Fractal Graph Optimization Algorithms*

James R. Riehl and João P. Hespanha†

***Abstract*— We introduce methods of hierarchically decomposing three types of graph optimization problems: all-pairs shortest path, all-pairs maximum flow, and search. Each method uses a partition on the graph to create a high level problem and several lower level problems. The computations on each level are identical, so the low level problems can be further decomposed. In this way, the problems become fractal in nature. We use these decomposition methods to establish upper and lower bounds on the optimal criteria of each problem, which can be achieved with much less computation than what is required to solve the original problem. Also, for each problem, we find an optimal number of partitions that minimizes computation time. As the number of hierarchical levels increases, the computational complexity decreases at the expense of looser bounds.**

I. Introduction

Graph optimization problems such as shortest path, maximum flow, and search problems are essential to a large number of control applications including navigation [5], path planning [4], and network routing [9], but for graphs with many nodes, the computation required to solve these problems can be impractical.

Consider a weighted directed graph $G(V, E)$ with n vertices and m edges.

- In the *all pairs shortest path* problem, each edge in G has an associated cost, and the goal is to find the shortest path between every pair of vertices in G. The best known computation for this problem is $O(nm + n^2 \log n)$ [1], which is equivalent to $O(n^3)$ for dense graphs, i.e. graphs for which $O(m) = O(n^2)$.
- In the *all pairs maximum flow* problem, each edge has a capacity constraint, and the goal is to find the maximum flow between every pair of vertices in G. This problem can be computed in $O(n^3 m \log \frac{n^2}{m})$ [2]. This is equivalent to $O(n^5)$ for dense graphs.
- In the *search* problem, each edge has an associated cost, and each vertex has a reward. The optimal search path is the path in G that maximizes the reward subject to a cost constraint. This problem is NP-hard even when significant structure is imposed on the graph [10].

When the computation of the exact solution would require an unreasonable amount of time, it is useful to approximate the solution. This paper introduces a framework of bounding the optimal solutions to these three problems by partitioning the graph, and generating best-case and worst-case solutions on a new smaller graph, whose vertices are subsets of the vertices in the original graph. For each problem, we show how to use the worst-case solution to generate an approximate solution on the original graph, and the best-case solution provides a bound on how far this approximation is from optimal. Additionally, the construction of the worst-case problem requires the solution of several identical problems of smaller dimension. These smaller problems can then be further decomposed, creating a hierarchical structure and making the overall problem fractal in nature. Increasing the number of hierarchical levels reduces computation, but generally results in looser bounds.

A. Shortest path

There is a rich literature on the hierarchical decomposition of the shortest path problem, especially as it relates to path planning for mobile robots [4], and navigation systems [5]. Holte *et al.* found that searching state spaces with hierarchical abstractions performed better than classical methods [6].

Shen and Caines presented results on hierarchically accelerated dynamic programming that is based on a similar idea to our shortest path decomposition [7]. Using state aggregation methods, they were able to speed up dynamic programming algorithms for finite state machines by orders of magnitude at the expense of some sub-optimality, for which they give bounds. Romeijn and Smith proposed an algorithm to solve an aggregated all pairs shortest path problem motivated by minimizing vehicle travel time [8]. Under the assumption that graphs in each level of aggregation have the same structure, they showed the computational complexity of their approximation to be $O(n \log n)$ for aggregation on two levels of sparse graphs, and $O(n^{\frac{2}{L}} \log n)$ for aggregation on L levels. Our treatment of the shortest path approximation will closely resemble that of [8] with a few modifications and expanded results. We give both upper and lower bounds on the optimal solution, and the choice of the specific partition to use is a degree of freedom for the user.

Numerical test results of the meta-shortest path problem show that hierarchical approximations are best-suited to graphs with a clustered structure, where there is a obvious way to partition the graph. These types of graphs are common to many applications such as transportation, computer networks, and integrated circuits. The worst-case diameter approximation for a test-graph having a clustered structure was within 7% of the actual diameter. We also show the approximation applied to a lattice graph, which has one of the worst possible structures for this method because there

*This material is based upon work supported by the National Science Foundation under Grants No. CCR-0311084, ANI-0322476.

†{jriehl,hespanha}@ece.ucsb.edu, Center for Control, Dynamical Systems, and Computation, Electrical and Computer Engineering Department, University of California, Santa Barbara, CA 93106-9560.

is no clear way to partition it. For the lattice test graph, the worst-case diameter approximation was within 60%.

B. Maximum flow

There has been considerably less work on hierarchical approaches to maximum flow problems. Bohacek *et al.* developed a technique to compute routing tables for max-flow network routing that involves hierarchically decomposing the network, and they showed the computational complexity to be about $O(n^{3.1})$ with a performance that is in some cases as good as the general flat max-flow routing problem [9]. Our results improve the computational complexity to $O(n^{2.8})$ for one hierarchical level.

C. Search

Search theory as we know it dates back to World War II, and the U.S. Coast Guard has applied it to search and rescue missions. The search problem formulated in this paper is based on a discretization of a continuous search problem solved by DasGupta, Hespanha, and Sontag [10]. We adapt this formulation to allow for hierarchical decomposition.

Our hierarchical search approximation is still NP-hard, but the computation is over much smaller sets of vertices. The result is a computational reduction that is exponential in n.

D. Organization

Section II presents the idea of partitions and meta-graphs, introducing concepts and notation that will be used throughout the paper. Sections III, IV, and V define each type of problem, develop upper and lower bounds for the solutions, and analyze computational complexity. These sections also include explicit procedures for constructing the approximate shortest path matrix, max-flow matrix, and optimal search path. Section VI presents some test results for the shortest path matrix problem, and the final section is a discussion of the results with suggestions for future research.

II. Graphs, meta-graphs, and subgraphs

Consider a graph $G := (V, E)$ with vertex set V and edge set $E \subset V \times V$.

A *partition* $\bar{V} := \{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_k\}$ of G is a set of disjoint subsets of V such that $\bar{v}_1 \cap \bar{v}_2 \cap \ldots \cap \bar{v}_k = V$. We call these subsets $\bar{v}_i$ *meta-vertices*.

For a given meta-vertex $\bar{v}_i \in \bar{V}$, we define the *subgraph of G induced by* $\bar{v}_i$ to be the subgraph $G|\bar{v}_i := (\bar{v}_i, E \cap \bar{v}_i \times \bar{v}_i)$.

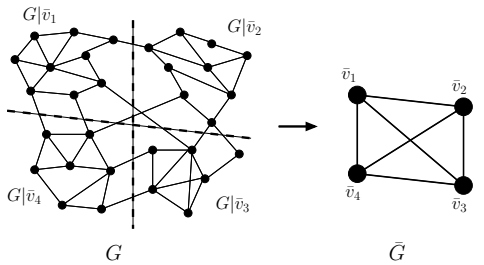


Fig. 1. Example of a graph partitioned into 4 *meta-vertices*.

Figure 1 shows an example of the graph partitioning process, where the dashed lines through G separate the partitioned subgraphs $G|v_i$, which are represented by meta-vertices in $\bar{G}$.

For the purposes of this paper, we will require that the subgraphs $G|\bar{v}$ be connected, that is, for every $v_i, v_j \in \bar{v}$, there must exist a path in $G|\bar{v}$ from v_i to v_j.

Given a partition $\bar{V}$ of the vertex set V, we define the *meta-graph of G induced by the partition* $\bar{V}$ to be the graph $\bar{G} := (\bar{V}, \bar{E})$ with an edge $\bar{e} \in \bar{E}$ between *meta-vertices* $\bar{v}_i, \bar{v}_j \in \bar{V}$ if and only if G has at least one edge $e \in E$ between vertices v and v' for some $v \in \bar{v}_i$ and $v' \in \bar{v}_j$. In general, there may exist several such edges $e \in E$ and we call these the *edges associated with the meta-edge* $\bar{e}$. With some abuse of notation, we write $e \in \bar{e}$ to express that the fact that e is associated with the meta-edge $\bar{e}$

III. Shortest Path Matrix Problem

The shortest path matrix problem involves constructing a $n \times n$ matrix of the minimum-cost path between all pairs of vertices in a graph. Our formulation is a slight variation on the conventional all-pairs shortest path (APSP) problem because in addition to assigning a cost to each edge, we also assign a cost to each vertex. This is crucial in facilitating the hierarchical decomposition of the problem, as will be explained in section III-A. Adding vertex costs does not increase the computational complexity of the problem, however.

Data Directed connected *graph* $G := (V, E)$; *edge cost* function $\eta : E \to [0, \infty)$; *vertex cost* function $\nu : V \to [0, \infty)$.

Path A *path in* G from $v_{\text{init}} \in V$ to $v_{\text{final}} \in V$ is a sequence of vertices (where $v_1 := v_{\text{init}}$, $v_f := v_{\text{final}}$)

$$p := (v_1, v_2, \ldots, v_{f-1}, v_f), \qquad (v_i, v_{i+1}) \in E.$$

The *path-cost* is given by

$$C(p) := \sum_{i=1}^{f-1} \eta(v_i, v_{i+1}) + \sum_{i=1}^{f} \nu(v_i).$$

Objective For every pair of vertices $(v_{\text{init}}, v_{\text{final}})$, compute the path p that minimizes the path-cost $C(p)$. We denote this path by p^* and the minimum cost $C(p^*)$ by $J^*_G(v_{\text{init}}, v_{\text{final}})$. The largest minimum cost over all possible pairs of vertices $(v_{\text{init}}, v_{\text{final}})$ is called the *diameter* of the graph and is denoted by $\|G\|$.

A. Worst-case meta-shortest path

Let $\bar{G}_{\text{worst}} := (\bar{V}, \bar{E})$ be a meta-graph of G, with edge cost and vertex cost defined by $\bar{\eta}_{\text{worst}}(\bar{e}) := \min_{e \in \bar{e}} \eta(e)$ and $\bar{\nu}_{\text{worst}}(\bar{v}) := \big\|G|\bar{v}\big\|$.

Note that to assign the vertex costs of $\bar{G}_{\text{worst}}$ one needs to compute the diameter of all subgraphs and therefore solve k smaller shortest-path matrix problems. This is the key step in the hierarchical decomposition of the problem.

B. Best-case meta-shortest path

Let $\bar{G}_{\text{best}} := (\bar{V}, \bar{E})$ be a meta-graph of G, with edge cost and vertex cost defined by $\bar{\eta}_{\text{best}}(\bar{e}) := \min_{e \in \bar{e}} \eta(e)$ and $\bar{\nu}_{\text{best}}(\bar{v}) := \min_{v \in \bar{v}} \nu(v)$.

Theorem 1 *For every partition $\bar{V}$ of G*

$$J^*_{\bar{G}_{\text{best}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}}) \leq J^*_G(v_{\text{init}}, v_{\text{final}}) \leq J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$$

where $v_{\text{init}} \in \bar{v}_{\text{init}}$, $v_{\text{final}} \in \bar{v}_{\text{final}}$.

The construction of the upper bound provides a procedure for generating an approximate shortest path between each pair of vertices in G, and the cost of this path lies between $J^*_G(v_{\text{init}}, v_{\text{final}})$ and $J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$.

Proof: To verify the upper bound, we demonstrate that one can use the shortest path from $\bar{v}_{\text{init}}$ to $\bar{v}_{\text{final}}$ in $\bar{G}_{\text{worst}}$ to construct an approximate shortest path from v_{init} to v_{final} in G. We do this by sequentially connecting v_{init}, the endpoints of the minimum cost edge between each meta-vertex in the optimal worst-case path, and v_{final} with the shortest paths between them in their respective subgraphs $G|\bar{v}$. The cost of this approximate shortest path is bounded below by $J^*_G(v_{\text{init}}, v_{\text{final}})$ and above by $J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$.

Similarly, we can check the lower bound by constructing a feasible path from $\bar{v}_{\text{init}}$ to $\bar{v}_{\text{final}}$ in $\bar{G}_{\text{best}}$ from the optimal path p^* connecting v_{init} to v_{final} in G. The cost of this path is bounded below by $J^*_{\bar{G}_{\text{best}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$ and above by $J^*_G(v_{\text{init}}, v_{\text{final}})$. The details of both upper and lower bound constructions can be found in [11]. ■

C. Computation

For a weighted directed graph with n vertices and m edges, Karger *et al.* showed that the all pairs shortest path problem can be solved with computational complexity $O(nm + n^2 \log n)$ [1]. In this analysis, we consider a graph to be dense if $O(m) = O(n^2)$ and sparse if $O(m) = O(n)$. For dense graphs, the above result is equivalent to $O(n^3)$, and for sparse graphs, it is $O(n^2 \log n)$. We will analyze the computation of the worst-case meta-shortest path matrix for dense graphs, but give results for both dense and sparse graphs. Here we analyze sequential computation, but see [11] for results on parallel computation.

Our formulation of the shortest path problem is slightly different than the conventional one because it includes vertex costs, but we can make small modifications in the algorithm to account for these vertex costs without increasing the computational complexity of the problem.

Partitioning a graph into k meta-vertices, and assuming that each meta-vertex contains roughly the same number of vertices, the computational complexity of the worst-case meta-shortest path problem will be $O(k(\frac{n}{k})^3 + k^3)$ for dense graphs, because there are k diameters to compute on subgraphs having approximately $\frac{n}{k}$ vertices each, and there is the top-level all-pairs shortest path problem over k meta-vertices. One can minimize this expression over k to obtain the following value of k that minimizes computation time: $k^* = \left(\frac{2}{3}\right)^{\frac{1}{5}} n^{\frac{3}{5}}$.

This results in a computational complexity of $O(n^{\frac{9}{5}})$ for the worst-case meta-shortest path problem using one hierarchical level. One can further decompose the problem by approximating each of the k diameter problems using the worst-case meta-shortest path method. In this way, the problem can be decomposed on many levels. As the number of hierarchical levels increases, the computation decreases at the obvious expense of less accurate bounds on the solution. Table I shows the approximate computational complexity for up to 3 hierarchical levels for dense and sparse graphs.

TABLE I
COMPUTATIONAL COMPLEXITY OF THE WORST-CASE META-SHORTEST PATH MATRIX FOR UP TO 3 HIERARCHICAL LEVELS

Levels	Dense	Sparse
0	$O(n^3)$	$O(n^2 \log n)$
1	$O(n^{1.8})$	$O(n^{1.43})$
2	$O(n^{1.42})$	$O(n^{1.24})$
3	$O(n^{1.25})$	$O(n^{1.19})$

Note that there is no need to analyze the computation of the best-case meta-shortest path problem because it can be computed much faster, since it does not involve diameter computations for the vertex costs.

IV. MAX-FLOW MATRIX PROBLEM

In the max-flow matrix problem, the goal is to construct an $n \times n$ matrix containing the maximum flow intensities between all pairs of vertices in a graph. The flow through each edge is limited by the bandwidth or capacity of that edge. In this formulation, the vertices are also assigned bandwidths. The vertex bandwidth is what allows for the hierarchical decomposition of this problem.

Data Directed connected *graph* $G := (V, E)$ with vertex set V and edge set $E \subset V \times V$; *edge bandwidth* function $\eta : E \to [0, \infty)$; *vertex bandwidth* function $\nu : V \to [0, \infty)$.

Flow A *flow in* G from $v_{\text{init}} \in V$ to $v_{\text{final}} \in V$ is a function $f : E \to [0, \infty)$ for which there exist some $\mu \geq 0$ such that

$$f_{out}(v) - f_{in}(v) = \begin{cases} \mu & v = v_{\text{init}} \\ -\mu & v = v_{\text{final}} \\ 0 & \text{otherwise,} \end{cases} \qquad \forall v \in V, \quad (1)$$

$$0 \leq f(e) \leq \eta(e), \qquad \forall e \in E \quad (2)$$

$$0 \leq f_{in}(v) \leq \nu(v), \qquad \forall v \in V \quad (3)$$

$$0 \leq f_{out}(v) \leq \nu(v), \qquad \forall v \in V \quad (4)$$

In the above,

$$f_{in}(v) := \sum_{e \in \text{In}[v]} f(e), \qquad f_{out}(v) := \sum_{e \in \text{Out}[v]} f(e), \quad (5)$$

where $\text{In}[v]$ denotes the set of edges that enter the vertex v and $\text{Out}[v]$ the set of edges that exit v. The constant μ is called the *intensity* of the flow.

Objective For every pair of vertexes $(v_{\text{init}}, v_{\text{final}})$, compute the flow f^* with maximum intensity μ from v_{init} to v_{final}.

The maximum intensity is denoted by $J^*_G(v_{\text{init}}, v_{\text{final}})$ and is called the *maximum flow from* v_{init} *to* v_{final}. The smallest maximum flow over all possible pairs of vertices is called the *bandwidth* of the graph and is denoted by $\|G\|$.

A. Worst-case meta-max flow

Let $\bar{G}_{\text{worst}} := (\bar{V}, \bar{E})$ be a meta-graph of G, with edge bandwidth and vertex bandwidth defined by $\bar{\eta}_{\text{worst}}(\bar{e}) := \sum_{e \in \bar{e}} \eta(e)$ and $\bar{\nu}_{\text{worst}}(\bar{v}) := \big\|G|\bar{v}\big\|$.

Note that to construct the graph $\bar{G}_{\text{worst}}$ one needs to compute the bandwidth of all subgraphs and therefore solve several smaller max-flow matrix problems.

B. Best-case meta-max flow

Let $\bar{G}_{\text{best}} := (\bar{V}, \bar{E})$ be a meta-graph of G, with edge bandwidth and vertex bandwidth defined by $\bar{\eta}_{\text{best}}(\bar{e}) := \sum_{e \in \bar{e}} \eta(e)$ and $\bar{\nu}_{\text{best}}(\bar{v}) := +\infty$.

Theorem 2 *For every partition* $\bar{V}$ *of* G

$$J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}}) \le J^*_G(v_{\text{init}}, v_{\text{final}}) \le J^*_{\bar{G}_{\text{best}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$$

where $v_{\text{init}} \in \bar{v}_{\text{init}}$, $v_{\text{final}} \in \bar{v}_{\text{final}}$.

The construction of the lower bound contains a procedure for generating an approximate maximum flow between each pair of vertices in G, and the intensity of this flow lies between $J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$ and $J^*_G(v_{\text{init}}, v_{\text{final}})$.

Proof: Given a worst-case maximum flow $\bar{f}^*_{\text{worst}}(\bar{e})$ in $\bar{G}_{\text{worst}}$, the idea is to construct a flow $f(e)$ in G that satisfies (1)–(4). This is possible because the subgraphs associated with each meta-vertex are connected and the total flow into and out of a meta-vertex is always no greater than the bandwidth of that meta-vertex. The intensity of this approximate maximum flow is bounded below by $J^*_{\bar{G}_{\text{worst}}}(\bar{v}_{\text{init}}, \bar{v}_{\text{final}})$, and above by $J^*_G(v_{\text{init}}, v_{\text{final}})$.

The proof for the best-case upper bound is straightforward because one can easily construct a flow in $\bar{G}_{\text{best}}$ from an optimal flow $f^*(e)$ in G. The intensity of this flow will be bounded below by $J^*_G(v_{\text{init}}, v_{\text{final}})$, and above by $J^*_{\bar{G}_{\text{best}}}(\bar{v}_{\text{init}}, v_{\text{final}})$. To see details of upper and lower bound flow constructions, see [11]. ∎

C. Computation

Given a directed graph $G(V, E)$ with n vertices and m capacitated edges, Goldberg and Tarjan [2] showed that one can compute the maximum flow between two vertices in $O(nm \log \frac{n^2}{m})$ time. To compute the max-flow between all pairs of vertices in an *undirected* graph, Gomory and Hu showed that one only needs to solve $n-1$ maximum flow problems [3], but since we are considering directed graphs, we must compute the flow between all $\frac{n(n-1)}{2}$ vertices. This results in a complexity of $O(n^3 m \log \frac{n^2}{m})$ to generate the complete max-flow matrix.

The addition of vertex bandwidths to the conventional max-flow problem does not increase computational complexity because in the worst case, we can simply model a vertex with bandwidth ν as two vertices without bandwidths connected by an edge with bandwidth ν.

In the case of a dense graph, the complexity of computing the max-flow matrix is $O(n^5)$. To solve the worst case meta-max-flow problem, we partition G into k subgraphs, where each subgraph contains roughly the same number of vertices. Then we will need to compute k subgraph bandwidths having complexity $O((\frac{n}{k})^5)$, and solve a maximum flow problem over the k meta-vertices which is $O(k^5)$. The resulting complexity of this problem is $O(k(\frac{n}{k})^5 + k^5)$.

Minimizing this expression over k yields the optimal number of partitions for one hierarchical level, $k^* = .98n^{\frac{5}{9}}$.

The complexity of the worst-case meta-max-flow problem that results from this choice of k is $O(n^{\frac{25}{9}})$. Table II shows the approximate computational complexity for up to 3 hierarchical levels on both dense and sparse graphs.

TABLE II
COMPUTATIONAL COMPLEXITY OF THE WORST-CASE META-MAX FLOW PROBLEM FOR UP TO 3 HIERARCHICAL LEVELS

Levels	Dense	Sparse
0	$O(n^5)$	$O(n^4 \log n)$
1	$O(n^{2.78})$	$O(n^{2.39})$
2	$O(n^{2.05})$	$O(n^{1.86})$
3	$O(n^{1.69})$	$O(n^{1.58})$

Note that there is no need to analyze the computation of the best-case max-flow problem because it can be computed much faster, since it does not involve bandwidth computations for the vertex costs.

V. SEARCH PROBLEM

For the search problem, each vertex in a graph is assigned some reward, generally based on the probability of finding a desired object, and each edge is assigned a cost, generally relating to travel time or energy spent. For the purposes of hierarchical decomposition, we also assign vertex costs. In solving the problem, one would like to find the path that maximizes the reward subject to the cost constraint.

Data Directed connected *graph* $G := (V, E)$; *edge cost* function $\eta : E \to [0, \infty)$; *vertex reward* function $r : V \to [0, \infty)$; *vertex cost* function $\nu : V \to [0, \infty)$; cost bound $L > 0$.

Search Path A *path in* G starting at $v_{\text{init}} \in V$ is a sequence of vertices

$$p := (v_1 := v_{\text{init}}, v_2, \ldots, v_{f-1}, v_f), \quad (v_i, v_{i+1}) \in E.$$

The *path-cost* is given by

$$C(p) := \sum_{i=1}^{f-1} \eta(v_i, v_{i+1}) + \sum_{i=1}^{f} \nu(v_i)$$

and the *path-reward* is given by

$$R(p) := \sum_{v \in p} r(v), \tag{6}$$

where the sum in (6) is taken with no repetitions, that is, if a vertex appears in p more than once, it is only included in the summation once. This represents the fact that the reward of a vertex can only be collected once.

Objective Let the cost bound $L > 0$ be given. For each vertex v_{init}, compute the path p^* that maximizes the path-reward $R(p)$ subject to the path-cost constraint $C(p) \leq L$. The maximum is denoted by $J^*_G(v_{\text{init}}; L)$ and is called the *maximum reward from* v_{init}. Let $J^*_G(L)$ denote the maximum reward from all possible $v_{\text{init}} \in V$.

A. Worst-case meta-graph search

Let $\bar{G}_{\text{worst}} := (\bar{V}, \bar{E})$ be a meta-graph of G with edge cost function defined by

$$\bar{\eta}_{\text{worst}}(\bar{v}, \bar{v}') = \max_{v \in \bar{v}, v' \in \bar{v}'} C^*[v, v'], \qquad \forall \bar{v}, \bar{v}' \in \bar{V}. \tag{7}$$

where $C^*[v, v']$ is the minimum cost of going from v to v'.

Let $l : \bar{V} \to [0, \infty)$ be any assignment of cost bounds to the set of meta-vertices (the specific assignment is up to the user). The vertex reward function is generated by solving search problems on each of the subgraphs $G|\bar{v}$ with cost bounded by $l(\bar{v})$:

$$\bar{r}_{\text{worst}}(\bar{v}) = J^*_{G|\bar{v}}(l(\bar{v})), \qquad \forall \bar{v} \in \bar{V}.$$

The setup of the search problem differs slightly from the previous two in that it requires the user to allocate a cost bound over the partitions of the graph. This is necessary so that the search problem on the subgraphs $G|\bar{v}$ is identical to the search problem on $\bar{G}$. The sub-problems can then be solved using the same worst-case meta-graph search method, creating a multi-level hierarchical decomposition of the problem. The choice of how to allocate the cost bound is left for future research. If information is known about which regions are likely to contain the most reward, this should be incorporated into the cost bound allocation.

Let p^*_i denote the search path that generates the maximum reward for each $\bar{v}_i$. Now assign the path-cost incurred in each of these sub-problems to the meta-vertex cost function as $\bar{\nu}_{\text{worst}}(\bar{v}_i) = C(p^*_i)$.

B. Best-case meta-graph search

Let $\bar{G}_{\text{best}} := (\bar{V}, \bar{E})$ be a meta-graph of G with edge cost function and vertex reward function defined by

$$\bar{\eta}_{\text{best}}(\bar{v}, \bar{v}') = \min_{v \in \bar{v}, v' \in \bar{v}'} C^*[v, v'], \qquad \forall \bar{v}, \bar{v}' \in \bar{V},$$

and $\bar{r}_{\text{best}}(\bar{v}) = \sum_{v \in \bar{v}} r(v)$. In the best case, we assume that all reward contained in a meta-vertex is collected immediately upon entry. The vertex cost function is defined by $\bar{\nu}_{\text{best}}(\bar{v}) = 0$.

Theorem 3 *For every partition $\bar{V}$ of G*

$$J^*_{\bar{G}_{\text{worst}}}(L) \leq J^*_G(L) \leq J^*_{\bar{G}_{\text{best}}}(L) \tag{8}$$

The construction of the lower bound contains a procedure for generating an approximate optimal search path on G whose total reward lies between $J^*_{\bar{G}_{\text{worst}}}(L)$ and $J^*_G(L)$.

Proof: To verify the lower bound, we construct a feasible search path p on G from the optimal worst-case path $\bar{p}^* = (\bar{v}_1, \bar{v}_2, ...\bar{v}_f)$ on $\bar{G}_{\text{worst}}$ that generates the reward $J^*_{\bar{G}_{\text{worst}}}(L)$. Since the worst-case solution involves solving the search problem on each partition subgraph, simply connect these sub-paths sequentially as they appear in the solution. Let $p^*_j = (v^j_1, v^j_2, \ldots, v^j_{f_j})$ denote the optimal search path on the subgraph $\bar{v}_j$. Now, let $\hat{p}$ be the connection of all $p_{\bar{v}_j}$ by the shortest path between each $v^j_{f_j}$ and v^{j+1}_1. Clearly, $C(\hat{p}) \leq C(\bar{p}^*)$ since their costs within meta-vertices are equal, and the cost between path segments p^*_j on $\hat{p}$ is no greater than the cost between meta-vertices in $\bar{p}^*$ because of (7). The reward $R(\hat{p})$ is bounded below by $J^*_{\bar{G}_{\text{worst}}}(L)$ because all reward collected by $\bar{p}^*$ is also collected by p. By the optimality of p^*, $R(\hat{p})$ must be bounded above by $J^*_G(L)$.

To verify the upper bound, we construct a path in $\bar{G}_{\text{best}}$ out of the optimal search path $p^* = (v_1, v_2, \ldots, v_f)$ that generates reward $J^*_G(L)$. Let $\tilde{p}$ consist of the sequence of meta-vertices $\bar{v}_x$ such that $p^*(i) \in \bar{v}_x$ for $i = 1, \ldots, f$, with no consecutive repetitions. The maximum cost of this path is $C(p^*)$ and the minimum reward is $J^*_G(L)$. The reward is also bounded above by $J^*_{\bar{G}_{\text{best}}}(L)$ since $\tilde{p}$ is a path in $\bar{G}_{\text{best}}$. ■

C. Computation

The search problem is computationally the worst of the three problems considered in this paper, because it is NP-hard even when significant structure is imposed on the graph [10]. For an exhaustive enumeration of all possible search paths on a graph with n vertices, the computation can be as high as $O(n!)$. Assuming this worst-case complexity, the search on a meta-graph partitioned into k subgraphs will have complexity $O(k! + k\,(\frac{n}{k})!)$. One can numerically solve for the optimal value of k from this expression.

While the meta-graph search problem is still NP-hard, its computation is a significant reduction on that of the full graph search problem. Let $C_{\text{meta}_1}(n)$ be the estimated computation of worst-case meta-graph search problem with one hierarchical level, and let $C_{\text{full}}(n)$ be the estimated computation of the full graph search problem. Figure 2 is a semi-log plot of $\log\left(\frac{C_{\text{meta}_1}(n)}{C_{\text{full}}(n)}\right)$ vs. the number of vertices n. The linear relationship shows that the computational reduction is exponential in n.

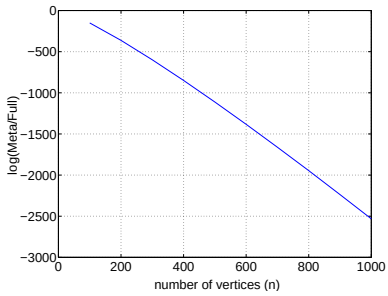
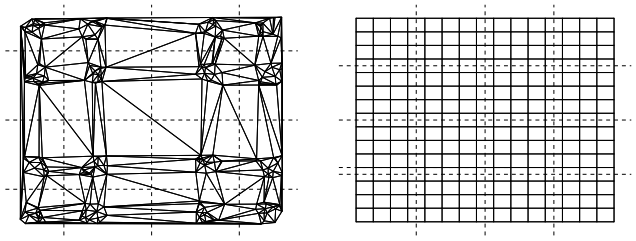


Fig. 2. Computational reduction of the worst-case meta-graph search problem over the full graph search problem.

VI. Test Results

This section presents some results of the meta-shortest path approximations on two test graphs: one which has a structure that naturally lends itself to partitioning, and a lattice graph that has no natural partition. One would expect the approximations to be good for the first graph and somewhat worse for the second graph.



(a) Delaunay graph of 16 vertex groups

(b) 16×16 lattice graph.

Fig. 3. These two test graphs were used to evaluate the diameter approximation

A. Grouped Delaunay Graph–256 vertices

Figure 3(a) was created by randomly distributing vertices over 16 1 unit $\times$ 1 unit regions centered in a block pattern and generating a Delaunay graph over this vertex set. The dashed lines indicate the partition used for these tests.

B. 16×16 Lattice Graph

Lattice structure graphs are probably the worst case for this approximation since there is no advantageous way to partition them. Figure 3(b) shows the 16×16 lattice used in these tests. The dashed lines indicate the partition that was used.

C. Diameter Approximation Results

The diameter is an appropriate metric for the tightness of the meta-shortest path bounds, because for each hierarchical level below the top, subgraph diameters are computed to assign the meta-vertex costs. Table III shows the results for best-case, worst-case, approximate, and actual diameters for each test graph partitioned into 16 vertex groups. The approximate diameter is computed using the procedure outlined in the proof of Theorem 1, and will always lie between the actual and worst-case diameters.

TABLE III
Results of Diameter Approximation for 16 partitions

	Best-case	Actual	Approximate	Worst-case
Grouped	9.0	14.1	14.5	15.1
Lattice	6	30	30	48

D. Discussion

As expected, the worst-case bounds are fairly tight for the grouped graph, but worse for the lattice graph. Although the approximation gets the correct diameter for the lattice graph, there is a large uncertainty. This shows that the meta-shortest path approximation is best suited to graphs with a clustered structure. For any graph, it is important to carefully choose a partition that will produce good results.

VII. Conclusions

We have introduced methods of decomposing three graph optimization problems to achieve upper and lower bounds with much less computation than would be required to solve the complete problems. Additionally, each problem is formulated in such a way that allows for multiple levels of hierarchical decomposition. As the number of levels increases, the computation decreases further at the expense of looser bounds on the optimal solution. Tests on the meta-shortest path method show good diameter approximation for a clustered graph. However, the bounds are looser if the graph is a lattice having no natural partition. The choice of which partition to use depends on the the graph structure and the individual problem. There may be an obvious partition as in the grouped graph of Figure 3(a), but this may not often be the case. Algorithms to generate graph partitions for which the upper and lower bounds are close is a good topic for future work. Another topic open for future research is the choice of how to allocate the cost bounds for the worst-case meta-graph search problem.

References

[1] D. R. Karger, D. Koller, and S. J. Phillips, "Finding the hidden path: Time bounds for all-pairs shortest paths," *SIAM J. Comput.*, vol. 22, pp 1199-1217, 1993.

[2] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," *Journal of the ACM*, vol. 35, pp 921-940, 1988.

[3] R. E. Gomory and Z. C. Hu, "Multi-terminal network flows," *SIAM J. Appl. Math.*, vol. 9, pp 551-570, 1961.

[4] J. A. Fernandez and J. Gonzalez, "Hierarchical Path Search for Mobile Robot Path Planning," *Proc. IEEE Int'l Conf. Robotics and Automation*, 1998.

[5] N. Jing, Y. Huang, and E. Rundensteiner, "Hierarchical Optimization of Optimal Path Finding for Transportation Applications," *Proc. Fifth Int'l Conf. Info. and Know. Mgmt.,* pp 261-268, 1996.

[6] R. C. Holte, C. Drummond, M. B. Perez, R. M. Zimmer, and A. J. MacDonald, "Searching with Abstractions: A Unifying Framework and New High-Performance Algorithm," *Proc. 10th Canadian Conf. Artificial Intelligence*, pp 263-270, 1994.

[7] G. Shen and P. E. Caines, "Hierarchically Accelerated Dynamic Programming for Finite-State Machines," *IEEE Transactions on Automatic Control*, vol. 47, no. 2, pp 271-283, 2002.

[8] H. E. Romeijn and R. L. Smith, "Parallel Algorithms for Solving Aggregated Shortest Path Problems," *Computers and Operations Research*, vol. 26, Issue 10-11, pp 941-953, 1999.

[9] S. Bohacek, J. Hespanha, C. Lim, and K. Obraczka. "Hierarchical Max-Flow Routing," *Submitted to publication,* Feb. 2005.

[10] B. DasGupta, J. Hespanha, and E. Sontag, "Computational Complexity of Honey-pot Searching with Local Sensory Information," *Proc. of the 2004 Amer. Contr. Conf.,* June 2004.

[11] J. Riehl and J. Hespanha, "Fractal Graph Optimization Algorithms," *Technical Report*, August 2005.