Clustering in Discrete Path Planning for Approximating Minimum Length Paths

Frank Imeson   Stephen L. Smith

Abstract— In this paper we consider discrete robot path planning problems on metric graphs. We propose a clustering method, Γ-Clustering for the planning graph that significantly reduces the number of feasible solutions, yet retains a solution within a constant factor of the optimal. By increasing the input parameter Γ, the constant factor can be decreased, but with less reduction in the search space. We provide a simple polynomial-time algorithm for finding optimal Γ-Clusters, and show that for a given Γ, this optimal is unique. We demonstrate the effectiveness of the clustering method on traveling salesman instances, showing that for many instances we obtain significant reductions in computation time with little to no reduction in solution quality.

I. INTRODUCTION

Discrete path planning is at the root of many robotic applications, from surveillance and monitoring for security, to pickup and delivery problems in automated warehouses. In such problems the environment is described as a graph, and the goal is to find a path in the graph that completes the task and minimizes the cost function. For example, in monitoring, a common problem is to compute a tour of the graph that visits all vertices and has minimum length [1]. These discrete planning problems are typically **NP**-hard [1], [2], and thus there is a fundamental trade off between solution quality and computation time. In this paper we propose a graph clustering method, called Γ-Clustering, that can be used to reduce the space of feasible solutions considered during the optimization. The parameter Γ serves to trade-off the feasible solution space reduction (and typically computation time) with the quality of the resulting solution.

The idea behind Γ-Clustering is to group vertices together that are in close proximity to each other but are also far from all other vertices. Figure 1 shows an example of Γ-Clustering in an office environment. Given this clustering, we solve the path planning problem by restricting the path to visit vertices within each cluster consecutively (i.e., no path can visit any cluster more than once). This restriction reduces the number of possible solutions exponentially and thus reduces the amount of computational time needed to find good solutions.

Unlike other clustering methods, Γ-Clustering does not accept as input a desired number of clusters. This means that some instances will have no clusters, while others will have many. In this way, Γ-Clusters only explore natural structures within the problem instances instead of imposing structures onto the instance. Additionally, when the graph is metric,
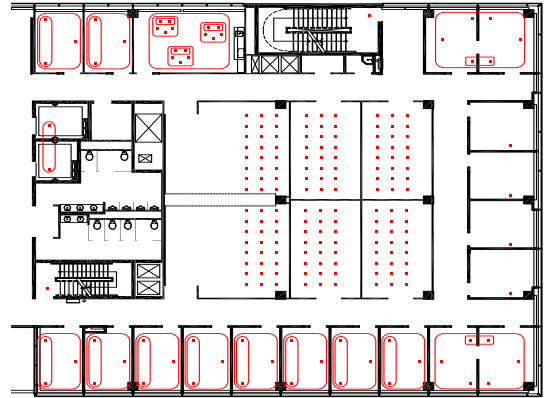
Fig. 1: The results of Γ-Clustering used on an office environment. The red dots represent locations of interest and the red rectangles show clusters of size two or greater.

we establish that for a given Γ-Clustering, the optimal path of the clustered planning problem is within a constant factor (dependent on Γ) of the true optimal solution.

Related work: There are a number of clustering methods for Euclidean [3], [4] and discrete [5], [6], [7] environments. Typically the objective of these algorithms is to find a set of equal (or roughly equal) non-overlapping clusters that are grouped by similarity (close in proximity, little to no outgoing edges, etc ...), where each location in the graph is assigned to one cluster. For these methods, the desired number of clusters is given as an input parameter. In contrast, in Γ-Clustering, the idea is to simply find a specific form of clustering within the environment, if it exists. These Γ-Clusters may be nested within one another.

There are other clustering methods that also look for specific structures within the graph such as community structures [8], which is based on a metric that captures the density of links within communities to that between communities. In contrast, our clustering method is specifically designed to find structures that yield desirable properties for path planning on road maps.

The use of clustering to save on processing/time is done in a variety of different fields such as data mining [9], parallel computer processing [10], image processing [11], and control [12] for path planning. In environments that have regions with a high degree of connectivity, such as electronic circuits, clustering is commonly used to identify these regions and then plan (nearly) independently in each cluster [13]. For path planning problems with repetitive tasks, one can cluster a set of popular robot action sequences into macros [14], [15], allowing the solver to quickly discover solutions that benefit from these action sequences. In applications such as sensor sweeping for coverage problems [16] or in the routing

of multiple agents [17], clustering has been used to partition the environment into regions that can again be treated in a nearly independent manner, reducing computation time.

There is some prior work on partitioning in discrete path planning. Multilevel refinement [18] is the process of recursively coarsening the graph by aggregating the vertices together to create smaller and smaller instances, for which a plan can be found more easily. The plan is then recursively refined to obtain a solution to the original problem. The idea in coarsening a graph is that the new coarse edges should approximate the transition costs in the original graph. This differs from Γ-Clustering, which preserves the edges within the graph. There are a number of clustering approaches that aim to reduced the complexity of Euclidean and or planar TSP problems [19], [20]. Γ-Clustering is more general, in that it works on any graph, while the solution quality guarantees only hold for metric graphs.

Contributions: The main contribution of this paper is the introduction of Γ-Clustering, a clustering method for a class of discrete path planning problems. We establish that the solution to the corresponding clustered problem provides a $\min\left(2, 1 + \frac{3}{2\Gamma}\right)$-factor approximation to the optimal solution. We give some insight into the reduction of the search space as a function of the amount of clustering, and we provide an efficient algorithm for computing the optimal Γ-Clustering. We then use an integer programming formulation of the TSP problem to demonstrate that for many problem instances the clustering method reduces the computation time while still finding near-optimal solutions.

Due to space constraints, we have omitted all proofs from this document. The omitted content can be found in our technical note [21].

II. PRELIMINARIES IN DISCRETE PATH PLANNING

In this section we define the class of problems considered in this paper, review some semantics of clusters, review the traveling salesman problem (TSP) [22] and define its clustered variant, the general clustered traveling salesman problem (CTSP*).

A. Discrete Path Planning

Given a graph $G = \langle V, E, w \rangle$, we define a *path* as a non-repeating sequence of vertices in V, connected by edges in E. A *cycle* is a path in which the first and last vertex are equal, and for simplicity we will also refer to cycles as paths. Let $\mathcal{P}$ represent the set of all possible paths in G. Then, abstractly, a path planning constraint defines a subset $\mathcal{P}_1 \subseteq \mathcal{P}$ of feasible paths. Given a set of constraints $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m$, the set of all feasible paths is $\mathcal{P}_1 \cap \mathcal{P}_2 \cap \cdots \cap \mathcal{P}_m$. In this paper we restrict our attention to the following class of constraints and planning problems.

Definition II.1 (Order-Free Constraints)**.** A constraint $\mathcal{P}_1$ is *order-free* if, given any $p \in \mathcal{P}_1$, then all paths obtained by permuting the vertices of p are also in $\mathcal{P}_1$.

Problem II.2 (Discrete Path Planning Problem)**.** Given a complete weighted graph $G = \langle V, E, w \rangle$ and a set of order-free constraints $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_m\}$, find the minimum length feasible path.

Many discrete path planning problems for single and multiple robots fall into this class, so long as they do not restrict the ordering of vertex visits (i.e., no constraints of the form "visit A before B"). Some examples include single and multi-robot traveling salesman problems, point-to-point planning, and patrolling. As a specific example the GTSP is a problem where a robot is required to visit exactly one location in each non-overlapping set of locations [23]. This is naturally expressed in the above framework by having one constraint for each set: for each cluster V_i we have a constraint stating that exactly one vertex in V_i must be visited in the path.

A *metric* discrete path planning problem is one where the edge weights in the graph G satisfy the triangle inequality: for $v_a, v_b, v_c \in V$, we have $w(v_a, v_c) \leq w(v_a, v_b) + w(v_b, v_c)$.

To describe the number of feasible paths for a given planning problem, we use the phrase *search space size*. For example, a problem where we must choose an ordering of n locations has a search space size of $n!$, since there are $n!$ combinations that a path may take. Note that as more constraints are added to the problem, the search space size can only be reduced, since a feasible path must lie in the intersection of all constraints.

B. Clusters

A *cluster* is a subset of the graph's vertices, $V_i \subset V$. Given the clusters V_1 and V_2, we say V_1 is *nested* in V_2 if $V_1 \subseteq V_2$. The clusters V_1 and V_2 are *overlapping* if $V_1 \cap V_2 \neq \emptyset$, $V_1 \not\subseteq V_2$, and $V_2 \not\subseteq V_1$. A *set of clusters* (or clustering) is denoted by $C = \{V_1, \ldots, V_m\}$. A clustering $C = \{V_1, V_2, \ldots, V_m\}$ is a nested if there exists some $V_i \subseteq V_j$ for $V_i, V_j \in C$.

In this paper we seek to add clustering constraints to a discrete planning problem that reduce its search space size, but also retains low-cost feasible paths. The clustering constraints we consider are of the following form.

Definition II.3 (Consecutive Visit Constraint)**.** Given a graph $G = \langle V, E, w \rangle$ and a cluster $V_i \subseteq V$, a feasible path p must visit the vertices within the cluster V_i *consecutively*. Formally, the vertices visited by p, $V[p]$ are visited consecutively if there exists a path segment p' of p that visits every vertex in $V_i \cap V[p]$ and is of length $|V_i \cap V[p]|$.

Note that in the above definition, it is not necessary for all of the vertices in V_i to be visited. It is just necessary to visit the vertices consecutively in V_i that are visited.

C. Traveling Salesman Problems

The traveling salesman problem (TSP) is defined as follows: given a set of cities and distances between each pair of cities, find the shortest path that the salesman can take to visit each city exactly once and return to the first city (i.e., the shortest tour). A tour in a graph that visits each vertex exactly once is called a Hamiltonian cycle (regardless of path cost). The general clustered version of TSP is the extension that

requires the solution to visit the vertices within the clusters consecutively. The definition of these problems is as follows:

Problem II.4 (Traveling Salesman Problem (TSP)). Given a complete graph $G = \langle V, E, w \rangle$ with edge weights $w : E \to \mathbb{R}_{\geq 0}$, find a Hamiltonian cycle of G with minimum cost.

Problem II.5 (General-CTSP). Given a complete and weighted graph $G = \langle V, E, w \rangle$ along with a clustering $C = \{V_1, \ldots, V_m\}$, find a Hamiltonian cycle of G with minimum cost such that the vertices within each cluster V_i are visited consecutively.

The traditional version of the CTSP restricts the clusters to be non-overlapping (and non-nested). For this paper we use the syntax CTSP* to emphasize when we are solving a General-CTSP problem CTSP to refer to the traditional problem.

III. Γ-CLUSTERING

In this section, we define Γ-Clustering and show that the Γ-Clustered path planning problem provides a $\min\left(2, 1 + \frac{3}{2\Gamma}\right)$ approximation of the original path planning problem. We then describe an algorithm for finding the optimal Γ-Clustering, and characterize the search space reductions.

A. Definition of Γ-Clustering

Below we define the notion of Γ-Clusters, Γ-Clusterings, and the clustered discrete path planning problem. Then we pose the clustering problem as one of maximizing the search space reduction.

Definition III.1 (Γ-Metric of a cluster). Given a graph $G = \langle V, E, w \rangle$ and a cluster $V_i \subset V$, we define the following quantities for V_i relative to G:

$$\alpha_i \equiv \min_{v_a \in V_i, v_b \in V \setminus V_i} (w(v_a, v_b), w(v_b, v_a))$$
$$\beta_i \equiv \max_{v_a, v_b \in V_i, v_a \neq v_b} w(v_a, v_b)$$
$$\Gamma_i \equiv \frac{\alpha_i}{\beta_i},$$

where α_i represents the minimum weight edge entering or exiting the cluster V_i, and β_i represents the maximum weight edge within V_i. The ratio Γ_i is a measure of how separated the vertices in V_i are from the remaining vertices in G.

Definition III.2 (Γ-Clustering). Given an input parameter $\Gamma \geq 0$ and a graph $G = \langle V, E, w \rangle$, a clustering $C = \{V_1, V_2, \ldots, V_m\}$ is said to be a Γ-Clustering if and only if V is covered by $V_1 \cup V_2 \cup \cdots \cup V_m$; each $V_i \in C$ has a separation $\Gamma_i \geq \Gamma$; and the clusters are either nested ($V_i \subseteq V_j$ or $V_j \subseteq V_i$ for all $V_i, V_j \in C$) or non-overlapping ($V_i \cap V_j = \emptyset$).

The search space reduction for path planning problems comes from restricting paths to visit the clusters consecutively and our goal is to maximize that reduction. Thus we are interested in the following two problems.

Definition III.3 (The Clustered Path Planning Problem). Given a discrete path planning problem P and a clustering C, the *clustered version of the problem P'* has the constraint

that the path must visit the vertices within each cluster consecutively, in addition to all the constraints of P.

Definition III.4 (The Clustering Problem). Given a graph $G = \langle V, E, w \rangle$ and a parameter $\Gamma > 0$, find a Γ-Clustering C^* such that the search space reduction is maximized.

Remark III.5 (Overlap). Note that in Definition III.2 overlapping clusters are not permitted. This is necessary for the problem in Definition III.3 to be well defined. In addition, we will see in the following section that clusters that have a separation of $\Gamma_i > 1$ cannot overlap.

B. Solution Quality Bounds

In this section, we show that when the graph G is metric and $\Gamma > 1$, then the solution to the Γ-clustered path planning problem provides a $\min\left(2, \left(1 + \frac{3}{2\Gamma}\right)\right)$-factor approximation to the optimal.

Theorem III.6 (Approximation Factor). *Given a metric discrete path planning problem P with optimal solution p^* and cost c^*, a Γ-Clustering $C = \{V_1, V_2, \ldots, V_m\}$ where $\Gamma > 1$, then the optimal solution $(p')^*$ to the clustered problem P' over the same set of vertices is a solution to P with cost $(c')^* \leq \min\left(2, 1 + \frac{3}{2\Gamma}\right) c^*$.*

The proof this Theorem III.6 and all subsequent proofs are found in [21].

C. Finding Γ-Clusters

Before we describe our method for finding optimal Γ-Clustering (s), we describe a few properties of Γ-Clusterings.

1) Overlap: A special property of Γ-Clustering is that when $\Gamma > 1$, there are no overlapping clusters.

Lemma III.7. *Given a graph G and clusters V_i and V_j with $\Gamma_i > 1$ and $\Gamma_j > 1$, then V_i and V_j do not overlap.*

2) Uniqueness: The non-overlapping property for Γ-Clusterings with $\Gamma > 1$ implies that there exists a unique maximal clustering set C^* (more clusters equals more reductions in the search space size).

Proposition III.8. *Given graph G and a parameter $\Gamma > 1$, the problem of finding a Γ-Clustering C^* that maximizes the search space reduction has a unique solution C^*. Furthermore C^* contains all clusters V_i with separation $\Gamma_i \geq \Gamma$.*

3) An MST Approach For Finding Γ-Clusters: Given an input $\Gamma > 1$, Algorithm 1 computes the optimal Γ-Clusterings, i.e., the Γ-Clustering with maximum search space reduction. Informally the algorithm deletes edges in the graph from largest to smallest (line 6-7) to look for Γ-Clusters. It uses a minimum spanning tree (MST) to keep track of when the graph becomes disconnected, and when it does the disconnected components are tested to see if they qualify as Γ-Clusters (line 9-11). Regardless, any non-trivial sized disconnected component (Γ-Cluster or not) is added back to the queue (line 13-14), so that it can be broken and tested again to find of all nested Γ-Clusters.

Theorem III.9. *Given G and a $\Gamma > 1$, Algorithm 1 finds the optimal Γ-Clustering C^* in $O(|V|^3)$ time.*

Algorithm 1: Γ-CLUSTERING(G, Γ)

---

1 **assert**$(\Gamma > 1)$
2 $C \leftarrow \{\}$
3 $M \leftarrow \{\text{MST}(G)\}$
4 **while** $|M| > 0$ **do**
5     $m \leftarrow M.pop()$
6     $\alpha \leftarrow$ largest edge cost in m
7     $M' \leftarrow$ disconnected trees after removing edge(s) of cost α from m
8   **for** $m' \in M'$ **do**
9         $G' \leftarrow$ graph induced by $V[m']$
10         $\beta \leftarrow$ max edge cost of G'
11   **if** G' *is a clique and* $\Gamma' \equiv \alpha/\beta \geq \Gamma$ **then**
12             $C \leftarrow C \cup \{V[m']\}$
13   **if** $|m'| > 1$ **then**
14             $M \leftarrow M \cup m'$

15 **return** C

D. Search Space Reduction

The last remaining question is to determine how much the clustering approach reduces the search space. In general, this is difficult to answer since it depends on the particular constraints of the path planning problem. However, to get an understanding of the search space reduction, consider the example of TSP with a non-nested clustering (nesting would result in further search space reductions). Let r be the ratio of the non-clustered search space size N_0, to the clustered search space size N_1, for a graph with vertices V and a clustering $C = \{V_1, V_2, \ldots, V_m\}$. Then, the ratio (derived from counting the number of solutions) is as follows:

$$r \equiv \frac{N_0}{N_1} = \frac{|V|!}{m! \prod_i^m |V_i|!}$$

To further simplify the ratio consider the case where all clusters are equally sized (for all $V_i, V_j \in C$, $|V_i| = |V_j|$):

Proposition III.10 (Search Space Reduction)**.** *Given a graph G and a clustering $C = \{V_1, V_2, \ldots, V_m\}$ such that $|V_i| = |V_j|$ for all $i, j \in [1, m]$ then the ratio r of the search space size for the original TSP problem to the cluster TSP problem is* [1]

$$r = \Omega\left((m!)^{x-1}\right),$$

where $x = |V|/m$.

To get an idea of the magnitude of r, consider an instance of size $|V| = 100$ divided into four equal clusters. The clustered problem has a feasible solution space of size $N_1 \approx 1.49 \times 10^{-56} N_0$, where N_0 is the feasible solution space size of the non-clustered problem. However, N_1 is still extremely large at about 1.39×10^{102}.

IV. Γ-CLUSTERING EXPERIMENTS

In this section we present experimental results that demonstrate the effectiveness of Γ-Clustering for solving discrete path planning problems. We focus on metric TSP instances

---

[1] The big-Ω notation states that for large enough $|V|$ the ratio r is at least $k(m!)^{x-1}$ for some constant k.

---

drawn from the established TSP library TSPLIB [24], which have a variety of TSP problem types (the first portion of the instance name indicates the type and the number indicates the size). The tests were conducted with $\Gamma = 1.000001$, for which Theorem III.6 implies that the solution to the clustered problem gives a $\min\left(2, 1 + \frac{3}{2\Gamma}\right)$-factor approximation to the TSP instance. However, we will see that the observed gap in performance is considerably smaller.

To test the effectiveness of the clustering method, we perform Γ-Clustering on each TSP instance and recorded both the runtime and the number of clusters found. This gives us an idea of whether or not instances from TSPLIB have a structure that can be exploited by Γ-Clustering. Then, we use standard integer programming formulations for both the original TSP instance and the general clustered version of the instance (CTSP*). We solve each instance three times using the solver Gurobi [25] and recorded the average solver time and solution quality. All instances were given a time budget of 900 seconds, after which they were terminated and the best solution found in that run was outputted. The TSP instances reported in this paper are the 37 instances that were solved to within 50% of optimal and have Γ-Clusters. The remaining instances are not reported as they would have required more than 900 seconds to provide a meaningful comparison.

Additionally we demonstrate Γ-Clustering on an office environment to gain insight into the structure of Γ-Clusters.

A. Integer Programming Formulation

The clustering algorithm was implemented in Python and run on an Intel Core i7-6700, 3.40GHz with 16GB of RAM. The integer programming (IP) expression of the TSP problem and clustered problem is solved on the same computer with Gurobi, also accessed through python. The results of both of these approaches is found in Table I and summarized in Figure 3.

The following is the IP expression used for the clustered and non-clustered path planning problems, where each variable $e_{a,b} \in \{0, 1\}$:

minimize

$$\sum_{a=1}^{|V|} \sum_{b=1}^{|V|} e_{a,b}\, w(v_a, v_b) \tag{1}$$

subject to

$$\sum_{b=1}^{|V|} e_{a,b} = 1, \quad \text{for each } a \in \{1, 2, \ldots, |V|\} \tag{2}$$

$$\sum_{a=1}^{|V|} e_{a,b} = 1, \quad \text{for each } b \in \{1, 2, \ldots, |V|\} \tag{3}$$

$$\sum_{\forall v_a \in V_i, v_b \notin V_i} e_{a,b} = 1, \quad \text{for each } i \in \{1, 2, \ldots, m\} \tag{4}$$

$$\sum_{\forall v_a \notin V_i, v_b \in V_i} e_{a,b} = 1, \quad \text{for each } i \in \{1, 2, \ldots, m\} \tag{5}$$

$$\sum_{e_{a,b} \in E'} e_{a,b} \leq |E'| - 1, \quad \text{for each subtour } E' \tag{6}$$

The formulation was adapted from a TSP IP formulation found in [26], where the Boolean variables $e_{a,b}$ represent the inclusion/exclusion of the edge $\langle v_a, v_b \rangle$ from the solution.

Constraints 2 and 3 restricts the incoming and outgoing degree of each vertex to be exactly one (the vertex is visited exactly once). Similarly constraints 4 and 5 restrict the incoming and outgoing degree of each cluster to be exactly one (these constraints are only present in the clustered version of the problem). Constraint 6 is the subtour elimination constraint, which is lazily added to the formulation as conflicts occur due to the exponential number of these constraints. For each instance, we seed the solver with a random initial feasible solution.

B. Results

Figure 2 shows the ratio of time spent finding clusters with respect to total solver time. In all instances this time is less than 6% and most is less than 1%. Additionally as the total solver time grows, the ratio gets smaller (Time02 is for instances that use the full 900 seconds). This approach is able to find Γ-Clusters on 63 out of 70 TSP instances. In total it found 3700 non-trivial Γ-Clusters (i.e., clusters with $|V_i| \geq 2$), which is promising since the TSPLIB library contains a variety of different TSP applications.

Figure 3 and Table I, shows that for instances that do not time out the solution path costs found by the clustering approach are close to optimal (Cost01 is close to 1 in the figure and the instances from burma14 to pr107 are almost all within 1% error as shown in the table). Furthermore when the solver starts to time out (exceeds its 900 second time budget) the solution quality of the Γ-Clustering approach starts to surpass the solution quality of the non-clustered approach (as shown by Cost02 in the figure). We attribute this trend to the fact that the clustered approach needs less time to search its feasible solution space and thus is able to find better quality solutions faster than its counterpart. In instances gr202, kroB200, pr107, tsp225, and gr229 the clustering approach does very well compared to the non-clustering approach, enabling the solver to find solutions within 1% of optimal for the first three instances and solutions within 6% of optimal for the latter two instances, while the non-clustered approach exceeds 8% of optimal in the first three instances and 28% of optimal for the latter two instances.

On average the Γ-Clustering approach is more efficient than the non-clustered approach, which is highlighted in Figure 3 and Table I. For the results that do not time out (Time01 in the figure) we often save more than 50% of the computational time, while maintaining a near optimal solution quality (Cost01 in the figure). For instances that require most or all of the 900 seconds (harder instances) the time savings can be quite large. This is particularly clear from the table when we compare the easy instances burma14 up to st70 that have an average time savings of around 60% to the harder instances kroA100, gr96, bier127, and ch130, which all have a time savings of more than 95%. For the instances that both time out (Time02 in the figure), the time savings is not present since both solvers use the full 900
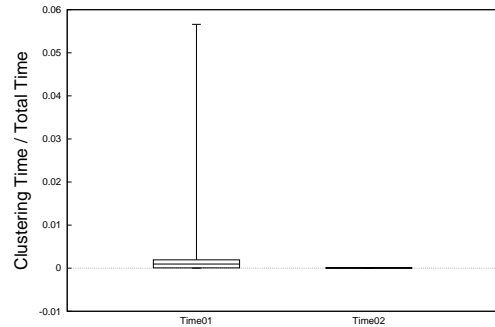

Fig. 2: Box plot of the clustering time ratio with respect to the Γ-Clustering approach. The data is categorized by instances that did not time out (Time01) and instances that did time out (Time02).

seconds.

From these results we can see that when the Γ-Clustering approach does not time out, we usually save time and when the approach does time out we often find better quality solutions as compared to the non-clustered approach.

Remark IV.1 (Discussion on using Γ-Clustering for TSP). It is worth emphasizing that we are not necessarily recommending solving TSP instances in this manner. We are simply using TSP as an illustrative example to show how Γ-Clustering can be used to reduce computation time in a given solver. Many discrete path planning problems are solved with IP solvers and as such we hope our results help provide some insight as to how Γ-Clustering would work on other path planning problems. In general unless the solver approach (IP or not) takes advantage of the clustering, there is no guarantee that a computation saving will be achieved.

C. Clustering Real-World Environments

As shown in Figure 1 we have also performed Γ-Clustering on real-world environments. The figure shows the floor-plan for a portion of one floor of the Engineering 5 building at the University of Waterloo. Red dots denote the locations of desks within the environment. We encoded the environment with a graph, where there is a vertex for each red dot and edge weights between vertices are given by the shortest axis-aligned obstacle free path between the locations (obstacle-free Manhattan distances). The figure shows the results of clustering for $\Gamma = 1.000001$. We see that locations that are close together are formed into clusters unless there are other vertices in close proximity.

A path planning problem on this environment could be robotic mail delivery, where a subset of locations must be visited each day. The clusters could then be visited together (or visited using the same robot).

V. CONCLUSION

In this paper we presented a new clustering approach called Γ-Clustering. We have shown how it can be used to approximate the discrete path planning problems to within a constant factor of $\min\left(2, 1 + \frac{3}{2\Gamma}\right)$, more efficiently than solving the original problem. We verify these findings with a set of experiments that show on average a time savings and a solution quality that is closer to the optimal solution than it
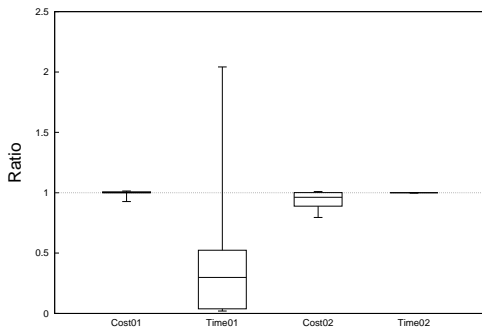
Fig. 3: Box plot of the solver cost and time ratios for the Γ-Clustering approach with respect to the TSP approach. The data is categorized by instances that did not time out (Cost01 and Time01) and instances that did time out (Cost02 and Time02).

is to the bound. For future directions we will be investigating other path planning applications, which includes online path planning with dynamic environments.

REFERENCES

[1] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.

[2] L. Gouveia and M. Ruthmair, "Load-dependent and precedence-based models for pickup and delivery problems," *Computers & Operations Research*, vol. 63, pp. 56–71, 2015.

[3] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[4] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[5] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[6] S. Guha, R. Rastogi, and K. Shim, "Rock: A robust clustering algorithm for categorical attributes," *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[7] D. Katselis and C. L. Beck, "Clustering fully and partially observable graphs via nonconvex optimization," in *American Control Conference*, 2016, pp. 4930–4935.

[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[9] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*. Springer, 2006, pp. 25–71.

[10] H. Meyerhenke, B. Monien, and T. Sauerwald, "A new diffusion-based multilevel algorithm for computing graph partitions," *Journal of Parallel and Distributed Computing*, vol. 69, no. 9, pp. 750–761, 2009.

[11] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *European Conference on Computer Vision*, 2010, pp. 1–14.

[12] X. Jin, S. Gupta, J. M. Luff, and A. Ray, "Multi-resolution navigation of mobile robots with complete coverage of unknown and complex environments," in *American Control Conference*, 2012, pp. 4867–4872.

[13] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[14] C. Bäckström, A. Jonsson, and P. Jonsson, "Automaton plans," *Journal of Artificial Intelligence Research*, vol. 51, no. 1, pp. 255–291, 2014.

[15] M. Levihn, L. P. Kaelbling, T. Lozano-Perez, and M. Stilman, "Foresight and reconsideration in hierarchical planning and execution," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 224–231.

[16] A. Das, M. Diu, N. Mathew, C. Scharfenberger, J. Servos, A. Wong, J. S. Zelek, D. A. Clausi, and S. L. Waslander, "Mapping, planning, and sample detection strategies for autonomous exploration," *Journal of Field Robotics*, vol. 31, no. 1, pp. 75–106, 2014.

	$|C|$	TSP		CTSP*	
		% Error	Time (t)	% Error	Time (t)
burma14	5	0.00	<1	0.39	<1
ulysses16	6	0.00	<1	0.73	<1
berlin52	17	0.00	<1	0.06	<1
swiss42	16	0.00	<1	0.94	<1
eil51	11	0.00	1	0.00	**<1**
eil76	13	0.00	3	0.00	**1**
rat99	28	0.00	6	0.83	13
eil101	16	0.00	8	0.00	**4**
ulysses22	10	0.00	8	0.00	<1
pr76	27	0.00	40	1.40	**11**
st70	23	0.00	45	0.44	**13**
lin105	42	0.00	225	0.00	**8**
kroE100	43	0.00	414	0.39	**20**
kroC100	46	0.00	445	0.00	**12**
kroA100	44	0.00	602	1.11	**23**
gr96	32	0.00	845	0.05	**38**
bier127	37	0.00	900	0.23	**24**
gr137	44	0.00	900	0.00	**154**
kroD100	42	0.00	900	0.57	**471**
kroB100	42	0.01	900	0.96	900
ch130	59	0.20	900	0.90	**29**
ch150	53	0.22	900	0.31	900
kroB150	65	0.35	900	0.35	**395**
kroA150	58	1.37	900	**0.15**	**492**
rat195	57	1.75	900	**0.89**	900
kroA200	82	5.59	900	**1.38**	900
pr124	18	5.78	900	**4.24**	900
gr202	74	8.56	900	**0.64**	**703**
pr136	48	9.78	900	**4.88**	900
kroB200	80	10.16	900	**0.67**	900
pr107	6	13.01	900	**0.40**	900
a280	11	20.11	900	20.46	900
pr144	40	24.40	900	**20.03**	900
tsp225	66	28.50	900	**5.76**	900
gr229	73	28.63	900	**2.24**	900
pr152	44	40.64	900	41.15	900
gil262	98	44.81	900	**21.00**	900

TABLE I: Experimental results for TSPLIB instances: $|C|$ reports the number of Γ-Clusters, % error reports the average error from optimal and time column reports the average solver time. The instances where the CTSP* approach outperforms the TSP approach are highlighted in bold. Results are sorted from least to most difficult for the non-clustered approach.

[17] M. R. K. Ryan, "Exploiting subgraph structure in multi-robot path planning," *Journal of Artificial Intelligence Research*, pp. 497–542, 2008.

[18] C. Chevalier and I. Safro, "Comparison of coarsening schemes for multilevel graph partitioning," in *International Conference on Learning and Intelligent Optimization*, 2009, pp. 191–205.

[19] R. M. Karp, "Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane," *Mathematics of operations research*, vol. 2, no. 3, pp. 209–224, 1977.

[20] Y. Haxhimusa, W. G. Kropatsch, Z. Pizlo, and A. Ion, "Approximative graph pyramid solution of the E-TSP," *Image and Vision Computing*, vol. 27, no. 7, pp. 887–896, 2009.

[21] F. Imeson and S. L. Smith, "Clustering in Discrete Path Planning for Approximating Minimum Length Paths," *ArXiv e-prints*, Feb. 2017. [Online]. Available: https://arxiv.org/abs/1702.08410

[22] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[23] C. E. Noon and J. C. Bean, "An efficient transformation of the generalized traveling salesman problem," *Information Systems and Operational Research (INFOR)*, vol. 31, no. 1, pp. 39–44, 1993.

[24] G. Reinelt, "TSPLIB–a traveling salesman problem library," *ORSA Journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[25] G. Optimization *et al.*, "Gurobi optimizer reference manual," 2012. [Online]. Available: http://www.gurobi.com

[26] L. Gouveia and J. M. Pires, "The asymmetric travelling salesman problem and a reformulation of the miller–tucker–zemlin constraints," *European Journal of Operational Research*, vol. 112, no. 1, pp. 134–146, 1999.