SIAM J. OPTIM.
Vol. 20, No. 1, pp. 54–77

CONVERGENT NETWORK APPROXIMATION FOR THE CONTINUOUS EUCLIDEAN LENGTH CONSTRAINED MINIMUM COST PATH PROBLEM*

RANGA MUHANDIRAMGE†, NATASHIA BOLAND‡, AND SONG WANG§

Abstract. In many path-planning situations we would like to find a path of constrained Euclidean length in $\mathbb{R}^2$ that minimizes a line integral. We call this the Continuous Length-Constrained Minimum Cost Path Problem (C-LCMCPP). Generally, this will be a nonconvex optimization problem, for which continuous approaches ensure only locally optimal solutions. However, network discretizations yield weight constrained network shortest path problems (WCSPPs), which can in practice be solved to global optimality, even for large networks; we can readily find a *globally* optimal solution to an *approximation* of the C-LCMCPP. Solutions to these WCSPPs yield feasible solutions and hence *upper bounds*. We show how networks can be constructed, and a WCSPP in these networks formulated, so that the solutions provide *lower bounds* on the global optimum of the continuous problem. We give a general convergence scheme for our network discretizations and use it to prove that both the upper and lower bounds so generated converge to the global optimum of the C-LCMCPP, as the network discretization is refined. Our approach provides a computable lower bound formula (of course, the upper bounds are readily computable). We give computational results showing the lower bound formula in practice, and compare the effectiveness of our network construction technique with that of standard grid-based approaches in generating good quality solutions. We find that for the same computational effort, we are able to find better quality solutions, particularly when the length constraint is tighter.

Key words. constrained shortest paths, Eikonal equations, optimal trajectories, network optimization, global optimization

AMS subject classifications. 65K10, 90B10, 90C35

DOI. 10.1137/070695356

1. Introduction.

Path-planning problems in networks have been widely studied, with numerous applications in diverse fields such as telecommunications routing (see, for example, [10]) and airline scheduling (see, for example, [1]). Continuous path-planning problems also arise in varied contexts, such as robotics [14], highway construction [9, 4], and military path planning [8, 12, 19, 18, 2, 21]. Our interest was motivated by the problem of planning a path through a naval minefield, which led us to formulate a path-planning problem in 2D Euclidean space, having the following form.

Let $F : \mathbb{R}^2 \mapsto [0, \infty)$ be a nonnegative, Hölder continuous function defined on a compact, convex domain $\Omega \subset \mathbb{R}^2$. We refer to F as the *cost function*. In a military application F may, for example, represent the risk distribution in a spatial domain of detecting an aircraft by radar, or of a ship detonating a mine in a naval minefield. Such applications are likely to yield functions F that are nonconvex, and indeed multimodal (see, for example, [19, 2, 21]). Note that the class of Hölder continu-

*Received by the editors June 25, 2007; accepted for publication (in revised form) October 29, 2008; published electronically March 13, 2009. Funding for this work was provided by the Hackett Foundation and the Defence Science and Technology Organisation.

http://www.siam.org/journals/siopt/20-1/69535.html

†School of Information Technology, Monash University, Caufield East, 3145, Melbourne, Australia (Ranga.Mulhandiramge@infotech.monash.edu.au).

‡Department of Mathematics and Statistics, University of Melbourne, Parkville, 3010, Melbourne, Australia (natashia@unimelb.edu.au).

§Department of Mathematics and Statistics, University of Western Australia, 35 Stirling Highway, Crawley, 6009, Perth, Australia (swang@maths.uwa.edu.au).

ous functions exclude functions that have discontinuities or singularities. However, many cost functions used in the literature satisfy the Hölder condition, for example, probability of detection for submarines used by Caccetta et al. [2] and the cost functions relating to weather disruptions and the reliability of the weather forecast used by Mitchell and Sastry [16]. The total cost for a path is obtained by integrating F along that path. While minimizing the total cost, we also wish to limit the Euclidean length of our path: This can be used to model a practical time or fuel constraint. The Continuous Length-Constrained Minimum Cost Path Problem (C-LCMCPP) can be stated as follows: Find a piecewise differentiable curve in Ω between a given start point a and end point b that minimizes the line integral of F subject to the constraint that the Euclidean length of the curve is less than or equal a prescribed value $\bar{L}$.

To be more precise, let $\mathcal{C}([0,1],\Omega)$ denote all piecewise differentiable curves parameterized by $s \in [0,1]$ such that for any $p \in \mathcal{C}([0,1],\Omega)$ we have $p(s) \in \Omega$ for all $s \in [0,1]$. Define

$$\Gamma = \{p \in \mathcal{C}([0,1],\Omega) : p(0) = a, p(1) = b\}.$$

Then the C-LCMCPP can be stated mathematically as:

$$\begin{aligned} \min_{p\in\Gamma} J[p] &= \int_0^1 F(p(s))||p'(s)||ds \\ s.t.\ Eu[p] &= \int_0^1 ||p'(s)||ds \le \bar{L}, \end{aligned} \tag{1.1}$$

where $||.||$ denotes the Euclidean norm. The piecewise differentiability of the paths in Γ make the path integrals in (1.1) well defined. An instance of the C-LCMCPP takes the form $(\Omega, F, \bar{L}, a, b)$. We are interested in the general case, with no further assumptions on F.

The C-LCMCPP could be approached directly as a continuous problem, but has more commonly been tackled via network discretization. We discuss the former approach first. The two principal continuous approaches that are applicable are (i) variational techniques, such as solution of the Euler–Lagrange equation, or (ii) solution of the Hamilton–Jacobi–Bellman equation. The former are discussed, for example, by Zabarankin et al. [21] and Caccetta et al. [2]. However, variational techniques can only ensure locally optimal solutions (see, for example, introductory remarks in Tsitsiklis [20], and references therein). Globally optimal solutions can, in principle, be obtained via the Hamilton–Jacobi–Bellman (HJB) equation (see [20] for an excellent exposition). These have been extensively explored in the case without the Euclidean length constraint, which we refer to as the C-MCPP. In this case, the problem is equivalent to solving the Eikonal equation,

$$||\nabla\tau|| = F(x,y), \tag{1.2}$$

where τ, the value function, is the time of arrival of a disturbance propagating from an initial set on which $\tau = 0$, travelling at a given "slowness" (the inverse of the speed of propagation), F, at each point. Numerical approaches to solution of this problem all involve discretization, and there have been several schemes proposed for which convergence to a global optimum has been proved; we believe Cristiani and Falcone [5] provide the most recent instance, and give a comprehensive review of previous approaches. The method in [5] is shown to converge under the relatively mild

assumption that the speed function (the pointwise reciprocal of our F) is Lipschitz continuous.

For the problem of interest to us, the C-MCPP *with* the length constraint, we believe no similar methods are known. Indeed, the only approach to constrained problems via the Eikonal equation that we are aware of is that of Mitchell and Sastry [16]. Their interest is finding paths for aircraft that minimize fuel consumption (i.e., path length) subject to constraints on the probability of encountering bad weather with a penalty relating to the reliability of the weather forecast in different regions. To handle constraints, they recast them as objectives, incorporating them in the objective function with a multiplier; this returns the problem to one of solving an Eikonal equation, where now the cost function, or speed, incorporates terms related to the constraints. Their method samples from possible multipliers, and so samples from the set of Pareto-optimal solutions for the multiobjective problem. As is noted in [16], this will not necessarily yield an optimal solution to the constrained problem.

We now discuss approaches based on network discretization. In such approaches, the spatial domain Ω is discretized, and represented by a set of points, including a and b, which are used as vertices in a network. The arcs in the network restrict the path to travel only between pairs of vertices connected by an arc, and the cost of each arc is taken to be the integral of F along the straight line between the two endpoints of the arc. The problem of finding a minimum cost path in the network from a to b is now a standard network shortest path problem, which is easily solvable, with techniques such as Dijkstra's algorithm [6] or the A* algorithm [11], to give globally optimal solutions. Problems with the additional Euclidean length constraint take the form of a Weight-Constrained Shortest Path Problem (WCSPP) in a network, which is also now very well solved for practical purposes, for example, using the recent approaches of Dumitrescu and Boland [7], Carlyle and Wood [3], or Muhandiramge and Boland [17]. In either case, solving the network shortest path problem provides a feasible solution to the continuous problem, and so yields an upper bound on its value.

For further detail of how continuous problems, particularly those arising from path planning in a threat environment, can be modeled as network path problems, we refer the reader to the paper of Zabarankin et al. [21], which gives an excellent exposition. Zabarankin et al. [21] also derive analytic solutions for the case of a single point threat, and so can demonstrate computationally that in such cases the upper bounds generated from network discretizations are very close to the exact global optimum. Most work along these lines rests with constructing the network discretization and solving the corresponding network path problem: The focus is on modeling other practical complications, such as curvature constraints. For example, Piatko et al. [19, 18] discretize with points on a square grid, with arcs from each point to either its four, or eight, nearest neighbors. Similar networks were used by Fahlen [8], Caccetta et al. [2], and Zabarankin et al. [21], although [8] and [21] both describe using sixteen neighbors in the 2D case, and [21] further considers the 3D case. Curvature (turning angle) constraints are considered in [8, 21], while [2] permits variable speeds for path traversal, selected from a finite set of possible speeds.

By contrast, Kim and Hespanha [12], in work on an anisotropic form of the problem (cost depends on direction) without the length constraint, focus on finding network constructions that provide better approximations. They develop a novel network construction method that they call "honeycomb" sampling. This method selects points at random from the spatial domain, according to a probability distribution that ensures either the points are close together, or that a term related to gradients of the cost function is small. The Voronoi diagram for these points is then constructed, and

nodes on the network are sampled from the edges of the Voronoi diagram. The honeycomb sampling is compared computationally with (i) a network with nodes selected uniformly at random from the spatial domain, and (ii) a network with nodes selected at random according to a probability distribution based on cost function gradients. Kim and Hespanha [12] report average reductions in the cost of the network paths found using honeycomb sampling of around 7.5% over the uniform sampling networks and around 11% over the gradient-based sampling method. Unfortunately, [12] does not say how their network nodes were connected (they don't define the arcs), so it is difficult to assess the relative computational effort for these approaches.

Network discretizations have also been explored by authors in contexts other than that of the C-LCMCPP or C-MCPP. Kimmel and Kiryati [13] used a grid network and local refinement procedure to find the minimum length path on a underlying 3D surface given a digitization of the surface. First, the surface was represented by a graph with a node for each surface voxel and an edge from each node to all the surface voxel nodes up to one unit away in each direction (a total of 26 possible different directions). Rather than weight these links by their length, they weighted them using a path length estimator which gives an unbiased estimate of the path length on the actual underlying surface. They then found the shortest path in this network using a standard network shortest path algorithm. Since grid networks suffer from discretization bias, [13] also used a curve shortening flow method [15] to shorten the path to a local optimum. Caccetta et al. [2] similarly combine an initial discretization step with a subsequent local optimization step based on variational techniques. They use a standard grid network, solve the corresponding network problem approximately, but then take the resulting path as an initial point for an optimal control solver, to derive a locally optimal solution.

As far as we are aware, the only work that considers the issue of how far the solution to the network discretization problem is from that of the original continuous problem, or considers the possibility of convergence to the globally optimal solution as the network is refined, is that of Kim and Hespanha [12]. As mentioned earlier, they tackle an anisotropic problem, and do not apply a length constraint. For this case, they provide a lower bound formula. Unfortunately, their formula involves a set of points in the spatial domain that they prove to exist, but which they don't explicitly show how to construct. They simply require the set to be "sufficiently dense". Thus they cannot readily use their formula to compute a lower bound from a path found in a given network. Furthermore, although their network construction is motivated by the theory they provide, it is not explicitly proved to converge to the globally optimal solution. Indeed, since their construction relies on randomized sampling, such a proof would have to include some kind of "almost surely" condition.

In this paper, we give a general scheme for convergence of network discretizations. With this scheme, we show that if we solve the corresponding WCSPP with path lengths constrained to $\bar{L}(1+\gamma)$, where γ depends on the network construction, then we can compute a lower bound on the global optimum of the C-LCMCPP. We prove furthermore that this bound converges to the global optimum as the network is refined in a way described later. We also prove that the solution to the WCSPP with path length constrained to $\bar{L}$ (an upper bound on the global optimum of the C-LCMCPP) also converges to the global optimum as the network is refined.

This is, of course, of theoretical interest, but from a practical point of view, we still need to construct good network discretizations. An advantage of network solution methods that make them useful for nonconvex problems is they find global optima within the network. However how accurately the network solution reflects

the continuous solution depends greatly on the structure of the network used. One point that is not hard to see is that the standard grid networks, with arcs only connecting points to a handful of their nearest neighbors, cannot, in general, converge to globally optimal solutions of the C-LCMCPP. With such networks, the set of gradients available to the network path is simply not rich enough to ensure it can well approximate the optimal continuous path; grid networks suffer from significant discretization bias. A complete network on a set of grid points would suffice, but a complete network on a fine grid has an enormous number of arcs, and even efficient shortest path algorithms are unlikely to be practical if we attempt to use complete networks. (We note that in the unconstrained case of the C-MCPP, the method of Cristiani and Falcone [5] implicitly considers a much larger set of tangent directions by updating node values using the multiple neighboring node values simultaneously. This allows them to prove convergence.)

Thus the challenge is to structure a network that is "just right". It needs to be rich enough to well approximate any optimal path, but not so dense as to make solution of the network problems impractical. By structuring our network carefully, we can overcome the discretization effects with a purely network method, avoiding the need for a local refinement procedure. We can also guarantee that our solution will converge to the true optimum as we refine our network. We have met this challenge with what we call a "cellular" network construction, based on triangular tessellation of the spatial domain, and hexagonal cells. This network is sparse, while still meeting the conditions for convergence.

We give the results of numerical computations, showing the effects of refining the network discretization on the lower and upper bounds computed. We also compare the upper bounds found with those found using the standard grid approach, using computational effort. This shows that the cellular network gives better solutions, particularly when the length constraint is tighter.

Thus our contribution in this work is what we believe is the first approach to a constrained continuous minimum cost path problem that is proved to converge to the globally optimal solution, under mild assumptions on the cost function. We also provide computable lower bounds, and a network construction that is sparse, while still providing better approximations to the continuous solution than standard approaches.

The paper is structured as follows: First we formalize the concept of using a network to approximate the C-LCMCPP; next we outline the properties of network that produce a convergent solution; and lastly we create a method of constructing networks with these properties and give numerical results.

2. Network formulation. To solve the C-LCMCPP using a network formulation we create a network $G = (V, A)$ consisting of nodes and directed edges in Ω such that nodes are located at the start point a and end point b and at least one *network path* exists that connects a and b. A network path p from node v_0 to node v_m in G is a sequence of arcs $p = ((v_0, v_1), (v_1, v_2), \ldots, (v_{m-1}, v_m))$ such that $(v_{k-1}, v_k) \in A$ for all $k \in \{1, \ldots, m\}$. For convenience, we will assume that the graph has a unique directed edge between any ordered pair of nodes so p can be equivalently written $p = (v_0, v_1, \ldots, v_{m-1}, v_m)$.

As the nodes in our network are also points in the Euclidean plane, we treat them both as abstract network nodes, e.g., $v \in V$ and also directly as coordinate in 2-space, e.g., $||v_i - v_{i+1}||$ and $v \in \Omega$. The context in which a node is mentioned indicates in what capacity it is to be treated.

We assign each edge a *cost* which is the line integral of F along the edge and a *weight* which is the Euclidean length of the edge. We then solve the corresponding weight constrained shortest path problem (WCSPP): Finding a network path from a to b that minimizes the sum of the costs of the edges in the path while keeping the total length of the path less than or equal to a given weight limit.

The quality of our network approximation depends a great deal on the structure of the network. In particular, we would like the difference between the optimal objective function value for the C-LCMCPP and the corresponding WCSPP to be as small as possible. We would also like this difference to shrink to zero as we refine our network. We formalize the network design into the concept of a *network construction* as follows.

DEFINITION 2.1. *A* network construction $\mathcal{G}$ *is a method that, given an instance* I *of the C-LCMCPP and a finite vector of real parameters* P *from a parameter domain* S*, will produce a finite directed network* $\mathcal{G}(I, P)$ *which includes* a *and* b *as nodes and in which a network path from* a *to* b *exists.*

The important point is that a network construction can take many different vectors of parameter values and thus produce many related networks for a given instance, e.g., many different grid spacings for a grid network. We would like to have a network construction that, given the right series of parameters, produces a series of networks whose WCSPP optimal objective function values converge to the objective function value of the C-LCMCPP. This is the focus of the next definition.

DEFINITION 2.2. *A* convergent network construction $\mathcal{G}$ *is a network construction for which for any instance* I *of the C-LCMCPP we can find a sequence of parameter sets* $(P_1, P_2, \dots)$ *with* $P_k \in S$ *for* $k \in \mathbb{Z}^+$ *such that the difference between the objective function value of the solution to* I *and the approximate WCSPP solution using the network* $\mathcal{G}(I, P_n)$ *goes to zero as* n *goes to infinity. The WCSPPs may use a different weight limit to the C-LCMCPP.*

In the following section, we formulate a convergent network construction and in the process obtain a calculable lower bound on the cost of the optimal solution of the C-LCMCPP.

3. $(\delta, \epsilon, \kappa)$-approximation networks. In this section, we outline the properties of a network that allow us to relate the solution of the WCSPP over the network to the corresponding C-LCMCPP. Such properties are given by Definition 3.1, and we call a network that satisfies these properties a $(\delta, \epsilon, \kappa)$-approximation network. We will give an example later of how such a network is constructed, but for now we concentrate on proving convergence using the abstract properties of the network without the distraction of outlining the full network construction method.

The motivation behind the definition is that to approximate a continuous path integral using a network path, we would ideally like the end points of the edges to be on the path, as standard for the Riemann sum definition of a path integral. In our case, however, we want to be able to approximate the integral for any reasonable path in our space using a finite network; thus we cannot guarantee that the approximating points will be directly on the path. Therefore, the best we can do is guarantee that the approximating points are within some distance of the path.

To make this guarantee, for any path $p \in \Gamma$, we have a sequence $p_G = (v_0, \dots, v_N)$, with $v_k \in V$ for $k \in \{0, 1, \dots, N\}$, which forms the network approximation to the path. To relate the network path to the continuous path it approximates, we have the corresponding sequence $(p(s_0), p(s_1), p(s_2), \dots, p(s_N))$ of points on the path p, in which each point, $p(s_k)$, is at most ϵ away from the nearest corresponding node, v_k, for each $k = 0, \dots, N$. The distance between these points on the path is bounded

below by δ and above by $\kappa\delta$. The distance $\kappa\delta$ corresponds to the maximum on the distance between points in a Riemann sum.

DEFINITION 3.1. *A $(\delta, \epsilon, \kappa)$-approximation network $G = (V, A)$ with $\delta > 0, \epsilon > 0, \kappa > 1 \in \mathbb{R}$ for an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP has the following properties:*

1. *the set V contains a and b;*
2. *for each node $v \in V \setminus \{b\}$ there is a closed, connected region $R_v \subseteq \Omega$ such that each R_v can be enclosed by a circle of radius $\kappa\delta$; and*
3. *for each $p \in \Gamma$ with $Eu[p] \leq \bar{L}$ there is an ordered sequence of points on the path p given by $(a = p(s_0), p(s_1), \ldots, p(s_{N-1}), p(s_N) = b)$ with $0 = s_0 < s_1 < \ldots < s_{N-1} < s_N = 1$, and a path $(a = v_0, v_1, \ldots, v_{N-1}, v_N = b)$ in the network G, i.e., a sequence with $v_k \in V$ and $(v_{k-1}, v_k) \in A$ for all $k \in \{1, \ldots, N\}$, such that:*
   (a) *$||p(s_k) - p(s_{k-1})|| \geq \delta$, for all $k \in \{1, \ldots, N\}$,*
   (b) *$||v_k - p(s_k)|| \leq \epsilon$, for all $k \in \{0, \ldots, N\}$, and*
   (c) *$p(s) \in R_{v_k}$ for all $s \in [s_k, s_{k+1}]$ and $\alpha v_k + (1-\alpha)v_{k+1} \in R_{v_k}$ for all $\alpha \in [0, 1]$, for each $k \in \{0, \ldots, N-1\}$.*

3.1. Length relationship. Consider a $(\delta, \epsilon, \kappa)$-approximation network for an instance I of the C-LCMCPP. For an optimal solution p^* of I, we have the sequence of points $(p^*(s_0), \ldots, p^*(s_N))$ on the path guaranteed by Definition 3.1 and clearly

$$Eu[p^*] \geq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})||. \tag{3.1}$$

We would now like to find, for any path $p \in \Gamma$, the relationship between the Euclidean length of the piecewise linear path formed by $(p(s_0), p(s_1), p(s_2), \ldots, p(s_N))$, and that formed by the corresponding network path $(v_0, v_1, v_2, \ldots, v_N)$ satisfying the conditions of Definition 3.1.

LEMMA 3.2. *Any $(\delta, \epsilon, \kappa)$-approximation network $G = (V, A)$ for an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP will have the property that for any path $p \in \Gamma$, there is a sequence of points on the path $(p(s_0), p(s_1), p(s_2), \ldots, p(s_N))$ and a sequence of nodes $(v_0, v_1, v_2, \ldots, v_N)$ with $0 = s_0 < s_1 < \ldots < s_{N-1} < s_N = 1$ and $v_k \in V$ for all $k \in \{0, \ldots, N\}$, such that $||v_k - v_{k-1}|| \leq ||p(s_k) - p(s_{k-1})||(1 + \gamma)$ for all $k \in \{1, \ldots, N\}$ and $\gamma \in \Phi_G$ where*

$$\Phi_G = \left[c_1 \frac{\epsilon}{\delta} + c_2 \frac{\epsilon^2}{\delta^2}, \infty \right) \tag{3.2}$$

for some $c_1, c_2 \in [0, 2]$ independent of p.

Proof. Let $G = (V, A)$ be a $(\delta, \epsilon, \kappa)$-approximation network for an instance I of the C-LCMCPP. For any path $p \in \Gamma$, let $(p(s_0),\ p(s_1)\ ,\ p(s_2),\ \ldots, p(s_N))$ and $(v_0, v_1, v_2, \ldots, v_N)$ be the sequences guaranteed to exist by Definition 3.1. Let $L_k = ||v_k - v_{k-1}||$, $\delta_k = ||p(s_k) - p(s_{k-1})||$ for $k \in \{1, \ldots, N\}$, and $\epsilon_k = ||p(s_k) - v_k||$ for $k \in \{0, \ldots, N\}$.

For $k \in \{1, \ldots, N\}$, if $p(s_{k-1}) \neq v_{k-1}$, let α_k be the angle defined by $p(s_k)$, $p(s_{k-1})$ and v_{k-1} measured anticlockwise from the segment $\{p(s_{k-1}), p(s_k)\}$, and if $p(s_k) \neq v_k$, let β_k be the angle defined by $p(s_{k-1})$, $p(s_k)$ and v_k also measured anticlockwise from the segment $\{p(s_{k-1}), p(s_k)\}$. If $p(s_{k-1}) = v_{k-1}$, i.e., $\epsilon_{k-1} = 0$, set α_k to 0 and similarly if $p(s_k) = v_k$, i.e., $\epsilon_k = 0$, set β_k to 0.

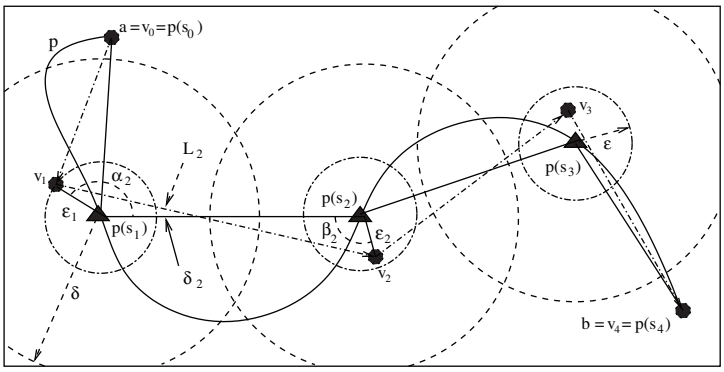


FIG. 3.1. *Representation of a path p and its network approximation. Parts of the diagram applicable to Lemma* 3.2 *for* $k = 2$ *are labelled. The small circles have radius* ϵ *and the large circles have radius* δ.

Then from Figure 3.1, we can deduce, by decomposing the segments $\{p(s_{k-1}), v_{k-1}\}$ and $\{p(s_k), v_k\}$ into components parallel and perpendicular to $\{p(s_{k-1}), p(s_k)\}$ and using Pythagoras, that

$$L_k^2 = (\delta_k - \epsilon_{k-1}\cos(\alpha_k) - \epsilon_k \cos(\beta_k))^2 + (\epsilon_{k-1}\sin(\alpha_k) + \epsilon_k \sin(\beta_k))^2.$$

To get an upper bound on how much longer L_k could be compared to δ_k, we take the absolute value of the component contributions. We also note that $0 \le \epsilon_{k-1}, \epsilon_k \le \epsilon$ due to Condition 3(b) of Definition 3.1. Using this, we then get

$$L_k^2 \le (\delta_k + \epsilon|\cos(\alpha_k)| + \epsilon|\cos(\beta_k)|)^2 + (\epsilon|\sin(\alpha_k)| + \epsilon|\sin(\beta_k)|)^2.$$

The effect of the absolute value signs on the sine and cosine function can be replicated if we make the following transformation which keeps the angles in the range $[0, \frac{\pi}{2}]$:

$$\alpha^k = \begin{cases} \alpha_k & \alpha_k \in [0, \frac{\pi}{2}], \\ \pi - \alpha_k & \alpha_k \in [\frac{\pi}{2}, \pi], \\ \alpha_k - \pi & \alpha_k \in [\pi, \frac{3\pi}{2}], \\ 2\pi - \alpha_k & \alpha_k \in [\frac{3\pi}{2}, 2\pi]. \end{cases}$$

Using the same transformation function for β_k, we then expand and simplify using trigonometric identities and then estimate L_k as follows:

$$\begin{aligned} L_k^2 &\le \delta_k^2 + 2\epsilon^2 + 2\delta_k\epsilon(\cos(\alpha^k) + \cos(\beta^k)) + 2\epsilon^2\cos(\alpha^k - \beta^k) \\ \implies L_k &\le \sqrt{\delta_k^2 + 2\epsilon^2 + 2\delta_k\epsilon(\cos(\alpha^k) + \cos(\beta^k)) + 2\epsilon^2\cos(\alpha^k - \beta^k)} \\ &= \delta_k\sqrt{1 + 2\left(\frac{\epsilon}{\delta_k}(\cos(\alpha^k) + \cos(\beta^k)) + \frac{\epsilon^2}{\delta_k^2}(1 + \cos(\alpha^k - \beta^k))\right)} \\ &\le \delta_k\sqrt{1 + 2\left(\frac{\epsilon}{\delta}(\cos(\alpha^k) + \cos(\beta^k)) + \frac{\epsilon^2}{\delta^2}(1 + \cos(\alpha^k - \beta^k))\right)} \\ &\le \delta_k\left(1 + \frac{\epsilon}{\delta}(\cos(\alpha^k) + \cos(\beta^k)) + \frac{\epsilon^2}{\delta^2}(1 + \cos(\alpha^k - \beta^k))\right) \\ &= \delta_k\left(1 + c_1^k\frac{\epsilon}{\delta} + c_2^k\frac{\epsilon^2}{\delta^2}\right), \end{aligned}$$

where we have used $\delta_k \geq \delta$ from Condition 3(a) of Definition 3.1 and the inequality $\sqrt{1+x} \leq 1 + \frac{x}{2}$, for any $x \geq 0$.

In the above, $c_1^k = \cos(\alpha^k) + \cos(\beta^k)$ and $c_2^k = 1 + \cos(\alpha^k - \beta^k)$. As $\alpha^k, \beta^k \in [0, \frac{\pi}{2}]$, this implies both $\cos(\alpha^k), \cos(\beta^k) \in [0, 1]$. Also $(\alpha^k - \beta^k) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ so $\cos(\alpha^k - \beta^k) \in [0, 1]$. Thus it is clear that $c_1^k, c_2^k \in [0, 2]$. We define $c_1(p)$ as $\max_{k\in\{1,\ldots,N\}} c_1^k$ and $c_2(p)$ as $\max_{k\in\{1,\ldots,N\}} c_2^k$ and note that $c_1(p), c_2(p) \in [0, 2]$.

Now, let c_1 and c_2 be the supremum of $c_1(p)$ and $c_2(p)$, respectively, over all paths $p \in \Gamma$. We see that $c_1, c_2 \in [0, 2]$. Thus

$$L_k \leq \delta_k \left(1 + c_1 \frac{\epsilon}{\delta} + c_2 \frac{\epsilon^2}{\delta^2}\right)$$
$$L_k \leq \delta_k(1 + \gamma),$$

where $\gamma \in [c_1 \frac{\epsilon}{\delta} + c_2 \frac{\epsilon^2}{\delta^2}, \infty)$.

If we return to our original definition of L_k and δ_k we get $L_k = ||v_k - v_{k-1}|| \leq ||p(s_k) - p(s_{k-1})||(1+\gamma) = \delta_k(1+\gamma)$ for all $k \in \{1, \ldots, N\}$, where $\gamma \in [c_1 \frac{\epsilon}{\delta} + c_2 \frac{\epsilon^2}{\delta^2}, \infty) = \Phi_G$ for some $c_1, c_2 \in [0, 2]$ independent of p. ∎

COROLLARY 3.3. *Let p^* be an optimal path of an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP and G be a $(\delta, \epsilon, \kappa)$-approximation network for I. Then the network approximation $p_G^* = (v_0, v_1, v_2, \ldots, v_N)$ to p^*, guaranteed to exist by Definition* 3.1*, satisfies $Eu[p_G^*] \leq \bar{L}(1 + \gamma)$ for $\gamma \in \Phi_G$ where Φ_G is defined in* (3.2)*.*

Proof. Let $(s_0, \ldots, s_N)$ be defined for p^* as per Definition 3.1. Then

$$\begin{aligned} Eu[p_G^*] &= \sum_{k=1}^{N} ||v_k - v_{k-1}|| \\ &\leq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})||(1 + \gamma) && \text{by Lemma 3.2} \\ &\leq Eu[p^*](1 + \gamma) && \text{by (3.1).} \end{aligned}$$

Now as p^* is feasible for the instance I of the C-LCMCPP, we have $Eu[p^*] \leq \bar{L}$ so

$$Eu[p_G^*] \leq \bar{L}(1 + \gamma). \quad ∎$$

Corollary 3.3 is important because it tells us that by relaxing the weight constraint in our network by the factor $1 + \gamma$, the network path p_G^* that approximates the continuous optimal solution will be a feasible path in our WCSPP approximation.

3.2. Lower bounds. Let G be a $(\delta, \epsilon, \kappa)$-approximation network for an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP. Then for p^* an optimal solution to I, the sequences with properties given by Definition 3.1, that is $(p^*(s_0), p^*(s_1), \ldots, p^*(s_N))$ with $0 = s_0 < s_1 < \ldots < s_N = 1$ and the network approximation $p_G^* = (v_0, v_1, \ldots, v_N)$ to p^*, are guaranteed to exist. Using the sequence $(p^*(s_0), p^*(s_1), \ldots, p^*(s_N))$ to put a lower bound on the optimal solution to I we get

$$J[p^*] \geq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| M^{\downarrow}(v_{k-1}), \tag{3.3}$$

where $M^{\downarrow}(v_{k-1})$ is the minimum value of F on the region $R_{v_{k-1}}$. Here we have used Condition 3(c) from Definition 3.1 that the path segment between p_{k-1}^* and p_k^* is

entirely in the region $R_{v_{k-1}}$. Remember $\kappa > 1$ and for each $v \in V$, R_v is a closed connected region around v that is contained in a circle of radius $\kappa\delta$.

Using the sequence $(v_0, v_1, \ldots, v_N)$ to put an upper bound on the cost of the network approximation to p^* we get

$$J[p^*_G] \leq \sum_{k=1}^{N} ||v_k - v_{k-1}|| M^{\uparrow}(v_{k-1}), \tag{3.4}$$

where $M^{\uparrow}(v_{k-1})$ is the maximum value of F on the region $R_{v_{k-1}}$. Here we again use Condition 3(c) from Definition 3.1 to guarantee that the arc between v_{k-1} and v_k is entirely in the region $R_{v_{k-1}}$.

Using inequalities (3.3) and (3.4) we get

(3.5)

$$\begin{aligned}
&J[p^*] - \frac{J[p^*_G]}{1+\gamma} \\
&\geq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| M^{\downarrow}(v_{k-1}) - \sum_{k=1}^{N} \frac{||v_k - v_{k-1}||}{1+\gamma} M^{\uparrow}(v_{k-1}) \\
&\geq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| M^{\downarrow}(v_{k-1}) - \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| M^{\uparrow}(v_{k-1}) \\
&\geq \sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| (M^{\downarrow}(v_{k-1}) - M^{\uparrow}(v_{k-1})),
\end{aligned}$$

where $\gamma \in \Phi_G$. At this stage we make the following definition.

DEFINITION 3.4. $\Delta_G = \max_{v \in V \setminus \{b\}} (M^{\uparrow}(v) - M^{\downarrow}(v))$ *or equivalently using the definition of* $M^{\uparrow}(v)$ *and* $M^{\downarrow}(v)$, $\Delta_G = \max_{v \in V \setminus \{b\}} (\max_{x \in R_v} F(x) - \min_{y \in R_v} F(y))$.

The parameter Δ_G represents the maximum over all $v \in V$ of the variation of the underlying F function over the regions R_v. Using this definition we proceed as follows:

(3.6)

$$\begin{aligned}
J[p^*] - \frac{J[p^*_G]}{1+\gamma} &\geq -\sum_{k=1}^{N} ||p^*(s_k) - p^*(s_{k-1})|| \Delta_G && \text{using Definition 3.4} \\
&\geq -Eu[p^*]\Delta_G && \text{by (3.1)} \\
&\geq -\bar{L}\Delta_G && \text{as } Eu[p^*] \leq \bar{L}.
\end{aligned}$$

Rearranging (3.6) gives us the relation

$$J[p^*] \geq \frac{J[p^*_G]}{1+\gamma} - \bar{L}\Delta_G. \tag{3.7}$$

If we consider the WCSPP for network G with a relaxed weight constraint, i.e., finding the network path in G between nodes a and b with length less than or equal to $\bar{L}(1+\gamma)$, we know that any network approximation p^*_G to the optimal path p^* of instance I is feasible for the relaxed WCSPP by Corollary 3.3. Thus if q^*_G is an

optimal solution to the relaxed WCSPP, then we know $J[q_G^*] \leq J[p_G^*]$ by the definition of the optimality of q_G^*. Then

$$J[p^*] \geq \frac{J[q_G^*]}{1+\gamma} - \bar{L}\Delta_G. \tag{3.8}$$

Letting $J^*(\bar{L}) = J[p^*]$ be the optimal objective function value of an instance I of the C-LCMCPP and $J_G^*(\bar{L}(1+\gamma)) = J[q_G^*]$ the optimal objective function value of the relaxed WCSPP using the network G, we get

$$J^*(\bar{L}) \geq \frac{J_G^*(\bar{L}(1+\gamma))}{1+\gamma} - \bar{L}\Delta_G. \tag{3.9}$$

Using this relation, we can calculate concrete lower bounds on the solution of the C-LCMCPP as we will demonstrate in section 5. Note that if we cannot find the optimal objective function value of the relaxed WCSPP, any lower bound on optimal value can replace $J_G^*(\bar{L}(1+\gamma))$ in the formula and produce a valid, if worse, lower bound on $J^*(\bar{L})$.

3.3. κ-regular network constructions and convergence. To create a convergent network construction we define κ-regular network constructions in Definition 3.5. When given the right series of parameters a κ-regular network construction produces a series of $(\delta, \epsilon, \kappa)$-approximation networks for which δ and $\frac{\epsilon}{\delta}$ approach zero. This property will help us show that κ-regular network constructions are convergent network constructions.

DEFINITION 3.5. *A κ-regular network construction is a network construction $\mathcal{G}$ with parameter domain S for which for any instance I of the C-LCMCPP there exists:*

1. *A sequence of parameter vectors $(P_1, P_2, \dots)$, $P_k \in S, \forall k \in \mathbb{Z}^+$,*
2. *A sequence $(\delta_1, \delta_2, \dots)$ with $\delta_k > 0, \forall k \in \mathbb{Z}^+$ such that $\lim_{k\to\infty} \delta_k = 0$ and,*
3. *A sequence $(\epsilon_1, \epsilon_2, \dots)$ with $\epsilon_k > 0, \forall k \in \mathbb{Z}^+$ such that $\lim_{k\to\infty} \frac{\epsilon_k}{\delta_k} = 0$,*

such that $\mathcal{G}(I, P_k)$ is a $(\delta_k, \epsilon_k, \kappa)$-approximation for all $k \in \mathbb{Z}^+$.

Before we prove convergence, we need to introduce the following theorem.

THEOREM 3.6. *Let $J^*(L)$ with $L \in [L_{min}, \infty)$ be the optimal objective function value of instance $I = (\Omega, F, L, a, b)$ of the C-LCMCPP, where $L_{min} = ||b - a||$ is the minimum distance between a and b. Then $J^*(L)$ is monotonically decreasing and continuous for $L \in [L_{min}, \infty)$ if F is Hölder continuous and Ω is convex.*

Proof. To show $J^*(L)$ is monotonically decreasing, we note that if $p^*(L_1)$ is an optimal solution to the C-LCMCPP for weight limit L_1, then for $L_2 > L_1$, $p^*(L_1)$ is a feasible solution to the C-LCMCPP with weight limit L_2. Thus if $L_1 < L_2$, $J[p^*(L_1)] = J^*(L_1) \geq J^*(L_2) = J[p^*(L_2)]$ so $J^*(L)$ must be monotonically decreasing. Due to space limitations, the proof that $J^*(L)$ is continuous is omitted. ∎

Note, however, that $J^*(L)$ may not be continuous if we allow obstacles as these would result in either a discontinuity in F or nonconvexity of Ω. In fact, it is easy to construct an example with obstacles in which the function $J^*(L)$ is discontinuous. In this paper we do not consider obstacles; recall our initial assumption that F is continuous and Ω is convex.

THEOREM 3.7. *A κ-regular network construction is a convergent network construction.*

Proof. Consider a κ-regular network construction $\mathcal{G}$. For any instance $I = (\Omega, F, \bar{L}, a, b)$, we know by Definition 3.5 that there exists a sequence of parameter vectors, $(P_1, P_2, \dots)$, such that $G(n) = (V(n), A(n)) = \mathcal{G}(I, P_n)$ is a $(\delta(n), \epsilon(n), \kappa)$-approximation network and $\lim_{n\to\infty} \delta(n) = 0$ with $\lim_{n\to\infty} \frac{\epsilon(n)}{\delta(n)} = 0$.

We see by rearranging (3.9) that

$$J^*_{G(n)}(\bar{L}(1+\gamma(n))) \leq (J^*(\bar{L}) + \bar{L}\Delta_{G(n)})(1+\gamma(n)), \tag{3.10}$$

where $\gamma(n) = 2\frac{\epsilon(n)}{\delta(n)} + 2\frac{\epsilon(n)^2}{\delta(n)^2} \in \Phi_{G(n)}$.

To show that the right-hand side of (3.10) converges to $J^*(\bar{L})$, we need to show that we can refine the network in such a way that $\lim_{n\to\infty} \Delta_{G(n)} = 0$ and $\lim_{n\to\infty} \gamma(n) = 0$.

Letting $R_v(n)$ be the region around the node $v \in V(n) \setminus \{b\}$ guaranteed to exist by Definition 3.1, we see clearly from Definition 3.4,

$$\Delta_{G(n)} = \max_{v\in V(n)\setminus\{b\}} (\max_{x\in R_v(n)} F(x) - \min_{y\in R_v(n)} F(y)),$$

that zero is a lower bound on $\Delta_{G(n)}$. We know from Definition 3.1 that each region $R_v(n)$ is contained in a disk of radius $\kappa\delta(n)$. Thus the maximum distance between points in the set $R_v(n)$ is $2\kappa\delta(n)$. Now as F is Hölder continuous, for any points x and y in Ω we have $|F(x) - F(y)| \leq K||x-y||^\sigma$ for some positive constant K and $0 < \sigma \leq 1$. Hence we have

$$\begin{aligned}
\lim_{n\to\infty} \Delta_{G(n)} &= \lim_{n\to\infty} \max_{v\in V(n)\setminus\{b\}} (\max_{x\in R_v(n)} F(x) - \min_{y\in R_v(n)} F(y)) \\
&\leq \lim_{n\to\infty} \max_{v\in V(n)\setminus\{b\}} K||\operatorname*{argmax}_{x\in R_v(n)} F(x) - \operatorname*{argmin}_{y\in R_v(n)} F(y)||^\sigma \quad \text{by Hölder condition} \\
&\leq \lim_{n\to\infty} K(2\kappa\delta(n))^\sigma \\
&\leq 0 \qquad\qquad \text{as } \lim_{n\to\infty} \delta(n) = 0.
\end{aligned}$$

Thus $\lim_{n\to\infty} \Delta_{G(n)} = 0$. Now

$$\lim_{n\to\infty} \gamma(n) = \lim_{n\to\infty} \left(c_1 \frac{\epsilon(n)}{\delta(n)} + c_2 \frac{\epsilon(n)^2}{\delta(n)^2} \right) = 0 \qquad \text{as } \lim_{n\to\infty} \frac{\epsilon(n)}{\delta(n)} = 0.$$

So finally, using $\lim_{n\to\infty} \Delta_{G(n)} = 0$ and $\lim_{n\to\infty} \gamma(n) = 0$, we get

$$\begin{aligned}
\lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n))) &\leq \lim_{n\to\infty} (J^*(\bar{L}) + \bar{L}\Delta_{G(n)})(1+\gamma(n)) \\
\lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n))) &\leq J^*(\bar{L}).
\end{aligned}$$

To show convergence we need a corresponding lower bound on $\lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n))$. We know $J^*(L)$ is continuous on $L \in [||b-a||, \infty)$ by Theorem 3.6. A continuous map of a convergent sequence is convergent, and thus

$$\begin{aligned}
\bar{L} &= \lim_{n\to\infty} \bar{L}(1+\gamma(n)) && \text{as } \lim_{n\to\infty} \gamma(n) = 0 \\
J^*(\bar{L}) &= \lim_{n\to\infty} J^*(\bar{L}(1+\gamma(n))) && \text{as } J^*(L) \text{ is continuous} \\
J^*(\bar{L}) &\leq \lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n))),
\end{aligned}$$

where we have used $J^*_{G(n)}(\bar{L}(1+\gamma(n))) \geq J^*(\bar{L}(1+\gamma(n)))$ by the optimality condition of the continuous optimal solution. Thus

$$J^*(\bar{L}) \leq \lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n)) \leq J^*(\bar{L}).$$

So

$$\lim_{n\to\infty} J^*_{G(n)}(\bar{L}(1+\gamma(n)) = J^*(\bar{L}).$$

This shows that if we choose the sequence of parameters $(P_1, P_2, \dots)$ guaranteed to exist by Condition 1 of Definition 3.5, then the optimal objective function values of the WCSPP's for the networks $G(n) = \mathcal{G}(I, P_n)$ will converge to the objective function value of the C-LCMCPP as $n \to \infty$. Thus we have shown a κ-regular network construction is convergent. ∎

Theorem 3.7 shows us that the WCSPP solutions for a κ-regular network construction using the relaxed weight constraint converge to the solution of the C-LCMCPP. However, we can also show that the solutions to the WCSPP using the same weight constraint as the C-LCMCPP also converge.

THEOREM 3.8. *Given an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP with $\bar{L} > ||a - b||$ and a κ-regular network construction $\mathcal{G}$, the solutions to the WCSPP using the network $\mathcal{G}(I, P_n)$ and the weight limit $\bar{L}$ will converge to the solution of I for some sequence of parameter sets $(P_1, P_2, \dots)$.*

Proof. Choose the parameter set $(P_1, P_2, \dots)$ guaranteed to exist by Definition 3.5 such that $G(n) = (V(n), A(n)) = \mathcal{G}(I, P_n)$ is a $(\delta(n), \epsilon(n), \kappa)$-approximation network with $\lim_{n\to\infty} \delta(n) = 0$ and $\lim_{n\to\infty} \frac{\epsilon(n)}{\delta(n)} = 0$. Let $\gamma(n) = 2\frac{\epsilon(n)}{\delta(n)} + 2\frac{\epsilon(n)^2}{\delta(n)^2} \in \Phi_{G(n)}$.

Choose a sequence L_n and integer N such that $\bar{L} = L_n(1+\gamma(n))$ and $L_n > ||a-b||$ for all $n \in \{N, N+1, \dots\}$. Note that $\lim_{n\to\infty} L_n = \bar{L}$ and $L_n < \bar{L}$ for all $n \in \mathbb{Z}^+$. As $G(n)$ is a $(\delta(n), \epsilon(n), \kappa)$-approximation network to problem instance $(\Omega, F, \bar{L}, a, b)$, then $G(n)$ is also a $(\delta(n), \epsilon(n), \kappa)$-approximation network for the problem instance (Ω, F, L_n, a, b); this is readily seen from Definition 3.1 noting $L_n < \bar{L}$. Hence, applying (3.9) for $n \in \{N, N+1, \dots\}$, we get

$$\begin{aligned} J^*(L_n) &\geq \frac{J^*_{G(n)}(L_n(1+\gamma(n)))}{1+\gamma(n)} - L_n\Delta_{G(n)} \\ &\geq \frac{J^*_{G(n)}(\bar{L})}{1+\gamma(n)} - L_n\Delta_{G(n)} \qquad \text{as } L_n(1+\gamma(n)) = \bar{L}. \end{aligned}$$

Rearranging, we get

$$J^*_{G(n)}(\bar{L}) \leq (J^*(L_n) + L_n\Delta_{G(n)})(1+\gamma(n)).$$

Taking limits and using reasoning similar to that used in the proof of Theorem 3.7, we obtain

$$\lim_{n\to\infty} J^*_{G(n)}(\bar{L}) \leq (J^*(L_n) + L_n\Delta_{G(n)})(1+\gamma(n)), \tag{3.11}$$

which implies

$$\lim_{n\to\infty} J^*_{G(n)}(\bar{L}) \leq J^*(\bar{L}) \qquad \text{as } J^*(.) \text{ continuous.} \tag{3.12}$$

Also, the optimization problem that defines $J^*_{G(n)}$ has a domain that is a subset of the domain of the optimization problem that defines J^* so $J^*(\bar{L}) \leq J^*_{G(n)}(\bar{L})$ for all $n \in \mathbb{Z}^+$. Thus $\lim_{n\to\infty} J^*_{G(n)}(\bar{L}) \geq J^*(\bar{L})$. So clearly $\lim_{n\to\infty} J^*_{G(n)}(\bar{L}) = J^*(\bar{L})$. This completes the proof. ∎

By solving the WCSPP corresponding to a given instance of the C-LCMCPP with the original rather than relaxed weight constraint we obtain feasible solutions to the optimization problem and an upper bound. Theorem 3.8 tells us that the upper bounds will converge to the true continuous optimal solution as we refine our framework.

Note that the convergence proof does not work if the length constraint is equal to $\|a-b\|$. In this case, there is only one possible path, being the straight line from a to b. However, in our successive network approximations to this problem this path may not appear in our network as all paths from a to b in our network may be slightly longer that $\|b-a\|$. The WCSPP would then have no feasible solution for $\bar{L} = \|b-a\|$ and thus $J^*_G(\|a-b\|)$ would be undefined. In this case, the successive approximations could not be said to converge.

4. A specific κ-regular network construction technique. In this section we create a κ-regular network construction $\mathcal{G}$ that satisfies Definition 3.5. We will do this using one network to create a scaffolding with cells of size of order δ and then placing a second network used to solve the WCSPP on this scaffolding. The nodes of the second network are placed on the boundaries of the cells with the nodes spaced at most 2ϵ apart. For reasons that will become clear later, we will call this a *cellular* network construction.

Our parameter space will be the set $S = \{(i,j,M) \in \mathbb{Z}^{\oplus} \times \mathbb{Z}^{\oplus} \times \mathbb{Z}^{+} : (i,j) \neq (0,0)\}$ where $\mathbb{Z}^{\oplus}$ is the set of nonnegative integers. We will first show that each network constructed is a $(\delta, \epsilon, \kappa)$-approximation network where $\delta = \frac{\sqrt{3}}{2}\frac{\|b-a\|}{\sqrt{i^2+j^2+ij}}$, $\epsilon = \frac{1}{\sqrt{3}M}\delta$ and $\kappa = \frac{4}{\sqrt{3}}$.

For an instance $I = (\Omega, F, \bar{L}, a, b)$ of the C-LCMCPP, we construct our network by first creating a tessellation of equilateral triangles covering all of $\mathbb{R}^2$ such that a and b (the start and end points) are located at triangle corners. The triangle size and orientation is specified by the parameters i and j. Specifically, we find the side length l_{ij} and unit vectors d_1 and d_2 such that d_2 points $\frac{\pi}{3}$ radians anticlockwise to the direction of d_1 and $a + il_{ij}d_1 + jl_{ij}d_2 = b$. Using the cosine rule and referring to Figure 4.1(a) we see that

$$\|b-a\|^2 = i^2 l_{ij}^2 + j^2 l_{ij}^2 - 2ijl_{ij}^2 \cos\left(\frac{2\pi}{3}\right),$$

which we simplify and rearrange to give

$$l_{ij} = \frac{\|b-a\|}{\sqrt{i^2+j^2+ij}}.$$

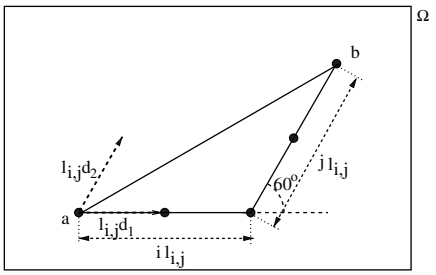
We can then find d_1 and d_2 using the sine rule. We construct our tessellation to align the vectors d_1 and d_2 with the side length of the triangles given by l_{ij}. Figure 4.1(b) shows the resulting tessellation for parameter vector $(i,j) = (2,2)$.

We define

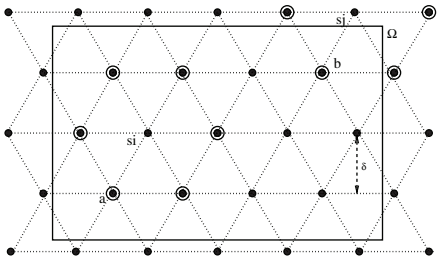
$$\delta = \frac{\sqrt{3}}{2} l_{ij} = \frac{\sqrt{3}}{2}\frac{\|b-a\|}{\sqrt{i^2+j^2+ij}}.$$

This definition makes δ the perpendicular height of the equilateral triangles that form our tessellation.

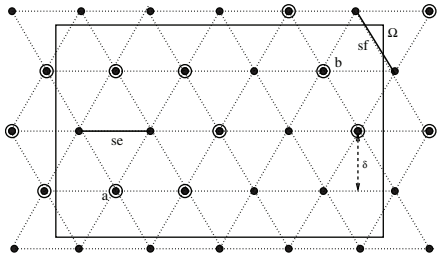
We will view this tessellation as a network which we call $T(i,j) = (SN, SE)$. To distinguish this network from the one over which the WCSPP is solved, we call the



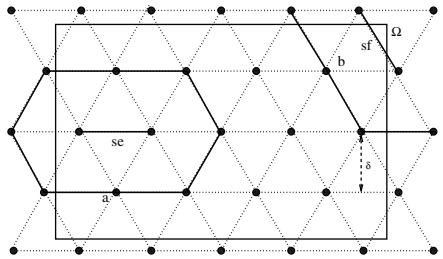
(a) Construction diagram for $T(i,j)$.



(b) Adjacent node sets for supernodes si and sj.



(c) Boundary supernode sets for superedges se and sf.



(d) Boundary superedge sets for superedges se and sf.

FIG. 4.1. *These diagrams show the construction of and examples of the various definitions for our scaffolding graph. The network was created using* $(i,j) = (2,2)$. *Knowing* (i,j) *and the position of* a *and* b, *we can easily find the length* l_{ij} *and the vectors* d_1, d_2. *These are used to construct our scaffolding graph* (SN_Ω, SE_Ω). *The supernode set* SN_Ω *are the black dots shown and the superedge set* SE_Ω *are dotted lines shown on the diagram.*

elements of SN *supernodes* and the elements of SE *superedges*. We place a supernode at every triangle vertex to form the set $SN = \{a + ml_{ij}d_1 + nl_{ij}d_2 : m,n \in \mathbb{Z}\}$. Then the set of superedges are defined by $SE = \{se = \{si, sj\} : si, sj \in SN, \|si - sj\| = l_{ij}\}$. Note that superedges are undirected.

DEFINITION 4.1. *We say the superedge* $se = \{si, sj\}$ contains *a point* $x \in \Omega$ *if there exists* $\lambda \in [0,1] \subset \mathbb{R}$ *such that* $x = \lambda si + (1-\lambda)sj$.

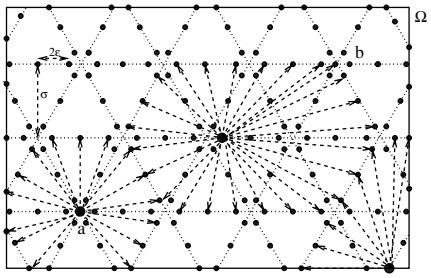
Naturally, we will only be interested in the part of the tessellation that covers Ω. Let the set of triangles in $T(i,j)$ that cover Ω, i.e., all triangles that intersect with Ω, be denoted by $Tri_\Omega = \{\triangle = \{\{si, sj\}, \{si, sk\}, \{sj, sk\}\} \subset SE : \exists se \in \triangle$ and $\exists x \in \Omega$ s.t. se contains $x\}$. We define a new set of supernodes $SN_\Omega = \{sn \in SN : \exists \triangle \in Tri_\Omega$ with $se \in \triangle$ s.t. $sn \in se\}$. We define $SE_\Omega = \{\{si, sj\} \in SE : si, sj \in SN_\Omega\}$. This gives us our scaffolding graph (SN_Ω, SE_Ω). We will also make the following definitions to ease the rest of the discussion.

DEFINITION 4.2. *The adjacent node set to a supernode* $si \in SN_\Omega$ *is the set* $Adj(si) = \{sj \in SN_\Omega : \|sj - si\| = l_{ij}\}$.

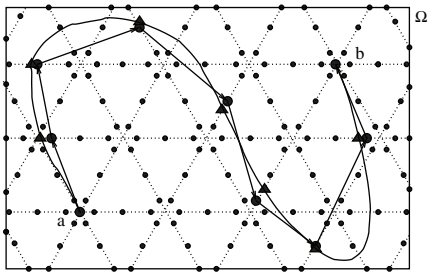
DEFINITION 4.3. *The* boundary supernode set *of a superedge* $se = \{si, sj\} \in SE_\Omega$ *is the set* $BndyNodes(se) = (Adj(si) \cup Adj(sj)) \setminus \{si, sj\}$.

DEFINITION 4.4. *The* boundary superedge set *of a superedge* $se \in SE_\Omega$ *is the set* $Bndy(se) = \{\{si, sj\} \in SE_\Omega : si, sj \in BndyNodes(se)\}$.

We will place the nodes of our network on the sections of the superedges inside Ω, and space the nodes such that each point in Ω contained by a superedge is at most ϵ from a node on that same superedge. We first choose an integer $M \geq 1$ and let



(a) Diagram of network showing how nodes (black dots) are placed and the leaving edges (dashed arrows) for selected nodes.



(b) Diagram of network approximation to a continuous path. The triangles are the $p(s_k)$'s and the larger circles are the corresponding v_k's.

FIG. 4.2. *These diagrams illustrate node placement, arc choice and path approximation in κ-regular networks.*

$\epsilon = \frac{1}{\sqrt{3}M}\delta$. This makes ϵ the length of the side of a tessellation triangle divided by $2M$. We then use the following procedure to place nodes on the superedges, producing our node set V:

For each superedge $se = \{si, sj\} \in SE_\Omega$,

1. **If** $si \in \Omega$, place a node at a distance ϵ along the superedge from si and place subsequent nodes at a distance 2ϵ as long as each node is placed inside Ω. If the next node to be placed is outside Ω and if the intersection of the superedge and boundary of Ω is a distance greater than ϵ from the last node, or it is the first node to be placed, we place a node on the intersection of the boundary of Ω and se; otherwise we do not place a node.
2. **Else If** $sj \in \Omega$, follow rule 1 but start at sj instead of si.
3. **Else If** $si, sj \notin \Omega$, and there exists $x \in \Omega$ such that se contains x, we start by placing a node at one of the intersections between the superedge and the boundary of Ω. We then place nodes at intervals of 2ϵ until the next node to be placed would be outside Ω. If the other intersection se and the boundary of Ω is a distance greater than ϵ from the last node, we place a node at the other intersection of the superedge and the boundary.
4. **Else If** there does not exist $x \in \Omega$ such that se contains x, then we do not place nodes on that super edge.

The above procedure defines our node set V. An example of the placement of nodes can be found in Figure 4.2(a). Next, we define the edge connectivity in our network but first we make the following definitions.

DEFINITION 4.5. *For node $i \in V \setminus \{a, b\}$ its* boundary superedge set*, $Bndy(i)$, is the set $Bndy(se)$ where i is contained by se. This is well defined as each node in $V \setminus \{a, b\}$ is contained by one and only one superedge. The boundary superedge set of a is $Bndy(a) = \{se = \{si, sj\} \in SE_\Omega : si, sj \in Adj(a)\}$. $Bndy(b)$ is not defined.*

DEFINITION 4.6. *A point $x \in \Omega$ is* contained *in $Bndy(j)$ for $j \in V \setminus \{b\}$ if there exists $se \in Bndy(j)$ such that x is contained by se.*

To create our edge set A we will place an edge from each node $i \in V \setminus \{b\}$ to every node $v \in V \setminus \{a\}$ contained by $Bndy(i)$. Note that no edges terminate at a and that there is an edge ending at b from any node which contains b in its boundary superedge set. Note, however, that no edges originate at b. Examples of edges are shown in Figure 4.2(a).

At this stage we have a network construction $\mathcal{G}$ that produces a network $G = (V, A)$ for a given instance I of the C-LCMCPP and parameter vector $P \in S$. We now wish to check that each network produced is a $(\delta, \epsilon, \kappa)$-approximation network.

For any path $p \in \Gamma$ from a to b, we will construct its network approximation in $G = (V, A)$ in the following manner and show it satisfies the properties of Definition 3.1.

1. Let $k = 0$, $v_0 = a$, and $s_0 = 0$ (meaning $p(s_0) = a$).
2. Let $k = k + 1$. Let $s_k \in (s_{k-1}, 1]$ be the smallest value such that $p(s_k)$ is contained in $Bndy(v_{k-1})$.
3. Let v_k be the closest node in $V \setminus \{a, b\}$ on the superedge containing $p(s_k)$. If $p(s_k)$ is located on a supernode, it will be contained by many superedges. In this case we choose v_k to be the closest node on any of the superedges in $Bndy(v_{k-1})$ containing $p(s_k)$, breaking ties arbitrarily.
4. If b is contained in $Bndy(v_{k-1})$ and if there is no $s \in [s_k, 1]$ such that $p(s)$ is contained in $Bndy(v_k)$, then let $v_k = v_N = b$ (replacing the last choice of v_k made in step 3) and $s_k = s_N = 1$ and stop. Otherwise go to step 2.

For any path $p \in \Gamma$ we have thus produced two sequences $(v_0, v_1, \ldots, v_N)$ and $(p(s_0), p(s_1), \ldots, p(s_N))$ with $v_k \in V$ for $k \in \{0, \ldots, N\}$ and $0 = s_0 < s_1 < \ldots < s_{N-1} < s_N = 1$. An example of the sequences $(v_0, v_1, \ldots, v_N)$ and $(p(s_0), p(s_1), \ldots, p(s_N))$ for a particular path and network is shown in Figure 4.2(b).

The Euclidean distance from any point on a superedge to any point on the boundary superedge set of that superedge is greater than or equal to δ; see Figure 4.1. We can see that as $p(s_k)$ lies on the boundary superedge set of the superedge which contains both v_{k-1} and $p(s_{k-1})$, we have $\|p(s_k) - p(s_{k-1})\| \geq \delta$ for $k \in \{2, N\}$. Also, all points contained by the boundary superedge set of node a are a distance greater than δ from a so $\|p(s_1) - p(s_0)\| \geq \delta$ as $p(s_1)$ is on the boundary superedge set on node a. Thus Condition 3(a) of Definition 3.1 is satisfied.

The point $p(s_k)$, $k \in \{1, \ldots, N-1\}$ will be approximated by the nearest node on the same superedge. Nodes are placed on superedges such that any point on the super edge is at most ϵ away from a node, and thus the spacing of nodes will satisfy the condition $\|v_k - p(s_k)\| \leq \epsilon$, $\forall k \in \{1, \ldots, N-1\}$. As $a = p(s_0) = v_0$ and $b = p(s_N) = v_N$, we satisfy Condition 3(b) of Definition 3.1.

By having edges run from each node $i \in V \setminus \{b\}$ to all the nodes contained by $Bndy(i)$, we can see that the edges required by a network approximation (v_0, v_1, v_2, $\ldots$, v_{N-1}, v_N) to any path $p \in \Gamma$, that is the edges (v_0, v_1), (v_1, v_2), $\ldots$, (v_{N-1}, v_N), are in A. This satisfies Condition 3 of Definition 3.1.

To define the regions R_v we make the following definition.

DEFINITION 4.7. *For supernode $si \in SN_\Omega$ the closed region $Reg(si) = \{\lambda si + (1-\lambda)(\mu sj + (1-\mu)sk) : \lambda, \mu \in [0,1], \{sj, sk\} \in SE_\Omega, \{sj, sk\} \subseteq Adj(si)\}$. For a set of supernodes, we will extend the definition of $Reg(.)$ to be the union of the set of regions for each supernodes, i.e., $Reg(\{si_1, si_2, \ldots, si_n\}) = \bigcup_{k=1}^{n} Reg(si_k)$.*

The region $Reg(si)$ will generally be a regular hexagon around supernode si, except where the network is truncated near the boundary of Ω.

The regions R_v, $v \in V \setminus \{a, b\}$ in Definition 3.1 are satisfied in our construction by the regions $Reg(se) \cap \Omega$ where se is the superedge containing node v with the exception of the case where b is contained by the boundary superedge set of se, in which case R_v is given by $Reg(se \cup sb_1 \cup sb_2) \cap \Omega$ where sb_1 and sb_2 are the superedges in the boundary superedge set of v that contain b. Examples of the regions near node b are given in Figure 4.3. For node a, $R_a = Reg(a) \cap \Omega$ except in the unlikely case that the boundary of a contains b. In this case $R_a = Reg(\{a\} \cup sb_1 \cup sb_2) \cap \Omega$ where

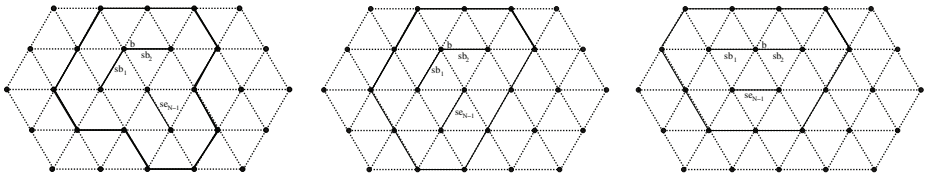


FIG. 4.3. *Diagrams of the shapes of the regions around end node b.*

sb_1 and sb_2 are the superedges in the boundary superedge set of a that contain b. Each of the regions can be enclosed in a circle of radius at most $\frac{4}{\sqrt{3}}\delta$. This means κ for our construction is $\frac{4}{\sqrt{3}}$.

For path p, the section of the path from $p(s_{k-1})$ to $p(s_k)$ and the points on the edge from v_{k-1} to v_k given by $\lambda v_{k-1} + (1-\lambda)v_k$ for $\lambda \in [0,1]$ are entirely contained in the region $R_{v_{k-1}}$ for $k \in \{1,\dots,N\}$. This satisfies the need for the regions R_v for $v \in V \setminus \{b\}$ and Condition 3(c) of Definition 3.1.

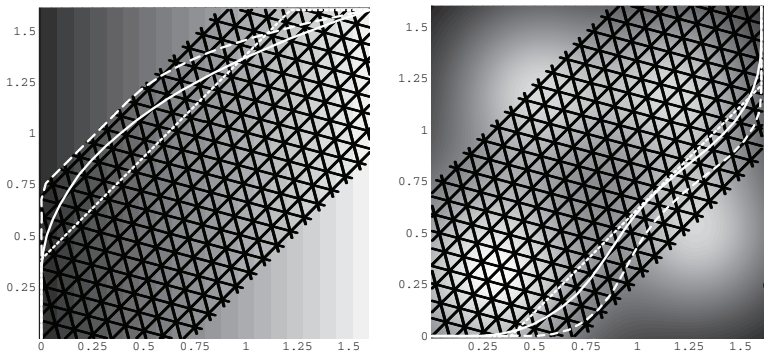
Thus for any instance $I = (\Omega, F, L, a, b)$ of the C-LCMCPP, the network $\mathcal{G}(I,P)$ for $P \in S$ produced by our cellular network construction will be a $(\delta, \epsilon, \kappa)$-approximation network.

We now wish to show that our cellular network construction is a κ-regular network construction. For any instance $I = (\Omega, F, L, a, b)$ of the C-LCMCPP, consider the network $\mathcal{G}(I,(k,0,k))$. This network will be a $(\delta_k, \epsilon_k, \kappa)$-approximation network for $\delta_k = \frac{\|a-b\|\sqrt{3}}{2k}$, $\epsilon_k = \frac{\|a-b\|}{2k^2}$, and $\kappa = \frac{4}{\sqrt{3}}$.

Noting the requirements of Definition 3.5, we see that for any instance I of the C-LCMCPP there is a sequence of parameter vectors $(P_1, P_2, P_3, \dots) = ((1,0,1), (2,0,2), (3,0,3), \dots)$, a sequence of δ values $(\delta_1, \delta_2, \delta_3, \dots) = \left(\frac{\|a-b\|\sqrt{3}}{2}, \frac{\|a-b\|\sqrt{3}}{4}, \frac{\|a-b\|\sqrt{3}}{6}, \dots\right)$ with $\lim_{k\to\infty} \delta_k = 0$, and a sequence of ϵ values $(\epsilon_1, \epsilon_2, \epsilon_3, \dots) = \left(\frac{\|a-b\|}{2}, \frac{\|a-b\|}{8} \frac{\|a-b\|}{18}, \dots\right)$ with $\lim_{k\to\infty} \frac{\epsilon_k}{\delta_k} = 0$, such that $\mathcal{G}(I,P_k)$ is a $(\delta_k, \epsilon_k, \kappa)$-approximation network. Thus the cellular network construction outlined in this section is a κ-regular network construction with $\kappa = \frac{4}{\sqrt{3}}$.

5. Numerical experiments. In this section we will test our κ-regular network construction method numerically. We implemented the lower bounds scheme in two different ways. Both schemes use Mathematica to calculate the node positions but differ in the method of calculating edge costs. In the Gaussian scheme, we used Mathematica to explicitly calculate the edge costs in the network using three-point Gaussian quadrature. These edges are then exported to a WCSPP solver written in C++. The trapezoidal scheme instead uses Mathematica to calculate function values at each node which are then exported to the WCSPP solver which calculates edge costs on the fly using the trapezoidal rule. The Gaussian scheme is more accurate, especially for smaller networks, whereas the trapezoidal scheme is faster as it utilizes the fact the edges share start and end positions and thus needs fewer function evaluations. The number of function evaluations is equal to the number of nodes for the trapezoidal scheme, whereas it is a multiple of the number of edges for the Gaussian scheme. The trapezoidal scheme also allows larger networks as the edges are not stored explicitly. Note that while it would be possible to calculate edge costs on the fly using Gaussian quadrature, this was not implemented.

For both schemes, Δ_G was found by numerically finding the maximum and minimum value of $F(x)$ for each region R_v, $v \in V \setminus \{b\}$ using Mathematica. Note that



(a) $F_1(x, y)$ (b) $F_2(x, y)$

FIG. 5.1. *Contour plot of functions $F_1(x, y)$ and $F_2(x, y)$ overlaid with the nodes of cellular network with parameters $(i, j, M) = (24, 0, 24)$. The lighter regions have higher function values. The shading scale on the two plots is not the same. Some parts of the region which are not length feasible have not been meshed. The edges are not shown to avoid cluttering the diagram. The paths corresponding to the upper bound in the cellular network are the solid white lines. The paths corresponding to the relaxed weight constraint $\bar{L}(1 + \gamma)$ are the long dashed lines. Both problems were solved using the Gaussian scheme. We also calculated the upper bound paths given by a grid network of* 721 *by* 721 *nodes which are shown as the short dashed lines. We can see that the cellular network produces a smoother path than the grid network. Note that the paths are all piecewise linear and have not been smoothed in anyway.*

all nodes on the same superedge share the same region R_v; thus only one calculation per superedge is required. We used a γ calculated by the formula

$$\gamma = \frac{2}{\sqrt{3}M} + \frac{2}{3M^2}, \tag{5.1}$$

where M is the number of nodes per side length. Note that we have used the pessimistic choice of $c_1 = c_2 = 2$ in (3.2) to calculate γ. The calculations were performed on a Pentium 4 2.4GHz with 512Mb RAM running under Linux.

We use two test functions. The first is $F_1(x, y) = x$. For the second we define a constituent function:

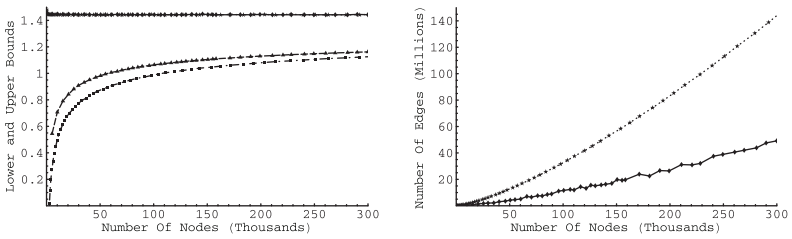
$$G_{\phi_1,\phi_2,\sigma}(x) = \frac{1}{\pi\sigma^2} e^{-\frac{(x_1-\phi_1)^2+(x_2-\phi_2)^2}{\sigma^2}},$$

and use the following as our test function,

$$F_2(x) = G_{.3,.3,.5}(x) + 0.5(G_{1.3,.4,.4}(x) + G_{.5,1.2,.4}(x) + G_{1.2,1.2,.4}(x)).$$

For both F_1 and F_2 our region Ω is the closed square with corners at $(0, 0)$ and $(1.6, 1.6)$. The start point a is $(0, 0)$ and the end point b is $(1.6, 1.6)$. The weight limit $\bar{L}$ is 1.1 times the distance between the start and end nodes or approximately 2.489 units. Both functions are plotted in Figure 5.1.

Figure 5.1 shows an example of a network and path that results from our cellular network construction for both $F_1(x, y)$ and $F_2(x, y)$. The solid white lines are the best upper bound paths, found by solving the WCSPP calculation using $\bar{L}$ as the length constraint. This is our approximate solution to the C-LCMCPP for this network. To obtain a lower bound, we use the relaxed weight constraint $\bar{L}(1 + \gamma)$ and solve the WCSPP to get a lower bound path, which are the long dashed lines. The objective



(a) Upper and lower bound.

(b) Number of nodes vs number of edges.

FIG. 5.2. *The upper and lower bounds for the C-LCMCPP for the function $F_1(x, y)$ vs the number of nodes. The squares are the lower bounds for $i = M$ and the triangles are the results when i and M are optimized for an approximately constant number of nodes. The stars are the upper bounds for the $i = M$ case and the diamonds are the upper bounds for the optimized i and M case. We can see that the optimization offers an improvement on the lower bound. We can also see a diminishing return on the improvement to the lower bound as we use more nodes. The number of edges vs the number of nodes is also given in* (b) *in which the stars indicate the $i = M$ case and the diamonds indicate the i and M optimized case.*

function values corresponding to these paths are $J_G^*(\bar{L}(1+\gamma))$ which are used in the lower bounds formula given by (3.9).

For comparison we have shown the paths that result from using a grid network with edges to the 8 nearest neighbors, shown as the short dashed line. We can see that the paths are not smooth compared to the ones obtained via the cellular construction. The number of nodes in the grid network was chosen to approximately equal the average node density of the cellular network. The objective function values of the grid network are also higher than that of the cellular network: 1.58602 vs 1.44257 or 9.9% higher for $F_1(x, y)$ 1.783 vs 1.734 or 2.8% higher for $F_2(x, y)$.

Figures 5.2(a) and 5.2(b) give the results of C-LCMCPP using successively larger networks to improve the lower bound. The trials were aborted at approximately 300,000 nodes and 150,000,000 edges when the WCSPP's became too big to solve effectively due to computational memory limitations.

Two methods of choosing i and M were tested. In the first, we set $i = M$ and in the second we chose i and M so that they provided the best lower bound for an approximately constant number of nodes. To do this we note that for our cellular network

$$|V| \approx K_v i^2 M \text{ and}$$
$$i \approx \sqrt{\frac{|V|}{K_v M}}. \tag{5.2}$$

We approximate Δ_G by

$$\Delta_G \approx \frac{K_\Delta}{i}. \tag{5.3}$$

Substituting (5.2), (5.3), and the formula for γ, (5.1), into the lower bounds formula, (3.9), gives

$$LB \approx \frac{J_G^*(\bar{L}(1+\gamma))}{1 + \frac{2}{\sqrt{3}M} + \frac{2}{3M^2}} - \bar{L}K_\Delta \sqrt{\frac{K_v M}{|V|}}. \tag{5.4}$$

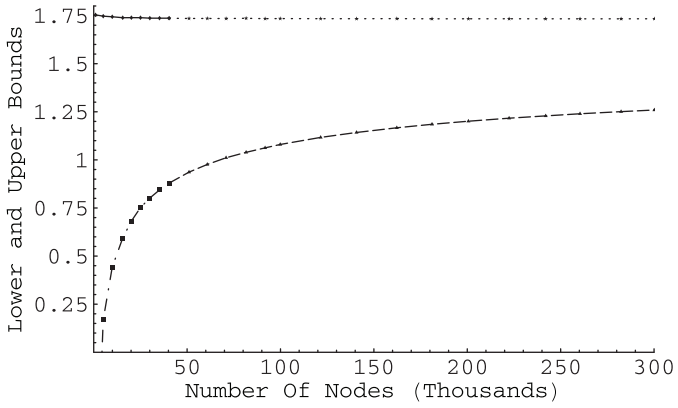


FIG. 5.3. *The upper and optimum lower bounds for the C-LCMCPP for the function $F_2(x, y)$ vs the number of nodes. The squares are the lower bounds calculated using the Gaussian scheme and the triangles are the lower bounds calculated using the trapezoidal scheme. The diamonds and the stars are the upper bounds calculated using Gaussian and trapezoidal scheme, respectively.*

Given previous lower bounds calculations we can estimate the values of $J_G^*(\bar{L}(1+\gamma))$, K_Δ and K_v, and then given a number of nodes $|V|$ we can calculate an approximately optimal value of M and use (5.2) to find i. We then vary i around this approximate optimal value to find a local optimum and report this value as the optimal i and M combination for a particular number of node in Figure 5.2.

The lower bound calculated by (3.9) steadily improves from being negative to becoming positive at 2,717 nodes and 227,909 edges when $i = M = 13$ and increases to within 21.5% of the upper bound at 318,946 nodes and 156,815,926 nodes when $i = M = 64$. Optimizing the choice of i and M results in a slight increase of the best lower bound to within 19.5% of the least upper bound using a network with $(i, M) = (108, 21)$ having 299,910 nodes and 49,165,143 edges.

Even though the smaller networks produce useless negative lower bounds, they produce competitive upper bounds. The upper bound for $i = M = 7$ calculated using a network of 404 nodes and 14,383 edges was 1.452 which was within 1% of the best upper bound of 1.442 produced for $i = M = 64$ with 318,946 nodes and 156,815,926 edges. We can see that the upper bound converges at a much faster rate than the lower bound. In light of the results of Zabarankin et al. [21], who compare grid network solutions to analytic solutions available in specific cases, we believe the upper bounds we compute to be very close to the corresponding global optima.

In Figure 5.3 we find the upper and lower bounds for a given number of nodes using the optimal choice of i and M for the function $F_2(x, y)$. The accuracy of the numerical integration is important in the smaller networks, which have longer edges; thus the Gaussian scheme was used for small numbers of nodes. When the size of the networks became prohibitively large for the Gaussian scheme we switched to the trapezoidal scheme. The accuracy of the trapezoidal scheme improves noticeably when the length of the edges is decreased; for example, for $(i, j, M) = (74, 0, 6)$, which produces a network with 40,358 nodes and 1,869,186 edges, the Gaussian scheme produces an upper bound of 1.73661 and the trapezoidal scheme produces an upper bound of 1.73614, a difference of less than 0.03%. The best lower bound found in this case was within 27.4% of the best upper bound. The trapezoidal scheme was also much faster with the above instance running in 1h51m using the trapezoidal scheme as opposed to 7h27m using the Guassian scheme.

TABLE 5.1

Percentage improvement in the upper bound when changing from a 8-nearest neighbor grid network to a cellular with similar number of edges. The results are averaged over 15 different functions. The standard deviation of the percentage improvement in the upper bound is given as std dev and N^o indicates the number of instances in which the cellular network produced a better result than the grid network out of the 15 instances.

	(i,j,M)	(6,0,3)	(12,0,6)	(24,0,12)	(30,15)
	grid width	21	88	364	572
	$\|A\|$ cell	3279	60508	1053426	2611897
	$\|A\|$ grid	3280	60900	1055604	2610612
$L = 2.489$	% mean UB gain	9.3	12.8	11.5	11.5
	std dev.	17.8	16.2	16.4	16.4
	N^o	9	15	15	15
$L = 2.715$	% mean UB gain	0.0	7.4	9.0	10.1
	std dev.	13.4	8.0	8.4	11.3
	N^o	8	15	15	15
$L = 2.942$	% mean UB gain	-6.1	0.5	2.2	1.9
	std dev.	8.3	1.9	1.7	1.9
	N^o	4	11	13	12

Surprisingly, the majority of the computational time was spent on evaluating $F(x,y)$ and/or evaluating line integrals, exporting data to the WCSPP solver, and calculating Δ_G rather than solving the resulting NP-hard WCSPP. As we were focused on pushing the lower bound as high as possible, we tested some problems with extremely long run times. For example, the time for a complete run, that is, calculating the node positions, calculating Δ_G, calculating the edges weights, exporting the edge data to the WCSPP solver, and solving two WCSPPs (for the upper and lower bound) for $F_2(x,y)$ for $(i,j,M) = (143,0,12)$ using 300,170 nodes and 28,302,752 edges was close to 10 hours giving an upper bound of 1.734 and a lower bound of 1.259. However, we were able to obtain reasonable upper bounds in much shorter times; for example, it took only 6min 7sec to do the same calculation with $(i,j,M) = (14,0,4)$ to get an upper bound of 1.754, though the network was not large enough to provide a positive lower bound.

Though calculating lower bounds can involve extreme computational effort, if we are looking only for feasible solutions, then we can use much smaller networks and get reasonable results. Given that the upper bound for cellular networks seems to converge quite rapidly, we compared the values of the upper bound, thus the best feasible solution found for the problem, to the upper bounds produced by 8 nearest neighbor grid network with a similar number of edges. We again set Ω to the closed square with corners at $(0,0)$ and $(1.6,1.6)$ and the start point to $a = (0,0)$ and the end point to $b = (1.6,1.6)$. We used three different weight limits for each function: 1.1, 1.2, and 1.3 times the distance from the start to end point, respectively. Besides using the functions F_1 and F_2, all the other functions we used were a sum of 15 Gaussians of the form $G_{\phi_1,\phi_2,\sigma}(x)$ with uniformly random centers, $(\phi_1,\phi_2) \in \Omega$, and σ uniformly random in the range $[.1,.3]$.

Table 5.1 shows us that, as we would expect, in the majority of cases, the upper bound produced by the cellular network is lower than that produced by a grid network with a similar number of edges. The clearest trend is the mean percentage improvement of the cellular network over the grid network improves as the weight constraint is made tighter. The cellular networks also tend to do better than the corresponding grid network when the networks are made larger.

We can also see that the variability of the improvement increases with the tighter weight constraint. This may be due to the weight constraint forcing the choice of high

cost arcs in the grid network as paths in the grid network are longer than they need be due to the restrictions in the number of directions available.

6. Conclusion. We have produced a network approximation method to the continuous length constrained minimum cost path problem (C-LCMCPP) for which we can show the network approximation converges to the continuous solution as the network is enlarged in an appropriate manner. We then defined $(\delta, \epsilon, \kappa)$-approximation networks and showed that for such a network, a lower bound can be calculated. We went on to define a κ-regular network construction, which can produce a sequence of $(\delta, \epsilon, \kappa)$-approximation networks such that the lower bound (and upper bound) converges to the continuous optimum.

Having developed the theory, we then created a specific example of a κ-regular network construction, which we dubbed a cellular network construction, and tested it computationally. We were able to calculate lower bounds that came within 19.5% of the best upper bound. We also found that the cellular networks produced upper bounds that converged rapidly and that corresponded to smoother paths with lower objective function values than the solutions produced by grid networks.

In the future, we wish to improve the lower bounds further. One possible method is a nonuniform triangulation of space so that node placement better reflects the contours of the underlying function. We also wish to explore the potential of iteratively eliminating regions of space using lower bounds; this would allow better lower bounds to be obtained for the same computational effort.

We may also look at restricting the underlying function to be a triangulated surface to simplify function evaluations, given that this is the way many surfaces in practical situations are represented. The function triangulation could be made to coincide with the triangulation of the cellular network. Implementing this method of function evaluation in C++ would offer a significant speed up to the algorithm over using analytical functions in Mathematica.

The cellular network concept may also be applied to higher dimensional spaces. One could imagine the cells becoming interlocking polytopes with nodes placed on the facets of these polytopes. While the theory underlying such a construction may be relatively straightforward, the number of nodes required by the discretization would grow enormously. However, given that the upper bounds for two dimensional networks converge rapidly, it may be possible that useful upper bounds for higher dimensional problems are attainable.

Lastly, we may wish to change the length constraint to a constraint with the same form of the objective function. This would allow more versatility in the application of the theory to practical situations. The main challenge here would be proving that suitably relaxing the constraint guarantees that a network approximation to an optimal path exists.

Acknowledgment. We would like to thank Tim Robinson for setting us on this path.

REFERENCES

[1] C. Barnhart, N. L. Boland, L. W. Clarke, E. L. Johnson, G. L. Nemhauser, and R. G. Shenoi, *Flight string models for aircraft fleeting and routing*, Transport. Sci., 32 (1998), pp. 208–220.

[2] L. Caccetta, I. Loosen, and V. Rehbock, *Modelling transit paths for military vehicles*, in Proceedings of the 16th Congress of the Modelling and Simulation Society of Australia and New Zealand, 2005, pp. 1751–1757.

[3] W. M. Carlyle and R. K. Wood, *Lagrangian relaxation and enumeration for solving constrained shortest-path problems*, in 38th Annual ORSNZ Conference, University of Waikato, Hamilton, New Zealand, November 2003.
[4] E. P. Chew, C. J. Goh, and T. F. Fwa, *Simultaneous optimization of horizontal and vertical alignments for highways*, Transport. Res. B-Meth., 23B (1989), pp. 315–329.
[5] E. Cristiani and M. Falcone, *Fast semi-Lagrangian schemes for the Eikonal equation and applications*, SIAM J. Numer. Anal., 45 (2007), pp. 1979–2011.
[6] E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numer. Math., 1 (1959), pp. 269–271.
[7] I. Dumitrescu and N. L. Boland, *Improved preprocessing, labelling and scaling algorithms for the weight-constrained shortest path problem*, Networks, 43 (2003), pp. 135–153.
[8] A. Fahlen, *Missile routing for a stand-off missile*, Master's Thesis, Optimeringslara, Matematiska institutionen, November 2000.
[9] C. J. Goh, E. P. Chew, and T. F. Fwa, *Discrete and continuous models for computation of optimal vertical highway alignment*, Transport Res. B-Meth., 22B (1988), pp. 399–409.
[10] L. Guo and I. Matta, *Search space reduction in QoS routing*, Comput. Network, 41 (2003), pp. 73–88.
[11] P. E. Hart, N. J. Nilsson, and B. Raphael, *A formal basis for the heuristic determination of minimum cost paths*, IEEE Trans. Syst. Sci. Cyb. SSC4, 2 (1968), pp. 100–107.
[12] J. Kim and J. P. Hespanha, *Discrete approximations to continuous shortest-path problems: Application to minimum-risk path planning for groups of UAVs*, in the 42nd Conference on Decision & Control, Hawaii, December 2003.
[13] R. Kimmel and N. Kiryati, *Finding shortest paths on surfaces by fast global approximation and precise local refinement*, Int. J. Pattern Recogn., 10 (1996), pp. 643–656.
[14] R. Kimmel and J. A. Sethian, *Optimal algorithm for shape from shading and path planning*, J. Math. Imaging Vision, 14 (2001), pp. 237–244.
[15] R. Kimmel and G. Shapiro, *Shortening three-dimensional curves via two-dimensional flows*, Comput. Math. Appl., 29 (1995), pp. 49–62.
[16] I. M. Mitchell and S. Sastry, *Continuous path planning with multiple constraints*, in the 42nd Conference on Decision and Control, Hawaii, December 2003.
[17] R. D. Muhandiramge and N. L. Boland, *Simultaneous solution of related Lagrangean dual problems with iterated preprocessing for solving the weight constrained shortest path problem*, Networks 2008, to appear.
[18] C. D. Piatko, C. P. Diehl, P. McNamee, C. Resch, and I. Wang, *Stochastic search and graph techniques for MCM path planning*, in Detection and Remediation Technologies for Mines and Minelike Targets VII, J. T. Broach, R. S. Harmon, and G. J. Dobeck, eds., SPIE, 2002.
[19] C. D. Piatko, C. Priebe, L. Cowen, I. Wang, and P. McNamee, *Path planning and mine countermeasures command and control*, in Detection and Remediation Technologies for Mines and Minelike Targets VI, SPIE, 2001.
[20] J. N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Contr., 40 (1995), pp. 1528–1538.
[21] M. Zabarankin, S. Uryasev, and R. Murphey, *Aircraft routing under the risk of detection*, Naval Res. Logist., 53 (2006), pp. 728–747.