution exists. In addition, an abstraction of the problem using matroids is presented. For the restricted case where each cluster contains at most three vertices, there is a linear time algorithm and a polyhedral description of all feasible solutions. A special case of the optimization *CST* problem, where the optimum spanning tree solution is required to span a complete star on each cluster, is presented in Korach and Stern (2008). A structure theorem which describes all feasible solutions and a polynomial algorithm for finding an optimum solution is presented, assuming the intersection graph is connected.

Another related optimization problem, called the clustering-*TSP*-path, arises when the optimum solution tree is required to be a *TSP*-path. The *TSP*-path is proven to be NP-hard in Christofides (1976). In Guttmann-Beck et al. (2018) several algorithms for the not necessarily disjoint clustered *TSP*-path are presented. For a restricted case of the problem an exact polynomial algorithm is presented. For other cases approximation algorithms are presented, whose efficiently depends on the structure of the hypergraph. A lot of research has also been investigated on the clustered *TSP*-path where the clusters are disjoint. A heuristic for this problem is presented in Chisman (1975), a branch and bound algorithm for solving this problem is presented in Lokin (1979) and bounded-approximation algorithms are presented in Arkin et al. (1997) and Guttmann-Beck et al. (2000). In Anily et al. (1999) the ordered disjoint clustered *TSP* is considered and an approximation algorithm is offered. In Potvin and Guertin (1998) a genetic algorithm for solving this problem is presented.

This paper is organized as follows. Section 2 introduces Algorithm Existence of Solution which decides whether a feasible solution tree exists for any given instance of the CST problem and finds a feasible solution when one exists. Section 3 contains theorems which state how the feasibility of subproblems of a given instance indicates whether a feasible solution tree exists. This section also uses information derived from the intersection graph of a given instance to determine its feasibility. When applicable, the methods introduced in this section have significantly better complexity than the algorithm introduced in Section 2. Section 4 considers hypergraphs with no feasible solution tree. For those cases, we characterize when adding vertices to exactly one cluster of the given hypergraph gains feasibility. The section contains proofs which characterize the hypergraphs that become feasible when such an addition is preformed, identify which clusters are appropriate for this change and what are the relevant sets of vertices whose addition creates feasibility.

2 Existence of Solution

This section introduces algorithm Existence of Solution (ES) which is a major result of the paper. The algorithm, presented in Figure 1, either finds a feasible solution tree for a hypergraph $H =< V, \mathcal{S} >$ or declares that there is no feasible solution. The algorithm creates a weighted graph denoted by $G_{ES} = (V_{ES}, E_{ES})$, where $V_{ES} = V$ and E_{ES} contains an edge (v, u) if there exists a cluster S_i such that $\{v, u\} \subseteq S_i$. The weight of each edge is equal to the number of clusters containing both endpoints of the edge. In this graph a maximum spanning tree T_{ES} is found. We prove that a hypergraph has a feasible solution tree if and only if the weight of T_{ES}, denoted by $w(T_{ES})$, is equal to $\sum_{i=1}^{m} |S_i| - m$. When a solution exists, T_{ES} is a feasible solution tree.

ES (Existence of Solution)
input
A hypergraph $H =< V, \mathcal{S} >$, where $\mathcal{S} = \{S_1, \ldots, S_m\}$.
returns
A feasible solution tree or a statement "No feasible solution".
begin
Construct the following weighted graph $G_{ES} = (V_{ES}, E_{ES})$:
$V_{ES} \equiv V$.
E_{ES} contains the edge (v, u) (for $v \neq u$) if there exists a cluster S_i such that $\{v, u\} \subseteq S_i$.
The weight of an edge (v, u) is set: $w(v, u) = |\{S_i : i \in \{1, \ldots, m\}, \{v, u\} \subseteq S_i\}|$.
Find T_{ES} - a maximum spanning tree of G_{ES}.
if *$w(T_{ES}) = \sum_{i=1}^{m} |S_i| - m$*
  then *Return T_{ES}*
  else *Return "No feasible solution".*
end if
end *ES*

Fig. 1: Algorithm ES

First, we define induced subtrees and the weighted graph G_{ES} which is used by Algorithms ES.

Definition 2.1 *Given a hypergraph $H =< V, \mathcal{S} >$, T a tree spanning V and $V' \subseteq V$, the subgraph of T induced by V', denoted by $\boldsymbol{T[V']}$, is defined to contain all the vertices of V' and all the edges of T whose both end-points are in V'.*

Definition 2.2 *Given a hypergraph $H =< V, \mathcal{S} >$:*

- *G_{ES} is the weighted graph with vertex set V_{ES} and edge set E_{ES}, where: $V_{ES} \equiv V$ and E_{ES} contains the edge (v, u) (for $v \neq u$) if there exists a cluster S_i such that $\{v, u\} \subseteq S_i$.*
- *For every edge $(v, u) \in E_{ES}$ and every cluster $S_i \in \mathcal{S}$:*

$$w_i(v, u) = \begin{cases} 1 & \text{if } v, u \in S_i \\ 0 & \text{otherwise} \end{cases}$$

- *For every edge* $(v,u) \in E_{ES}$ *:* $w(v,u) = \sum_{i=1}^{m} w_i(v,u) = |\{S_i : i \in \{1,\ldots,m\}, \{v,u\} \subseteq S_i\}|$. *This is the weight used in Algorithm* ES *(Figure 1).*
- *For every tree* T *of* G_{ES} *and every cluster* $S_i \in \mathcal{S}$*:* $w_i(T) = \sum_{(v,u)\in E(T)} w_i(v,u)$*, where* $E(T)$ *is the set of edges in* T.
- *For every tree* T *of* G_{ES}*:* $w(T) = \sum_{i=1}^{m} w_i(T)$.

Lemma 2.3 *Given a hypergraph* $H =< V, \mathcal{S} >$*, the inequality* $w_i(T) \leq |S_i| - 1$ *holds for every spanning tree* T *of* G_{ES} *and every cluster* $S_i \in \mathcal{S}$*. An equality holds if and only if* $T[S_i]$ *is a subtree spanning all the vertices in* S_i*.*

Proof: According to Definition 2.2, the weight $w_i(T)$ is equal to the number of edges in $T[S_i]$. Since T is cycle-less, the number of edges in $T[S_i]$ is at most $|S_i| - 1$ which proves that $w_i(T) \leq |S_i| - 1$. $T[S_i]$ is a tree if and only if it contains exactly $|S_i| - 1$ edges, each edge e with weight $w_i(e) = 1$. Hence, $T[S_i]$ is a spanning tree of S_i if and only if $w_i(T) = w_i(T[S_i]) = |S_i| - 1$. □

Corollary 2.4 *Given a hypergraph* $H =< V, \mathcal{S} >$*, for every* T *a spanning tree of* G_{ES} *the weight* $w(T) = \sum_{i=1}^{m} w_i(T) \leq \sum_{i=1}^{m} |S_i| - m$*.*

Theorem 2.5 *Given a hypergraph* $H =< V, \mathcal{S} >$ *and* T_{ES} *a maximum spanning tree of* G_{ES}*,* $w(T_{ES}) = \sum_{i=1}^{m} |S_i| - m$ *if and only if* T_{ES} *is a feasible solution tree.*

Proof: By Corollary 2.4, $w(T) = \sum_{i=1}^{m} w_i(T) \leq \sum_{i=1}^{m} |S_i| - m$, for every T a spanning tree of G_{ES}. Using Lemma 2.3, the weight of the maximum spanning tree is $w(T_{ES}) = \sum_{i=1}^{m} |S_i| - m$ if and only if $w_i(T_{ES}) = |S_i| - 1$ for every $i \in \{1,\ldots,m\}$. In this case, $T[S_i]$ is a spanning tree of S_i for every cluster $S_i \in \mathcal{S}$. Therefore, T_{ES} is a solution tree for H. □

In Figure 2 we give two examples of hypergraphs. The left most item in this figure is the hypergraph suited for $\mathcal{S} = \{\{1,2\},\{2,4\},\{1,2,3\},\{2,3,4\}\}$. The second item on the left is the appropriate weighted graph created for this hypergraph by Algorithm ES. The solid lines describe the maximum spanning tree of this graph. The weight of this tree is $6 = \sum_{i=1}^{m} |S_i| - m$ which indicates, according to Theorem 2.5, that the hypergraph has a feasible solution tree with the maximum spanning tree as its solution. The two right items of this figure describe the hypergraph and the weighted graph suited for $\mathcal{S} = \{\{1,2\},\{2,4\},\{1,2,3\},\{1,3,4\}\}$. The weight of a maximum spanning in this case is $5 \neq \sum_{i=1}^{m} |S_i| - m$ and indeed the hypergraph has no feasible solution tree.

Theorem 2.6 *The complexity of Algorithm* ES *(Figure 1) is* $O(|V|^2 m)$.

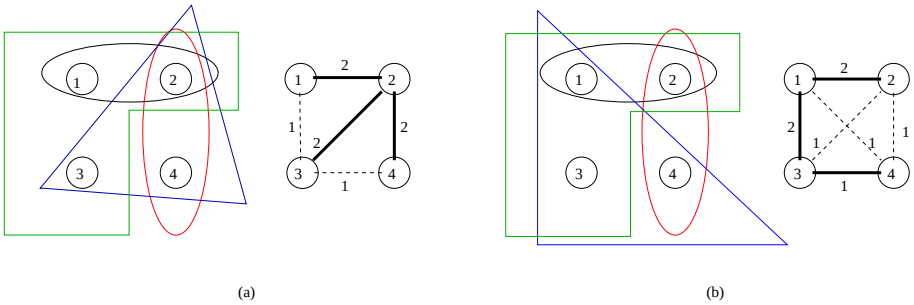

Fig. 2: Example of a hypergraph (a) with a feasible solution tree and (b) without a feasible solution tree

Proof: The first step of the algorithm is to construct G_{ES}. For every two nodes v, u the algorithm decides whether the edge (v, u) is included in E_{ES} and when it does, computes the weight of this edge. Thus constructing G_{ES} takes $O(|V|^2 m)$ steps. The second step is to find the maximum spanning tree of this graph. The dominating step with respect to complexity in finding a maximum spanning tree requires sorting the edges by their weights. Since the edge weights are integers and bounded by m, this can be done in $O(m + |V|^2)$ time. Hence the complexity of finding the required maximum tree is dominated by the complexity of the first step. Therefore, the complexity of Algorithm ES is $O(|V|^2 m)$. □

3 Induced Hypergraphs

Consider a hypergraph $H =< V, \mathcal{S} >$ which is an instance for the *CST* problem. New instances of the problem are created by considering induced hypergraphs defined by subsets of $\mathcal{S}$. In this section we prove that when H has a feasible solution tree T^H, then induced subtrees of T^H are feasible solution trees for the corresponding induced subproblems. Furthermore, when an induced hypergraph does not have a feasible solution tree, neither does H. Section 3.1 considers the case where the intersection graph of H has a cut-edge. In this case, we prove that the hypergraph has a feasible solution tree if and only if the induced hypergraphs, corresponding to the connected components created by removing the cut-edge, have feasible solution trees. Furthermore, a feasible solution tree for H can be constructed from the solution trees of the induced hypergraphs. Section 3.2 considers the special case where each vertex of V is contained in at most two clusters. In this case, we prove that H has a feasible solution tree if and only if the corresponding intersection graph is a tree.

The following definitions will be used throughout the paper.

Definition 3.1 *Given a hypergraph $H =< V, \mathcal{S} >$ where $\mathcal{S}$ is a set of not necessarily disjoint clusters $\{S_{i_1}, \ldots, S_{i_p}\}$ of V. The* ***intersection graph of*** $\boldsymbol{\{S_{i_1}, \ldots, S_{i_p}\}}$*, denoted by* $\boldsymbol{G_{int}(\{S_{i_1}, \ldots, S_{i_p}\})}$*, is defined to be a graph whose set of nodes is $\{s_{i_1}, \ldots, s_{i_p}\}$, where s_{i_j} corresponds to S_{i_j}, and an edge (s_{i_j}, s_{i_k}) exists if $S_{i_j} \cap S_{i_k} \neq \phi$.*

Definition 3.2 *Let $H =< V, \mathcal{S} >$ be a hypergraph and let $\mathcal{S}' \subset \mathcal{S}$ be a set of clusters. We define* $\boldsymbol{H[\mathcal{S}']}$

to be the hypergraph whose vertex set is $\boldsymbol{V(\mathcal{S}')} = \bigcup_{S_i \in \mathcal{S}'} S_i$ *and its clusters set is* $\mathcal{S}'$.

Property 3.3 *Let* $H =< V, \mathcal{S} >$ *be a hypergraph, if* $G_{int}(\mathcal{S})$ *is the intersection graph of* H*, then the induced graph* $G_{int}(\mathcal{S})[(\bigcup_{S_i \in \mathcal{S}'} S_i)]$ *is the intersection graph of* $H[\mathcal{S}']$ *and therefore can be denoted as* $G_{int}(\mathcal{S}')$.

Theorem 3.4 *Let* $H =< V, \mathcal{S} >$ *be a hypergraph with a connected intersection graph* $G_{int}(\mathcal{S})$ *and a feasible solution tree* T^H*. If* $G_{int}(\mathcal{S}')$ *is connected for* $\mathcal{S}' \subset \mathcal{S}$*, then* $T^H[V(\mathcal{S}')]$ *is a feasible solution tree for* $H[\mathcal{S}']$.

Proof: First we prove that $T^H[V(\mathcal{S}')]$ is a tree. Since $T^H[V(\mathcal{S}')]$ is a subgraph of T^H clearly it has no cycles.

To prove that this subgraph is connected, consider $v_1, v_2 \in V(\mathcal{S}')$. There are two clusters $S_{i_1}, S_{i_2} \in \mathcal{S}'$, not necessarily distinct, such that $v_1 \in S_{i_1}$ and $v_2 \in S_{i_2}$.

If $S_{i_1} \neq S_{i_2}$ then since $G_{int}(\mathcal{S}')$ is connected it contains a simple path $(s_{i_1}, w_1), (w_1, w_2), \ldots, (w_k, s_{i_2})$ with s_{i_1} representing S_{i_1}, s_{i_2} representing S_{i_2} and w_i representing a cluster $W_i \in \mathcal{S}'$, $\forall i \in \{1, \ldots, k\}$. Hence, there are vertices $u_1, \ldots, u_{k+1} \in V(\mathcal{S}')$ such that:

1. $u_1 \in S_{i_1} \cap W_1$,
2. $u_i \in W_{i-1} \cap W_i, \forall i \in \{2, \ldots, k\}$,
3. $u_{k+1} \in W_k \cap S_{i_2}$.

Since T^H is a feasible solution tree it contains the following simple paths:

1. a path $v_1 - u_1$ whose vertices are all contained in S_{i_1}.
2. $\forall i \in \{2, \ldots, k\}$ a path $u_{i-1} - u_i$ whose vertices are all contained in W_i,
3. a path $u_{k+1} - v_2$ whose vertices are all contained in S_{i_2}.

$T^H[V(\mathcal{S}')]$ contains all these paths and their union gives a path connecting v_1 and v_2 in $T^H[V(\mathcal{S}')]$.

Similarly, if $S_{i_1} = S_{i_2}$ then T^H contains a path connecting v_1 and v_2 whose vertices are all contained in S_{i_1} which is also in $T^H[V(\mathcal{S}')]$.

Combining the above claims we get that $T^H[V(\mathcal{S}')]$ is cycle-less and connected and therefore a tree.

Since T^H is a feasible solution tree, the induced tree $T^H[V(\mathcal{S}')]$ on any $S_i \in \mathcal{S}'$ is equal to the induced tree of T^H on S_i and hence is a subtree spanning S_i. Thus, $T^H[V(\mathcal{S}')]$ is a feasible solution tree for $H[\mathcal{S}']$. □

Corollary 3.5 *Let $H =< V, \mathcal{S} >$ be a hypergraph with a connected intersection graph $G_{int}(\mathcal{S})$. If $G_{int}(\mathcal{S}')$ is connected for $\mathcal{S}' \subset \mathcal{S}$ and $H[\mathcal{S}']$ has no feasible solution tree, then H has no feasible solution tree.*

3.1 Breaking the Intersection Graph

This section introduces a strategy to decide whether it is possible to divide the problem into smaller subproblems by a cut-edge of the intersection graph. We prove that a feasible solution tree of the given problem exists if and only if each subproblem has a feasible solution tree. Since the instances for the subproblems may be significantly smaller, the complexity of deciding whether a subproblem has a feasible solution decreases according to the size of the subproblem.

Theorem 3.6 *Let $H =< V, \mathcal{S} >$ be a hypergraph with a connected intersection graph $G_{int}(\mathcal{S})$. If the intersection graph contains a cut-edge (bridge) which divides the intersection graph into two connected components $G_{int}(\mathcal{S}')$ and $G_{int}(\mathcal{S}\backslash\mathcal{S}')$, then H has a feasible solution tree if and only if each one of $H[\mathcal{S}']$ and $H[(\mathcal{S}\backslash\mathcal{S}')]$ has a feasible solution tree.*

Proof: Without loss of generality, denote the cut-edge as (s_1, s_2) with s_1 representing cluster $S_1 \in \mathcal{S}'$ and s_2 representing cluster $S_2 \in \mathcal{S}\backslash\mathcal{S}'$. According to theorem's assumption $G_{int}(\mathcal{S}) = G_{int}(\mathcal{S}') \cup G_{int}(\mathcal{S}\backslash\mathcal{S}') + (s_1, s_2)$.

If H has a feasible solution tree then since both $G_{int}(\mathcal{S}')$ and $G_{int}(\mathcal{S}\backslash\mathcal{S}')$ are connected, according to Theorem 3.4, $H[\mathcal{S}']$ and $H[\mathcal{S}\backslash\mathcal{S}']$ both have feasible solution trees.

Suppose, on the other end, that $H[\mathcal{S}']$ and $H[\mathcal{S}\backslash\mathcal{S}']$ have feasible solution trees T_1 and T_2 respectively. Since T_1 is a feasible solution tree for $H[\mathcal{S}']$, $T_1[S_1]$ is a connected subtree. According to the theorem's assumption, the only edge in G_{int} between $\mathcal{S}'$ and $\mathcal{S}\backslash\mathcal{S}'$ is (s_1, s_2). Therefore, $(S_1 \cap S_2) \cap V(\mathcal{S}\backslash\{S_1, S_2\}) = \phi$. Hence, the vertices contained in $T_1[S_1]$ can be reordered to obtain a connected subtree on $T_1[S_1 \cap S_2]$. T_2 can be reordered in a similar way to satisfy that $T_2[S_1 \cap S_2]$ is a connected subtree of $T_2[S_2]$. This can be performed also to satisfy $T_2[S_1 \cap S_2] = T_1[S_1 \cap S_2]$. After the described adjustments, the union of T_1 and T_2 is a feasible solution tree for H. □

3.2 Bounded Containment of Vertices

In this section a special type of hypergraphs is considered, when every vertex in V is contained in at most 2 clusters from $\mathcal{S}$. We prove that in this type of hypergraphs, the CST problem has a feasible solution if and only if the intersection graph is a tree.

First, some definitions and properties are presented.

Definition 3.7 *For every $v \in V$, we define $\boldsymbol{nc(v)} = |\{S_i : i \in \{1,\ldots,m\}, v \in S_i\}|$ (the number of clusters which contain vertex v).*

Observation 3.8 *In an intersection graph, a vertex $v \in V$ which belongs to k clusters ($nc(v) = k$) creates a k-size clique. Hence, if the intersection graph of a hypergraph is a tree or a chordless cycle with at least 4 nodes, then $nc(v) \leq 2\ \forall v \in V$.*

Definition 3.9 *We denote the following subclusters:*

- *For every $i \in \{1,\ldots,m\}$, define the subcluster $\boldsymbol{K_{\{i\}}} = \{v : v \in S_i, nc(v) = 1\}$. $K_{\{i\}}$ contains all the vertices which belong only to S_i.*
- *For every $i \neq j \in \{1,\ldots,m\}$, define the subcluster $\boldsymbol{K_{\{i,j\}}} = \{v : v \in S_i \cap S_j\}$. $K_{\{i,j\}}$ contains all the vertices which belong both to S_i and to S_j.*

The following Property follows from Definition 3.9.

Property 3.10 *All clusters and subclusters satisfy:*

- $K_{\{i\}} \cap K_{\{i,j\}} = \phi$, $\forall i \neq j \in \{1,\ldots,m\}$.
- $S_i = K_{\{i\}} \cup (\bigcup_{j \neq i} K_{\{i,j\}})$, $\forall i \in \{1,\ldots,m\}$.
- *When $nc(v) \leq 2$ for every $v \in V$, then $K_{\{i,j_1\}} \cap K_{\{i,j_2\}} = \phi$ for $j_1 \neq j_2$ and $\forall i \in \{1,\ldots,m\}$.*

Theorem 3.11 *Let $H =< V, \mathcal{S} >$ be a hypergraph with $nc(v) \leq 2\ \forall v \in V$ and a connected intersection graph. H has a feasible solution tree if and only if its intersection graph is a tree.*

Proof: Consider the weighted graph created for H in Algorithm ES (Figure 1). Using Observation 3.8 and Property 3.10, the graph contains edges (v,u) with $w(v,u) \in \{1,2\}$, and $w(v,u) = 2$ when there are $i,j \in \{1,\ldots,m\}$ such that $v,u \in K_{\{i,j\}}$.

A maximum spanning tree T_{ES} contains a maximum number of edges of weight 2. Inside each subcluster $K_{\{i,j\}}$ a subtree is chosen, using $|K_{\{i,j\}}| - 1$ edges. By property 3.10, these subclusters are disjoint, giving that altogether T_{ES} contains $\sum_{i=1}^{m} \sum_{\substack{j>i \\ K_{\{i,j\}} \neq \phi}}^{m} (|K_{\{i,j\}}| - 1)$ edges of weight 2. The rest of the edges are of weight 1. Since any spanning tree on n vertices contains $n-1$ edges, there are $(n-1) - (\sum_{i=1}^{m} \sum_{\substack{j>i \\ K_{\{i,j\}} \neq \phi}}^{m} (|K_{\{i,j\}}| - 1))$ edges of weight 1.

Hence, the weight of a maximum spanning tree returned by Algorithm ES:

$$\begin{aligned} w(T_{ES}) &= 2*(\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}(|K_{\{i,j\}}|-1))+1*((n-1)-(\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}(|K_{\{i,j\}}|-1))) \\ &= (n-1)+\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}(|K_{\{i,j\}}|)-\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}1 \end{aligned} \quad (1)$$

In an intersection graph, the degree of s_i (the vertex representing S_i) is $d_i = |\{S_j|S_i \cap S_j \neq \phi, j \neq i\}| = |\{K_{\{i,j\}}|K_{\{i,j\}} \neq \phi, j \neq i\}|$. Therefore:

$$w(T_{ES}) = (n-1)+\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}(|K_{\{i,j\}}|)-0.5\sum_{i=1}^{m}d_i \quad (2)$$

By Property 3.10 each vertex of the graph belongs to exactly one subcluster, hence:

$$n = \sum_{i=1}^{m}|K_{\{i\}}|+\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}(|K_{\{i,j\}}|) \quad (3)$$

Using Equations (2) and (3) and Property 3.10:

$$\begin{aligned} w(T_{ES}) &= \sum_{i=1}^{m}|K_{\{i\}}|+2\sum_{i=1}^{m}\sum_{\substack{j>i \\ K_{\{i,j\}}\neq\phi}}^{m}|K_{\{i,j\}}|-1-0.5\sum_{i=1}^{m}d_i \\ &= \sum_{i=1}^{m}|S_i|-1-0.5\sum_{i=1}^{m}d_i \end{aligned} \quad (4)$$

According to Theorem 2.5, hypergraph H has a feasible solution tree if and only if $w(T_{ES}) = \sum_{i=1}^{m}|S_i|-m$. Using Equation (4), hypergraph H has a feasible solution tree if and only if

$$0.5\sum_{i=1}^{m}d_i = m-1$$

Since d_i is also the degree of s_i in the intersection graph, $0.5\sum_{i=1}^{m}d_i$ is equal to the number of edges of this intersection graph. The number of edges in a connected intersection graph with m nodes is equal to $m-1$ if and only if the intersection graph is a tree, thus proving the correctness of the theorem. □

Theorem 3.11 proves the following remark, which can also be deduced from Theorem 1.1:

Remark 3.12 *Let $H =< V, \mathcal{S} >$ be a hypergraph with a connected intersection graph $G_{int}(\mathcal{S})$. If $G_{int}(\mathcal{S})$ contains a chordless cycle with at least 4 nodes, then H has no feasible solution tree.*

Proof: Denote by $s_{i_1}, \ldots, s_{i_p}, p \geq 4$, the nodes of a chordless cycle in $G_{int}(\mathcal{S})$ and let $\mathcal{S}_C = \{S_{i_1}, \ldots, S_{i_p}\}$. Consider the hypergraph $H[\mathcal{S}_C]$, its intersection graph $G_{int}(\mathcal{S}_C)$ is a chordless cycle and therefore it is

connected. According to Observation 3.8, $\forall v \in V(\mathcal{S}_C)$ it holds that $nc(v) \leq 2$. By Theorem 3.11, since $G_{int}(\mathcal{S}_C)$ is a cycle, $H[\mathcal{S}_C]$ has no feasible solution tree. Therefore, by Corollary 3.5, H has no feasible solution tree. □

4 Inserting Vertices to Gain Feasibility

In this section we discuss hypergraphs with no feasible solution tree. We consider adding vertices to only one cluster of the given hypergraph. We characterize the hypergraphs that become feasible when such an addition is preformed. We find an appropriate cluster and all the set of vertices which could be added to it.

Definition 4.1 *Let $H =< V, \mathcal{S} >$ with $\mathcal{S} = \{S_1, \ldots, S_m\}$, be a hypergraph with no feasible solution tree. An **insertion cluster** is a cluster $S_i \in \mathcal{S}$, such that there exists a set of vertices $U \subset V$, for which the hypergraph $H' =< V, \{S_1, \ldots, S_{i-1}, \mathbf{S_i} \cup \mathbf{U}, S_{i+1}, \ldots, S_m\} >$ has a feasible solution tree.*

Definition 4.2 *Let $H =< V, \mathcal{S} >$ be a hypergraph, $U \subset V$ is an **intersection cover set** of $\mathcal{S}' \subset \mathcal{S}$ if for every set of clusters $\{S_{i_1}, \ldots, S_{i_q}\} \subseteq \mathcal{S}'$, $q \geq 2$, when $\bigcap_{j=1}^{q} S_{i_j} \neq \phi$, there is at least one node $u \in U$ such that $u \in \bigcap_{j=1}^{q} S_{i_j}$. In this case we say that u **covers** the intersection $\bigcap_{j=1}^{q} S_{i_j}$.*

Definition 4.3 *Let $H =< V, \mathcal{S} >$ be a hypergraph, H is a **cycled hypergraph** if the clusters in $\mathcal{S}$ satisfy the Helly Property and if $G_{int}(\mathcal{S})$ is composed only of chordless cycles, each one of size ≥ 4.*

Definition 4.4 *Let $H =< V, \mathcal{S} >$ be a hypergraph, for every $v \in V$ and $i \in \{1, \ldots, m\}$ define $\boldsymbol{MI(v, i)}$ to be the maximal set of indices contained in $\{1, \ldots, i-1, i+1, \ldots, m\}$, such that $v \in \bigcap_{j \in MI(v)} S_j$.*

Note that $MI(v, i) \geq 1$ for every $v \in \bigcup_{j \neq i} S_j$.

Definition 4.5 *Let $H =< V, \mathcal{S} >$ be a hypergraph, and let $U \subset V$ define $S_j^U = S_j \cap U \; \forall S_j \in \mathcal{S}$.*

Lemma 4.6 *In Algorithm IC (Figure 3), with U an intersection cover set of $\mathcal{S} \backslash \{S_i\}$, for every $j, k \in \{1, \ldots, i-1, i+1, \ldots, m\}$, $j \neq k$, $S_j \cap S_k \neq \phi$ if and only if $S_j^U \cap S_k^U \neq \phi$.*

Proof: Suppose that $S_j \cap S_k \neq \phi$. According to definition 4.2 and since U is an intersection cover set of $\mathcal{S} \backslash \{S_i\}$, there exists $u \in U$ such that $u \in S_j \cap S_k$. According to the way S_j^U and S_k^U are defined, $u \in S_j^U \cap S_k^U$ which proves that $S_j^U \cap S_k^U \neq \phi$.

Suppose that $S_j^U \cap S_k^U \neq \phi$. According to the way these clusters were defined $S_j \supseteq S_j^U$ and $S_k \supseteq S_k^U$, hence $S_j \cap S_k \supseteq S_j^U \cap S_k^U \neq \phi$. □

IC (Insertion Cluster)
input
A hypergraph $H =< V, \mathcal{S} >$, where $\mathcal{S} = \{S_1, \ldots, S_m\}$.
A set $U \subset V$ which is an intersection cover set of $\mathcal{S} \backslash \{S_i\}$.
Assumptions
H is a cycled hypergraph with no feasible solution tree.
All the cycles in $G_{int}(\mathcal{S})$ contain node s_i.
returns
A feasible solution tree T for $H' =< V, \{S_1, \ldots, S_{i-1}, S_i \cup U, S_{i+1}, \ldots, S_m\} >$.
begin
for every $(j \neq i)$
    Define $S_j^U = S_j \cap U$.
end for
Define a hypergraph $H^U =< U, \{S_1^U, \ldots, S_{i-1}^U, S_{i+1}^U, \ldots, S_m^U\} >$.
Find T a feasible solution tree for H^U (T exists by Lemma 4.7).
for every $(v \in ((\bigcup_{j \neq i} S_j) \backslash U))$
    if $(|MI(v,i)| == 1)$
      then
        Find $k \neq i$ such that $v \in S_k$ and choose a node $u \in S_k^U$.
      else
        Find $u \in U$ that covers the intersection $\bigcap_{j \in MI(v,i)} S_j$.
    end if
    Add to T the edge (v, u).
end for
Choose a node $u \in U$.
Add to T a star whose center is u and spans the vertices of $S_i \backslash (U \cup (\bigcup_{j \neq i} S_j))$.
return T.
end IC

Fig. 3: Algorithm IC

Lemma 4.7 *In Algorithm IC (Figure 3),* $H^U =< U, \{S_1^U, \ldots, S_{i-1}^U, S_{i+1}^U, \ldots, S_m^U\} >$ *has a feasible solution tree.*

Proof: According to Lemma 4.6, two nodes in $G_{int}(\{S_1^U, \ldots, S_{i-1}^U, S_{i+1}^U, \ldots, S_m^U\})$ intersect if and only if the corresponding nodes intersect in $G_{int}(\{S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_m\})$. According to the assumption of the algorithm, s_i is contained in every chordless cycle of $G_{int}(\mathcal{S})$.

Therefore, $G_{int}(\{S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_m\})$ does not contain any chordless cycle and neither does $G_{int}(\{S_1^U, \ldots, S_{i-1}^U, S_{i+1}^U, \ldots, S_m^U\})$.

In addition, since H is cycled the clusters in $\{S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_m\}$ satisfy the Helly Property. According to Lemma 4.6 the clusters in $\{S_1^U, \ldots, S_{i-1}^U, S_{i+1}^U, \ldots, S_m^U\}$ also satisfy the Helly Property.

By Theorem 1.1 H^U has a feasible solution tree. □

Lemma 4.8 *The output* T *of Algorithm IC (Figure 3) is a tree which spans* V.

Proof: In the algorithm, T is initialized to be a feasible solution tree of H^U. In this stage T is a spanning tree of the vertices in U.

Consider a vertex $v \in V \backslash U$. There are two options:

1. $v \in (\bigcup_{j \neq i} S_j) \backslash U$, in this case the algorithm finds the set of indices $MI(v, i)$ and a vertex $u \in U$ such that $u \in \bigcap_{j \in MI(v,i)} S_j$. Vertex u exists since U is an intersection cover set. Next, the algorithm adds v to T as a leaf.
2. $v \in S_i \backslash (U \cup (\bigcup_{j \neq i} S_j))$, in this case v is added to T as a leaf in the last step of the algorithm.

Hence, T is a tree which touches all the vertices of V. □

Lemma 4.9 *In the output tree* T *returned by Algorithm IC (Figure 3) ,* $T[S_k]$ *is a connected subtree for every* $k \in \{1, \ldots, m\}$, $k \neq i$.

Proof: In Algorithm IC, T is initialized to be a feasible solution tree of H^U. Therefore, at this stage $T[S_k^U]$ is a subtree spanning $S_k \cap U$. In all the following steps of the algorithm $T[S_k^U]$ does not change.

Consider a vertex $v \in S_k \backslash U$, in this case $v \in ((\bigcup_{j \neq i} S_j) \backslash U)$.

If $|MI(v, i)| = 1$ then S_k is the only cluster to contain v. Since $G_{int}(\mathcal{S})$ is cycled, S_k intersects with at least one other cluster $S_j \neq S_i$. U is an intersection cover set, therefore there exists $u \in U$ which covers $S_j \cap S_k$, proving that Algorithm IC can find $u \in S_k^U$.

If $|MI(v,i)| > 1$ and since U is an intersection cover set of $\mathcal{S}\backslash\{S_i\}$, there exists $u \in U$ which covers $\bigcap_{j \in MI(v,i)} S_j$. Since $\bigcap_{j \in MI(v,i)} S_j \subset S_k$, it follows that also in this case $u \in S_k^U$.

In both cases Algorithm IC connects v to $T[S_k^U]$ by an edge (v,u) with $u \in S_k^U$. At the end of the algorithm, all the vertices of S_k are either in $T[S_k^U]$ or connected to it by an edge whose both endpoints are in S_k, thus proving the required subtree. □

Lemma 4.10 *In the output tree T returned by Algorithm IC (Figure 3), $T[S_i \cup U]$ is a connected subtree.*

Proof: In Algorithm IC, T is initialized to be a feasible solution tree of H^U, therefore it is a spanning tree of U.

Consider a vertex $v \in S_i\backslash U$. If there is $k \neq i$ such that $v \in S_k$, by Lemma 4.9 $T[S_k]$ is a connected subtree and thus v is connected by an edge to $u \in S_k^U \subset U$. Otherwise, S_i is the only cluster to contain v and $v \in S_i\backslash(U \cup (\bigcup_{j \neq i} S_j))$. In the last step, Algorithm IC connects v by an edge to a vertex $u \in U$.

In both cases, Algorithm IC connects v to $T[U]$ by an edge (v,u) with $u \in U$. At the end of the algorithm, all the vertices of $S_i \cup U$ are either in $T[U]$ or connected to it by an edge whose both endpoints are in $S_i \cup U$, thus proving the required subtree. □

Corollary 4.11 *Let $H =< V, \mathcal{S} >$ be a hypergraph. If H is a cycled hypergraph and there is a node s_i which is contained in all the chordless cycles of $G_{int}(\mathcal{S})$, then T returned by Algorithm IC (Figure 3) is a feasible solution tree for $H' =< V, \{S_1, \ldots, S_{i-1}, S_i \cup U, S_{i+1}, \ldots, S_m\} >$ and S_i is an insertion cluster of H.*

Theorem 4.12 *Let $H =< V, \mathcal{S} >$ be a hypergraph with no feasible solution tree. If the clusters of H satisfy the Helly Property and if there is at least one node which is contained in every chordless cycle of $G_{int}(\mathcal{S})$, then H has an insertion cluster.*

Proof: Let $\{S_{i_1}, \ldots, S_{i_p}\} \subseteq \mathcal{S}$ be the set of clusters constructing the chordless cycles of $G_{int}(\mathcal{S})$. Consider the induced hypergraph $H^C = H[\{S_{i_1}, \ldots, S_{i_p}\}]$. H^C is a cycled hypergraph and all the cycles in the corresponding intersection graph $G_{int}(\{S_{i_1}, \ldots, S_{i_p}\})$ share a node, denote this node s_i. According to Corollary 4.11, S_i is an insertion cluster of H^C and there is $U \subset V$ such that the hypergraph, created by adding U to S_i in H^C has a feasible solution tree. By Theorem 1.1, this hypergraph has a chordal intersection graph and its clusters satisfy the Helly Property.

Therefore, hypergraph $< V, \{S_1, \ldots, S_{i-1}, S_i \cup U, S_{i+1}, \ldots, S_m\} >$ has a chordal intersection graph and its clusters satisfy the Helly property. By Theorem 1.1, it has a feasible solution tree. □

Lemma 4.13 *Let $H =< V, \mathcal{S} >$ be a hypergraph with no feasible solution tree. If $G_{int}(\mathcal{S})$ has no node which is contained in all its chordless cycles, then H does not have an insertion cluster, even if its clusters satisfy the Helly Property.*

Proof: Suppose by contradiction that H has an insertion cluster S_i. By the lemma's assumption, $G_{int}(\mathcal{S})$ has a chordless cycle C, which does not contain node s_i. Therefore, adding vertices to S_i does not add chords to C. Hence, after the vertices insertion, the intersection graph still contains cycle C which is chordless. By Theorem 1.1, the hypergraph has no feasible solution tree, contradicting the assumption that S_i is an insertion cluster. □

Another important question is achieving minimum cardinality set of vertices to be added to an insertion cluster, in order to achieve feasibility. In the following Lemma we prove that the set of added vertices is an intersection cover set. The discussion which follows Lemma 4.14 bellow, considers the required minimality.

Lemma 4.14 *Let $H =< V, \mathcal{S} >$ be a hypergraph with no feasible solution tree. Suppose that H is a cycled hypergraph with a node s_i which is contained in all the chordless cycles of $G_{int}(\mathcal{S})$. If adding $U \subset V$ to S_i creates a hypergraph $H' =< V, \{S_1, \ldots, S_{i-1}, S_i \cup U, S_{i+1}, \ldots, S_m\} >$ which has a feasible solution tree, then U is an intersection cover set of $\mathcal{S} \backslash \{S_i\}$.*

Proof: First we prove that $S_j \cap U \neq \phi \; \forall j \in \{1, \ldots, m\}, j \neq i$. Suppose by contradiction that there exists some cluster S_j with $S_j \cap U = \phi$. Consider a cycle C in $G_{int}(\mathcal{S})$ containing S_j. Since H is a cycled hypergraph, its clusters satisfy the Helly Property and C contains at least 4 nodes. Let S_{j_l} and S_{j_r} be two clusters in C (not necessarily distinct), each one on a different path in C between S_i and S_j. Choose S_{j_l} (respectively Sj_r) to be the cluster closest to S_j which satisfy $S_{j_l} \cap U \neq \phi$ ($S_{j_r} \cap U \neq \phi$) if it exists, otherwise let $S_{j_l} = S_i$ ($S_{j_r} = S_i$). Let s'_i be the node representing $S_i \cup U$ in the intersection graph of H'. This intersection graph contains the cycle $C' = s'_i \cdots s_{j_l} \cdots s_j \cdots s_{j_r} \cdots s'_i$. According to the Lemma's assumption, H' has a feasible solution tree. Therefore, according to Theorem 1.1, C' contains exactly 3 nodes. Without loss of generality, suppose that C' contains nodes $s'_i = s_{j_l}, s_j$ and s_{j_r}. According to Theorem 1.1, the clusters of H' also satisfy the Helly Property, therefore there exists $v \in (S_i \cup U) \cap S_j \cap S_{j_r}$. Since $S_j \cap U = \phi$ it follows that $v \in S_i \cap S_j \cap S_{j_r}$, contradicting the fact that cycle C contains at least 4 nodes.

Consider $q \geq 2$ distinct clusters $S_{j_1}, \ldots, S_{j_q}$ for $\{j_1, \ldots, j_q\} \subseteq \{1, \ldots, i-1, i+1, \ldots, m\}$ with $S_{j_1} \cap \ldots \cap S_{j_q} \neq \phi$ and $S_{j_1} \cap \ldots \cap S_{j_q} \not\subset S_i$. Since $S_{j_1} \cap U \neq \phi$, …, $S_{j_q} \cap U \neq \phi$, in H' the $q+1$ clusters $S_{j_1}, \ldots, S_{j_q}, S_i \cup U$ pairwise intersect. Since the clusters in H' satisfy the Helly Property $S_{j_1} \cap \ldots \cap S_{j_q} \cap (S_i \cup U) \neq \phi$ giving that $S_{j_1} \cap \ldots \cap S_{j_q} \cap U \neq \phi$. Thus, there exists $u \in U$ which covers the intersection $S_{j_1} \cap \ldots \cap S_{j_q}$. □

Theorem 4.15 *Let $H =< V, \mathcal{S} >$ be a hypergraph with no feasible solution tree. Suppose the clusters of H satisfy the Helly Property and there is at least one node s_i which is contained in every chordless cycle of $G_{int}(\mathcal{S})$. If adding $U \subset V$ to S_i creates a hypergraph which has a feasible solution tree, then U is an intersection cover set of all the clusters, excluding S_i, which are contained in the chordless cycles in the intersection graph.*

Proof: Similar to the proof of Theorem 4.12, using Lemma 4.14. □

According to Theorems 4.12 and 4.15, when $H =< V, \mathcal{S} >$ is a hypergraph whose clusters satisfy the Helly Property and s_i is a node which is contained in all the chordless cycles of $G_{int}(\mathcal{S})$, then S_i is an insertion cluster. Adding the vertex set $U \subset V$ to S_i creates a hypergraph with a feasible solution tree if and only if U is an intersection cover set of all the clusters which are contained in the chordless cycles, excluding S_i. Thus, finding the minimum number of added vertices to S_i is equivalent to finding the minimum cardinality intersection cover set. Since the hypergraph satisfy the Helly Property, this is equivalent to finding all the maximal cliques of the intersection graph induced on the chordless cycles, which may require exponential complexity in the general case.

However, for the more common case when $nc(v) \leq k$, for a constant k, for all vertices in the hypergraph induced on the chordless cycles, the question of finding minimum cardinality intersection cover set becomes polynomial. If $G_{int}(\mathcal{S})$ contains exactly one chordless cycle of p nodes, every vertex v of the induced hypergraph on this cycle satisfies that $nc(v) \leq 2$. In this case, the minimum intersection cover set will contain $p - 2$ vertices.

All the above discussion assumes that the clusters satisfy the Helly Property. Adding vertices to an insertion cluster may increase $nc(v)$ of a vertex v by at most 1. Therefore, if the intersection graph contains a clique on the nodes $s_{i_1}, s_{i_2}, \ldots, s_{i_p}$ and $\max\{nc(v) | v \in \cup_{j=1}^{q} S_{i_j}\} < p-1$ then the hypergraph has no insertion cluster. Furthermore, if the clusters do not satisfy the Helly Property, H has S_i as an insertion cluster if the following two conditions are satisfied:

- In the intersection graph the corresponding node s_i is contained in all the chordless cycles.
- Adding vertices to S_i causes the clusters of every clique of the intersection graph to satisfy the Helly Property.

5 Summary and Further Research

In this paper we introduce different algorithms for the *CST* problem. The first algorithm introduced in this paper creates a weighted graph where the weight of the maximum spanning tree of this graph indicates whether a feasible solution tree exists. Furthermore, the maximum spanning tree offers a feasible solution,

if one exists. The other methods introduced in this paper decide whether a feasible solution exists and find such a tree when it exists, for some special structures of the intersection graph.

For those instances where no feasible solution tree exists, we characterize when adding vertices to exactly one cluster will gain feasibility. This approach finds the appropriate cluster and the vertices that should be added.

Further research may be applied to find possible vertices insertion to more than one cluster of the given hypergraph. It is of special interest to define which insertions to perform and how to measure their minimality.

Acknowledgements

We thank Ephraim Korach for introducing us this problem. We would also like to thank Refael Hassin for his careful reading and helpful suggestions.

References

S. Anily, J. Bramel, and A. Hertz. A $\frac{5}{3}$-approximation algorithm for the clustered traveling salesman tour and path problems. *Oper. Res. Lett.*, 24(1-2):29–35, 1999. ISSN 0167-6377. doi: 10.1016/S0167-6377(98)00046-7. URL http://dx.doi.org/10.1016/S0167-6377(98)00046-7.

E. M. Arkin, R. Hassin, and L. Klein. Restricted delivery problems on a network. *Networks*, 29(4):205–216, 1997. ISSN 0028-3045. doi: 10.1002/(SICI)1097-0037(199707)29:4⟨205::AID-NET3⟩3.3.CO;2-X. URL http://dx.doi.org/10.1002/(SICI)1097-0037(199707)29:4<205::AID-NET3>3.3.CO;2-X.

K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. System Sci.*, 13(3):335–379, 1976. ISSN 0022-0000. Working Papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N. M., 1975).

J. A. Chisman. The clustered traveling salesman problem. *Computers & Operations Research*, 2(2): 115–119, 1975.

N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

M. C. Dourado, F. Protti, and S. J. L. Complexity aspects of the helly property: graphs and hypergraphs. *Electronic J. of Combinatorcs, Dymanic Surveys*, DS 17, 2009.

P. Duchet. Propriété de Helly et problèmes de représentation. *Colloqu. Internat. CNRS, Problemes Conbinatoires et Theorie du Graphs, Orsay, France*, 260:117–118, 1976.

C. Flament. Hypergraphes arborés. *Discrete Math.*, 21(3):223–227, 1978. ISSN 0012-365X. doi: 10.1016/0012-365X(78)90154-1. URL https://doi.org/10.1016/0012-365X(78)90154-1.

N. Guttmann-Beck, R. Hassin, S. Khuller, and B. Raghavachari. Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. *Algorithmica*, 28(4):422–437, 2000. ISSN 0178-4617. doi: 10.1007/s004530010045. URL http://dx.doi.org/10.1007/s004530010045.

N. Guttmann-Beck, E. Knaan, and M. Stern. Approximation algorithms for not necessarily disjoint clustered tsp. *Journal of Graph Algorithms and Applications*, 22:555–575, 01 2018. doi: 10.7155/jgaa.00478.

E. Korach and M. Stern. The clustering matroid and the optimal clustering tree. *Math. Program.*, 98 (1-3, Ser. B):385–414, 2003. ISSN 0025-5610. doi: 10.1007/s10107-003-0410-x. URL http://dx.doi.org/10.1007/s10107-003-0410-x. Integer programming (Pittsburgh, PA, 2002).

E. Korach and M. Stern. The complete optimal stars-clustering-tree problem. *Discrete Appl. Math.*, 156 (4):444–450, 2008. ISSN 0166-218X. doi: 10.1016/j.dam.2006.12.004. URL http://dx.doi.org/10.1016/j.dam.2006.12.004.

F. C. J. Lokin. Procedures for travelling salesman problems with additional constraints. *European Journal of Operational Research*, 3(2):135–141, 1979. URL http://doc.utwente.nl/68455/.

T. A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999. ISBN 0-89871-430-3. doi: 10.1137/1.9780898719802. URL https://doi.org/10.1137/1.9780898719802.

J. Y. Potvin and F. Guertin. A genetic algorithm for the clustered traveling salesman problem with a prespecified order on the clusters. In *Advances in computational and stochastic optimization, logic programming, and heuristic search*, volume 9 of *Oper. Res./Comput. Sci. Interfaces Ser.*, pages 287–299. Kluwer Acad. bl., Boston, MA, 1998. doi: 10.1007/978-1-4757-2807-1_11. URL http://dx.doi.org/10.1007/978-1-4757-2807-1_11.

P. J. Slater. A characterization of soft hypergraphs. *Canad. Math. Bull.*, 21(3):335–337, 1978. ISSN 0008-4395. doi: 10.4153/CMB-1978-058-5. URL https://doi.org/10.4153/CMB-1978-058-5.

R. Swaminathan and D. K. Wagner. On the consecutive-retrieval problem. *SIAM J. Comput.*, 23(2):398–414, 1994. ISSN 0097-5397. doi: 10.1137/S0097539792235487. URL https://doi.org/10.1137/S0097539792235487.