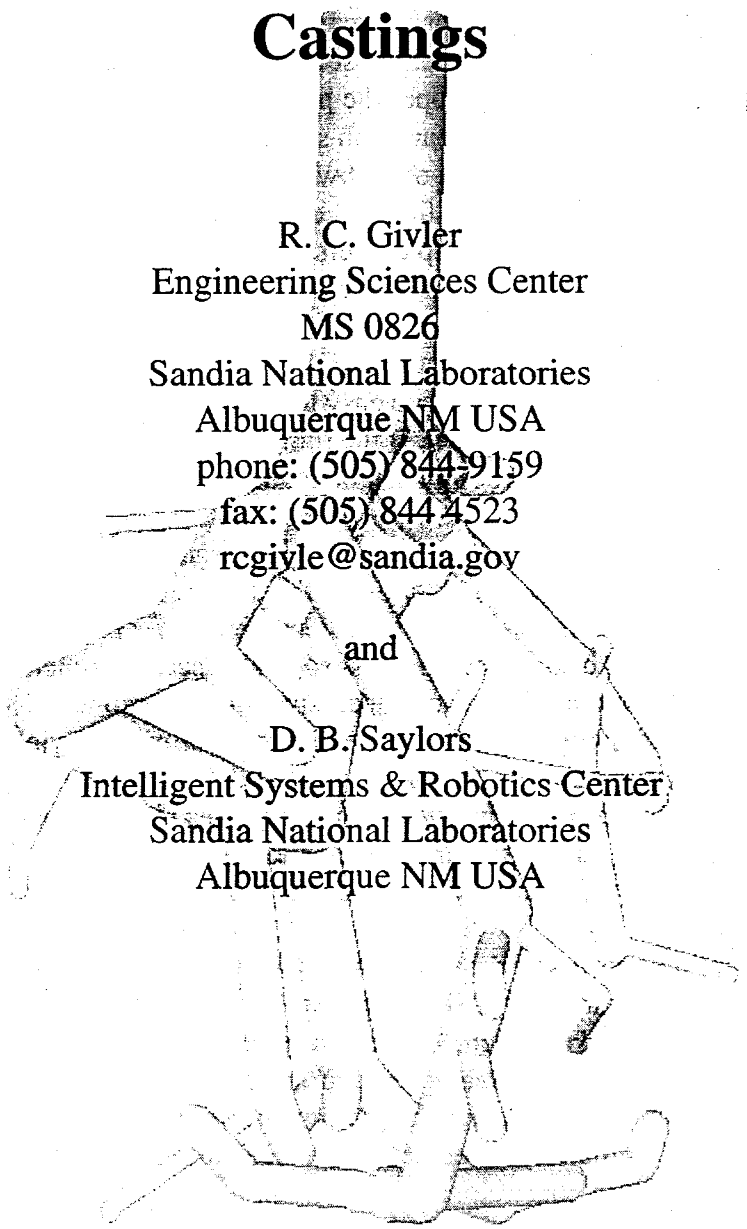
SAND2000-1758J

Efficient Runner Networks for Investment Castings

RECEIVED
AUG 17 2000
OSTI



R. C. Givler
Engineering Sciences Center
MS 0826
Sandia National Laboratories
Albuquerque NM USA
phone: (505) 844-9159
fax: (505) 844-4523
rcgivle@sandia.gov

and

D. B. Saylors
Intelligent Systems & Robotics Center
Sandia National Laboratories
Albuquerque NM USA

Summary: We present a computational method that finds an efficient runner network for an investment casting, once the gate locations have been established. The method seeks to minimize a cost function that is based on total network volume. The runner segments are restricted to lie in the space not occupied by the part itself. The collection of algorithms has been coded in C and runner designs have been computed for several real parts, demonstrating substantial reductions in rigging volume.

Keywords: *combinatorial optimization, investment casting, runner network, shortest paths*

1. Introduction

The intuitive design process of a satisfactory runner system for investment castings is an art. An experienced engineer usually begins this process by choosing the best orientation of a gated part to facilitate investing the part, burning out the wax, filling the cavity and fabricating the runner system itself. Having made this determination the gates are connected so that there is a continuous path from a common pouring cup to each gate. For some parts with many gates, the corresponding rigging often appears as a cage which envelopes the part. One recurring theme in runner design is the use of one or more rings encircling the part with spokes connecting the ring to individual gates. In two dimensions, this is reminiscent of highway design in the vicinity of a large city. Past experience and a knowledge of liquid metal flow in conduits can guide the engineer in making these connections. For example, flow in sharp corners or T-junctions will be inhibited more than that in smooth bends. More often than not, the runner system design is impacted by its ease of fabrication. Many connections are made at right angles because it is easy to cut and butt-join extruded wax stock in this manner. These facts will influence the manner in which a designer makes runner connections. Thus, it has not been easy to codify the process of runner design — as one can see, many considerations influence it. If the runner system is inadequate during the casting process, *a posteriori* changes to its design can be made, but at a cost that is often significant.

Another feature of a well-designed rigging system is the enlarged cross-sectional area of runner segments that feed multiple members at a juncture — this feature ensures that sufficiently molten metal will reach the lower branches of the network; it also promotes pressurized filling, *i.e.* all runners remain filled. In runner designs that possess several cascading junctures, however, liberally increasing the cross-sectional area at each successive juncture can lead to fairly inefficient runner networks. Casting efficiency can be measured in terms of product yield; here, yield is defined as the cast weight ratio between the part and metal delivery system (not including the pouring cup). In some parts, especially thin-walled castings, this ratio can be as low as 10%. Such a ratio implies that most of the poured metal ends up in the runner network. This is especially problematic for expensive alloys or alloys that can not be reconstituted or recycled.

We have recognized that the process of designing an efficient runner network can be formulated as a problem in combinatorial optimization. Simply posed, can one devise an algorithm to compute the shortest "weighted" network [1] which connects the gates to the pouring cup? This problem is similar to the construction of low-cost drainage networks, an application first explored by Lee [2]. He devised a two-pass algorithm that gave an approximate solution to this plumbing design problem. In Lee's work, as well as that described herein, we seek to minimize the weighted length of the network — subject to certain constraints. In the case of a low cost drainage network for a multi-story building, it is desirable to have all piping in the walls and floors. Thus, the network connections are confined to a three-dimensional lattice. Others have included additional geometrical constraints when seeking optimal networks. Drezner and Wesolowsky [3] computed optimal facility locations on the surface of a sphere and Hansen, Peeters and Thisse [4] devised an algorithm to locate a facility, in two-dimensions, with polygonal exclusion regions.

When designing practical runner systems we restrict any runner segment from piercing the part. This implies that we compute a fundamental quantity: the shortest distance between two points on the surface of the part. This quantity is difficult to compute for many complex castings. In Section

4 we present a schema for computing the shortest paths between two points in obstructed space. This fundamental calculation is used extensively by the algorithm presented in Section 3 to find low-cost runner networks. Finally, a comparative study of efficient runner design for an actual casting is discussed in Section 5.

2. The Rigging/Network Model

In this section we endeavor to build a mathematical model using graph theory from which one can compute an efficient rigging design for a gated part. Gate locations for investment castings are typically specified by a design engineer with consideration given to both filling and feeding the casting. For the purposes of this work, gate locations can be viewed as the intersection points between the liquid metal delivery system and the part itself.

We assume that the coordinate locations of the gates define the *nodes* of a graph. These nodes are classified as *sinks*, that is, there is one connection directed toward each such node. If there are no connections directed away from a sink it is defined as a *proper sink*. To facilitate graph construction, the gate coordinates may be extracted from a solid model description (*e.g.,* Pro/ENGINEER®) of the gated part, if such an electronic data base exists. In addition to the gate locations, the position of the pouring cup is added to the collection of nodes and is called the *proper source* (there are no connections directed toward this node). For most investment castings, the number of nodes in the associated graph will not exceed 20; we have worked on a few castings in which the number of sinks has been in the thirties. Additional nodes may be added to the graph which serve as junctures among three or more nodes. These intermediate nodes are analogous to the so-called Steiner points [5-6] used in minimum spanning networks or Weber points [7-8] often used in spatial economics problems. These juncture nodes facilitate connections among the sinks and can reduce the overall network cost. By definition, juncture nodes are sinks which have both incoming and outbound connections.

Connections between nodes are called *edges*. Each edge has associated with it a diameter *d*, a length *l* and a direction vector — from one node towards a different node. The direction of an edge corresponds to the gravity-induced, flow direction of the liquid metal in the runners. Even though we choose to associate a diameter with each edge, this does not necessarily infer cylindrical runner segments. The optimization algorithm is not dependent on the shape of cross-section, only its area. The diameter is convenient, in that, it is a single parameter that is indicative of cross-sectional area. The diameter of an edge that emanates from a proper sink is taken to be the effective diameter of the gate — one that produces the same cross-sectional area. Gate areas are specified by the casting engineer and, thus, are assumed to be given for our purposes. This raises an interesting question. Since we are designing a filling/draining network, how should the diameters of the various branches change at a juncture, say, one edge bifurcating into two or three branches? As a rule, the diameters of tree edges will increase as one traverses a path from a proper sink to proper source (the root). At each juncture we propose that the diameters of joining segments satisfy the following equation

$$d_i^p = \sum_{j=1}^{N} d_j^p, \qquad \text{EQ(1)}$$

where the subscript i denotes a single incoming edge, j cycles through the outgoing edges and the exponent p controls the degree to which the diameters change across the juncture. The flow-carrying capacity of the network, in general, is quite sensitive to the choice of p. Figure 1 graphically

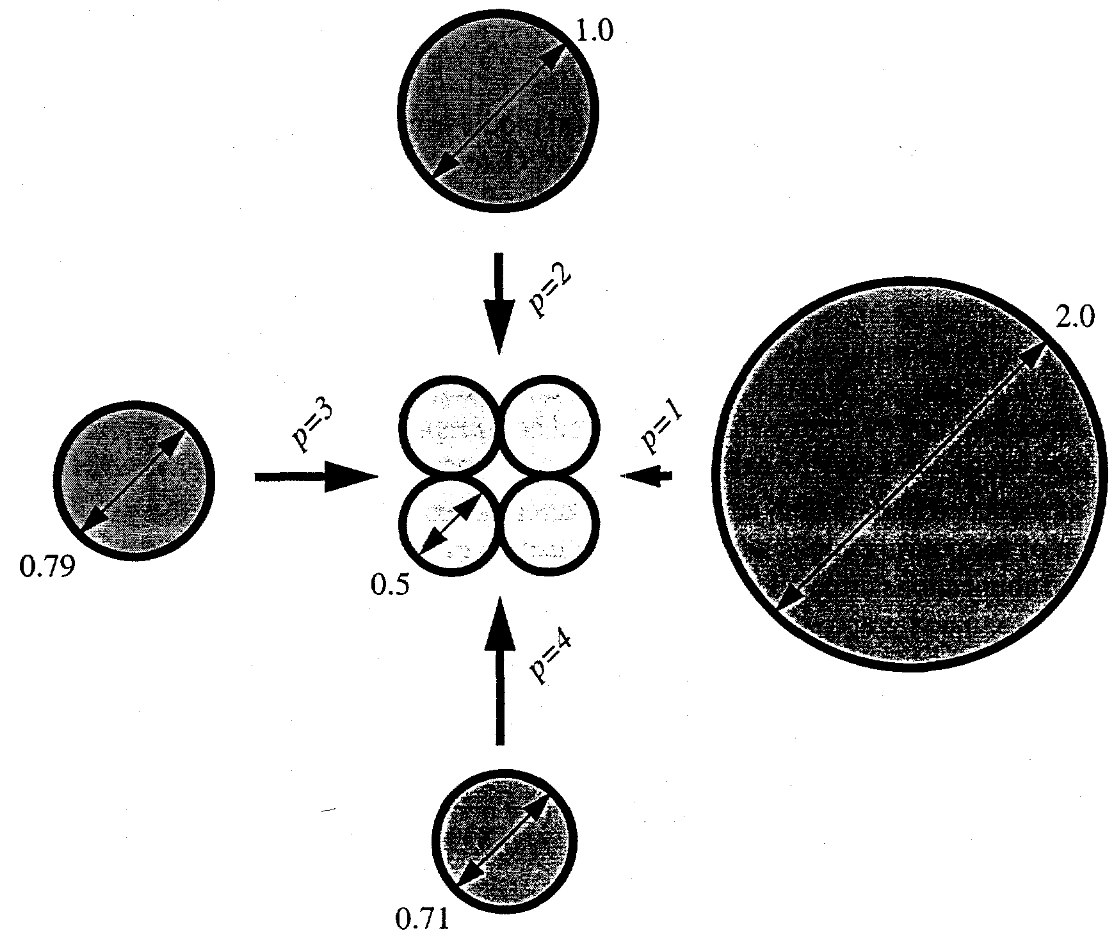


Figure 1: Diameter changes in network branches across a juncture. We illustrate the various possibilities when one branch (dark gray cross-sections) divides into four branches (light gray cross-sections). Four cases shown are for different values of the exponent p in EQ(1).

illustrates the relationship defined by EQ(1); given four downstream branches (each with a diameter of 0.5), we illustrate how the diameter of the incoming branch can change, for various p, according to EQ(1). The four cases depicted in Figure 1 correspond to the integer values of p such that $1 \leq p \leq 4$; this parameter range produces a wide variation in the diameter of the incoming segment. We typically choose a value of the exponent between $2.0 \leq p \leq 2.5$. To support our choice for p, we note that $p=2$ preserves the cross-sectional area on both sides of a juncture. This geometric condition corresponds to mass conservation of flow and is a nominal benchmark for good manifold design.

Taken together, the nodes and edges define a graph G. We seek an optimal branching network that contains no circuits. This means that our graph must be acyclic — no directed path leads from any node to itself. To permit such a feature would lead to an inefficient and non-optimal runner network. Finally, for complete filling we seek a graph in which a path can be found from the source to every sink. The set of criteria that we have asserted will impart monotonicity to G. When drawn, a monotonic graph resembles that of an inverted tree: a network that possesses a main trunk, branches and petioles. In constructing such a graph we seek to minimize the volume of the network given by the cost function C defined as

$$C = \sum_{i=1}^{M} d_i^2 l_i \qquad \text{EQ(2)}$$

where the index i ranges over the number M of network segments. While this cost function appears to be rather simple, we believe it produces good rigging designs. We have already noted that we seek to minimize the rigging volume. This is a good design feature from an economic point of view, especially when casting expensive alloys. Less material is wasted in the rigging system. Moreover, the algorithm used to construct a suitable network to satisfy EQ(2) also seeks to minimize the path lengths from source (pouring cup) to sinks (gates). This aids the casting process, in that, both loss of super heat and dynamic pressure will be kept to a minimum.

3. A Low-Cost Network Algorithm

An algorithm that constructs a low-cost network, and that was inspired by the work of Lee [2], is presented in this section. The algorithm consists of two separate parts; we denote them, as did Lee, as PASS(A) and PASS(B). In PASS(A) we construct a spanning tree which connects all gate locations to the pouring cup.

(A1) Among sink nodes, yet untagged, find the node i furthest from the proper source and construct a link directed from i's closest neighbor to i itself. Put i into a tagged group, record the link attributes and repeat this step until no untagged nodes remain or a circuit is established.

By successively choosing the sink node furthest from the proper source to construct each new link, a directed network will be formed (see Figure 2). The resulting tree will be inverted — its trunk up and branches down — this assumes that the pouring cup is positioned above the gates. From repeated application of step(A1) it is apparent that every sink node will have at least one link directed toward itself and some of the sink nodes will have a link (perhaps more than one) directed away from itself; these latter nodes are juncture nodes. The diameter of the incoming link to a juncture node is computed from EQ(1).

Occasionally, successive applications of step(A1) will form a closed circuit. Figure 2 illustrates this situation. Here, two separate circuits have formed — the first one among nodes 2, 3 and 4 and a second one between adjacent nodes 5 and 6. To see how this can happen consider the circuit formation of the latter case. During the repeated application of step(A1) we arrive at node 5 (currently the furthest untagged node from the source). We find its closest neighbor (node 6) and construct a link from node 6 to 5 and subsequently tag node 5. Now we consider node 6 — the last untagged node. Its nearest neighbor is node 5 so we construct a link (dashed link) from node 5 to 6, thereby, creating a two-member circuit. A similar discussion will verify the formation of a circuit among nodes 2, 3 and 4 — it is assumed that these nodes are equidistant from one another.

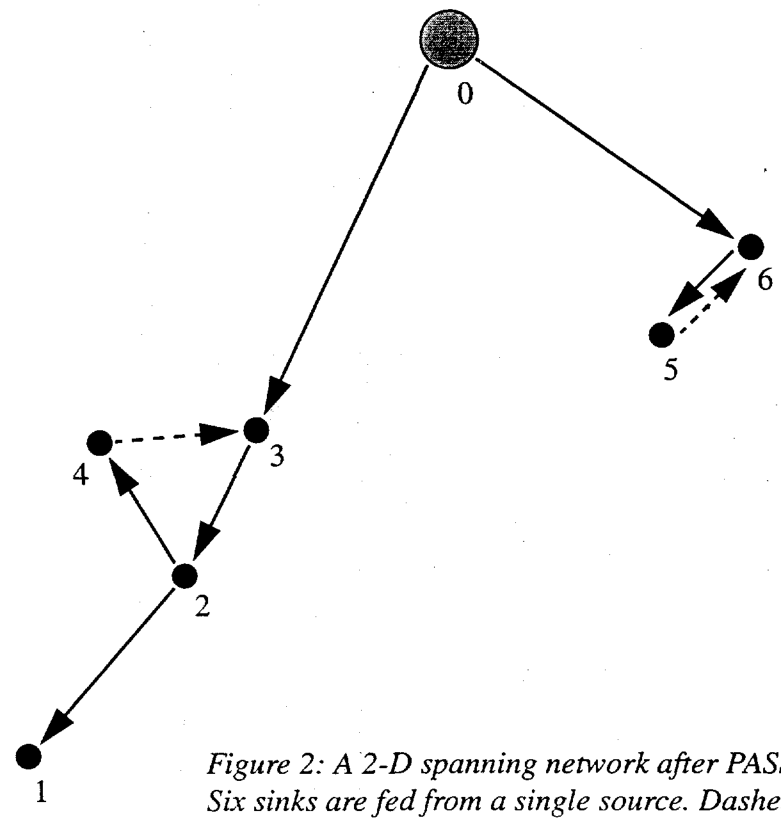


Figure 2: A 2-D spanning network after PASS(A). Six sinks are fed from a single source. Dashed links are removed from the network by step(A2)

As soon as a circuit is detected it is resolved (broken) by executing the following instruction.

(A2) Break a circuit by removing one branch at a time. Find the minimum cost connection (using EQ(2)) into the broken circuit from each of the remaining untagged nodes. This is accomplished by full enumeration over the sequentially broken circuit and untagged nodes.

Again, Figure 2 may be used to illustrate how a closed circuit may be resolved. Consider the case of the circuit formed between adjacent nodes 5 and 6. Application of step(A2) will remove first, let's say, link $[6 \rightarrow 5]$. The broken circuit is then joined, in turn, to all untagged nodes. For the graph depicted in Figure 2 the only untagged node, when the circuit is formed, is the proper source. The proper source is included in the list of untagged nodes from which a connection into a circuit can be made. It is not, however, included in the list of untagged sinks used in step(A1). The cost of this subarborescence $C(I) = [0 \rightarrow 5 \rightarrow 6]$ is computed from EQ(2) and stored. Alternatively, the original closed circuit can be broken by removing link $[5 \rightarrow 6]$ (dashed link). As before the minimum cost connection into this newly broken circuit is accomplished with link $[0 \rightarrow 6]$. The cost of the subarborescence $C(II) = [0 \rightarrow 6 \rightarrow 5]$ is less than realization I and, therefore, is chosen to resolve the original circuit. In proclaiming that $C(II) < C(I)$ we have assumed that node 6 is closer to the root than node 5 and that the gate diameters corresponding to nodes 5 and 6 are equal.

Step(A2) of the algorithm uses full enumeration to resolve a circuit, *i.e.,* we connect each broken circuit realization with each untagged sink node. At first, this may seem prohibitively expensive. It turns out not to be the case. Full enumeration proceeds rather rapidly since both sets are usually small. The total number of gates (nodes) for most investment castings is usually less than 20 and most of the circuits encountered will include only 2 or 3 nodes.

The cost of a spanning network that results from PASS(A), represented graphically in Figure 2 by the solid links, may be further reduced by the introduction of Weber points [7]. These are interme-

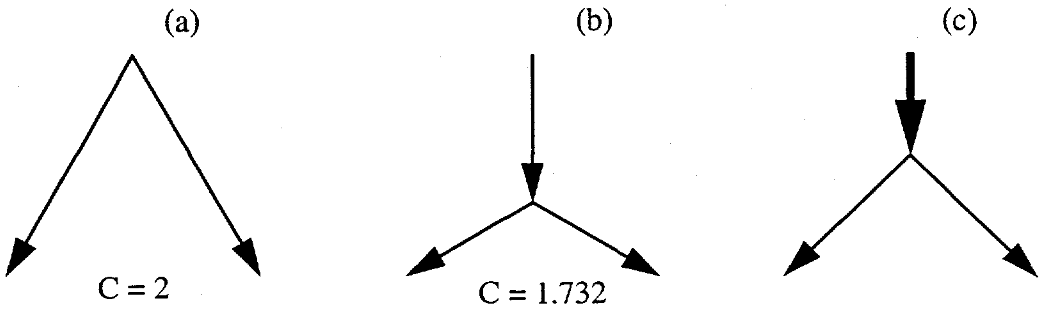


Figure 3: Reducing network costs with the introduction of Weber points.

diate branch points added to the network that reduce the overall network cost by virtue of their location. Their effect can best be illustrated with a simple example. Consider node locations that coincide with the vertices of an equilateral triangle — unit length on a side. Figure 3a shows a network that connects all three nodes with two links. If we assume that each node has unit weight (*i.e., d =1* in EQ(2)) then the cost of connecting the vertices is C = 2. Figure 3b illustrates how one may reduce the cost of the network with the introduction of an additional node. The optimal location for this extra node (Weber point) assuming equal node weights is at the centroid of the triangle. As before, the network depicted in Figure 3b connects all vertices, but does so at least cost, C = 1.732. Figure 3c illustrates a case in which the nodal weighting is not uniform. Here, we assume that the upstream node weight (*i.e.* diameter) is greater than 1, but less than 2. In the case of our runner network design this would account for an upstream link diameter being greater than the downstream members. The increased nodal weight tends to pull the position of the Weber point closer to that node. Figure 3c is only qualitative because the optimal location of the Weber point is dependent upon the weights assigned to the nodes. PASS(B) is comprised of three steps which seeks to distribute Weber points to minimize the network cost.

> *(B1) Identify all juncture nodes that result from PASS(A); for each juncture node compile a list of incoming and outgoing branches.*

Recall, from the definition, that a juncture node is a node that has both incoming and outbound links. Nodes 2, 3 and 6 represent juncture nodes of the example network in Figure 2. Also, associated with the juncture nodes is the concept of link-level. Simply stated, the link-level of a juncture node is the number of discrete links to the proper source. Again referencing Figure 2, nodes 3 and 6 are one link-level removed from the source while node 2 is two link-levels downstream from the pouring cup.

(B2) Cycle through the juncture nodes; start with the one that is the most link levels removed from the source. Compute the Weber location for the subarborescence associated with each juncture node. If the cost of the subarborescence (with the newly computed Weber node) is less than the original subarborescence, accept the new subarborescence. Update juncture node and link lists, if necessary.

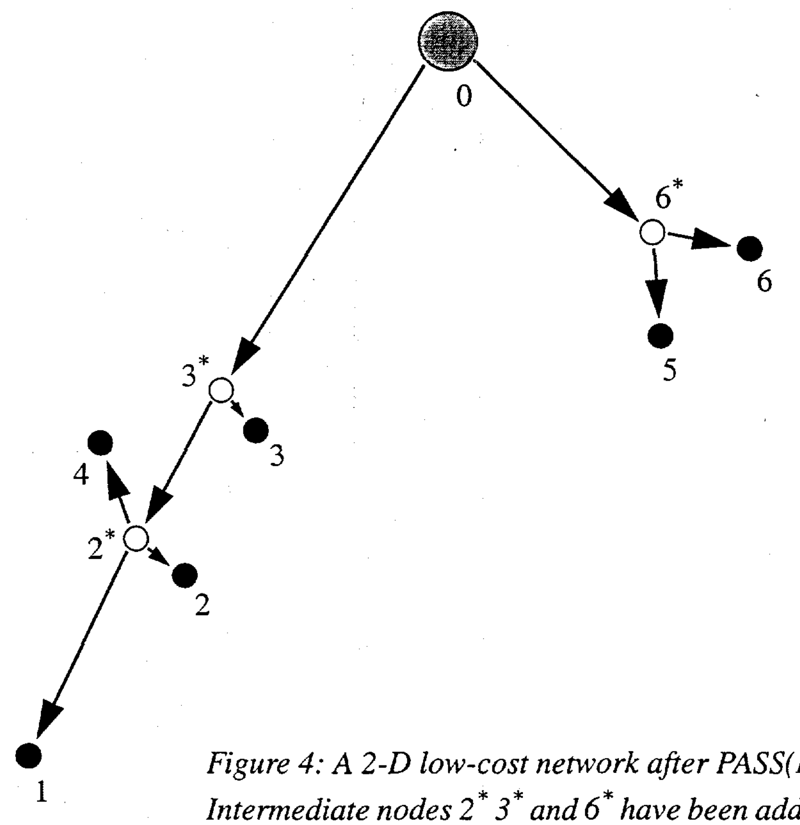


Figure 4: A 2-D low-cost network after PASS(B). Intermediate nodes 2^* 3^* and 6^* have been added to the network to reduce its total cost.

Application of step(B2) to the spanning network produced by PASS(A) is illustrated in Figure 4. New nodes 2^*, 3^* and 6^* (unfilled circles) have been added to the network because in each case the subarborescence cost associated with each juncture node was reduced. In some cases the calculated Weber point will not be far from the original juncture node. This is due to nodal weighting of the local subarborescence. In fact, in some cases, it will not be possible to improve upon the location of the juncture node either because of large upstream weighting or rearrangement among connecting subarborescences. Notice that step(B2) only locally minimizes network cost. However, by repeating this step, slight rearrangements in neighboring subarborescences will influence the calculation of Weber locations in junctures closer to the proper source. In this manner the global characteristics of the network are accounted for in the minimization scheme.

(B3) Compute (new network cost); if (new network cost) - (old network cost) > tol, repeat step (B2). If not, exit.

Step(B3) defines a stopping criterion for the algorithm. After each complete cycle of step(B2), we compute the total network cost from EQ(2) and compare its cost with the previous iteration. When the difference falls below a predefined tolerance, we accept the current network and terminate the algorithm. The tolerance value, *tol,* is usually initialized to a few percent of the tree cost that results from PASS(A). Our experience has been that three or four iterations of (B2) is sufficient to satisfy (B3) when *tol=0.005*(network cost from PASS(A)).*

finding Weber points

The critical operation of PASS(B) is the calculation of Weber points. The coordinates $\underset{\sim}{\rho}$ of a Weber point can be found from the solution of a local cost minimization problem on a subarborescence (see Figure 5). We seek a solution to

$$min \sum_{i=1}^{N} d_i^2 \|\underset{\sim}{\rho} - \underset{\sim}{x}_i\| , \qquad \text{EQ(3)}$$

where i ranges over the number of nodes with a direct link to the juncture node; $N = 4$ for the subarborescence illustrated in Figure 5a. The distance between the juncture node and the ith subarborescence endpoint is expressed by $\|\underset{\sim}{\rho} - \underset{\sim}{x}_i\|$. Figure 5a pictorially shows how the

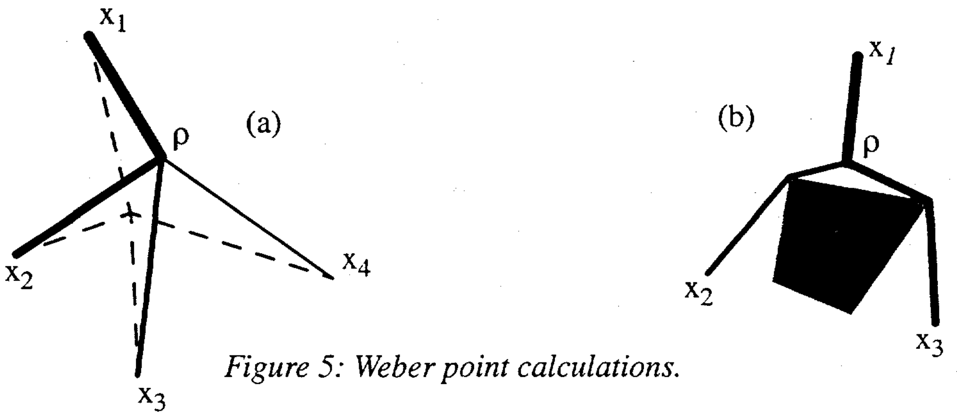


Figure 5: Weber point calculations.

subarborescence changes when the juncture point moves. The dashed network depicts an alternative realization of the subarborescence as the best location of ρ is sought. Line thickness is representative of segment diameters which are assigned to the nodes as weights. For the curious reader, Wesolowsky [8] has written a historical review of this most interesting problem; it includes solutions to EQ(3) by both geometric construction techniques and mechanical devices. However, what is crucial to our work is a method to compute these points efficiently; one such algorithm that accomplishes this task is described in the paper by Kuhn and Kuenne [9]. They devised an iterative algorithm to find Weber points when the distance between two points is given by the Euclidean (straight-line) metric (see Figure 5a),

$$\|\underset{\sim}{\rho} - \underset{\sim}{x}\| = \sqrt{(\rho_1 - x)^2 + (\rho_2 - y)^2 + (\rho_3 - z)^2} \qquad \text{EQ(4)}$$

where x, y and z are the Cartesian coordinates of a given segment endpoint $\underset{\sim}{x}_i$. EQS(3-4), taken together, define a problem that is often referred to in the archival literature as the unconstrained Weber problem. It can be verified that EQ(4) will render the function given by EQ(3) continuous and convex. Consequently, the success of the K&K algorithm is intimately tied to the analytical form of EQ(4); *i.e.*, it is differentiable with respect to ρ and it can be manipulated algebraically. We have used the K&K algorithm with good results to compute Weber points when the distances between points can be computed explicitly from EQ(4).

Figure 6 illustrates one such example problem and will help to demonstrate various aspects of the algorithm. Figure 6(i) depicts a perspective view of randomly placed nodes in three-space — 16 points dispersed in a box. The overall dimensions of the box are $W(1) \times D(1) \times H(1.5)$. These nodes represent sink locations and for this example, all sink locations have equal weight, *i.e* equal gate diameters.Those points that appear slightly larger are, in fact, closer to the front of the box while the slightly smaller nodes are further to the rear of the box. We have designated the node on the lid of the box to be the proper source.

PASS(A) of our network algorithm produces a spanning tree that is illustrated in Figure 6(ii). The cost associated with the network at this stage is C = 3.16 — the absolute value of this cost is not important but we will use this number to identify cost reductions that result from PASS(B). Notice the characteristic structure imparted to the network from PASS(A). It has a preferred direction, the diameters of the segments are increasing with decreasing link-level and no circuits are unresolved. Figure 6(iii) illustrates the network after the first iteration of Weber point calculations. Qualitatively, it is apparent that many of the sharp kinks in the network are removed by the addition of these extra branch points. Also of significance is the relative reduction in network cost, C = 2.80 in Figure 6(iii). This example is typical of our experience in the reduction of network cost; the largest reduction coming after the first cycle through step(B2). The converged network is shown in Figure 6(iv). This final network is the result of three additional cycles of step(B2) producing a network with cost C = 2.72. Notice that the network appears to become more streamlined as the algorithm cycles through step(B2). The effect of this added computational effort has been to iteratively rearrange the Weber sites as the entire network relaxes toward the lowest cost. The exponent, p, used in EQ(1) was 3.0 for this example.

The exponent, p, that controls diameter growth at the juncture nodes can have a profound effect on the structure of the resulting network. To illustrate this fact consider the series of networks displayed in Figure 7. The sink positions of these networks are exactly the same as those of the previous example shown in Figure 6, however, the viewing direction is different; images in Figure 7 are left side views. Clockwise from top left, the networks in Figure 7 corresponds to different values for the exponent p used in EQ(1); *viz.* 2.05, 2.10, 2.25 and 3.00, respectively. The difference in network structure is due solely to a changing growth model for the diameters at a juncture node. From the results plotted in Figure 1 it is apparent that the diameter of the upstream branch at a Weber point grows inversely with the value for the exponent p. Because a large diameter segment contributes so overwhelmingly to the overall network cost, especially if it is far from the proper source, the algorithm will necessarily try to shift all Weber locations toward its upstream node, or

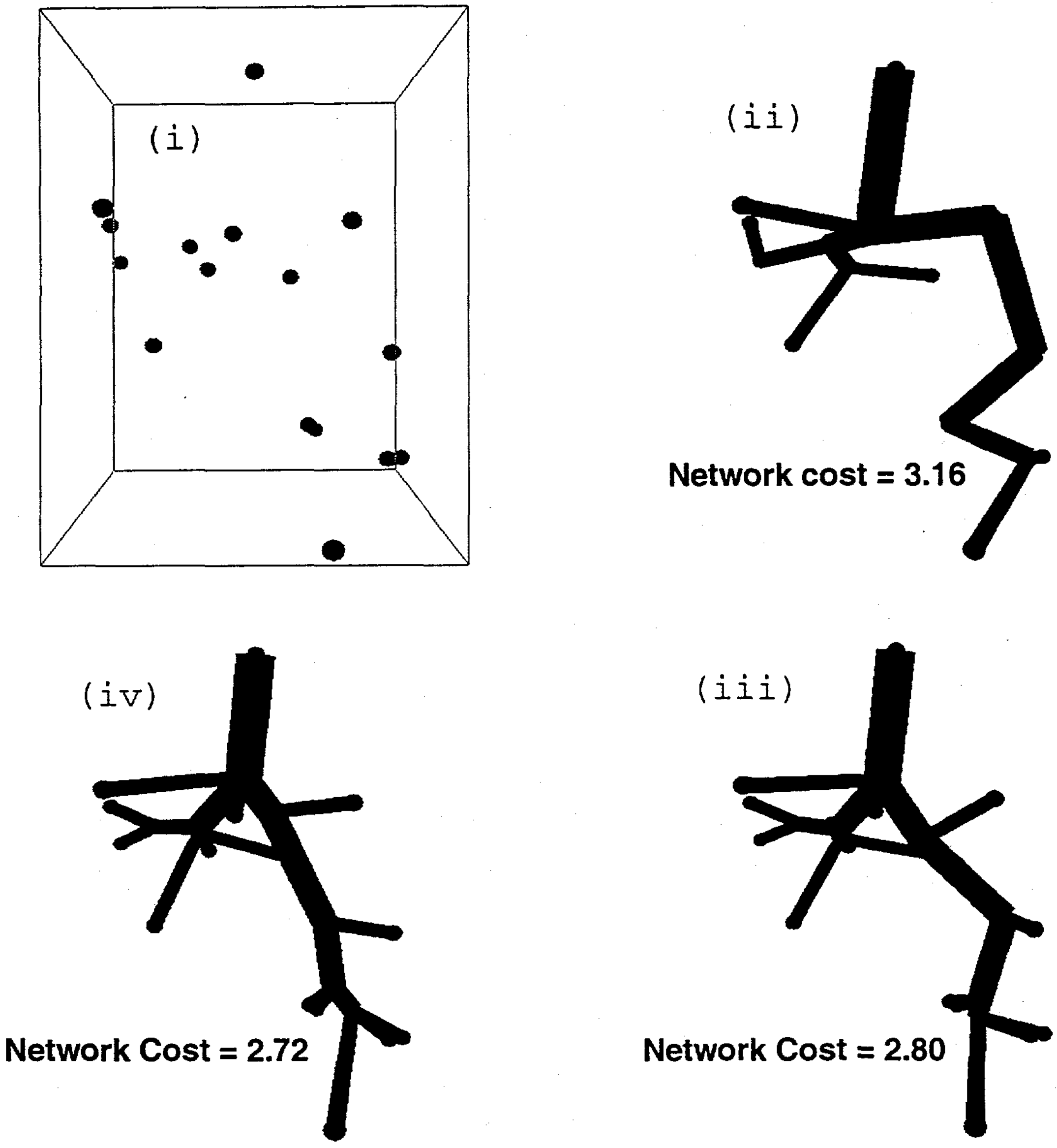


Figure 6: Finding a low-cost network. Clockwise from top left: (i) perspective view of 16 points in three-space that are to be connected, (ii) minimum spanning network resulting from PASS(A), (iii) network after completing first iteration of PASS(B), (iv) network resulting from converged algorithm. Network costs are computed from EQ(2). The segment diameters of the arborescence increase as the root is approached (p = 3.0 in EQ(1)).

beyond (see Figure 3c). Thus, for small p, there will be many segments trying to make connections as high up on the tree as possible — this trend lowers the cost for a given subarborescence and necessarily the overall network cost. This is clearly the case if one scrutinizes the top networks in Figure 7. These networks are illustrative but not practical for runner systems. Conversely, as the exponent p is increased, networks with more branching will result as coalescing of downstream segments becomes proportionately less costly. These bottom networks in Figure 7 represent higher values for the parameter p. For comparative purposes the overall costs for these networks are given in the caption of Figure 7. We have not determined the optimal value of the parameter, *p*, for runner networks based upon flow and heat-loss models. This will require further study.

When the distance between two points is no longer directly computable from EQ(4), the calculation of Weber points becomes more difficult. Under what conditions might EQ(4) no longer apply? Suppose, for example, that there are obstacles that force connecting paths to bend or deviate from the usual straight-line (see Figure 5b). How can one find solutions to EQ(3)? What if the Weber node is forbidden to lie within some exclusion region? This is precisely the case encountered during runner design. The forbidden zone is the cast part itself, therefore, one must restrict the placement of branch nodes or network connections from intersecting with the gated part.

In the next section we will describe an algorithm that computes the shortest path between two points in the presence of obstacles. In the general case, this particular calculation is required for both PASS(A) and PASS(B) of our network algorithm. For now, let's assume that such a computation exists and complete our discussion on finding so-called "constrained" Weber points.

We reconsider the problem of finding a solution to EQ(3) when, in general, the distance between two points can be computed, but not explicitly from EQ(4). Nelder and Mead [10] devised a simplex method for function minimization which can be used to find Weber points in the presence of exclusion regions. This situation is illustrated in Figure 5b. Here, we seek to find the coordinates of ρ, subject to constraints, such that the cost of the local subarborescence is minimized. The essence of the N&M algorithm compares the functional values (evaluated with EQ(3)) at *N* vertices of a general simplex — the vertex with the largest value is repeatedly replaced by a point whose functional value is smaller. New points are computed by deforming the original simplex. Parameters specific to the N&M algorithm that control reflection, contraction and expansion characteristics, which we list here for completeness are: $\alpha = 1$, $\beta = 0.5$ and $\gamma = 2$, respectively. The N&M algorithm contracts the simplex around the local minimum by adaptively fitting itself to the functional landscape. As the original authors point out, exclusion regions are easily handled by setting the value of the function arbitrarily high when a vertex of the collapsing simplex trespasses into a forbidden zone. When this happens the N&M scheme will retract that vertex until it resides in an accessible region. It follows that Weber points on the surface of exclusion regions are inaccessible, in general, though arbitrarily close approaches can be made.

The N&M algorithm is a local minimization algorithm. It cannot guarantee that a global minimization value will be found. Despite this apparent limitation, we have used the N&M algorithm with good success in actually finding global extrema; we believe this is due to the shape of the functional landscape. More often than not, the functional landscape of EQ(3) is fairly smooth, shaped like the bottom of a shallow bowl. In this instance the algorithm does indeed find the best global position for the Weber point.

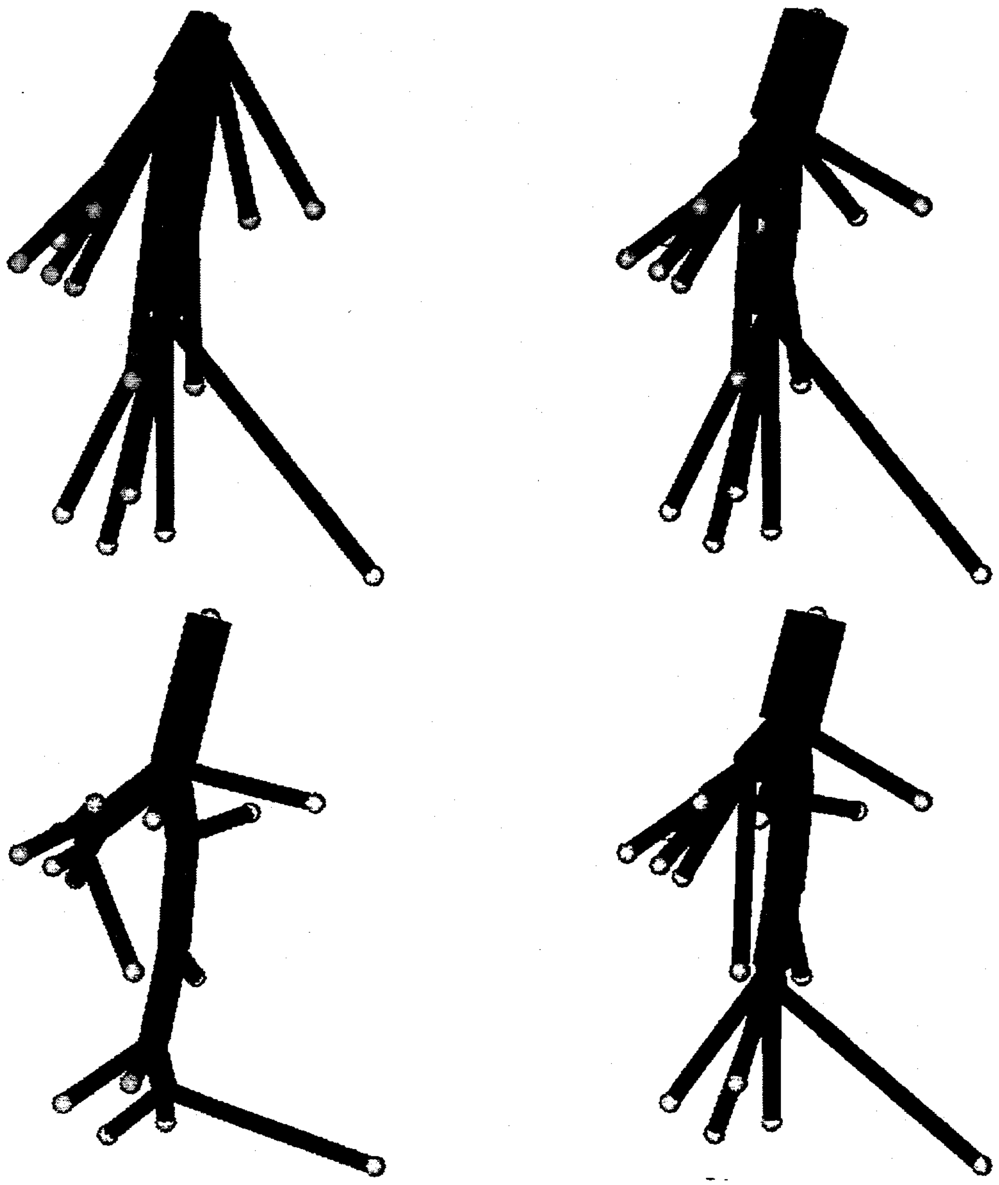


Figure 7: Structural changes to a network by varying the rules at a juncture; these networks all connect the same points. Clockwise from top left the exponent p, in EQ(1), is assigned values of 2.05, 2.10, 2.25 and 3.00, respectively. The corresponding costs of the networks are C= 3.83, 3.87, 3.60 and 2.72.

We have added one enhancement to our implementation of the N&M algorithm which enlarges the initial simplex. If one begins the N&M algorithm with only 3 vertices (the case in which one branch feeds two branches downstream), the algorithm will only search for minimum points in the plane defined by the initial vertices. This follows directly from the N&M rules for manipulating the simplex. To enlarge the initial simplex and force N&M to search in 3D, we introduce a fourth vertex out of the plane. This preconditioning improves the placement of the branch points especially when the exclusion regions are geometrically complex.

4. Shortest Paths Through Obstructed Space

Finding the shortest path between two points in three-space, while avoiding an arbitrarily-shaped exclusion region, is currently an unsolved problem in computational geometry. Recognizing this fact we describe a procedure, in this section, that gives a close approximation to this shortest path problem. It will be helpful to refer to the schematic drawing in Figure 8 as we describe this three-step process. We seek the shortest paths from point *O* to points *A* and *B*. The paths may not intrude into the restricted region — in this example the forbidden area is the shaded circle missing a wedge-shaped cutout. Points *A* and *B* happen to lie on the surface of the exclusion region; point *O* does not. Recalling our underlying motivation of efficient runner design point *O* represents the proper source, or pouring cup, and points *A* and *B* correspond to gate locations on the surface of the casting. While Figure 8 is a two-dimensional representation of the shortest path problem, the sequence of steps described next are equally valid, yet more tedious to implement, in three dimensions.

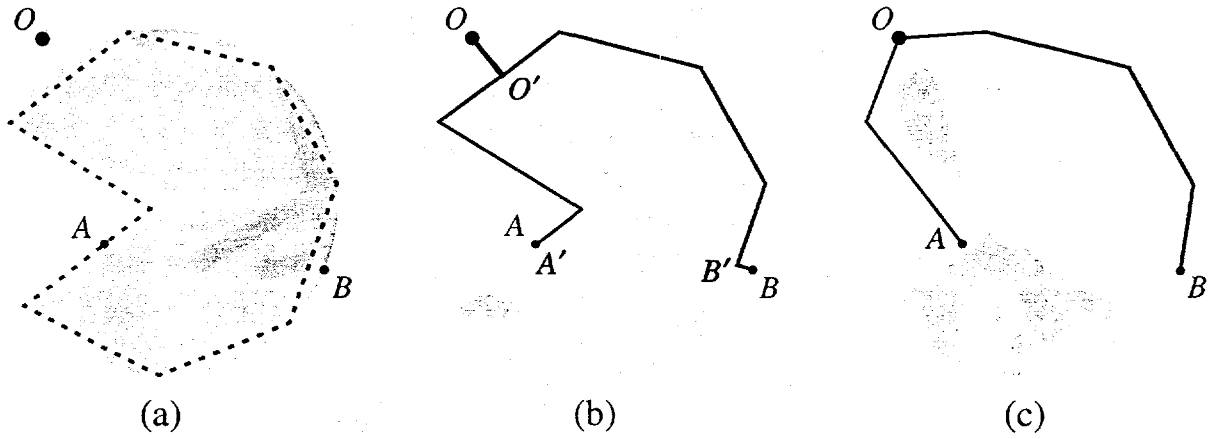


Figure 8: A sequence of steps used to illustrate the calculation of shortest paths through obstructed space.

We begin with a faceted description of the exclusion region (effectively the cast part). Figure 8a shows a coarsely-segmented perimeter using a dotted line. In three-dimensional situations a solid model of the part (typically constructed with Pro/ENGINEER®) is faceted with triangular faces. The number of surface triangles that results from the facetting operation is controlled by user input. One of two criteria can be specified to limit the number of facets on non-planar surfaces: (i) maximum angle between normals of adjacent facets or (ii) maximum offset distance between a facet centroid and the original surface. It is best to work with enough facets that accurately represent the surface of the original solid model, but not too many, as the computer run-times for the shortest-path computations are proportional to the number of surface triangles. As a point of reference, the surface description of the fireset housing shown in Figure 9b contains 642 triangles.

Notice upon close inspection of Figure 8a that two points, namely *O* and *B*, do not lie on the *facetted* surface. This will be the case more often than not since gate locations are specified on the surface of the solid model and not the segmented surface. When this happens, we find the nearest location on the faceted surface to points like *O* and *B* [1]— we call these new points surface endpoints; they are denoted with a prime($'$) in Figure 8b. For this example point A was initially on the faceted surface and, therefore A coincides with A'. Surface endpoints will usually lie within an existing facet. Because of the introduction of these surface endpoints additional local tiling is performed so that all surface endpoints coincide with a vertex of the now-enhanced surface triangulation.

The next step in the algorithm sequence is illustrated in Figure 8b. We construct the shortest path between a beginning point and any destination point along the surface of the faceted body — in our case, one path between O' and A' and another between O' and B'. Given a general three-dimensional surface, this calculation is not trivial. Mitchell, Mount and Papadimitriou [11] describe an algorithm to find the shortest path between two points on a polyhedral surface. With very little alteration we have successfully adapted that algorithm for our purposes here.

To calculate the shortest path on a faceted surface the MM&P algorithm effectively "unfolds" the entire surface by mapping it onto a plane. In this transformed reference frame shortest paths between two points become straight lines. Two important consequences result from this unfolding procedure. Firstly, shortest paths from one start vertex to every other vertex can be obtained with one, albeit expensive, unfolding process. We have structured the logic used in PASS(A)[2] and PASS(B)[3], to take advantage of this fact. Secondly, the unfolding process proceeds from the start vertex much like a spreading brushfire. Again, because the unfolding process is expensive, we terminate the unfolding process and compute the desired paths as soon as the brushfire has passed our destination vertices.

After using the MM&P algorithm to compute the shortest surface paths from start vertex O' to target points A' and B' additional segments are appended to the paths to join points that did not originally lie on the facetted surface. In Figure 8b the extra segments O-O' and B-B' are added to form the complete path O-B. At this point of the algorithm execution we have complete paths from point O to endpoints A and B. However, these paths may contain a series of segments that form concave contours; hence, the path length could be further reduced. The length of each path is lessened, if possible, with a "line-tensioning" procedure illustrated in Figure 8c. Interior points of a path are removed if, and only if: (i) the resulting path does not intersect the faceted body and (ii) its overall length is reduced. The length of this shortened path is used in all calculations described in Section 3 and the description of the path is retained for later use in plotting the final network.

5. A Comparative Study

We have chosen to demonstrate our runner optimization software on a part that has many gates

1. Geometrical computations like this, and others, are performed with function calls to the ACIS® test harness.
2. In PASS(A) when we need distances between the proper source and all sinks, we take the pouring cup location as the start vertex and all sinks as destination vertices.
3. EQ(3), which requires path lengths, is computed repetitively in PASS(B). We can compute all needed paths in one MMP unfolding if we associate $\underset{\sim}{\rho}$ with the start vertex and $\underset{\sim}{x}_i$ with the destination vertices.

and is geometrically non-trivial. Moreover, it was important for comparative purposes to choose an example problem that has been gated, rigged and cast in our foundry. Using the actual casting, we can make quantitative assessments of the differences between runner networks — in one case, gates were connected by an experienced casting engineer and in another an efficient network was calculated with the software described in Sections 3 and 4.

Now it may be argued that certain runner networks produced by the software will not work in practice. For example, optimized runner networks may be difficult to remove from the part once it is cast. This could be caused by exceedingly small clearances between the runners and the part itself or regions that are now inaccessible to cut-off saws or torches. Also, some designs could make the investing and sanding operations difficult. Or perhaps, a nearby runner will alter the solidification rate of the part in its vicinity and, thereby, affect the casting quality. Each of these situations would be recognized by an experienced design engineer.

When these singular design considerations are important, we believe that either the cost function or rules used to generate the runner network could be altered to account for these additional design concerns. If it turns out that just one or two runner segments violate some design 'rules of thumb' then these branches might be more easily altered selectively. Despite the possible limitations we believe that there is great intrinsic benefit to a casting engineer to be able to visualize one or more efficient runner designs. Doing so may impart an entirely new thought process to a runner design that would not have been considered otherwise. For example, the network software may identify the path for a major trunk line that was not intuitively obvious. Moreover, this software may be used to generate low-cost runner networks for various casting orientations of the part. This is accomplished by merely changing the location of the pouring cup to produce an entirely different network.

One final observation is offered before closing this general discussion. Building runner systems has traditionally been a hand lay-up procedure. This fact, at first glance, would suggest that constructing complex runner systems, as predicted by the network software, might prove too difficult. However, adopting recent advances in rapid manufacturing processes it seems quite possible to build the runner system concurrently with the part using, say, a selective laser sintering process.

fireset housing

A fireset housing is a thin-walled casting used as a mounting platform for electrical components in the nose cone of a missile. The one shown in Figure 9 has nominal dimensions of 12 inches in diameter and 4 inches high Two large through-holes provide passageways for electrical and mechanical connections; these cut-outs also provide a good test for the optimization software. We have simplified the model of the fireset housing slightly by omitting small bosses, ribs and fillets that were part of the actual casting. Omission of these features does not alter the topology of the model, but does allow the surface of the part to be represented with fewer triangles.

Solid model renditions of the fireset housing are shown in Figure 9. A feature-based wireframe model, constructed with Pro/ENGINEER®, is shown on the left; a translucent view of the faceted solid model is displayed in 9b. In both images the 24 gate locations on the surface of the part are

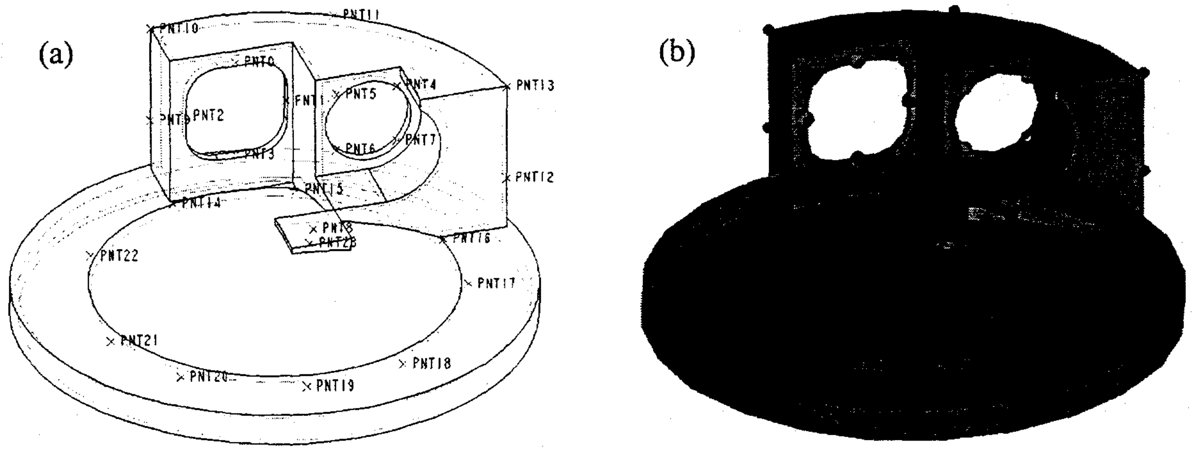


Figure 9: Solid model of the fireset housing with highlighted gate locations.

indicated. The diameter of the gate assigned to each location is given in Table 1. Notice that the gate locations are distributed all over the model surface. The pouring cup is located approximately 5 inches off the concave surface near its middle. This specification will determine the orientation of the part during casting. It is the same orientation used for the actual casting. Figure 10 shows the cast fireset housing and runner system after the ceramic mold was removed. Gate locations and diameters for the model in Figure 9 correspond to those used to cast the part. Actual gate cross-sections were recomputed as equivalent circles for use in the model.

node ID	*d (in)*	node ID	*d (in)*	node ID	*d (in)*
PNT0	0.56	PNT8	0.56	PNT16	0.60
PNT1	0.56	PNT9	0.40	PNT17	0.60
PNT2	0.56	PNT10	0.40	PNT18	0.60
PNT3	0.56	PNT11	0.40	PNT19	0.60
PNT4	0.40	PNT12	0.40	PNT20	0.60
PNT5	0.40	PNT13	0.40	PNT21	0.60
PNT6	0.40	PNT14	0.60	PNT22	0.60
PNT7	0.40	PNT15	0.60	PNT23	0.85

Table 1: Gate diameters for the fireset housing.

A few statistics regarding the rigging design are worth noting. Total volume in the runner network was 105 in^3. This value was calculated by measuring the volume of each runner segment with a flexible rule; the volume of the 8 inclined vents was not included in this estimate. The volume of the fireset housing itself was 37 in^3 computed internally by the solid modeling software. In addition, its surface area was 373 in^2 which yields a ratio of surface area to volume of about 10 — a value characteristic of a thin-walled structure. Finally, we note that there is nearly 3 times the volume in this runner network as in the part itself!

The runner network generated by the optimization software for the fireset housing is illustrated in Figure 11. Two views of the same network are shown — (a) one view from above and (b) one looking from the underside. The network exhibits the characteristic structure of an inverted tree. The diameter of the downsprue, which is calculated by the program, was found to be 1.6 inches; the value of p in EQ(1) was 3.0.

The runner network is computed as a sequence of connected line segments. Thus, they will in some instances lie directly on the surface of the part. This will always be the case when the runners are tracking around a convex surface. This does not violate any of our assumptions used to construct the network as long as the runner segments do not pierce the part. In order to generate the plots shown in Figure 11, these line segments were transformed into cylinders. This explains why a few runner segments appear to overlap the part. These segments must be offset from the part to allow for the mold layer. This problem still needs to be addressed before this technology can be used in a turn-key mode for investment castings. This troublesome dilemma is somewhat alleviated if we begin the optimization software using a gated part. The presence of the gates serve as *de facto* offsets. In this case, the resulting runner system will be less prone to exactly trace the part surface. In related applications for this network optimization software, *i.e.*, path calculations that minimize robot motion, the offset problem may not be an issue.

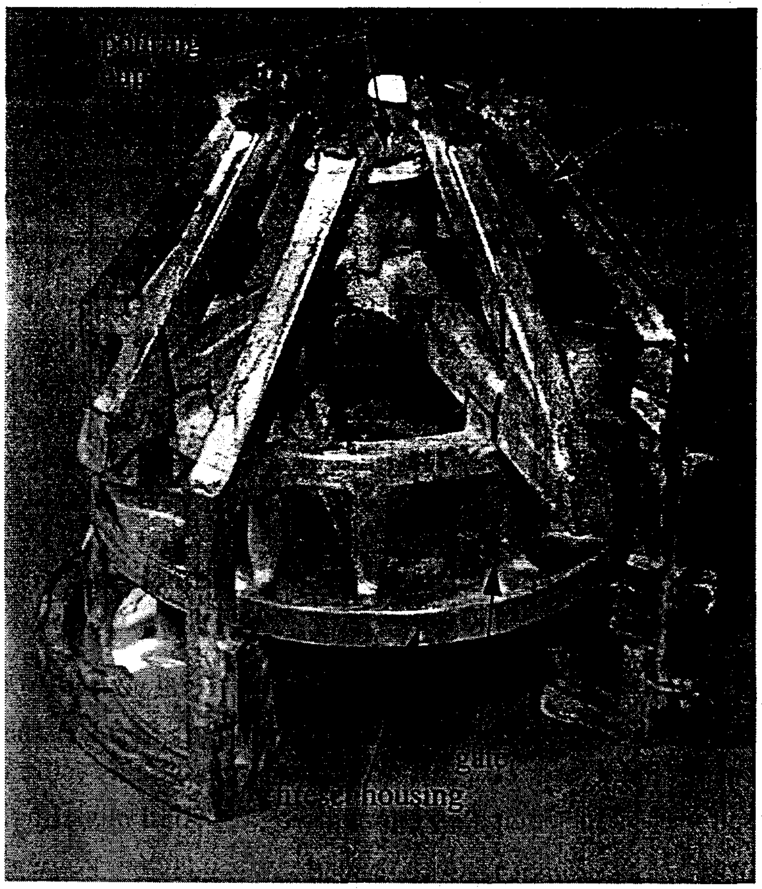


Figure 10: Actual rigging used to cast the fireset housing.

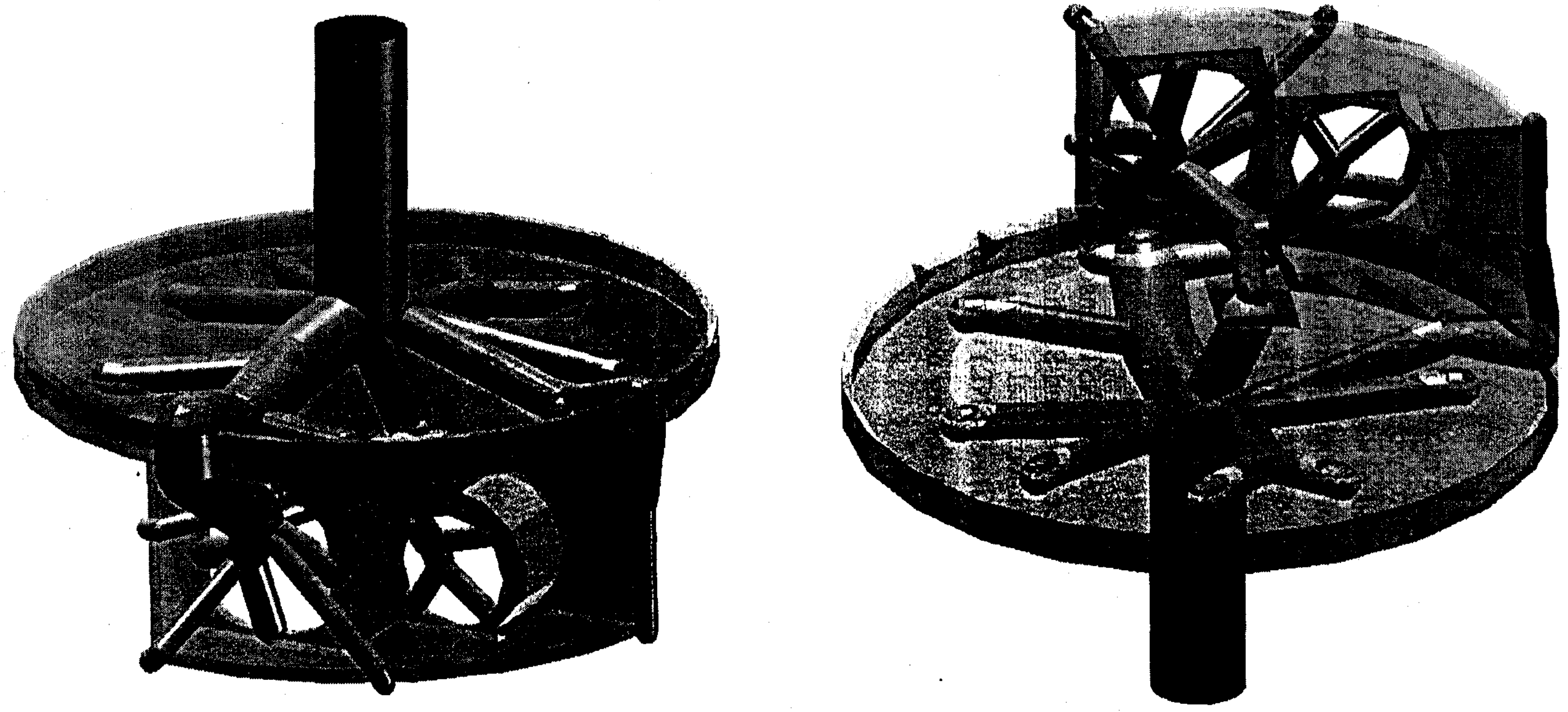


Figure 11: Optimal rigging design (blue) computed for the fireset housing (grey). Twenty four (24) gates on the surface of the casting are fed with a runner network containing 36 in^3. This represents a reduction in rigging volume of 66% when compared to a rigging design in production (105 in^3).

The volume of the runner network illustrated in Figure 11 was calculated to be 36 in^3. This represents a reduction in rigging volume of 66% when compared to the rigging design featured in Figure 10!

There were 642 triangular facets in the surface model of the fireset housing. Five cycles through PASS(B) required 22 hours on an UltraSPARC workstation. Our experience has been that, on average, 10-15 iterations of the N&M algorithm were needed to find an acceptable estimate for a given Weber point. Obviously, the number of simplex iterations depends on the tolerance limits used for Weber point convergence and the local geometry of the part. The code was terminated when the percent change in network cost between successive cycles of PASS(B) was 0.05.

6. Acknowledgments

Funding for this project was provided by the FASTCAST consortium — a partnership among foundries, companies that support casting activities and Sandia National Laboratories. Their support is gratefully acknowledged. The three-dimensional visual displays of runner networks were made possible by the creative work of Mike Glass (SNL) who developed a graphics application package, using the AVS/Express® environment, for our use.

7. References

[1] Bern MW, Graham RL. The shortest network problem. *Scientific American* 1989; January: 84-89.

[2] Lee DH. Low cost drainage networks. *Networks* 1976; **6**:351-371.

[3] Drezner Z, Wesolowsky GO. Facility location on a sphere. *Journal of Operational Research Society* 1978; **29**:997-1004.

[4] Hansen P, Peeters D, Thisse JF. An algorithm for a constrained Weber problem. *Management Science* 1982; **28**:1285-1295.

[5] Winter P. Steiner problem in networks: a survey. *Networks* 1987; **17**:129-167.

[6] Smith W D. How to find Steiner minimal trees in Euclidean d-space. *Algorithmica* 1992; **7**:137-177.

[7] Weber A. *Uber den standort der Industrien*. Tubingen, 1909. (Friedrich CJ. trans.) *Theory of the location of industries*. University of Chicago Press: Chicago 1929.

[8] Wesolowsky GO. The Weber problem: history and perspectives. *Location Science* 1993; **1**:5-23.

[9] Kuhn HW, Kuenne RE. An efficient algorithm for the numerical solution of the generalized

Weber problem in spatial economics. *Journal of Regional Science* 1962; **4**:21-33.

[10] Nelder JA, Mead R. A simplex method for function minimization. *The Computer Journal* 1965; **9**:308-313.

[11] Mitchell JB, Mount DM, Papadimitriou CH. The discrete geodesic problem. *SIAM Journal of Computing* 1987; **16**:647-668.