Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 1, pp. 517–541 (2026)
DOI: 10.7155/jgaa.v30i1.3172

Achieving Feasibility for Clustered TSP Using Intersection Graphs

Nili Guttmann-Beck[1] *Hadas Meshita-Sayag*[1] *Michal Stern*[1,2]

[1]Academic College of Tel-Aviv Yaffo, Yaffo, Israel
[2]Caesarea Rothschild Institute, University of Haifa, Haifa, Israel

Submitted: January 2026 Accepted: October 2026 Published: Nov. 2026

Article type: Regular Paper Communicated by: A. I. Tomescu

Abstract. Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph, where V is a set of vertices and $\mathcal{S}$ is a set of clusters $S_1, \ldots, S_m$, $S_i \subseteq V$, such that the clusters in $\mathcal{S}$ are not necessarily disjoint. This paper explores the feasibility version of the Clustered Traveling Salesman Problem, which involves determining the existence of a simple path that visits every vertex precisely once, where the vertices contained in each cluster are visited consecutively. The focus is on hypergraphs without a feasible solution path. We explore the idea of updating the hypergraph by removing vertices from clusters, such that the resulting hypergraph has a feasible solution path. We use specific structures of the corresponding intersection graph to find the minimum number of vertices that need to be removed, to gain feasibility.

1 Introduction

Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph, where V is a set of vertices and $\mathcal{S}$ is a set of clusters, also known as hyperedges, $S_1, \ldots, S_m$, $S_i \subseteq V$, for $i \in \{1, \ldots, m\}$, such that the clusters in $\mathcal{S}$ are not necessarily disjoint. The Feasibility Clustered Traveling Salesman Problem aims to decide whether there exists a simple path that visits each vertex exactly once, such that the vertices of each cluster are visited consecutively.

The main emphasis of this paper is to identify feasible solution paths by removing vertices from clusters within a given hypergraph. This is accomplished by considering different structures of the intersection graph of a given hypergraph. We will demonstrate that even for simple structures of the intersection graph, the corresponding hypergraph has no feasible solution path. Thus, the removal of vertices is required to attain a feasible solution path. In this paper we prove that when the intersection graph is a simple path, the hypergraph has a feasible solution path. However, for hypergraphs whose intersection graphs are a simple cycle, a star with paths or a caterpillar with

E-mail addresses: becknili@mta.ac.il (Nili Guttmann-Beck) hadas.sayag@gmail.com (Hadas Meshita-Sayag) stern@mta.ac.il (Michal Stern)

This work is licensed under the terms of the CC-BY license.

paths (which is a generalization of the known caterpillar structure), we prove that the hypergraph has no feasible solution path. For these cases, we present algorithms that find a minimum number of vertices to be removed, to gain feasibility. In addition, we analyze the time complexity of these algorithms. After removing the appropriate vertices, we can use the algorithm of Booth and Lueker [3] to find a feasible solution path for the updated hypergraph.

The Feasibility Clustered Traveling Salesman Problem is the feasibility problem of the better known optimization problem, the Clustered Traveling Salesman Problem (*CTSP*). Given lengths between every two vertices, the objective is to compute a shortest Hamiltonian path that visits each of the vertices once, such that the vertices of each cluster are visited consecutively. *CTSP* may be considered as a generalization of the classic Traveling Salesman Problem (*TSP*), where $S_i = \{v_i\}$ and $m = |V|$. *CTSP* may also be considered as a generalization of the problem where the clusters are disjoint. While for every instance of the problem with disjoint clusters, there exists a feasible solution path, considering the problem where the clusters are not necessarily disjoint, may impose additional constraints creating instances with no feasible solution path. *TSP* and the disjoint-clustered *TSP* are known to be NP-hard [6].

The feasibility Clustered Traveling Salesman Problem is equivalent to testing the Consecutive Ones Property(*COP*). It can easily be shown that an instance of *COP* has a solution if and only if the corresponding hypergraph has a feasible solution path. Booth and Lueker presented in [3] a data structure referred to as a PQ-Tree, along with an algorithm that uses this data structure to efficiently address the *COP* in a linear time complexity. In instances where a given hypergraph has no feasible solution path, an extended version of the Booth-Lueker algorithm is introduced by Guttmann-Beck, Meshita-Sayag, and Stern in [10]. This modified algorithm proposes a list of vertices whose removal gains feasibility.

Extensive research has been conducted on the disjoint-clustered *TSP*, where the clusters are vertex disjoint. In [4] a heuristic approach is proposed to tackle this problem. Additionally, in [12], a branch and bound algorithm is presented as a solution strategy. Bounded-approximation algorithms can be found in [2] and [9]. Considering the ordered disjoint-clustered *TSP*, a study conducted in [1] introduces a $\frac{5}{3}$-approximation algorithm. In [14] they present a genetic algorithm specifically designed to address this problem. However, it is important to note that these algorithms are not applicable to *CTSP*. The presence of vertices within the intersections of clusters imposes additional constraints on the required path, making the abovementioned algorithms insufficient for solving the *CTSP*.

In [12] the authors offer numerous applications of the Feasibility Clustered Traveling Salesman Problem within the robotics field. One potential application involves the management of customer orders in a warehouse system. These orders often comprise multiple commodities, which are collected by either a motorized truck or a robot. The robot has the capability to handle multiple orders simultaneously, with the constraint that each order must be processed consecutively. The main challenge lies in determining a route that covers all the orders. In the feasibility variant of this problem, a hypergraph representation can be employed. Each order is represented by a cluster, while each commodity is represented as a vertex. In cases where a feasible solution path is not achievable, a suggestion is made to enhance feasibility by removing certain commodities from specific orders.

The field of bioinformatics offers another potential application for the Feasibility Clustered Traveling Salesman Problem, particularly in the construction of physical maps of chromosomes, as discussed in [5]. In this context, each chromosome is mapped using probes that correspond to distinct points on the chromosome, and clones that represent intervals of the chromosome. By conducting hybridization experiments, it is possible to determine whether a given clone contains a

specific probe. The primary challenge is to reconstruct the probes, to satisfy the condition that all probes hybridizing to a particular clone appear consecutively in the final solution. In this problem, each probe is represented as a vertex, and each clone is represented by a cluster in an instance of the Feasibility Clustered Traveling Salesman Problem. In cases where a feasible solution path cannot be found, one possible approach is to gain feasibility by removing certain probes from specific clones.

Throughout this paper we assume that there are no two identical clusters in a given hypergraph. In cases where such identical clusters exist, when addressing the question of feasibility, we can disregard one of these clusters.

The paper is organized as follows: Section 2 contains some basic definitions. Section 3 presents some general claims that are used throughout the paper. These claims focus on induced hypergraphs and their feasible solutions paths. Section 4 refers to clusters that are either containing or contained. We prove that one can ignore these clusters when verifying whether an instance of the problem has a feasible solution path. Section 5 focuses on hypergraphs whose intersection graphs are simple chordless paths. Section 6 focuses on hypergraphs whose intersection graphs are chordless cycles. Section 7 focuses on hypergraphs whose intersection graphs are stars with paths. Section 8 focuses on hypergraphs whose intersection graphs are caterpillars with paths.

2 Definitions

This section includes the basic definitions used throughout this paper.

Definition 2.1. *Let $H = \langle V, \mathcal{S} \rangle$ 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 $i \in \{1, \ldots, m\}$, such that the clusters in $\mathcal{S}$ are not necessarily disjoint. The* **Feasibility Clustered Traveling Salesman Problem** *is to decide whether there exists a simple path, in a complete graph on V, that visits each vertex exactly once, such that the vertices of each cluster are visited consecutively.*

Hypergraph $H = \langle V, \mathcal{S} \rangle$ can be represented by the matrix $\boldsymbol{M_H}$. In M_H each row represents a vertex $v \in V$ and each column represents a cluster $S_i \in \mathcal{S}$. The value of cell $[v, S_i]$ in M_H is equal to 1 if $v \in S_i$, otherwise the value is set to 0.

Remark 2.2. *Throughout this paper, we assume that every vertex belongs to at least one cluster, that is: $V = \bigcup_{S_i \in \mathcal{S}} S_i$. Vertices that do not belong to any cluster impose no consecutiveness constraints and may therefore be appended arbitrarily to any feasible solution path.*

Definition 2.3. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with vertex set V, $|V| = n$, and $\mathcal{S} = \{S_1, \ldots, S_m\}$, $S_i \subseteq V$ for $i \in \{1, \ldots, m\}$, a set of not necessarily disjoint clusters. The* **intersection graph of $\boldsymbol{H}$**, *denoted by $\boldsymbol{G_{int}(\mathcal{S})}$, is defined to be a graph whose set of nodes is $\{s_1, \ldots, s_m\}$, where s_i corresponds to $S_i \in \mathcal{S}$, and an edge (s_i, s_j) exists if and only if $S_i \cap S_j \neq \emptyset$ for $S_i, S_j \in \mathcal{S}$, $i \neq j$.*

$G_{int}(\mathcal{S})$ is represented by:

- *M_G - An adjacency matrix of size $m \times m$, where each row and each column represents a node in $G_{int}(\mathcal{S})$. The value of cell $[s_i, s_j]$ in M_G is equal to $|S_i \cap S_j|$. Hence, for $i \neq j$, the value of a cell is greater than zero if and only if there exists an edge between the corresponding nodes, s_i and s_j, in $G_{int}(\mathcal{S})$. For $i = j$, we set $[s_i, s_i] = 0$, since diagonal entries do not correspond to edges of the intersection graph. Note that, M_G is a symmetric matrix.*

- *V_D - A degree vector of size m. The value of $V_D[s_i]$ is the degree of node s_i in $G_{int}(\mathcal{S})$.*

- V_S - *A cluster size vector of size* m. *The value of* $V_S[s_i]$ *is* $|S_i|$.

Note that, creating a representation of an intersection graph, by calculating M_G, V_D *and* V_S, *can be done in* $O(nm^2)$.

Definition 2.4. *Let* $H = \langle V, \mathcal{S} \rangle$ *be a hypergraph and let* $\mathcal{S}' \subseteq \mathcal{S}$ *be a set of clusters. We define* $\boldsymbol{H[\mathcal{S}']}$ *to be the* **induced hypergraph on** $\boldsymbol{\mathcal{S}'}$, *whose vertex set is* $\boldsymbol{V(\mathcal{S}')} = \bigcup_{S_i \in \mathcal{S}'} S_i$ *and cluster set is* $\mathcal{S}'$. *Furthermore, the induced intersection graph on* $\mathcal{S}'$ *is the intersection graph of* $H[\mathcal{S}']$ *and therefore can be denoted by* $\boldsymbol{G_{int}(\mathcal{S}')}$.

Definition 2.5. *Let* $H = \langle V, \mathcal{S} \rangle$ *be a hypergraph, consider the complete graph on* V *and let* P *be a path on the vertices of* V. *We define* $\boldsymbol{P[V']}$, *for* $V' \subseteq V$, *as the collection of subpaths that contain all the vertices of* V' *and all the edges of* P *whose both endpoints are in* V'. *Furthermore, we define* $\boldsymbol{P[\mathcal{S}']}$, *for* $\mathcal{S}' \subseteq \mathcal{S}$, *to be* $P[V(\mathcal{S}')]$.

Definition 2.6. *Let* $H = \langle V, \mathcal{S} \rangle$ *be a hypergraph.* RL *is a* **removal list of** $\boldsymbol{H}$ *if* RL *is a list of pairs:* $RL = \{(v_1, S_{i_1}), \ldots, (v_k, S_{i_k})\}$ *with* $v_j \in S_{i_j}$, *such that the removal of vertex* v_j *from cluster* S_{i_j}, *for every pair* $(v_j, S_{i_j}) \in RL$, *creates a new instance of the hypergraph, denoted by* $\boldsymbol{H \setminus RL}$. *We also denote by* $\boldsymbol{\mathcal{S} \setminus RL}$ *the cluster set of the new instance and by* $\boldsymbol{G_{int}(\mathcal{S} \setminus RL)}$ *the corresponding intersection graph.*
If $H \setminus RL$ *has a feasible solution path, we say that* RL *is a* **feasible removal list of** $\boldsymbol{H}$.
Consider a minimum cardinality feasible removal list of H *and denote* $\boldsymbol{mRL(H)}$= *min* $\{|RL| \mid RL$ *is a feasible removal list of* $H\}$, *the cardinality of such a list.*
Note that, if a given hypergraph H *has a feasible solution path, then it has an empty feasible removal list and* $mRL(H) = 0$.
For simplicity of notation, we denote a removal list to contain pair $\boldsymbol{(U, S_i)}$ *where* $U \subseteq S_i$, *when we remove all the vertices of* U *from* S_i. *Note that, notation* (U, S_i) *may be used as a removal list.*

Definition 2.7. *Let* $H = \langle V, \mathcal{S} \rangle$ *be a hypergraph. If* $RL = \{(v_1, S_{i_1}), \ldots, (v_k, S_{i_k})\}$ *is a removal list and* $\mathcal{S}' \subseteq \mathcal{S}$ *is a set of clusters, we define the* **induced removal list** $\boldsymbol{RL[\mathcal{S}']}$ *to be* $\{(v, S') \mid (v, S') \in RL, S' \in \mathcal{S}'\}$.
Denote by $\boldsymbol{VRL[S_i]}$= $\{v \mid (v, S_i) \in RL\}$, *the vertices removed from* S_i *by* RL.

Definition 2.8. *Given two sets* X *and* Y *we define* $\boldsymbol{SetMin\{X, Y\}}$ *as a set with minimum cardinality among* X *and* Y. *That is:*

$$SetMin\{X, Y\} = \begin{cases} X, & \text{if } |X| \leq |Y|, \\ Y, & \text{otherwise.} \end{cases}$$

Similarly, given a finite collection of sets $X_1, \ldots, X_t$, *for* $t \geq 2$, *we define* $\boldsymbol{SetMin\{X_1, \ldots, X_t\}}$ *as a set of minimum cardinality among* $X_1, \ldots, X_t$. *That is,* $SetMin\{X_1, \ldots, X_t\} = X_{j^\#}$, *where* $j^\# \in \underset{1 \leq j \leq t}{argmin} |X_j|$. *If the minimum is attained by more than one set, one of the minimizing sets is chosen arbitrarily.*

Definition 2.9. *Let* S *and* R *be two intersecting clusters, equivalently, their corresponding nodes are neighbors in the intersection graph. We define* $\boldsymbol{minIntDif(R, S)}$ *as the smaller, in terms of cardinality, of the two sets* $R \cap S$ *and* $R \setminus S$. *That is* $minIntDif(R, S) = SetMin\{R \cap S, R \setminus S\}$.

3 General Claims

In this section we refer to solutions of induced hypergraphs and cases where the intersection graph is composed of more than one connected component. We prove that the feasible solution path induced on the vertices of induced hypergraph is a feasible solution path of the induced hypergraph. Furthermore, we prove that each connected component induces a connected path in any feasible solution path, when it exists. On the other hand, a union of feasible solution paths of different components of the intersection graph compose a solution of the given hypergraph. In addition, we prove that an induced feasible removal list forms a feasible removal list of the induced hypergraph, and that the union of minimum cardinality feasible removal lists, of the different components, creates a minimum cardinality feasible removal list of the given hypergraph.

Lemma 3.1. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with a feasible solution path P, whose intersection graph $G_{int}(\mathcal{S})$ contains a simple path $s_{i_1} - s_{i_2} - \ldots - s_{i_k}$. For every $v \in S_{i_1}, u \in S_{i_x}$, $x \in \{1, \ldots, k\}$, the subpath between v and u in P contains only vertices from $\bigcup_{j=1}^{x} S_{i_j}$.*

In [11] they prove this claim for the problem of finding a spanning tree such that each cluster induces a path. The proof presented in [11] is also valid for the problem discussed in this paper.

Lemma 3.2. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with a feasible solution path P. For every $\mathcal{S}' \subseteq \mathcal{S}$, if $G_{int}(\mathcal{S}')$ is connected, then $P[\mathcal{S}']$ is a connected path which is a feasible solution path of $H[\mathcal{S}']$.*

Proof: Suppose by contradiction that $P[\mathcal{S}']$ is not a connected path. In this case, there are two vertices $v \in V(\mathcal{S}')$ and $u \in V(\mathcal{S}')$ such that the subpath between v and u in P contains a vertex $w \notin V(\mathcal{S}')$. Since $v \in V(\mathcal{S}')$, there exists a cluster $S_v \in \mathcal{S}'$ such that $v \in S_v$. Similarly, a cluster $S_u \in \mathcal{S}'$ exists such that $u \in S_u$. Since $G_{int}(\mathcal{S}')$ is connected, it contains a path $s_v - s_{i_2} - \cdots - s_{i_{k-1}} - s_u$, for $k \geq 2$, which connects s_u and s_v, such that $s_{i_j} \in S_{i_j} \in \mathcal{S}'$ for every $j \in \{2, \ldots, k-1\}$. According to Lemma 3.1, the subpath between v and u in P contains only vertices from $S_v \cup (\bigcup_{j=2}^{k-1} S_{i_j}) \cup S_u$. Hence, all the vertices in P between v and u are contained in $V(\mathcal{S}')$, contradicting the assumption that $w \notin V(\mathcal{S}')$, proving that $P[\mathcal{S}']$ is a connected path.

Since $P[\mathcal{S}']$ does not change the order of the vertices with respect to P, by the feasibility of P, $P[S]$ is a connected path for every $S \in \mathcal{S}'$. Proving that $P[\mathcal{S}']$ is a feasible solution path of $H[\mathcal{S}']$. ■

Corollary 3.3. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with a feasible solution path P. For every $\mathcal{S}' \subseteq \mathcal{S}$, $P[\mathcal{S}']$ is a collection of paths, such that each one is a feasible solution path of a connected component in $H[\mathcal{S}']$.*

Corollary 3.4. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If there exists $\mathcal{S}' \subseteq \mathcal{S}$ such that $H[\mathcal{S}']$ does not have a feasible solution path, then H does not have a feasible solution path.*

Theorem 3.5. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph $G_{int}(\mathcal{S})$ contains k, $k \geq 2$, pairwise disjoint connected components with corresponding cluster sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$. If $H[\mathcal{S}_i]$ has a feasible solution path P_i, for every $i \in \{1, \ldots, k\}$, then the concatenation of $P_1, \ldots, P_k$, in any arbitrary order, is a feasible solution path of H.*

Proof: Let P be the path created by the concatenation of $P_1, \ldots, P_k$. Every cluster $S \in \mathcal{S}$ has exactly one index $i \in \{1, \ldots, k\}$ such that $S \in \mathcal{S}_i$. Since P_i is a feasible solution path of $H[\mathcal{S}_i]$, $P_i[S]$ is a connected path spanning all the vertices of S. P contains P_i, and therefore it also satisfies $P[S] = P_i[S]$, which is a connected path spanning all the vertices of S. Since this is true for every cluster $S \in \mathcal{S}$, P is a feasible solution path of H. ■

Lemma 3.6. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If RL is a feasible removal list of H and $\mathcal{S}' \subseteq \mathcal{S}$, then $RL[\mathcal{S}']$ is a feasible removal list of $H[\mathcal{S}']$.*

Proof: Since RL is a feasible removal list of H, hypergraph $H \setminus RL$ has a feasible solution path, denote it $P^{H \setminus RL}$.

The clusters in $H \setminus RL$ are $\{S_1 \setminus VRL[S_1], S_2 \setminus VRL[S_2], \ldots, S_m \setminus VRL[S_m]\}$, and $P^{H \setminus RL}[S_i \setminus VRL[S_i]]$ is a path, for every $i \in \{1, \ldots, m\}$.

Let $\mathcal{S}' = \{S_{i_1}, \ldots, S_{i_k}\}$. According to Corollary 3.3, $P^{H \setminus RL}[(S_{i_1} \setminus VRL[S_{i_1}]) \cup \ldots \cup (S_{i_k} \setminus VRL[S_{i_k}])]$ is a set of connected subpaths of $P^{H \setminus RL}$, which spans a path on $S_{i_j} \setminus VRL[S_{i_j}]$, for every $j \in \{1, \ldots, k\}$. Furthermore, according to Corollary 3.3, this set of subpaths induces a connected path on every connected component of $G_{int}(\mathcal{S}' \setminus RL[\mathcal{S}'])$. According to Theorem 3.5, these subpaths can be concatenated into a path which is a feasible solution path of $H[\mathcal{S}'] \setminus RL[\mathcal{S}']$, thus proving that $RL[\mathcal{S}']$ is a feasible removal list of $H[\mathcal{S}']$. ∎

Theorem 3.7. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph $G_{int}(\mathcal{S})$ contains k, $k \geq 2$, pairwise disjoint connected components with corresponding cluster sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$. If $RL_1, \ldots, RL_k$ are minimum cardinality feasible removal lists of $H[\mathcal{S}_1], \ldots, H[\mathcal{S}_k]$, respectively, then $\bigcup_{i=1}^{k} RL_i$ is a minimum cardinality feasible removal list of H.*

Proof: Since RL_i is a feasible removal list of $H[\mathcal{S}_i]$, $H[\mathcal{S}_i] \setminus RL_i$ has a feasible solution path, for every $i \in \{1, \ldots, k\}$. According to Theorem 3.5, $\bigcup_{i=1}^{k}(H[\mathcal{S}_i] \setminus RL_i) = H \setminus \bigcup_{i=1}^{k} RL_i$ has a feasible solution path and therefore $\bigcup_{i=1}^{k} RL_i$ is a feasible removal list of H.

Since $\mathcal{S}_1, \ldots, \mathcal{S}_k$ correspond to different connected components of the intersection graph and since $RL_1, \ldots, RL_k$ are minimum cardinality feasible removal lists, $RL_1, \ldots, RL_k$ are pairwise disjoint. Therefore, $|\bigcup_{i=1}^{k} RL_i| = \sum_{i=1}^{k} |RL_i|$.

Suppose by contradiction that $\bigcup_{i=1}^{k} RL_i$ is not a minimum cardinality feasible removal list of H. Hence, there exists a feasible removal list L with $|L| < |\bigcup_{i=1}^{k} RL_i|$. According to Lemma 3.6, $L[\mathcal{S}_i]$ is a feasible removal list of $H[\mathcal{S}_i]$, and therefore $|L[\mathcal{S}_i]| \geq |RL_i|$, for every $i \in \{1, \ldots, k\}$. Since $\mathcal{S}_1, \ldots, \mathcal{S}_k$ correspond to different connected components of the intersection graph, $|L| = \sum_{i=1}^{k} |L[\mathcal{S}_i]| \geq \sum_{i=1}^{k} |RL_i| = |\bigcup_{i=1}^{k} RL_i|$, contradicting the assumption that $\bigcup_{i=1}^{k} RL_i$ is not a minimum cardinality feasible removal list of H. ∎

In [11] they prove the following claim for the problem of finding a spanning tree such that each cluster induces a path. The proof presented in [11] is also valid for the problem discussed in this paper.

Lemma 3.8. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with a feasible solution path P. For every $X = \bigcap_{S_i \in \mathcal{S}'} S_i$, $\mathcal{S}' \subseteq \mathcal{S}$ in H, $P[X]$ is a connected subpath of P.*

4 Containment of Clusters

In this section we consider cases in which the hypergraph has clusters that are containing clusters, contained clusters sets or contained clusters. In these cases, we can solve the problem without the containing clusters set or contained clusters, and then easily update the solution path to get a feasible solution path of the given hypergraph.

Definition 4.1. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. Cluster $S_i \in \mathcal{S}$ is* **a containing cluster**, *if every $S_j \in (\mathcal{S} \setminus \{S_i\})$ satisfies $S_j \subseteq S_i$.*

Definition 4.2. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. $\mathcal{S}' \subsetneq \mathcal{S}$ is* **a contained clusters set** *if all of the following conditions hold:*

1. *The nodes corresponding to $\mathcal{S}'$ induce one connected component in $G_{int}(\mathcal{S})$.*
2. *There exists a cluster $S_i \in \mathcal{S} \setminus \mathcal{S}'$ such that $V(\mathcal{S}') \subseteq S_i$.*
3. *For every cluster $S_j \in \mathcal{S}'$ and every cluster $S_k \in \mathcal{S} \setminus (\mathcal{S}' \cup \{S_i\})$ we have $S_j \cap S_k = \emptyset$.*

Note that, the algorithms presented throughout the paper may change the state of a single cluster to become contained in another cluster, therefore, we use the following definition to define separately a contained clusters set with only one cluster.

Definition 4.3. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. Cluster $S_j \in \mathcal{S}$ is* **a contained cluster**, *if both of the following conditions hold:*

1. *There exists a cluster $S_i \in (\mathcal{S} \setminus \{S_j\})$ such that $S_j \subseteq S_i$.*
2. *For every cluster $S_k \in (\mathcal{S} \setminus \{S_i, S_j\})$ we have $S_j \cap S_k = \emptyset$.*

Theorem 4.4. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph, and let $S_i \in \mathcal{S}$ be a containing cluster in H. $H[\mathcal{S} \setminus \{S_i\}]$ has a feasible solution path if and only if H has a feasible solution path.*

Proof: Denote $\mathcal{S}' = \mathcal{S} \setminus \{S_i\}$, $H' = H[\mathcal{S}']$. Suppose H' has a feasible solution path denoted by P'. Since every cluster $S \in \mathcal{S}'$ also satisfies $S \subseteq S_i$, all the vertices in P' are contained in S_i.

Arrange all the vertices of $(S_i \setminus (V(\mathcal{S} \setminus \{S_i\})))$ in a path, denote this path by P''. Concatenate P' and P'', and denote the new path by P. For every $S \in \mathcal{S}', P[S] = P'[S]$, and therefore is a connected path. P is a path spanning all the vertices of S_i and is obviously connected. Therefore, P is a feasible solution path of H.

The correctness of the opposite direction is obtained directly from Corollary 3.3 and Theorem 3.5. ∎

Theorem 4.5. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph, and let $\mathcal{S}' \subseteq \mathcal{S}$ be a contained clusters set in H, such that $H[\mathcal{S}']$ has a feasible solution path. $H[\mathcal{S} \setminus \mathcal{S}']$ has a feasible solution path if and only if H has a feasible solution path.*

Proof: Let P' be a feasible solution path of $H[\mathcal{S}']$. Since $\mathcal{S}'$ is a contained clusters set, there exists $S_i \in \mathcal{S}$ such that $V(\mathcal{S}') \subseteq S_i$. Furthermore, $S_j \cap S_k = \emptyset$, for every $S_j \in \mathcal{S}'$ and $S_k \in \mathcal{S} \setminus (\mathcal{S}' \cup \{S_i\})$.

Suppose that $H[\mathcal{S} \setminus \mathcal{S}']$ has a feasible solution path denoted by P. Since P is a feasible solution path, $P[S_i]$ is a connected path. In addition, for every $S_k \in \mathcal{S} \setminus (\mathcal{S}' \cup \{S_i\})$, all the vertices of $V(\mathcal{S}')$ which appear in $P[S_i]$ are not contained in $P[S_k]$. Let v be the first vertex from $V(\mathcal{S}')$ to appear inside P. Remove from P all the vertices of $V(\mathcal{S}') \setminus \{v\}$, replace v by P' and concatenate all the created subpaths according to their order in P. Denote the created path by $\tilde{P}$.

By the way it is constructed, $\tilde{P}$ is a connected path. Since P' is a feasible solution path of $H[\mathcal{S}']$, $P'[S_j]$ is a connected path for every $S_j \in \mathcal{S}'$. According to the way $\tilde{P}$ is constructed, $\tilde{P}[S_j] = P'[S_j]$, and is therefore a connected path. The changes performed on P to create $\tilde{P}$, change the internal arrangement of the vertices of S_i, but keep them as a connected path, and therefore $\tilde{P}[S_i]$ is a connected path. For every cluster $S_k \in \mathcal{S} \setminus (\mathcal{S}' \cup \{S_i\})$, the order of the vertices of S_k is not changed and satisfies $\tilde{P}[S_k] = P[S_k]$, therefore $\tilde{P}[S_k]$ is a connected path. Hence, $\tilde{P}$ is a feasible solution path of H.

The correctness of the opposite direction is obtained directly from Corollary 3.3 and Theorem 3.5. ∎

Corollary 4.6. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph, and let $S_i \in \mathcal{S}$ be a contained cluster in H. $H[\mathcal{S} \setminus \{S_i\}]$ has a feasible solution path if and only if H has a feasible solution path.*

Remark 4.7. *Following Theorems 4.4, 4.5 and Corollary 4.6, in the algorithms presented throughout this paper, we assume that H does not have containing clusters, contained clusters sets and contained clusters.*

5 Path Intersection Graphs

This section focuses on hypergraphs whose intersection graphs are simple chordless paths. For simplicity of writing, we will refer to these paths as simple paths. First, we present Algorithm SolvePath, which constructs a feasible solution path for a given hypergraph, whose intersection graph is a simple path. Thus, proving that a feasible solution path always exists. Furthermore, this section contains a time complexity analysis for this algorithm.

Without loss of generality, in constructing the intersection graph, we assume that the nodes representing the clusters are arranged along the path as $s_1, s_2, \ldots, s_m$, where node s_i represents cluster S_i. We also assume that $m \geq 2$.

Procedure SolvePath
Given $H = \langle V, \mathcal{S} \rangle$ a hypergraph with an intersection graph that is a simple path, whose nodes are ordered $s_1, s_2, \ldots, s_m$.
Find the following subsets of vertices and paths:

1. Let $X_1 = S_1 \setminus S_2$ and let P_{X_1} be a path spanning X_1.
2. $\forall j \in \{1, \ldots, m-1\}$ let $Y_j = S_j \cap S_{j+1}$ and let P_{Y_j} be a path spanning Y_j.
3. $\forall j \in \{2, \ldots, m-1\}$ let $W_j = S_j \setminus (S_{j-1} \cup S_{j+1})$ and let P_{W_j} be a path spanning W_j.
4. let $X_m = S_m \setminus S_{m-1}$ and let P_{X_m} be a path spanning X_m.

Concatenate the paths in this order: $P_{X_1}, P_{Y_1}, P_{W_2}, P_{Y_2}, P_{W_3}, \ldots, P_{W_{m-1}}, P_{Y_{m-1}}, P_{X_m}$
to create path P.
Return P.

Theorem 5.1. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If its intersection graph $G_{int}(\mathcal{S})$ is a simple path, then H has a feasible solution path.*

Proof: Since $G_{int}(\mathcal{S})$ is a simple path, it does not contain any cliques of size 3. Therefore, every vertex $v \in V$, is contained in at most two clusters. The returned path P of Procedure SolvePath, spans all the vertices of H, and every cluster appears consecutively in the path, see Figure 1.

Hence, Procedure SolvePath constructs a feasible solution path of H. ∎

Theorem 5.2. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with an intersection graph that is a simple path, whose nodes are ordered $s_1, s_2, \ldots, s_m$. The time complexity of constructing a feasible solution path of H is $O(nm)$, $n = |V|, m = |\mathcal{S}|$.*

Proof: Using Procedure SolvePath:

- Calculate X_1 and Y_1: process the columns of S_1, S_2 in M_H, and look for cells with value 1 only in column of S_1 or in both columns, respectively. The time complexity of this step is $O(n)$.

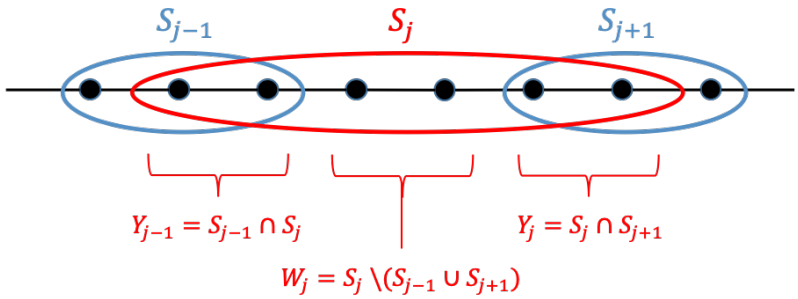


Figure 1: The subclusters of cluster S_i in the proof of Theorem 5.1.

- Calculate Y_j and W_j: process columns of S_{j-1}, S_j, S_{j+1} in M_H. The time complexity of calculating Y_j and W_j is $O(n)$. Performing this stage, for $j \in \{2, \ldots, m-1\}$, requires $O(mn)$ time complexity.
- Finally, calculate X_m. The time complexity of this step is $O(n)$.

Hence, the total time complexity is $O(mn)$. ∎

6 Chordless Cycle Intersection Graphs

This section focuses on hypergraphs with chordless cycles intersection graphs, containing at least four nodes. Using Corollary 6.2, we prove that these hypergraphs have no feasible solution path. We also offer a minimum cardinality feasible removal list with time complexity analysis.

Without loss of generality, in constructing the intersection graph, we assume that the nodes of the intersection graph are arranged along the chordless cycle as $s_1 - s_2 - \cdots - s_m - s_1$, where node s_i represents cluster S_i.

The following theorem, summarized in [13], gives sufficient and necessary conditions for the feasibility decision problem for a given instance of the Clustered Spanning Tree by Trees problem. This problem is: given a hypergraph decide whether there exists a spanning tree of its vertices, such that each cluster is spanned by a subtree.

Theorem 6.1. *([7], [8], [15]) A hypergraph $H = \langle V, \mathcal{S} \rangle$ has a feasible solution tree for the Clustered Spanning Tree by Trees problem if and only if it satisfies the Helly property and its intersection graph is chordal.*

Since the Feasibility Clustered Traveling Salesman Problem is a restricted case of the Clustering Spanning Tree by Trees problem, we have the following corollary:

Corollary 6.2. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$, if its intersection graph contains a chordless cycle with at least four nodes, H has no feasible solution path for the Feasibility Clustered Traveling Salesman Problem.*

We note that when the intersection graph of a hypergraph is a chordless cycle, each cluster in the hypergraph has a non-empty intersection with exactly two other clusters. Also, removing all the vertices of a non-empty intersection from one of the corresponding clusters, removes exactly one edge from the intersection graph, transforming the intersection graph into a simple path.

Definition 6.3. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with an intersection graph that is a chordless cycle, whose nodes are $s_1 - s_2 - \cdots - s_m - s_1$. Let Define* $\boldsymbol{minIntCyc(H)} = j^{\#} \in \underset{1 \le k \le m}{argmin}(|S_k \cap S_{next(k)}|)$, *where* $\boldsymbol{S_{next(k)}}$ *denotes the cluster corresponding to the successor of s_k in the cyclic order. Thus, $j^{\#}$ is an index attaining the minimum size of an intersection between two consecutive clusters in the cycle.*
For $minIntCyc(H) = j^{\#}$, define also $\boldsymbol{CycRL(H)} = (S_{j^{\#}} \cap S_{next(j^{\#})}, S_{j^{\#}})$.

Theorem 6.4. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If its intersection graph is a chordless cycle, then $CycRL(H)$ is a minimum cardinality feasible removal list of H.*

Proof: First, we prove that $CycRL(H)$ is a feasible removal list.

Let $j^{\#} = minIntCyc(H)$. In $G_{int}(\mathcal{S} \setminus CycRL(H))$ edge $(S_{j^{\#}}, S_{next(j^{\#})})$ is removed. Since $G_{int}(\mathcal{S})$ is a chordless cycle, $G_{int}(\mathcal{S} \setminus CycRL(H))$ is a simple path. According to Theorem 5.1, $H \setminus CycRL(H)$ has a feasible solution path, and therefore $CycRL(H)$ is a feasible removal list.

To prove minimality, suppose by contradiction that there exists a feasible removal list RL' with $|RL'| < |CycRL(H)|$. According to Definition 6.3, $|RL'| < |S_k \cap S_{next(k)}|$, for every $1 \le k \le m$. Therefore, none of the edges of $G_{int}(\mathcal{S})$ is removed according to RL' and $G_{int}(\mathcal{S}) = G_{int}(\mathcal{S} \setminus RL')$ is a chordless cycle. According to Corollary 6.2, $H \setminus RL'$ has no feasible solution path, contradicting the assumption that RL' is a feasible removal list. ∎

Theorem 6.5. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph with an intersection graph that is a chordless cycle, whose nodes are $s_1 - s_2 - \cdots - s_m - s_1$. The time complexity of finding $CycRL(H)$, which is a minimum cardinality feasible removal list, is $O(m^2 + n)$, $n = |V|, m = |\mathcal{S}|$.*

Proof: Each cell in M_G stores the size of the appropriate intersection. Finding an intersection with a minimum positive cardinality can be done by scanning off diagonal positive entries in M_G and looking for a cell with a minimum value. This can be done in $O(m^2)$ time complexity. Denote by $S_{j^{\#}}, S_{next(j^{\#})}$ the clusters of the minimum intersection, that is $j^{\#} = minIntCyc(H)$. To find the vertices in this intersection, we process the columns of $S_{j^{\#}}$, $S_{next(j^{\#})}$ in M_H, looking for columns with value 1 in both columns, which can be done in $O(n)$ time complexity. Hence, the total time complexity is $O(m^2 + n)$. ∎

7 Star with Paths Intersection Graphs

This section focuses on hypergraphs whose intersection graphs are stars with paths (see Definition 7.1 and Figure 2). First, we identify a special structure: a cluster that intersects three pairwise disjoint, non-contained clusters. We prove that if a hypergraph contains this structure, then it has no feasible solution path. Next, we show that every hypergraph whose intersection graph is a star with paths contains such a structure, and therefore has no feasible solution path. Finally, we construct a minimum removal list that gains feasibility, and analyze its time complexity.

Definition 7.1. *A graph $G = (U, E)$ is a* **star with paths centered at node u**, *for a node $u \in U$ whose degree is k, for $k \ge 3$, if removing u from G disconnects G into k connected components, such that each component is a simple path, see Figure 2.*

Remark 7.2. *We focus on stars whose centers have degree at least 3, although the standard definition also allows centers of degree 2. This is because the special case in which the center has degree 2 yields a simple path, which is substantially different and is treated separately in Section 5.*

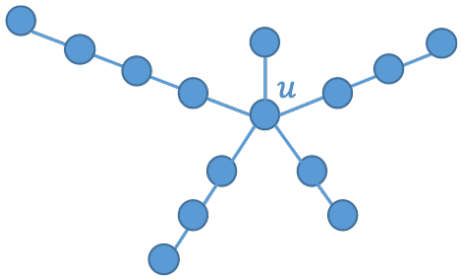


Figure 2: An example of a star with paths centered at node u, with $k = 5$ paths.

We start by stating the following theorem, which focuses on a specific structure of the intersection graph, where the corresponding hypergraph has no feasible solution path. The structure considered in Theorem 7.3 may occur as part of a hypergraph whose intersection graph is a star with paths, where the center has degree $k \geq 3$. Since the theorem shows that such a hypergraph has no feasible solution path, any hypergraph that contains this structure as an induced hypergraph also has no feasible solution path, as stated and proved in Theorem 7.4.

Theorem 7.3. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If there exist four different clusters $S_i, S_{j_1}, S_{j_2}, S_{j_3} \in \mathcal{S}$ such that:*

1. *$S_i \cap S_{j_1} \neq \emptyset$, $S_i \cap S_{j_2} \neq \emptyset$, $S_i \cap S_{j_3} \neq \emptyset$.*
2. *$S_{j_1} \not\subseteq S_i$, $S_{j_2} \not\subseteq S_i$, $S_{j_3} \not\subseteq S_i$.*
3. *S_{j_1}, S_{j_2} and S_{j_3} are pairwise vertex disjoint.*

Then H has no feasible solution path.
(See Figure 3 for the induced intersection graph $G_{int}(\{S_i, S_{j_1}, S_{j_2}, S_{j_3}\})$.)

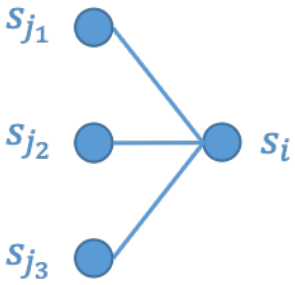


Figure 3: $G_{int}(\{S_i, S_{j_1}, S_{j_2}, S_{j_3}\})$ for Theorem 7.3.

Proof: Suppose by contradiction that H has a feasible solution path P. Denote by $X_1 = S_i \cap S_{j_1}, X_2 = S_i \cap S_{j_2}, X_3 = S_i \cap S_{j_3}$. According to Lemma 3.8, $P[X_1], P[X_2], P[X_3]$ are connected subpaths. Since $X_1 \neq \emptyset, X_2 \neq \emptyset, X_3 \neq \emptyset$, subpaths $P[X_1], P[X_2], P[X_3]$ are non-empty. Since S_{j_1}, S_{j_2} and S_{j_3} are pairwise vertex disjoint, subpaths $P[X_1], P[X_2], P[X_3]$ are also pairwise vertex disjoint.

Without loss of generality, assume that subpaths $P[X_1], P[X_2], P[X_3]$ are ordered in this order, not necessarily consecutively, in P. Since P is a feasible solution path, $P[S_i]$ is a connected path, which contains subpaths $P[X_1], P[X_2]$ and $P[X_3]$, in this order. Since $X_2 \neq \emptyset$, there exists a vertex $v \in X_2$ which appears between $P[X_1]$ and $P[X_3]$ in P. Since $S_{j_2} \not\subseteq S_i$, there exists a vertex $u \in S_{j_2} \setminus S_i$, and therefore $u \notin P[S_i]$. In this case, all vertices of X_1 or all vertices of X_3 appear between v and u in P. However, since $X_1 \cap S_{j_2} = \emptyset$ and $X_3 \cap S_{j_2} = \emptyset$, there exists at least one vertex not contained in S_{j_2}, which appears between v and u. Therefore, $P[S_{j_2}]$ is not a connected path which contradicts the assumption that P is a feasible solution path. ∎

Theorem 7.4. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph. If its intersection graph is a star with paths centered at node s^*, corresponding to cluster $S^* \in \mathcal{S}$, then H has no feasible solution path.*

Proof: Let $s_{i_1}, s_{i_2}, s_{i_3}$ be three different neighbor nodes of s^* in $G_{int}(\mathcal{S})$ and let $S_{i_1}, S_{i_2}, S_{i_3}$ be the corresponding clusters. According to Definition 7.1:

- Since $s_{i_1}, s_{i_2}, s_{i_3}$ are neighbor nodes of s^*, $S_{i_1} \cap S^* \neq \emptyset$, $S_{i_2} \cap S^* \neq \emptyset$ and $S_{i_3} \cap S^* \neq \emptyset$.
- If s_{i_1} has another neighbor node $s' \neq s^*$ then $S_{i_1} \not\subseteq S^*$, otherwise s' is also a neighbor node of s^*, contradicting the structure of a star with paths. If s^* is the only neighbor node of s_{i_1}, then according to Remark 4.7, there are no contained clusters in H and $S_{i_1} \not\subseteq S^*$. Thus, in any case, $S_{i_1} \not\subseteq S^*$. A similar argument also proves that $S_{i_2} \not\subseteq S^*$ and $S_{i_3} \not\subseteq S^*$.
- $G_{int}(\mathcal{S})$ does not contain any edge connecting between two of the nodes $s_{i_1}, s_{i_2}, s_{i_3}$ and therefore S_{i_1}, S_{i_2} and S_{i_3} are pairwise vertex disjoint.

According to Theorem 7.3, H has no feasible solution path. ∎

Intuitively, let P be a path in a star with paths centered at node s^*, which corresponds to cluster S^*. Let r be the first node of P, and let R be the corresponding cluster. Thus, r and s^* are neighbor nodes in the intersection graph. There are two ways to disconnect P from s^*. One option is to remove $R \cap S^*$ from R, thereby removing the edge between r and s^* from the intersection graph. The other option is to remove $R \setminus S^*$ from R, and thus R becomes contained in S^*. In this case the rest of path P is disconnected from s^*. Set $minIntDif(R, S)$ chooses the minimum cardinality of these two options.

Notation 7.5. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph is a star with paths centered at node s^*, which corresponds to a cluster $S^* \in \mathcal{S}$. Let $\boldsymbol{R_1, \ldots, R_k}$ be the clusters corresponding to the neighbors of s^* in the intersection graph. Equivalently, they are the clusters that correspond to the nodes nearest to the center in the paths of the star.*

The indexing of the clusters is chosen so that the values of $|minIntDif(R_j, S^*)|$ are nondecreasing, with ties broken arbitrarily:

$$|minIntDif(R_1, S^*)| \leq |minIntDif(R_2, S^*)| \leq \cdots \leq |minIntDif(R_k, S^*)|.$$

By Theorem 7.4 a hypergraph whose intersection graph contains a star with paths centered at s^* with $k \geq 3$ branches has no feasible solution path. Therefore, to gain feasibility, at least $k-2$ paths must be disconnected from s^*, leaving at most two paths attached to the center. The removal list $StarRL(H, S^*)$ defined below, disconnects the $k-2$ cheapest paths and leaves the two most expensive paths unchanged.

Definition 7.6. *Define*

$$\boldsymbol{StarRL(H, S^*)} = \{(minIntDif(R_1, S^*), R_1), \ldots, (minIntDif(R_{k-2}, S^*), R_{k-2})\}.$$

This is a removal list that does not remove any vertices from S^*.

Theorem 7.7. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph $G_{int}(\mathcal{S})$ is a star with paths centered at node s^*, corresponding to a cluster $S^* \in \mathcal{S}$. The removal list $StarRL(H, S^*)$ is a minimum cardinality feasible removal list of H.*

Proof: For simplicity of reading, we omit H and S^* from $StarRL(H, S^*)$. Let $P_1, \ldots, P_k$ be the simple paths obtained in $G_{int}(\mathcal{S})$ after removing s^*. The paths are indexed so that $r_j \in P_j$ for $j \in \{1, \ldots, k\}$, where node r_i corresponds to cluster R_i. In addition, let $\mathcal{S}_j$ be the set of clusters corresponding to the nodes of path P_j.

Feasibility of *StarRL*:

First, we prove that *StarRL* is a feasible removal list. In $H \setminus StarRL$, for every cluster in R_j for $j \in \{1, \ldots, k-2\}$, exactly one of the following options is correct:

- All vertices of $R_j \cap S^*$ are removed from R_j and therefore $G_{int}(\mathcal{S} \setminus StarRL)$ does not contain edge (r_j, s^*).

- All vertices of $R_j \setminus S^*$ are removed from R_j. In this case, R_j becomes a contained cluster in S^*. In addition, if r_j has a neighbor node $r' \neq s^*$ in $G_{int}(\mathcal{S})$ whose corresponding cluster is R', then all the vertices of $R_j \cap R'$ are contained in $R_j \setminus S^*$ and therefore are removed from R_j. Thus, $G_{int}(\mathcal{S} \setminus StarRL)$ does not contain edge (r_j, r').

After considering all the clusters $R_1, \ldots, R_{k-2}$, for every $j \in \{1, \ldots, k-2\}$, if $minIntDif(R_j, S^*) = R_j \setminus S^*$, then R_j becomes a contained cluster in S^* in $H \setminus StarRL$. In this case, remove node r_j from path P_j and remove cluster R_j from $\mathcal{S}_j$. Remove from $H \setminus StarRL$ all the contained clusters and let H_{RL} be the created hypergraph and $\mathcal{S}_{RL}$ the corresponding clusters set. In $G_{int}(\mathcal{S}_{RL})$, s^* remains with degree exactly 2. Moreover, $G_{int}(\mathcal{S}_{RL})$ is a disjoint union of simple paths created from paths $P_1, \ldots, P_{k-2}$ and the path created by the concatenation of P_{k-1}, s^*, P_k.

Since P_j is a simple path, for every $j \in \{1, \ldots, k-2\}$, according to Theorem 5.1, $H[\mathcal{S}_j]$ has a feasible solution path. Similarly, since the concatenation of P_{k-1}, s^*, P_k is a simple path, $H[\mathcal{S}_{k-1} \cup \{S^*\} \cup \mathcal{S}_k]$ has a feasible solution path. According to Theorem 3.5, H_{RL} has a feasible solution path. Finally, according to Corollary 4.6, $H \setminus StarRL$ has a feasible solution path, thus proving that *StarRL* is a feasible removal list.

Minimality of *StarRL*:

To prove minimality, suppose by contradiction that there exists a feasible removal list RL' with $|RL'| < |StarRL|$. For each path adjacent to s^*, eliminating the obstruction associated with that path requires either disconnecting the path from s^* or making its first cluster contained in S^*. If the node closest to s^* in the path is r, and R is the cluster corresponding to r, then the minimum cost of achieving this is $|minIntDif(R, S^*)|$. By the way *StarRL* is constructed it has minimum cardinality among all removal lists that handle $k-2$ paths. Hence, since $|RL'| < |StarRL|$, the removal list RL' can handle at most $k-3$ paths. Therefore, in $G_{int}(\mathcal{S} \setminus RL')$ node s^* has at least three different neighbor nodes whose corresponding clusters are not contained in S^*. It follows that $G_{int}(\mathcal{S} \setminus RL')$ contains a star with paths centered at s^*. By Theorem 7.4, the hypergraph induced on this star has no feasible solution path. Consequently, by Corollary 3.4, $H \setminus RL'$ has no feasible solution path, contradicting the assumption that RL' is a feasible removal list. ∎

Remark 7.8. *Note that $StarRL(H, S^*)$ is a minimum cardinality feasible removal list of H which does not remove any vertices of S^*. There may be other feasible removal lists of minimum cardinality that do remove vertices from S^*.*

Theorem 7.9. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph $G_{int}(\mathcal{S})$ is a star with paths centered at node s^*, corresponding to a cluster $S^* \in \mathcal{S}$. The time complexity of finding $StarRL(H, S^*)$, which is a minimum cardinality feasible removal list, is $O(mn)$, $n = |V|, m = |\mathcal{S}|$.*

Proof: Finding s^*, the center of the star, can be done by parsing V_D and searching for a node whose degree is greater than 2, in $O(m)$ time complexity.

The neighbor nodes of s^* can be found by processing the row of s^* in M_G, looking for cells with a positive value. This can be done in $O(m)$ time complexity.

Next, we calculate $minIntDif(R_j, S^*)$, for every cluster R_j, which corresponds to a neighbor node of s^*, by processing the columns of every S_j and S^* in M_H. For every R_j, this can be done in $O(n)$ time complexity. Since the number of neighbor nodes of S^* is bounded by m, calculating these values for all the relevant clusters can be done in $O(mn)$ time complexity.

Since $StarRL(H, S^*)$ removes $k-2$ paths while keeping two paths attached to s^*, it suffices to select the two paths whose corresponding values of $|minIntDif(R, S^*)|$ are largest and omit them from the removal list. The two largest values can be found in $O(m)$ time complexity.

Hence, the total time complexity is $O(mn)$. ∎

8 Caterpillar with Paths Intersection Graphs

This section considers hypergraphs whose intersection graphs are caterpillars with paths. Such graphs consist of a central path, together with several paths attached to nodes of the central path (see Definition 8.1 and Figure 4). Since a caterpillar with paths contains the special structure presented in Theorem 7.3, the corresponding hypergraph has no feasible solution path.

We present Algorithm RmCaterpillar (see Figure 6), which is a dynamic programming algorithm that finds a minimum cardinality feasible removal list for hypergraphs whose intersection graphs are caterpillars with paths. The algorithm scans the central path from left to right. At each node of the central path, the branches attached to that node form a local star with paths structure, and the algorithm uses the minimum removal lists for such stars, as developed in Section 7. The dynamic program keeps two types of partial solutions, depending on whether the edge to the next central node must remain. We first introduce the notation for the local branches, then define the dynamic programming states, present the recurrence, and prove correctness and running time.

8.1 Structure and Notation

Definition 8.1. *A graph $G = (U, E)$ is a* **caterpillar with paths** *if it is a tree, not a simple path, that contains a path called the central path, such that removing the central path (nodes and edges) yields a forest of simple paths.*

Note that this is a generalization of the traditional definition of a caterpillar tree, in which the paths touching the central path are all of length 1.

When considering a hypergraph whose intersection graph is a caterpillar with paths, we assume without loss of generality that, in the construction of the intersection graph, the nodes representing the clusters are ordered along the central path as $s_1, s_2, \ldots, s_k$, where node s_i corresponds to cluster S_i.

Notation 8.2. *Consider a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, with $\boldsymbol{s_1, \ldots, s_k}$ the nodes of the central path of $G_{int}(\mathcal{S})$.*

For $i \in \{1, \ldots, k\}$:

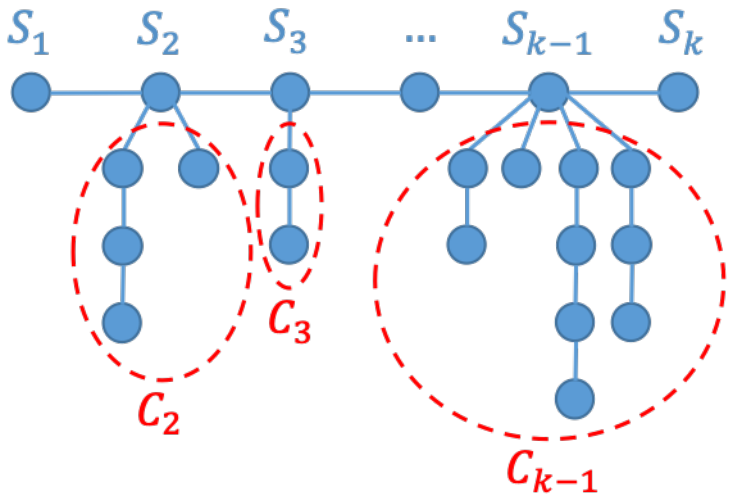


Figure 4: An example of a caterpillar with paths, see Definition 8.1

- *Let $\boldsymbol{C_i}$ be the set of clusters corresponding to the non-central nodes that belong to the paths attached to the central path through s_i. Thus, C_i consists of all non-central clusters in the branches incident to s_i.*
- *Let $\boldsymbol{R_{i,1}, \ldots, R_{i,k_i}}$ be the clusters in C_i, that intersect S_i. Equivalently, these are the clusters corresponding to the first nodes of the branches attached to s_i, one from each such branch. The clusters are indexed so that:*

$$|minIntDif(R_{i,1}, S_i)| \leq |minIntDif(R_{i,2}, S_i)| \leq \cdots \leq |minIntDif(R_{i,k_i}, S_i)|,$$

with ties broken arbitrarily.

8.2 Dynamic Programming States and Local Removal Lists

The algorithm we present uses dynamic programming to process the hypergraph, using the order of the nodes in the central path. Note that the algorithm processes each node in the central path, even if its degree in $G_{int}(\mathcal{S})$ is two. In this case $C_i = \emptyset$.

The algorithm aims to transform the caterpillar with paths into a disjoint union of simple paths. At each central node s_i, the key decision is how many incident branches may remain attached to s_i. By Theorem 7.3, three such incident paths create a condition that prevents feasibility. Hence, the algorithm ensures that at most two incident paths remain attached to each central node. The dynamic program records whether the edges to the previous and next central nodes are preserved, because these choices determine how many of the branches attached to s_i must be disconnected.

During the algorithm, for every $i \in \{1, \ldots, k\}$, we compose both L^- and L^+, see Definition 8.3.

Definition 8.3. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path. For every $i \in \{1, \ldots, k\}$ define the removal list $L^-(i)$ and for every $i \in \{1, \ldots, k-1\}$ define the removal list $L^+(i)$:*

- $\boldsymbol{L^-(i)}$ - *a minimum cardinality feasible removal list of* $H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)]$.

  Intuitively, $L^-(i)$ *is the best removal list for the hypergraph corresponding to the prefix of the central path ending at* S_i*, with no requirement that edge* (s_i, s_{i+1}) *remains in the intersection graph.*

- $\boldsymbol{L^+(i)}$ - *a minimum cardinality feasible removal list of* $H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j) \cup \{S_{i+1}\}]$*, subject to the following conditions:*

  1. $S_i \cap S_{i+1}$ *is not removed from either* S_i *or* S_{i+1}*,*
  2. *at least one vertex of* $S_i \setminus S_{i+1}$ *is not removed from* S_i
  3. *at least one vertex of* $S_{i+1} \setminus S_i$ *is not removed from* S_{i+1}*.*

  Thus, after applying this removal list, edge (s_i, s_{i+1}) *remains in the intersection graph.*

  Intuitively, $L^+(i)$ *is the best removal list for the hypergraph which corresponds to the prefix of the central path ending at* S_i *together with the next central cluster* S_{i+1}*, under the requirement that edge* (s_i, s_{i+1}) *remains in the intersection graph, available for connecting the prefix to the rest of the central path.*

The idea of the algorithm is that when processing S_i, for $i \in \{1, \ldots, k\}$, we consider whether edge (s_{i-1}, s_i) is removed from the intersection graph. When (s_{i-1}, s_i) is removed, after considering a few options which may create this removal, the algorithm uses $L^-(i-1)$ or $L^-(i-2)$ as the removal list used in the previous stages. When (s_{i-1}, s_i) is not removed, the algorithm uses $L^+(i-1)$ as the removal list used in the previous stage.

Composing the removal lists $L^-(i)$ and $L^+(i)$ in the algorithm, requires the removal lists defined in the following Definition (see also Table 1).

Definition 8.4. *Given a hypergraph* $H = \langle V, \mathcal{S} \rangle$ *whose intersection graph is a caterpillar with paths, where clusters* $S_1, \ldots, S_k$ *correspond to the nodes of the central path.*

In the notation below, $*$ *indicates that the local hypergraph is a star with paths centered at* s_i*;* B *indicates the central path edge* **B***efore* s_i*, namely* (s_{i-1}, s_i) *is preserved, and* A *indicates the central path edge* **A***fter* s_i*, namely* (s_i, s_{i+1}) *is preserved.*

For simplicity of notation, we define $S_0 = \emptyset$*. This convention is used to make the set expressions in the recurrence well-defined at the endpoint of the central path. In addition, at* s_1 *there is no edge* (s_0, s_1) *and at* s_k *there is no edge* (s_k, s_{k+1})*.*

For every $i \in \{1, \ldots, k\}$ *let:*

- $\boldsymbol{L^*(i)}$*= a minimum cardinality feasible removal list of the induced hypergraph* $H[\{S_i\} \cup C_i]$.

- $\boldsymbol{L^{*B}(i)}$ *= a minimum cardinality feasible removal list of the induced hypergraph* $H[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$*, under the condition that edge* (s_i, s_{i-1}) *is not removed from the intersection graph.*

- $\boldsymbol{L^{*A}(i)}$ *= a minimum cardinality feasible removal list of the induced hypergraph* $H[\{S_i\} \cup C_i \cup \{S_{i+1}\}]$*, under the condition that edge* (s_i, s_{i+1}) *is not removed from the intersection graph.*

- $\boldsymbol{L^{*BA}(i)}$*= a minimum cardinality feasible removal list of the induced hypergraph* $H[\{S_i\} \cup C_i \cup \{S_{i-1}\} \cup \{S_{i+1}\}]$*, under the condition that neither edge* (s_i, s_{i-1}) *nor edge* (s_i, s_{i+1}) *is removed from the intersection graph.*

List	Local hypergraph	Required preserved edge(s)
$L^*(i)$	$H[\{S_i\} \cup C_i]$	none
$L^{*B}(i)$	$H[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$	(s_{i-1}, s_i)
$L^{*A}(i)$	$H[\{S_i\} \cup C_i \cup \{S_{i+1}\}]$	(s_i, s_{i+1})
$L^{*BA}(i)$	$H[\{S_i\} \cup C_i \cup \{S_{i-1}\} \cup \{S_{i+1}\}]$	(s_{i-1}, s_i), (s_i, s_{i+1})

Table 1: The local removal lists used by Algorithm RmCaterpillar.

Definition 8.5. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path. For every $i \in \{1, \ldots, k\}$ and every integer t let:*

$$\boldsymbol{B_i(t)} = \begin{cases} \emptyset, & t \leq 0, \\ \{(minIntDif(R_{i,1}, S_i), R_{i,1}), \ldots, (minIntDif(R_{i,t}, S_i), R_{i,t})\}, & 1 \leq t \leq k_i. \end{cases}$$

Thus, $B_i(t)$ consists of the t cheapest removal lists that disconnect branches from the central node s_i. This removal list does not remove any vertex from S_i.

Property 8.6. *Since the clusters $R_{i,1}, \ldots, R_{i,k_i}$ are ordered by nondecreasing values of $|minIntDif(R_{i,j}, S_i)|$, the list $B_i(t)$ has the minimum cardinality among all removal lists that disconnect exactly t branches attached to s_i.*

Property 8.7. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path.*

The local removal lists defined in Definition 8.4 can be constructed as follows:

- *Since s_1 is the first node of the central path, there is no edge preceding s_1. Therefore, we define $L^{*B}(1) = L^*(1)$. By Theorem 7.7 $L^{*B}(1) = L^*(1) = B_1(k_1 - 2)$.*

  Similarly, we define $L^{*BA}(1) = L^{*A}(1)$ and an analogous reasoning gives: $L^{*BA}(1) = L^{*A}(1) = B_1(k_1 - 1)$.

- *For $i \in \{2, \ldots, k-1\}$, by Theorem 7.7:*

$$L^*(i) = B_i(k_i - 2).$$

- *For $i \in \{2, \ldots, k-1\}$, an analogous reasoning to the one presented in Theorem 7.7 can demonstrate that:*

$$L^{*B}(i) = B_i(k_i - 1).$$

- *For $i \in \{2, \ldots, k-1\}$, again by an analogous reasoning to the one presented in Theorem 7.7:*

$$L^{*A}(i) = B_i(k_i - 1).$$

- *For $i \in \{2, \ldots, k-1\}$, when constructing $L^{*BA}(i)$, the two central-path edges (s_i, s_{i-1}) and (s_i, s_{i+1}) are preserved. Hence, by Theorem 7.3, all other branches connected to s_i must be disconnected. Therefore:*

$$L^{*BA}(i) = B_i(k_i).$$

- *Since s_k is the last node in the central path, there is no edge following s_k. Therefore, we define $L^{*A}(k) = L^{*}(k) = B_k(k_k - 2)$ and $L^{*BA}(k) = L^{*B}(k) = B_k(k_k - 1)$.*

All the removal lists above are minimum cardinality feasible removal lists under their corresponding boundary conditions. Moreover, none of these lists removes any vertex from the central cluster S_i.

8.3 Algorithm RmCaterpillar

Before presenting the algorithm, we briefly describe its structure. Algorithm RmCaterpillar processes the central path from left to right. At each central cluster S_i, the branches attached to s_i form a local star with paths structure. The lists $L^{*}(i)$, $L^{*B}(i)$, $L^{*A}(i)$, and $L^{*BA}(i)$ describe the minimum removals lists required for this star, depending on whether the edge to the previous central node, the edge to the next central node, or both, must remain in the intersection graph.

The algorithm is a dynamic program that combines these local removal lists with the best removal lists already computed for the prefix of the central path. The list $L^{-}(i)$ represents the best solution for hypergraph which corresponds to the prefix of the central path ending at S_i, with no other requirement. The list $L^{+}(i)$ represents the best solution for the same hypergraph together with cluster S_{i+1}, under the requirement that edge (s_i, s_{i+1}) remains in the intersection graph. Therefore, each transition in the recurrence corresponds to one of the possible ways to handle the connection between S_{i-1} and S_i: either disconnect the two clusters, make one of the clusters contained in a neighboring central cluster, or keep the edge (s_{i-1}, s_i).

Remark 8.8. *In Algorithm RmCaterpillar (Figure 6) there are four options considered when calculating $L^{-}(i)$, for $i \in \{2, \ldots, k\}$:*

1. *Disconnect the central node s_{i-1} from the central node s_i. This is achieved by removing $S_{i-1} \cap S_i$ from S_i.*
2. *Make the cluster S_{i-1} contained in either S_{i-2} or S_i, thereby disconnecting the corresponding central node s_{i-1} from all its branches. This is achieved by removing either $S_{i-1} \setminus S_{i-2}$ or $S_{i-1} \setminus S_i$ from S_{i-1}.*
3. *Make the cluster S_i to become contained in S_{i-1}, thereby disconnecting the corresponding central node s_i from all its branches. This is achieved by removing $S_i \setminus S_{i-1}$ from S_i.*
4. *Keep edge (s_{i-1}, s_i) in the intersection graph, without removing any vertex from cluster S_i.*

See Figure 5 for all four options.

Remark 8.9. *There are three options considered when calculating $L^{+}(i)$, for $i \in \{2, \ldots, k-1\}$, in Algorithm RmCaterpillar (Figure 6):*

1. *Disconnect the central node S_{i-1} from the central node S_i. This is achieved by removing $S_{i-1} \cap S_i$ from S_i.*

2. *Make the cluster S_{i-1} contained in either S_{i-2} or S_i, thereby disconnecting the corresponding central node s_{i-1} from all its branches. This is achieved by removing either $S_{i-1} \setminus S_{i-2}$ or $S_{i-1} \setminus S_i$ from S_{i-1}.*

3. *Keep edge (s_{i-1}, s_i) in the intersection graph, without removing any vertex from cluster S_i.*

Note that, since edge (s_i, s_{i+1}) should remain in the intersection graph, at least one vertex $v \in S_i \cap S_{i+1}$ must not be removed from S_i. Since this vertex satisfies $v \notin S_i \cap S_{i-1}$, not all the vertices of $S_i \setminus S_{i-1}$ can be removed from S_i, and S_i cannot become a contained cluster in S_{i-1}.

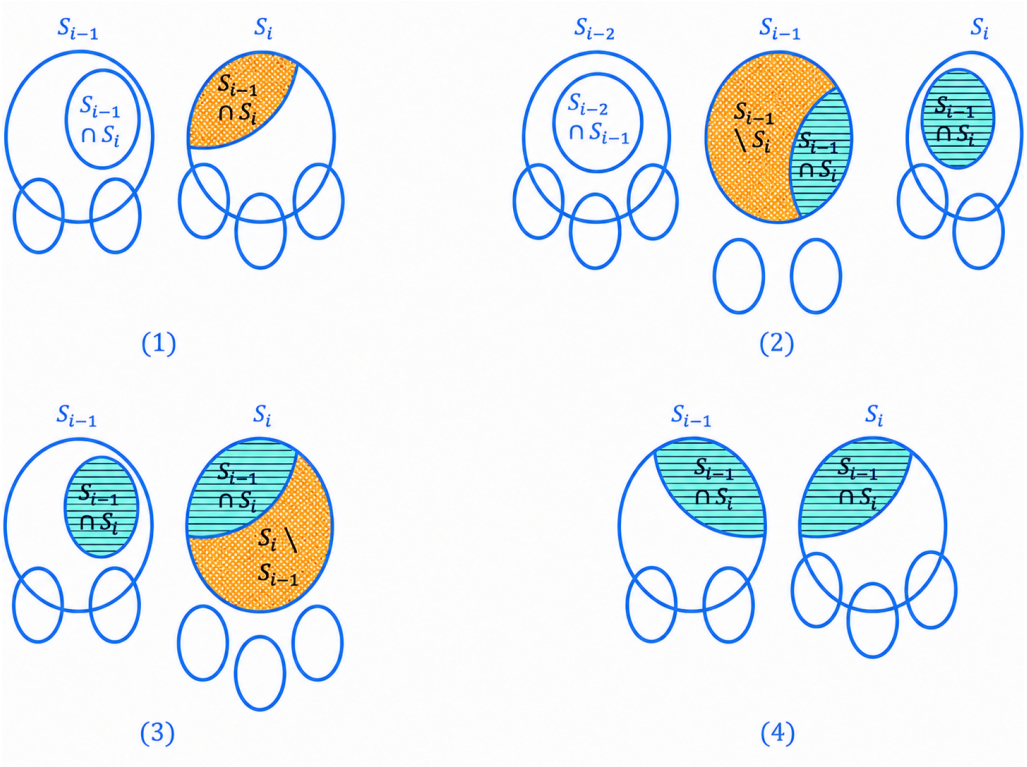


Figure 5: The four transition types used in the recurrence for $L^-(i)$ in Algorithm RmCaterpillar. The dotted (orange) parts are removed, and the striped (turqoise) parts stay as intersections. In (1) central node s_{i-1} is disconnected from s_i by removing $S_{i-1} \cap S_i$ from S_i. In (2) cluster S_{i-1} is made contained in one of its neighboring central clusters, thereby disconnecting s_{i-1} from all its branches. In (3) cluster S_i is made contained in S_{i-1}, thereby disconnecting s_i from all its branches. In (4) edge (s_{i-1}, s_i) is kept in the intersection graph.

8.4 Correctness of the Algorithm

The next lemma proves that the lists constructed by the recurrence are feasible, the following theorem then proves that, among all feasible removal lists satisfying the corresponding boundary

RmCaterpillar (An Algorithm to find a minimum cardinality feasible removal list of a hypergraph whose intersection graph is a caterpillar with paths)

input

hypergraph $H = \langle V, \mathcal{S} \rangle$ *whose intersection graph* $G_{int}(\mathcal{S})$ *is a caterpillar with paths.*

returns

A minimum cardinality feasible removal list L *of* H.

begin

Let $\boldsymbol{R_{i,1}, \ldots, R_{i,k_i}}$ *be the clusters that intersect* S_i *in* C_i,
ordered in increasing order of $|minIntDif(R, S_i)|$ *for* $R \in \{R_{i,1}, \ldots, R_{i,k_i}\}$.

$L^*(1) = L^{*B}(1) = B_1(k_1 - 2)$.

$L^{*A}(1) = L^{*BA}(1) = B_1(k_1 - 1)$.

for $(i \in \{2, \ldots, k-1\})$

$L^*(i) = B_i(k_i - 2)$.

$L^{*B}(i) = L^{*A}(i) = B_i(k_i - 1)$.

$L^{*BA}(i) = B_i(k_i)$.

end for

$L^*(k) = L^{*A}(k) = B_k(k_k - 2)$. $L^{*B}(k) = L^{*BA}(k) = B_k(k_k - 1)$.

$S_0 = \emptyset$, $L^-(0) = L^+(0) = \emptyset$.

$L^-(1) = L^*(1)$. $L^+(1) = L^{*A}(1)$.

for $(i \in \{2, \ldots, k-1\})$

$$L^-(i) = SetMin\{L^-(i-1) \cup (S_{i-1} \cap S_i, S_i) \cup L^*(i),$$
$$L^-(i-2) \cup (SetMin\{S_{i-1} \setminus S_{i-2}, S_{i-1} \setminus S_i\}, S_{i-1}) \cup L^*(i),$$
$$L^-(i-1) \cup (S_i \setminus S_{i-1}, S_i),$$
$$L^+(i-1) \cup L^{*B}(i)\}.$$

See Remark 8.8 for an explanation about the different options considered for $L^-(i)$.

$$L^+(i) = SetMin\{L^-(i-1) \cup (S_{i-1} \cap S_i, S_i) \cup L^{*A}(i),$$
$$L^-(i-2) \cup (SetMin\{S_{i-1} \setminus S_{i-2}, S_{i-1} \setminus S_i\}, S_{i-1}) \cup L^{*A}(i),$$
$$L^+(i-1) \cup L^{*BA}(i)\}.$$

See Remark 8.9 for an explanation about the different options considered for $L^+(i)$.

end for

$$L = L^-(k) = SetMin\{L^-(k-1) \cup (S_{k-1} \cap S_k, S_k) \cup L^*(k),$$
$$L^-(k-2) \cup (SetMin\{S_{k-1} \setminus S_{k-2}, S_{k-1} \setminus S_k\}, S_{k-1}) \cup L^*(k),$$
$$L^-(k-1) \cup (S_k \setminus S_{k-1}, S_k)\},$$
$$L^+(k-1) \cup L^{*B}(k)\}.$$

See Remark 8.8 for an explanation about the different options considered for $L^-(k)$.

return L

end *RmCaterpillar*

Figure 6: Algorithm RmCaterpillar

conditions, they have minimum cardinality. Together, these two results establish the correctness and optimality of Algorithm RmCaterpillar.

Lemma 8.10. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path. $L^-(i)$, for every $i \in \{1, \ldots, k\}$, and $L^+(i)$, for every $i \in \{1, \ldots, k-1\}$, calculated in Algorithm RmCaterpillar, satisfy:*

- *$L^-(i)$ is a feasible removal list of $H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)]$.*
- *$L^+(i)$ is a feasible removal list of $H[(\bigcup_{j=1}^{i}(\{S_j\} \cup C_j) \cup \{S_{i+1}\}]$, which does not remove $S_i \cap S_{i+1}$ from either S_i or S_{i+1} and where $S_{i+1} \setminus S_i$ and $S_i \setminus S_{i+1}$ each contains at least one vertex in $H \setminus L^+(i)$.*

Proof: The proof is by induction on i.

For $i = 1$, $H[\{S_1\} \cup C_1]$ is a hypergraph whose intersection graph is a star. According to Theorem 7.7 and Property 8.7, $L^-(1)$ and $L^+(1)$ are feasible removal lists.

Suppose the assumption of the lemma is correct for all $j < i$, and prove it for i.

We first prove that $L^-(i)$, calculated in the algorithm, is a feasible removal list of $H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)]$, by proving that each one of the four options considered in the algorithm creates a feasible removal list.

1. $L^-(i) = L^-(i-1) \cup (S_{i-1} \cap S_i, S_i) \cup L^*(i)$. In this case, $L^-(i)$ contains $(S_{i-1} \cap S_i, S_i)$. Thus, edge (s_{i-1}, s_i) is deleted and $G_{int}(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j))$ is disconnected from $G_{int}(\{S_i\} \cup C_i)$. According to the induction hypothesis, $L^-(i-1)$ is a feasible removal list of $H[\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)]$. According to Theorem 7.7, $L^*(i)$ is a feasible removal list of $H[\{S_i\} \cup C_i]$. Thus, according to Theorem 3.7, $L^-(i)$ satisfies the claim of the lemma. See Figure 5 (1).

2. $L^-(i) = L^-(i-2) \cup (SetMin\{S_{i-1} \setminus S_{i-2}, S_{i-1} \setminus S_i\}, S_{i-1}) \cup L^*(i)$. In this case, all the vertices in the intersections of S_{i-1} with all its neighbors in C_{i-1} are removed from S_{i-1}, such that S_{i-1} becomes a contained cluster in either S_{i-2} or S_i. The intersection graph of $H[\{S_{i-1}\} \cup C_{i-1}] \setminus L^-(i)[\{S_{i-1}\} \cup C_{i-1}]$ is a collection of simple paths. According to Theorems 5.1 and 3.5, $H[\{S_{i-1}\} \cup C_{i-1}] \setminus L^-(i)[\{S_{i-1}\} \cup C_{i-1}]$ has a feasible solution path and therefore an empty feasible removal list. According to the induction hypothesis, $L^-(i-2)$ is a feasible removal list of $H[\bigcup_{j=1}^{i-2}(\{S_j\} \cup C_j)]$. According to Theorem 7.7, $L^*(i)$ is a feasible removal list of $H[\{S_i\} \cup C_i]$. Thus, according to Theorem 3.7, $L^-(i)$ satisfies the claim of the lemma. See Figure 5 (2) for the case of removing $S_{i-1} \setminus S_i$ from S_{i-1}.

3. $L^-(i) = L^-(i-1) \cup (S_i \setminus S_{i-1}, S_i)$. In this case, all the vertices in the intersections of S_i and all its neighbors in C_i are removed from S_i, thus S_i becomes a contained cluster in S_{i-1}. The intersection graph of $H[\{S_i\} \cup C_i] \setminus L^-(i)[\{S_i\} \cup C_i]$ is a collection of simple paths. According to Theorems 5.1 and 3.5, $H[\{S_i\} \cup C_i] \setminus L^-(i)[\{S_i\} \cup C_i]$ has a feasible solution path and therefore an empty feasible removal list. According to the induction hypothesis, $L^-(i-1)$ is a feasible removal list of $H[\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)]$. Thus, according to Theorem 3.7, $L^-(i)$ satisfies the claim of the lemma. Note that in this case, since we removed all the vertices of $S_i \setminus S_{i-1}$ from S_i, all the vertices of $S_i \cap S_{i+1}$ are also removed from S_i. See Figure 5 (3).

4. $L^-(i) = L^+(i-1) \cup L^{*B}(i)$. According to the induction hypothesis, $L^+(i-1)$ is a feasible removal list of $H[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}]$. According to Property 8.7, $L^{*B}(i)$ is a feasible

removal list of $H[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$. None of these two lists remove a vertex from $S_{i-1} \cap S_i$, either from S_{i-1} or from S_i. Thus, the corresponding feasible solution paths can agree on the order of $S_{i-1} \cap S_i$ at the end of the paths. Hence, the feasible solution paths can be combined into one solution path, and the union of $L^+(i-1)$ and $L^{*B}(i)$ satisfies the claim of the lemma. See Figure 5 (4).

The proof of feasibility for $L^+(i)$ is similar to the proof of $L^-(i)$, where:

1. The proof for $L^+(i) = L^-(i-1) \cup (S_{i-1} \cap S_i, S_i) \cup L^{*A}(i)$ is similar to case (1).
2. The proof for $L^+(i) = L^-(i-2) \cup (SetMin\{S_{i-1} \setminus S_{i-2}, S_{i-1} \setminus S_i\}, S_{i-1}) \cup L^{*A}(i)$ is similar to case (2).
3. The proof for $L^+(i) = L^+(i-1) \cup L^{*BA}(i)$ is similar to case (4).

Note that, for $L^+(i)$ we do not consider the removal $S_i \setminus S_{i-1}$ from S_i, as in $L^+(i)$ the vertices of $S_i \cap S_{i+1} \subseteq S_i \setminus S_{i-1}$ are not removed from S_i. ∎

Theorem 8.11. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path. $L^-(i)$, for every $i \in \{1, \ldots, k\}$, and $L^+(i)$, for every $i \in \{1, \ldots, k-1\}$, calculated in Algorithm RmCaterpillar, satisfy:*

- *$L^-(i)$ is a minimum cardinality feasible removal list of $H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)]$.*
- *$L^+(i)$ is a minimum cardinality feasible removal list of $H[(\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)) \cup \{S_{i+1}\}]$, which does not remove $S_i \cap S_{i+1}$ from either S_i or S_{i+1}, and where $S_{i+1} \setminus S_i$ and $S_i \setminus S_{i+1}$ each contains at least one vertex in $H \setminus L^+(i)$.*

Proof: According to Lemma 8.10, $L^-(i)$ and $L^+(i)$ are feasible removal lists. We will prove by induction on i that they are minimum cardinality feasible removal lists.

For $i = 1$, $H[\{S_1\} \cup C_1]$ is a hypergraph whose intersection graph is a star. According to Theorem 7.7 and Property 8.7, $L^-(1)$ and $L^+(1)$ are minimum cardinality feasible removal lists.

Suppose the assumption of the lemma is correct for all $j < i$, and prove it for i.

Suppose by contradiction that L' is a feasible removal list of $H' = H[\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)]$ with $|L'| < |L^-(i)|$. Note that, H' does not contain clusters $S_{i+1}, \ldots, S_k$. However, the vertices of $S_{i+1} \cap S_i$ are contained in S_i, and according to Lemma 3.8, these vertices have to appear consecutively in any feasible solution path, so they can be considered as a contained cluster in S_i. Thus, according to Corollary 4.6, we can ignore them when constructing any feasible removal list of H'.

Suppose by contradiction that $(S_{i-1} \setminus S_i, S_{i-1}) \subseteq L'$. In this case, S_{i-1} becomes a contained cluster in S_i. According to Corollary 4.6, we can ignore S_{i-1} in constructing a feasible removal list of $H' \setminus (S_{i-1} \setminus S_i, S_{i-1})$. The intersection graph $G_{int}((\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)) \setminus L')$ is composed from the following connected components: $G_{int}(\bigcup_{j=1}^{i-2}(\{S_j\} \cup C_j))$, the clusters in C_{i-1} which form a collection of simple paths, and $G_{int}(\{S_i\} \cup C_i)$, where the latter intersection graph is a star. According to Theorem 5.1, the simple paths of C_{i-1} have feasible solution paths, and no additional removals are required. According to Lemma 3.6, $L'[\bigcup_{j=1}^{i-2}(\{S_j\} \cup C_j)]$ is a feasible removal list for $H'[\bigcup_{j=1}^{i-2}(\{S_j\} \cup C_j)]$. Similarly, $L'[\{S_i\} \cup C_i]$ is a feasible removal list of $H'[\{S_i\} \cup C_i]$. Obviously, $L'[\bigcup_{j=1}^{i-2}(\{S_j\} \cup C_j)]$ and $L'[\{S_i\} \cup C_i]$ are disjoint. Using the induction hypothesis and Theorem 7.7, $|L'| \geq |(S_{i-1} \setminus S_i, S_{i-1})| + |L^-(i-2)| + |L^*(i)|$, contradicting the assumption that $|L'| < |L^-(i)|$.

A similar argument proves that L' contains neither $(S_{i-1} \setminus S_{i-2}, S_{i-1})$ nor $(S_i \setminus S_{i-1}, S_i)$.

Suppose by contradiction that $(S_i \cap S_{i-1}, S_i) \subseteq L'$. In this case, the intersection graph $G_{int}((\bigcup_{j=1}^{i}(\{S_j\} \cup C_j)) \setminus L')$ is composed from two connected components: $G_{int}(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j))$ and $G_{int}(\{S_i \setminus (S_i \cap S_{i-1})\} \cup C_i)$, where the latter intersection graph is a star. According to Lemma 3.6, $L'[\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)]$ is a feasible removal list of $H'[\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)]$. Similarly, $L'[\{S_i \setminus (S_i \cap S_{i-1})\} \cup C_i]$ is a feasible removal list of $H'[\{S_i\} \cup C_i]$. Obviously, $L'[\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)]$ and $L'[\{S_i \setminus (S_i \cap S_{i-1})\} \cup C_i]$ are disjoint. Using the induction hypothesis and Theorem 7.7, $|L'| \geq |(S_i \cap S_{i-1}, S_i)| + |L^-(i-1)| + |L^*(i)|$, contradicting the assumption that $|L'| < |L^-(i)|$.

A similar argument proves that L' does not contain $(S_i \cap S_{i-1}, S_{i-1})$.

Hence, L' does not remove $S_i \cap S_{i-1}$ from either S_{i-1} or S_i, at least one vertex of $S_{i-1} \setminus S_i$ is not removed from S_{i-1} and at least one vertex of $S_i \setminus S_{i-1}$ is not removed from S_i.

Consider $L'_1 = L'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}]$ and $L'_2 = L'[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$. By Lemma 3.6, L'_1 is a feasible removal list of $H'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}]$. Let P' be a feasible solution path of $H'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}] \setminus L'_1$ and let $U_i = \{u \mid u \in S_i \setminus S_{i-1}, (u, S_i) \in L'_1\}$ be the set of vertices from $S_i \setminus S_{i-1}$ removed from S_i by L'_1. Since at least one vertex $v_i \in S_i \setminus S_{i-1}$ is not removed from S_i by L', and since in $H'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}]$ the vertices of $S_i \setminus S_{i-1}$ are contained only in S_i, we may replace v_i in P' by a path spanning $\{v_i\} \cup U_i$. This produces a feasible solution path for $H'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}] \setminus (L'_1 \setminus (U_i, S_i))$. Hence, $\widehat{L'_1} = L'_1 \setminus (S_i \setminus S_{i-1}, S_i)$ is a feasible removal list of $H'[(\bigcup_{j=1}^{i-1}(\{S_j\} \cup C_j)) \cup \{S_i\}]$. In addition, $\widehat{L'_1}$ satisfies all the conditions in the definition of $L^+(i-1)$, and according to the induction hypothesis, $|\widehat{L'_1}| \geq |L^+(i-1)|$. Similarly, L'_2 is a feasible removal list of $H'[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$. Since at least one vertex from $S_{i-1} \setminus S_i$ is not removed from S_{i-1} in L', and since in $H'[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$ the vertices of $S_{i-1} \setminus S_i$ are contained only in S_{i-1}, $\widehat{L'_2} = L'_2 \setminus (S_{i-1} \setminus S_i, S_{i-1})$ is a feasible removal list of $H'[\{S_i\} \cup C_i \cup \{S_{i-1}\}]$. Furthermore, since L' does not remove $S_i \cap S_{i-1}$ from either S_{i-1} or S_i, neither does $\widehat{L'_2}$, and therefore $|\widehat{L'_2}| \geq |L^{*B}(i)|$. Since $\widehat{L'_1}$ and $\widehat{L'_2}$ are contained in L', $L' \supseteq \widehat{L'_1} \cup \widehat{L'_2}$. Since L' does not remove $S_i \cap S_{i-1}$ from either S_{i-1} or S_i, $\widehat{L'_1}$ and $\widehat{L'_2}$ are disjoint and therefore:

$$|L'| \geq |\widehat{L'_1}| + |\widehat{L'_2}| \geq |L^+(i-1)| + |L^{*B}(i)|,$$

contradicting the assumption that $|L'| < |L^-(i)|$.

A similar argument proves that $L^+(i)$ is a minimum cardinality feasible removal list which satisfies the conditions of the theorem. ∎

Corollary 8.12. *Given a hypergraph $H = \langle V, \mathcal{S} \rangle$ whose intersection graph is a caterpillar with paths, where clusters $S_1, \ldots, S_k$ correspond to the nodes of the central path. L returned by Algorithm RmCaterpillar is a minimum cardinality feasible removal list of H.*

Proof: Follows from Theorem 8.11 and since $L \equiv L^-(k)$ produced by the algorithm. ∎

8.5 Time Complexity

Theorem 8.13. *Let $H = \langle V, \mathcal{S} \rangle$ be a hypergraph whose intersection graph $G_{int}(\mathcal{S})$ is a caterpillar with paths. The time complexity of Algorithm RmCaterpillar (Figure 6), which finds a minimum cardinality feasible removal list, is $O(n(k+b))$, where $n = |V|$, k is the number of nodes in the central path and b is the total number of branches incident to the central path,*

Proof: The first stage of the algorithm calculates each one of the lists in $L^*(i), L^{*B}(i), L^{*A}(i)$ and $L^{*BA}(i)$, for every $i \in \{1, \ldots, k\}$. As a preprocessing step, we first calculate the cardinality of intersections and differences of subsets for all the clusters representing neighbor nodes of the central path clusters. For central node s_i, Calculating each individual intersection and difference takes $O(n)$, doing it for all the branches of s_i takes $O(nb_i)$ time complexity, where b_i is the number of branches attached to s_i. In addition, this process considers each cluster in the central path twice, as a neighbor of other clusters in the central path, and each branch of a central node exactly once. Thus, the total time complexity of this calculation is bounded by $O(n(k+b))$.

The subsequent task involves performing the actual calculation of the lists. For these calculations, the algorithm needs to find, for each cluster corresponding to a node in the central path, the two neighbors with the highest value of *minIntDif*. Since each list may contain at most n pairs, for every cluster in the central path, $O(n)$ time complexity is required. Hence, the total time complexity of calculating the lists is $O(nk)$.

At the next step of the algorithm, lists L^- and L^+ are calculated. Since each list may include at most n pairs, calculating each list requires $O(n)$ time complexity and the total time for all the lists is $O(nk)$ time complexity.

The last calculation of L requires $O(n)$ time complexity.

Hence, the total time complexity is $O(n(k+b))$, when b is the total number of branches touching all central nodes. ∎

9 Summary and Future Research

In this paper we consider the feasibility question of the Clustered Traveling Salesman Problem. We focus on hypergraphs with no feasible solution path and consider removing vertices from clusters, such that the hypergraph with the updated clusters has a feasible solution path. For special structures of the intersection graph, we offer minimum cardinality feasible removal lists, to gain feasibility.

Further research may consider other structures of the intersection graph. Alternatively, one can explore different target functions, such as minimizing the number of clusters that need to be modified.

References

[1] S. Anily, J. Bramel, and A. Hertz. A $\frac{5}{3}$-approximation algorithm for the clustered traveling salesman tour and path problems. *Operations Research Letters*, 24(1–2):29–35, 1999. doi:10.1016/S0167-6377(98)00046-7.

[2] E. M. Arkin, R. Hassin, and L. Klein. Restricted delivery problems on a network. *Networks*, 29(4):205–216, 1997. doi:10.1002/(SICI)1097-0037(199707)29:4<205::AID-NET3>3.0.CO;2-J.

[3] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. Working Papers presented at the ACM-SIGACT Symposium on the Theory of Computing (Albuquerque, N. M., 1975). doi:10.1016/S0022-0000(76)80045-1.

[4] J. A. Chisman. The clustered traveling salesman problem. *Computers & Operations Research*, 2(2):115–119, 1975. doi:10.1016/0305-0548(75)90015-5.

[5] T. Christof, M. Jünger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping of chromosomes by unique end-probes. *Journal of Computational Biology*, 4(4):433–447, 1997. doi:10.1089/cmb.1997.4.433.

[6] 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, Pittsburgh, PA, 1976. Reprinted in *Operations Research Forum*, 3(1):20, 2022. doi:10.1007/s43069-021-00101-z.

[7] P. Duchet. Propriété de Helly et problèmes de représentation. In *Problèmes Combinatoires et Théorie des Graphes (Colloque International CNRS, Orsay, 1976)*, volume 260 of *Colloques Internationaux du CNRS*, pages 117–118, Paris, 1978. CNRS.

[8] C. Flament. Hypergraphes arborés. *Discrete Mathematics*, 21(3):223–227, 1978. doi:10.1016/0012-365X(78)90154-1.

[9] 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. doi:10.1007/s004530010045.

[10] N. Guttmann-Beck, H. Meshita-Sayag, and M. Stern. Achieving feasibility for clustered traveling salesman problems using PQ-trees. *Networks*, 82(2):153–166, 2023. doi:10.1002/net.22164.

[11] N. Guttmann-Beck and M. Stern. Decomposing the feasibility of Clustered Spanning Tree by Paths. *Discrete Applied Mathematics*, 354:160–180, 2024. 18th Cologne-Twente Workshop on Graphs and Combinatorial Optimization (CTW 2020). doi:10.1016/j.dam.2022.11.009.

[12] F. C. J. Lokin. Procedures for travelling salesman problems with additional constraints. *European Journal of Operational Research*, 3(2):135–141, 1979. doi:10.1016/0377-2217(79)90099-7.

[13] 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. doi:10.1137/1.9780898719802.

[14] J. Y. Potvin and F. Guertin. A genetic algorithm for the clustered traveling salesman problem with a prespecified order on the clusters. In D. L. Woodruff, editor, *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search*, volume 9 of *Operations Research/Computer Science Interfaces Series*, pages 287–299. Kluwer Academic Publishers, Boston, MA, 1998. doi:10.1007/978-1-4757-2807-1_11.

[15] P. J. Slater. A characterization of soft hypergraphs. *Canadian Mathematical Bulletin*, 21(3):335–337, 1978. doi:10.4153/CMB-1978-058-5.