Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 29-31, 2011, Washington, DC, USA

DETC2011/47906

PIPESYNTH: AN ALGORITHM FOR AUTOMATED TOPOLOGICAL AND PARAMETRIC DESIGN AND OPTIMIZATION OF PIPE NETWORKS

William R. Patterson Jr. and Matthew I. Campbell*
Department of Mechanical Engineering
Automated Design Lab
The University of Texas at Austin
Austin, Texas 78712, U.S.A.

ABSTRACT

This paper describes a design automation approach that combines various optimization research and artificial intelligence methods for synthesizing fluid networks. Unlike traditional software tools available today, this approach does not rely on having any predefined network topology to design and optimize its networks. PipeSynth generates its designs by using only desired port locations, and the desired fluid properties at each of those ports. An ideal network is found by optimizing the number and connectivity of pipes and pipe fittings, the size and length of each pipe, and the size and orientation of each fitting. A Uniform-Cost-Search is used for topology optimization along with a combination of non-gradient based optimization methods for parametric optimization. PipeSynth demonstrates how advances in automated design can enable engineers to manage much more complex fluid network problems. PipeSynth uses a unique representation of fluid networks that synthesizes and optimizes networks one pipe at a time, in three-dimensional space. PipeSynth has successfully solved several problems containing multiple interlaced networks concurrently with multiple inputs and outputs. PipeSynth shows the power of automated design and optimization in producing solutions more effectively and efficiently than traditional design approaches.

1. INTRODUCTION

This study focuses on a design approach, known as PipeSynth, that automatically synthesizes and optimizes fluid networks. Cost and performance are typically the driving factors for pipe network optimization and therefore the algorithm seeks to concurrently minimize the total cost and the overall pressure drop of each network. These two objectives are coupled with several constraint functions to ensure that only feasible solutions are generated. The first constraint verifies that all the specified inputs and outputs are connected, with no additional open ports present in the network. The remaining two constraints prevent the pipes from intersecting with each other, and from passing through physical objects, and boundaries. These spatial constraints are enforced by using rigorous computational geometry techniques to ensure realistic designs are produced. After several possible solutions are found, the best ones are presented to the user.

For network synthesis, PipeSynth focuses on constructing designs using rigid circular pipes and fittings and is designed to correctly evaluate a large variety of common fluid flows.

Existing fluid network software analysis tools can approximate the pressures and flow-rates throughout of user designed networks quite admirably [1, 2]. While some of these software tools can perform parametric optimization on parameters such as the pipe diameters, none of these programs can design and optimize a network from scratch, using real pipe fittings.

A unique feature of PipeSynth is its ability to concurrently construct and optimize multiple networks that share the same physical design space, but have separate, non-interacting fluids. These disjoined networks, such as separate hot and cold water lines, will be referred to as separate channels during the remainder of this report. By synthesizing and optimizing these channels in parallel, this approach assures that the best possible design can be found. Optimizing each channel successively would restrict the possible solutions and make the resulting solutions dependent upon the order in which the channels were optimized.

2. RELATED WORK

There has been a wide variety of approaches used to optimize fluid networks, everything from direct numerical solutions [3, 4], to genetic algorithms [5-7] and simulated annealing [6]. Numerous topology optimization studies have had success when constraining the network joint locations to either a

* Corresponding Author, Phone: (512) 232-9122, Fax: (512) 471-6356, email: mc1@mail.utexas.edu

predefined or a randomly generated grid [3,5-9]. However, all these approaches limit the optimization search space by constricting each network junction point to lie on one of roughly 10 to 100 predefined nodes. Then by their specific optimization technique, they determine if and how each of these nodes should be connected to each other. Many of the nodes are left unused, as long as the inlets (sources) and outputs (sinks) are connected. Other studies have focused on the parametric optimization of existing topologies [4, 10]. These papers focused on minimizing cost by balancing the pressure drops between different pipes and by controlling the pressure drops within their networks by optimizing the individual pipe diameters. One common characteristic for all these studies is that they do not account for the cost and geometric constraints encountered when using real pipe fittings.

Many of these same ideas, previously studied for pipe flow analysis, led to the development of commercially available fluid network software tools [1, 2]. Compared to traditional computational fluid dynamics (CFD), these software tools do not focus on individual particle flow behavior, rather they primarily consider the behavior of the mean flow throughout the networks. These commercial pipe flow analysis tools are aimed at large-scale civil and agricultural planning where predefined networks (topologies) exist [1, 2]. However, these programs still leave it to the user to decide on the system of fittings that will join the intersection points of the network found by the software analysis tool.

3. REPRESENTATION

The optimization problem takes the form of minimizing two objectives subjected to three constraints.

$$\begin{aligned} &\textit{minimize} && f_{Cost}(\vec{x}),\ f_{Head\ loss}(\vec{x}) && && (1a)\\ &\textit{subject to} && h_{PortsConnected}(\vec{x}) && = 0 && (1b)\\ & && g_{PipeIntersection}(\vec{x}) && \leq 0 && (1c)\\ & && g_{ObstacleIntersection}(\vec{x}) && \leq 0 && (1d) \end{aligned}$$

The design variables vector, $\vec{x}$, is composed of both properties of a given topology and the parametric variables that orient and size the components of the topology. The cost objective, $f_{\mathrm{Cost}}(\vec{x})$, is the result of individual costs associated with both the topology and parametric design variables. The second objective, $f_{\mathrm{Head\ loss}}(\vec{x})$, minimizes the total head loss as a form of the total pressure drop across all the input and outlet ports.

The one equality constraint and two inequality constraints are all spatial constraints in order to prevent the optimization functions from trying to solve the networks with physically impossible designs. The first constraint, $\mathrm{h_{PortsConnected}}(\vec{x})$, forces the optimization methods to produce networks that have an open pipe within some specified tolerance of each output location. The second constraint, $\mathrm{g_{PipeIntersection}}(\vec{x})$, prevents intersecting pipes by specifying a clearance distance that they must maintain between each other. The third constraint, $\mathrm{g_{ObsaclestIntersection}}(\vec{x})$, prevents components of the network from intersecting with any objects and any defined virtual boundaries residing in the world around the network.

The fluids are assumed to be Newtonian fluids and to have constant density and viscosity over their working conditions. The flow is also assumed to be incompressible and moving at a steady rate. The incompressible flow assumption is safe for ideal or perfect gases, in terms of viscous effects, up to speeds of 0.3Mach, where the Mach number is the speed of sound in that fluid [11]. It is also assumed that there is a minimal amount of heat transfer to or from the walls of the pipes due to a temperature changes. Depending on the application, careful analysis may make it possible to show that the temperature gradient is insignificant to the design of an optimal pipe network from a fluid flow standpoint.

Efficient data structures and minimization of variables are important to the performance of any algorithm. First, every pipe and pipe fitting has a starting point and an endpoint in three-dimensional space. These can be simply stored as values in Cartesian coordinates. Furthermore, the diameter and type of each fitting must be identifiable. The representation and evaluation were written specifically for this project in a total of approximately 2500 lines of C#.

PipeSynth represents the topology of fluid networks by using graph theory. Each network's fittings are represented by nodes, while the interconnecting straight pipes are represented as arcs. Fluid flows between each node through each arc, with each node having a position in three-dimensional space. Each node and arc also contains a data structure to retain information associated with real objects that they represent.

Each node's location is defined by one point at the intersection of the centerline axis of the ports of each fitting. Additionally, each pipe fitting also has an orientation in three-dimensional space that determines the directions of each of its ports. Rather than storing the location of the center of each port in every fitting, or representing the orientation with direction vectors, the location and orientation is determined with only two parameters. This is not only convenient, but necessary because if the pipe fittings were free to be moved and rotated in space independently of each other, then the fittings would rarely be oriented correctly to fit a straight pipe between them. Therefore, the location of each fitting is determined relative to the *previous* fitting, starting at a specified input node. The length of pipe between two fittings is represented by the parameters α, while the angle that the previous fitting is attached is represented by the parameter Θ. Figure 1 shows how a simple topology comprised of one 90 degree elbow and two straight pipes is updated in three-dimensional space based on the design variables.

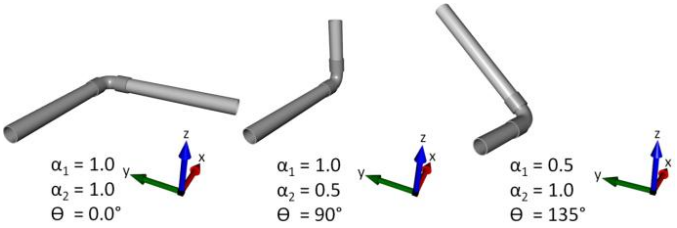

Figure 1. One topology generates different networks based on design variables

The variables α_1 and α_2 scale the lengths of the first and second pipes respectively. The variable Θ, ranges from zero to 360 degrees and rotates the elbow fitting about its input port (each fitting has exactly one defined input port). Additional fittings and pipes follow this same generation methodology.

3.1 Seeds

In generative grammars, the seed is the starting graph that represents a particular design problem. Sometimes, this can be very abstract, but for PipeSynth it is fairly representative of the real world problem. Each problem, or seed, is fundamentally defined by location of the inlet and outlet ports in Cartesian coordinates. As shown in Figure 2, these ports are represented by nodes in GraphSynth and contain pertinent data about their particular port such as the predefined pressure, flow-rate, and the fluid channel that they belong to. Inlet ports are also used to define key fluid material properties—most importantly density and viscosity.

Branched networks in PipeSynth are created by starting at one inlet and working towards all the remaining inlet and outlet ports. For consistency and performance, if more than one input is required per channel, then all the additional inputs are represented as outputs but with negative flow-rates.

3.2 Grammar Rules

PipeSynth uses grammar rules to construct the topology by adding one applicable pipe fitting at a time. Each grammar rule in PipeSynth consists of three or more nodes that represent the inlet port, the fitting origin and type, and the outlet port. An example rule for a half-inch 90° nominal pipe size, NPS, elbow fitting is shown in Figure 3.

In order to expand a network, the grammar rules recognize where they can be applied by first finding a node and arc pair that represents an open, or free, straight pipe. The local variables of the open pipe arc and the inlet pipe arc are compared to the rule to determine if they are equivalent sizes. If a match is found, then it is considered to be a possible option in building onto the current candidate. When a grammar rule is applied, it removes the open node and arc pipe pair and replaces them with the new nodes and arcs of the grammar rule.

4 OBJECTIVES

The values of the objectives are determined by both the topology and the design parameters. The first objective,

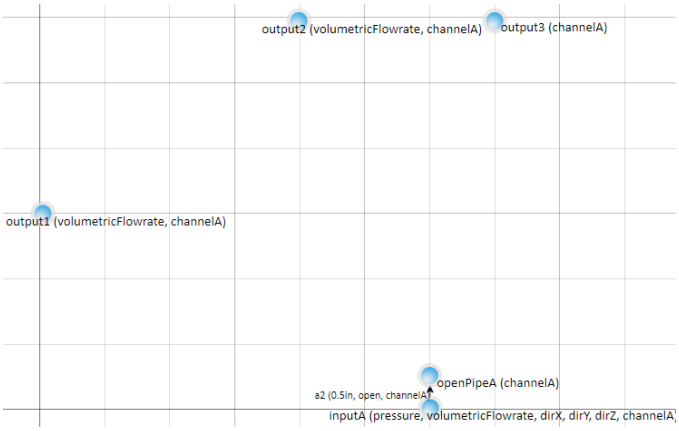

Figure 2. Example seed for PipeSynth with one fluid channel with one input and three outputs

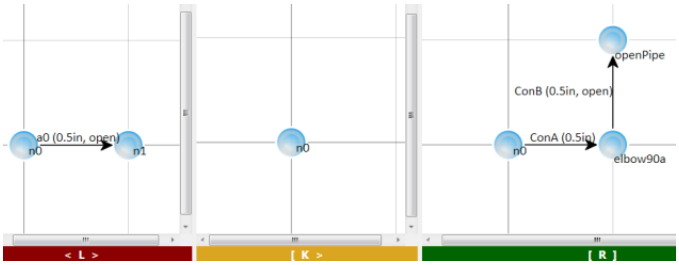

Figure 3. A PipeSynth Grammar Rule for a half inch NPS 90° elbow fitting

$f_{Cost}(\vec{x})$, seeks to minimize the total construction cost of network and its components.

These costs are dependent both on the topology and the parametric design variables. The topology dictates the number of each specific fitting and straight pipe elements. Each straight pipe has a material cost per linear foot also specific to its size, schedule, and material type. This length is determined by the α design variables which are determined during the parametric optimization stage. The total cost can be viewed as a summation of the topology's costs and the costs dependent on parametric design variables.

$$minimize \qquad f_{cost,total} = f_{cost,topology} + f_{cost,parametric} \quad (2)$$

The topology cost is the summation of each fitting cost, and a nominal assembly cost of each fitting:

$$f_{cost,topology} = \sum_{i=1}^{m} \left(f_{cost,material,fitting}(fitting_i) + f_{cost,Assembly}(fitting_i) \right) \quad (3)$$

where m is the total number of fittings in a specific topology. The material costs function simply looks up the part cost based on size and material of the fitting passed into the function. The assembly cost is based on the assembly cost of each port and the associated set assembly cost for that fitting class and diameter. The parametric cost determines the cost of the straight pipes used to join the fittings contained in the topology. After a specific topology has been optimized

parametrically, the lengths of each pipe in a candidate are known.

$$f_{cost,Parametric}(topology) = \sum_{i=1}^{m}(pipe_i.lenght * linearPipeCost_i) \quad (4)$$

This function sums all straight pipe material costs by multiplying each individual length by its respective linear cost. The second objective, $f_{\text{Head loss}}(\vec{x})$, has the goal to minimize the resistance of the network which is measured by calculating the total pressure drop, Δp_i, across each inlet and outlet ports for every channel.

$$minimize \qquad f_{Head\ loss}(\vec{x}) = \sum(\Delta p_i(\vec{x})) \quad (5)$$

The fluid dynamics determine how efficient or resistive a network is to a fluid traversing through it. In order to properly determine the pressure drop, the state of several intermediate points is calculated. The state of every fluid element in the algorithm is found by using the conservation of mass and the conservations of momentum.

The conservation of mass principle states that for any fluid element, mass cannot be created or destroyed. If the flow can be assumed to behave as an incompressible fluid then the volumetric flow-rate is also be conserved.

$$\Delta\dot{V}_{system} = 0 = \sum_{i=1}^{n_{ports}} \dot{V}_{port,i} \quad (6)$$

This equation can be applied to an entire network or any sub-section, including individual fittings, to aid in calculation of unknown flow-rates.

The conservation of momentum can be used to derive an equation for the pressure drop across a section of pipe. The resulting relation can be converted to the popular Darcy-Weisbach head loss equation used to calculate major head loss in pipes [12].

$$h_l = \frac{8C_f L\dot{V}^2}{\pi^2 g D^5} \quad (7)$$

The friction factor, C_f, is determined by specific correlations depending on the current flow regime determined by the Reynolds number. These friction factors are all used to calculate what are known as the major head losses of a network, which occur in the straight pipes. The fittings and joints are locations of minor head losses and, with their complex geometry, have much more complicated velocity profiles and friction factors. However, good correlations have been developed which scale major friction factors to equivalent minor head loss friction factors based on the individual fitting geometries. All of these are appropriately used in our custom fluidic simulation.

When solving a network, it is iteratively broken down into smaller subsections. Each section has a constant flow-rate, and a unique change of pressure due to viscous friction and gravity. These sections are represented by objects known internally to PipeSynth as Bernoulli-objects. They are named after Daniel Bernoulli, because they relate the pressure change between two points by the change of kinetic and potential energy.

The fluid state at each Bernoulli-object endpoint is partially determined by at least one other neighboring Bernoulli-object, with the exception of the network port nodes. The state of the fluid at the points shared by a pair of Bernoulli-objects is known as a subnode and these subnodes are always associated with one node and one arc. By iteratively solving the fluid mechanics equations for each Bernoulli-object, the state of each of these subnodes and the mean state of the fluid at any point in the entire network can be determined.

To find the states of the fluid at every subnode in the network, variations of equations 6 and 7 are used to solve each Bernoulli-object iteratively until the entire network is solved. This object-oriented approach is intelligent in that is solves for the easiest states first, working towards more difficult. Using these techniques alone, no additional matrix inversion methods are needed to analyze the fluid dynamics. At the ports it is possible to under- or over-specify the boundary conditions needed for this analysis. In the current implementation, these boundary conditions must be comprised of one known pressure at any inlet or outlet port and n-1 flow-rates, where n is the sum of inlet and outlet ports. The developed algorithm for performing this fluid mechanics simulation is written in C# and has 592 lines of code.

5 CONSTRAINTS

The one equality and two inequality constraints are each evaluated using various methods of computational geometry. Full three-dimensional solid object intersections can be unnecessarily complicated and resource intensive. To improve the performance of the constraint evaluations, the pipes are first converted into parametric line segments. Then a simple calculation yields the closet distance, d, between a parametric line and a point of interest, $\vec{P}_x$.

$$d = \frac{|(\vec{P}_{1,2})\times(\vec{P}_x - \vec{P}_1)|}{\|\vec{P}_{1,2}\|} \quad (8)$$

The intersection of a line segment and an infinite plane equation can be found by knowing the normal to the plane, $\vec{N}$, and any point $\vec{P}_0$ that lies in the plane.

$$t = \frac{-\vec{N}\cdot(\vec{P}_1 - \vec{P}_0)}{\vec{N}\cdot\vec{P}_{1,2}} \quad (9)$$

If this equation yields a value between 0 and 1, then there is an intersection with the line segment.

The first constraint, $h_{PortsConnected}(\vec{x})$, is a way to test if the output ports have been reached by a particular candidate network. Specifically, this constraint tests if each design output location resides within some tolerance distance of the desired output location.

$$h_{PortsConnected}(\vec{x}) = \sum(max(d_{output,i}(\vec{x}) - tolerance, 0)) = 0 \quad (10)$$

The function $d_{output,i}(\vec{x})$ is the distance between a corresponding output port and its paired open pipe. As long as $d_{output,i}(\vec{x})$ is less than the specified tolerance, then that output will not contribute to anything in the summation because the *max(x,0)* function will return 0. The *max(x,0)* function is important because it keeps each designed output location test independent of the next. If the outputs are not within in the tolerance, $h_{PortsConnected}(\vec{x})$ increases with the increased distance.

The port connection constraint will always seek to minimize the distance between each open pipe and the nearest output location. The distance between each pair of nodes is simply the magnitude of the vector that joins the two. For each of these distances that are above the specified tolerance, a penalty proportional to that distance is added to return a value of this constraint. Otherwise if the nodes are within the tolerable distance of each other, no penalty is contributed by that pair.

The second constraint, $g_{PipeIntersection}(\vec{x})$, prevents the pipes from colliding with each other.

$$g_{PipeIntersection}(\vec{x}) = \sum_{i=0}^{n_{pipes}} \left[\sum_{j=i+1}^{n_{pipes}} \left(PIT\left(pipe_i, pipe_j\right)\right)\right] \leq 0 \qquad (11)$$

The function $PIT\left(pipe_i, pipe_j\right)$ evaluates two specific pipes and returns the results of a pipe intersection test on them. The constraint's value increases by an amount proportional to the intersection magnitude of each pair of pipes. For each pair of pipes that do not intersect, the pipe intersection test will return zero. The summations starting index ($j = i + 1$) ensures that every pair of pipes, up to the total number of pipes, n_{pipes}, are tested exactly once.

If the goal is just to prevent collisions from happening, then it does not matter where the intersection occurs, only if and by how much. With this consideration, the minimum distance between the central axes of the two pipes will tell whether or not an intersection will occur. To account for the fittings that will be larger than the diameter pipes and have more complex geometry, the pipes are approximated as cylinders that run all the way beyond the center point of the fittings, which occurs at the intersection between the centerlines of all pipes connected to a particular fitting. Then by adding a large enough clearance value to outer diameter of the pipes, adequate clearance can be assured for all the pipes and fittings as long as $g_{PipeIntersection}(\vec{x}) \leq 0$.

The third constraint, $g_{ObstacleIntersection}(\vec{x})$, prevents any part of the network from residing in the same physical space as define known objects. The obstacle constraint function guides the pipes to stay within predefined areas by calculating a magnitude of intersection violation, similar to the pipe intersection test.

$$g_{ObstacleIntersection}(\vec{x}) = \sum_{i=0}^{n_{pipes}} \left[\sum_{j=i+1}^{n_{objects}} \left(OIT\left(pipe_i, object_j\right)\right)\right] \leq 0 \qquad (12)$$

The function $OIT\left(pipe_i, object_j\right)$ is the object intersection test that checks for intersections between each pipe and each defined object/boundary. Similar to $g_{PipeIntersection}(\vec{x})$, the value of $g_{\mathrm{ObstacleIntersection}}(\vec{x})$ is proportional to the sum of the magnitudes of the intersections for each pipe.

The three primary obstacle shapes that PipeSynth can represent are half-space planes, parallelepipeds, and spheres. Each one can either block a network intersection with a wall or object, or can contain the network in a region like a duct or crawlspace. Obstacles are generally defined by an origin point, direction vector(s) (or radius), and whether they impose an interior or exterior boundary.

In order to evaluate if, and by how much, a pipe intersects an object, the obstacles are either externally or internally padded, depending on their constraining side. This inflated, or padded, distance is equal to the external radius of the pipe in question and some set obstacle clearance value. This technique allows for a much simpler intersection test of line segments and solids.

6. OPTIMIZATION

There is unique interaction between the two tiers of optimization in PipeSynth beginning with the topology optimization level. Starting with a seed, a new topology is generated with the application of a grammar rule. This topology is then subjected to a parametric optimization subroutine. The parametric optimization method uses the evaluation techniques discussed in sections 4 and 5 to determine the fitness and feasibility of each solution candidate. After the parametric optimization method converges, the design is tested for feasibility. If it passes, then it is compared to the list of best solutions (nondominated or Pareto set) found thus far and is added if it is not dominated – worse in both $f_{\mathrm{Cost}}(\vec{x})$, and $f_{\mathrm{Head\ loss}}(\vec{x})$ – to those already on the list. If no stopping criteria is met for the topology optimization, a new topology is generated and the cycle repeats.

6.1 Topology Optimization

PipeSynth primarily uses a uniform cost search for topology optimization. In the branched tree search space, every element in a level is searched, in the order from least transition cost to most. The process converges when no new solutions have been added to the Pareto front after a specified number of levels. The amount of levels it searches below the youngest Pareto solution should be scaled according to the number of channels (independent networks) in the design problem. This is because each channel needs one level each to add one additional pipe to its level.

With each grammar rule in PipeSynth representing a unique pipe type and size, there is one option corresponding to each grammar rule that has an inlet port size that matches the size of each open port. The numbered arrows in Figure 4 indicate the order in which candidates are searched for an example problem.

Uniform-cost-search selects the order based on the transition cost of each pipe at each tree level. This transition cost is based on the fitting type and size, with the smaller and more significant geometry changes having the lower costs. Fittings that change the network more radically are preferred to be searched first because they are more likely to find a feasible network earlier. For example, several 90 degree elbows can typically get around obstacles and allow a network to reach the output ports with fewer fittings than using a mix of simpler geometry fittings, such as straight couplings, could. For illustrative purposes, this problem contains just four fittings and prefers to use tees over elbows because they expand the number open pipes, and thus the possibility of connecting to multiple outlets with fewer fittings. After each candidate is parametrically optimized, the branch is determined to be at one of three conditions. Condition 1 represents that a feasible solution has been found, but the branch will continue to be searched for better solutions. Condition 2 is determined to be an infeasible solution, but it could lead to a feasible solution with more fittings. Therefore, this branch will also be kept and searched further. Condition 3 is a termination to a branch and occurs when a stopping condition is met.

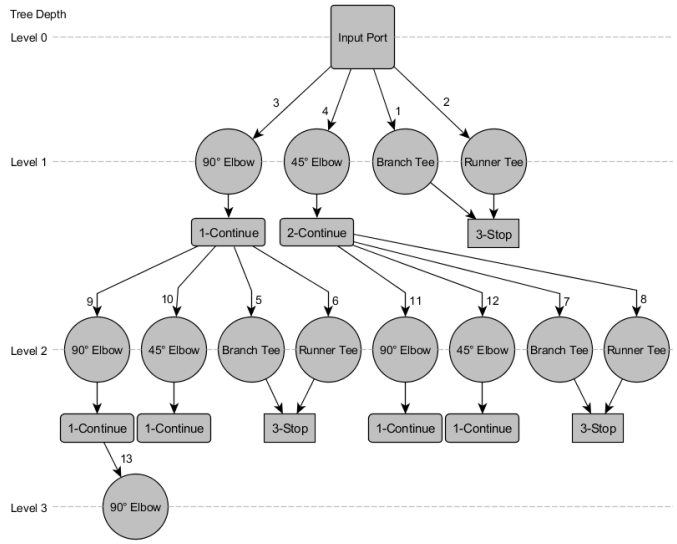

Figure 4. Example PipeSynth Topology Optimization

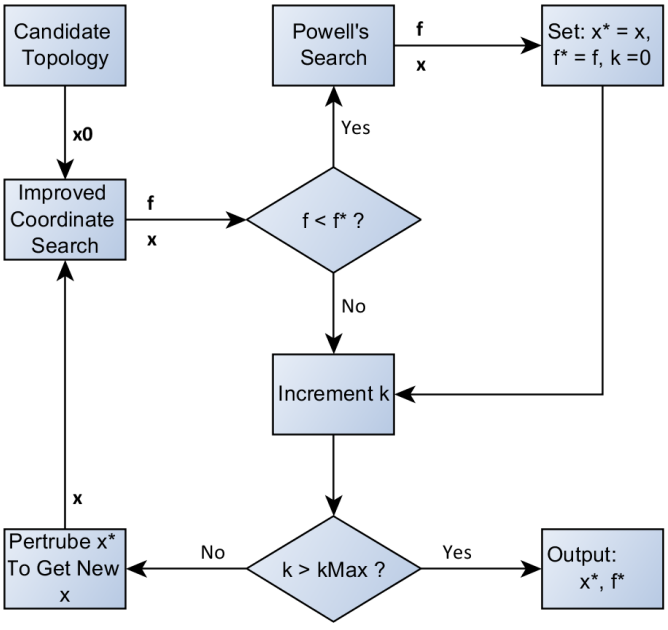

Figure 5. Parametric optimization flowchart.

When a particular channel in a candidate has branched to contain more open pipes than outlets, the branch also stops at Condition 3. This state also dictates that the current candidate is infeasible and that adding more pipes will never make it feasible again. Consequently, the branch is eliminated and does not get expanded any further. Condition 3 is a direct result of PipeSynth being designed to create only branched type networks, and as a result, there is no way for additional fittings to merge open pipes back together to decrease the number of open pipes.

After parametric optimization of a candidate is complete, all the children are generated for that candidate and added to that sorted list, including an additional cost proportional to the depth of the tree that the child resides on. Thus, all the candidate's children are searched at one level before moving on to the next. The quality of each of these candidates as a feasible and better solution is determined during the parametric optimization stage.

6.2 Parametric Optimization

Each candidate created from the rules represents a fixed topology – the number of fittings and the order in which they are connected is known. The parametric optimization attempts to adjust each topology in a way to produce the best feasible solution possible. Each pipe length, α, and fitting attachment angle, Θ, make up the set of design variables, $\vec{x}$, that describe the final layout of the network. By optimizing these design variables, the best possible network(s) for a given topology can be found.

PipeSynth uses a combination of cyclic coordinate search, Powell's Method, and random hill climbing to find the global optimum. The unique combination provides the benefits of speed, accuracy, and robustness from each of the three methods. The implementation of these methods is shown in the flowchart (Figure 5).

Cyclic coordinate search is used first because it is very quick at making big improvements to $\vec{x}$, and it can be used quickly to test if a point is in a region with a better minimum than the best known minimum. Unfortunately, this method gets stuck easily at local minima and saddle points. Actually, it was found for some problems that in addition to searching in the direction of every coordinate, searching along directions of every pair of coordinates had a significant improvement on avoiding local minima. This diagonal search is labeled improved coordinate search and not only searches in the positive direction of the unit vectors, but also every combination of positive and negative unit vectors for every

pair. The performance improvement is a direct result of the way PipeSynth represents the networks from the parametric design variables. By increasing one variable and decreasing another at the same rate, it allows the length, α, or angle of attachment, Θ of one pipe to increase while the length (angle) of an adjacent pipe decreases. This allows fittings with a pair of collinear ports to "slide" or "spin" without affecting the design of the rest of the network during the parametric optimization stage.

Every time after a new best function evaluation has been found, Powell's method is used to further exploit the space and more closely reach the local minima. Powell's method was able to fine tune $\vec{x}$ better than coordinate search, but at the cost of being anywhere from three to eight times computationally slower than cyclic coordinate search.

If a better point is not found with Coordinate Search, and if k is still less than $kMax$, then a random disturbance is added to the current best $\vec{x}^*$. This is done by selecting a random direction and step size in attempts at getting from the current best point over a hill and into a deeper valley. After this disturbance is added, the cyclic coordinate search method is used to quickly test $\vec{x}$ again. If the disturbance successfully found a significantly deeper valley, then the objective function's value will be the best found so far. This disturbance and improvement cycle continues until either a better point is found and Powell's method is employed to optimize it, or the limit of disturbance attempts, $kMax$, is reached. In order to adequately sample the space around x, the integer $kMax$ is proportional to the number of design variables.

This unique hybrid approach proved to be more robust than any of three methods individually. Nelder-Mead's simplex method was also extensively tested and while it had a fast convergence rate, it failed to be any more efficient than cyclic coordinate search, and was especially unreliable at higher dimensions (larger topologies of fittings and pipes). The algorithm details of the parametric optimization come from the Object-Oriented-Optimization-Toolbox, OOOT, created by members of the Automated Design Lab at the University of Texas at Austin [13].

6.3 Pareto Optimality

A well-designed multi-objective optimization problem can be problematic because improving one objective often compromises another [14]. Each new problem in PipeSynth generates a two-dimensional Pareto curve that is comprised of the best trade-off solutions found for that application.

A simple way of managing multi-objective problems is to minimize or maximize the total sum of the objectives—taking care to note the sign for each individual object which will determine if the individual objectives will be minimized or maximized. Using the linear sum method, while easy to implement and computationally efficient, often provides a poor distribution of solutions due to wildly varying ratios of objective functions for different problems and topologies. The way multiple objectives in PipeSynth are managed is by using a targeting method. This approach only minimizes one objective while the other is set as a target constraint.

$$\begin{aligned} &minimize \quad \vec{f}_i(\vec{x}) && (14a)\\ &subject\ to: \quad f_j(\vec{x}) \le C, \\ &\qquad\qquad \vec{g}(\vec{x}) \le 0, \\ &\qquad\qquad \vec{h}(\vec{x}) = 0 && (14b) \end{aligned}$$

This is only applied during the parametric optimization, because the topology optimization is not dictated by continuous design variables and thus not evaluable on the aforementioned objective functions and constraints. To apply the targeting method, PipeSynth first finds the extrema of the Pareto curve by minimizing each of the objectives independently, while leaving the alternative objective unconstrained. This finds the least expensive and least restrictive feasible solutions for each topology. Then an even distribution of cost targets is found between these limits to help approximate the Pareto curve. To generate these new targeted solutions points, each topology's head loss objective is parametrically optimized at each of those cost targets.

$$\begin{aligned} &min\,imize \ f_{head\ loss}(\vec{x}) && (15a)\\ &subject\ to\ g_{cost} = f_{cost,total}(\vec{x}) - targetCost_i \le 0 && (15b) \end{aligned}$$

The new inequality constraint, g_{cost} maintains the total cost below the targeted cost. Therefore $\vec{x}$ must minimize the head loss but without making the total network cost exceed the targeted cost. By incrementing the target cost, several Pareto solutions can be generated for a specific topology. This method does not guarantee an even distribution of points along the Pareto curve but is very robust because the distribution will be generated regardless if the curve is concave or convex. It is also efficient because every parametric optimization returns a new useful point to aide in the approximation of the Pareto curve. Some methods have the ability to generate the Pareto curve in much finer detail and with a better distribution of points, but typically at the cost of requiring several more optimization executions. If a finer resolution of curve is desired, it is easy to increase the number of cost targets per topology.

7 RESULTS AND DISCUSSION

Two different types of example networks are presented and discussed in this section. These are created to showcase the capabilities of PipeSynth for handling multiple channels, ports, and obstacles. Robustness and flexibility were placed above performance during the design and development of the algorithms used in PipeSynth.

7.1 Plant Example with Multiple Separate Channels

This first example displays PipeSynth's ability to handle multiple, separate channels that each contain a different type of fluid. Each channel path is constrained by the two processing objects, a central obstacle, and the floor, all the while being contained within a bounding box. The two dimensional seed graph is shown in Figure 6.

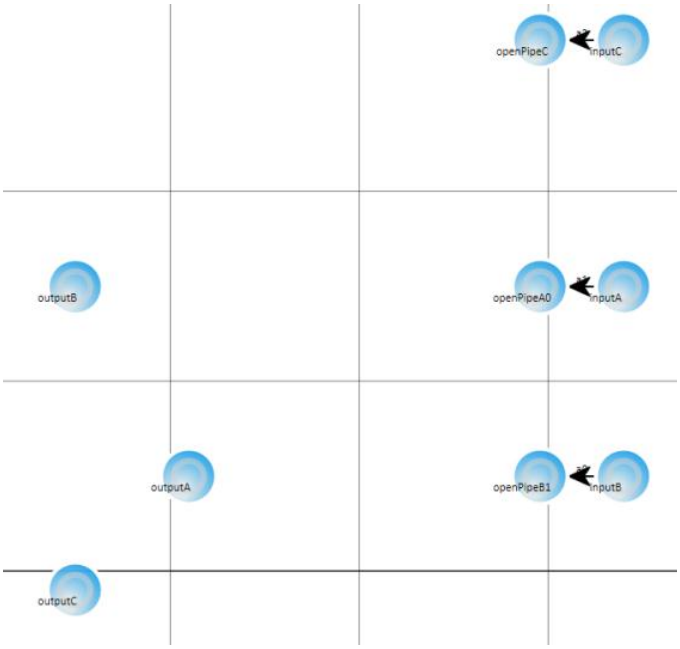

Figure 6. Plant example seed graph showing three independent channels

The test ran for approximately 39 hours and began finding solutions at level 7 of the tree. It proceeded through level 9 where it continued to find feasible solutions. The topology search terminated at the end of level 9 because no superior solutions had been found since level 7. The large tree was growing quickly and additional pipes would unlikely decrease both total cost and head loss below what was found at level 7. The resulting solutions are shown in Figure 7 below.

Three solutions from level 7 dominate the remaining 52 feasible solutions found. These solutions form the Pareto curve, but the scatter makes it difficult to predict the complete curve. The discontinuity in the Pareto curve is likely attributed to the discrete nature of the different topologies that produce a wide variety of unique solutions. Each parametric optimization only returned one unique solution for each feasible topology. This shows that only allowing one pipe size per channel caused tradeoff solutions to occur only at the topological level.

The three-dimensional models representing the networks and the objects were created with the assistance of the Helix 3D Toolkit, as seen in Figure 8 [15].

In each of the solutions (one shown), the best the network designs correctly attach all the fittings and do not violate any constraints. While these solutions satisfy the design criteria, some of pipes appear to not be quite optimized to their full potential. In the other Pareto solutions, not shown, there are some channels that appear to naked eye to take a less than optimal route. Balancing the algorithms' performance and robustness of finding the very good parametric solutions proved very challenging with the multiple channels and obstacles of this problem. Driving the channels towards their respective outputs while minimizing their objective functions

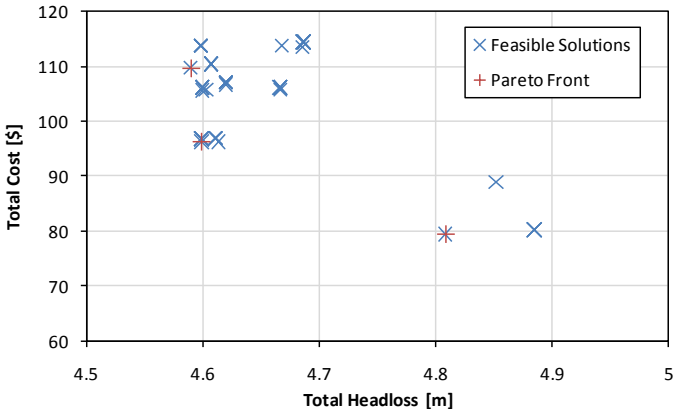

Figure 7. Pareto Plot of feasible solutions found for Plant Example

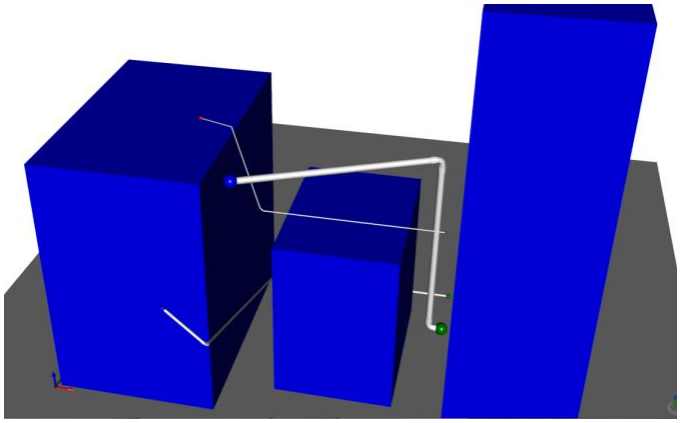

Figure 8. Lowest head loss plant example modeled with Helix 3D Toolkit

all while preventing interferences required a lot fine tuning. With even further refinement to the algorithms, slightly better parametric solutions could likely still be found.

7.2 Food Processing Fire Suppression Example

The second demonstration is for the routing of a liquid fire suppressant to 4 workstations. The example liquid suppressant, PRX™, is produced by the ANSUL® company for grease and ventilation fires [16]. This example was setup with one input located near a wall and the ceiling of a rectangular building with the need to provide suppressant to four separate workstations. Four various size boxes were created to represent the different workstations requiring the suppressant and also act as obstacles to the network. The four outputs are placed on top of their respective boxes to allow connection to each unit's independent fire suppression sprinkler system. To contain the network within the room, a bounding box is used. The results show how one topology found at level 4 of the tree dominated all the other designs found.

Again, the objectives were found to not be competing at the parametric level and always converged to one best point for each topology. Only a single pareto point is found as shown in Figure 9 and the three-dimensional model of this solution is shown Figure 10.

The best solution, Figure 10, shows a branch tee (having its input port on the branched port side) and a 90° elbow as the first two fittings. The reason these fittings are used in conjunction instead of using a single tee is because the input pipe has its direction constrained to remain horizontal and therefore cannot reach the farthest output without a change of direction. This can be seen more clearly in Figure 11.

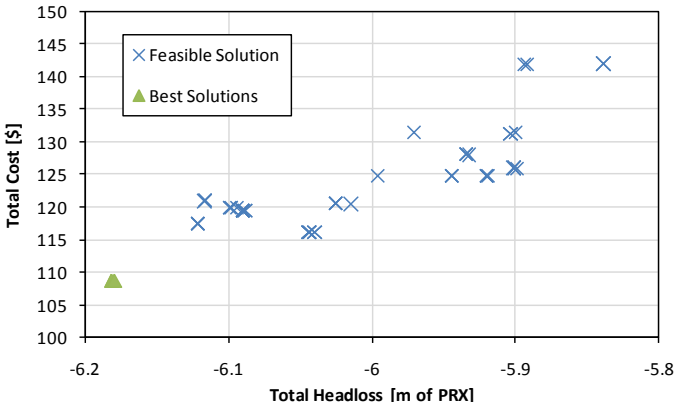

Figure 9.Pareto Plot of feasible solutions found for Plant Example

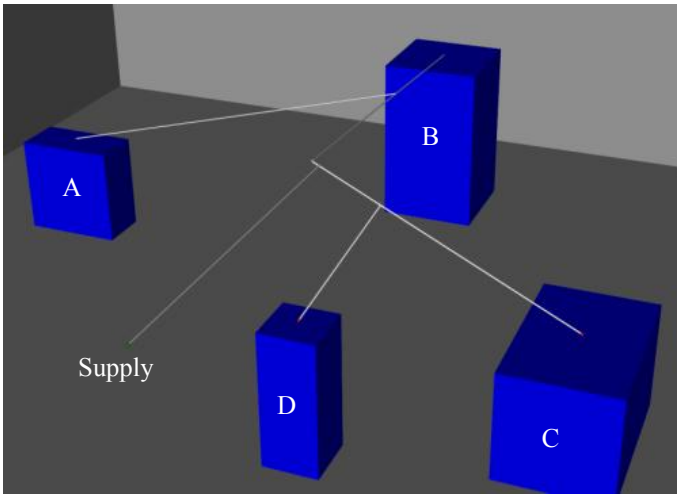

Figure 10. Best solution for fire suppression example modeled with Helix 3D Toolkit

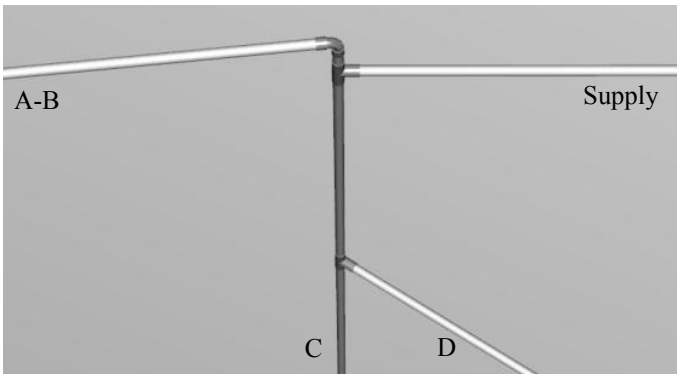

Figure 11. Tee and elbow fittings arrangement attached to input pipe in Figure 10.

The horizontal pipe exiting towards the right of Figure 11 attaches to the input port of the problem, while the angled pipe leaving towards the left points in the direction output port four. The pipes exiting the bottom attach to outputs two and three. In this solution, every pipe attaches properly to each fitting and each port, without violating any constraints.

The branching factor of the tree increases proportionally to the number of ports. Therefore, because this example had five ports, the topology optimization was limited to only two elbows and two tees in order to minimize the total candidates per level of the tree. This allowed level 6 to be reached with 24 hours.

7.3 Discussion

Several challenges and accomplishments were found during the study of this project. Increasing complexity became more and more resource intensive. Each additional level of the tree represents the addition of more pipe fittings. Each pipe fitting increases the number of design parameters by an amount equal to the number of ports the fitting contains. For example, an elbow adds one additional pipe length, α, and one additional angle of attachment, Θ, while a tee adds two additional pipe length design parameters along with its angle of attachment.

The plant example has four obstacles and three separate channels. It was found that the amount of obstacles and channels added to the parametric optimization time considerably, and required much more fine tuning to produce good feasible solutions in a reasonable amount of time. Adding 7 fittings requires traversing to level 7 of the tree, which may be prohibitively time intensive if the tree has a large branching factor.

It was discovered that when a problem contains multiple channels, normal grammar rule application will produce confluent candidates. This can happen when, for example starting from the seed, rule 1 is applied to channel A, then rule 2 is applied to channel B. Then starting from the seed again, rule 2 could be applied to channel B, followed by rule 1 on channel A. To prevent time from being wasted searching these candidates with duplicate topologies, a data structure was created to keep track of every combination of rules that have been applied to generate candidates. However, it also takes into consideration which channels the rules were applied to, so that if a new candidate is created with a duplicate topology, it can be removed immediately.

8. CONCLUSION

PipeSynth was quite successful in tackling many of the development goals and design considerations when designing pipe networks. It can handle a wide variety of fluids at a wide variety of flow conditions. It can build networks that conform to real world spatial constraints while retaining the correct fitting geometries between each and every pipe. With the computational resources available, Uniform Cost Search was an effective topological method when the optimal network resides within about the first 500 candidates. For more

complex problems, stochastic tree search methods must be employed to traverse the tree adequately.

Penalty functions were used to handle constraint violations which in turn caused the n-dimension parametric search space to become very multi-modal. The most effective way of overcoming these local minima was the introduction of a random disturbance process to "kick" the search out of local minima. After each random kick, the improved cyclic coordinate search is used to see if the local minima has been escaped. When a better region is found, Powell's method is used to fine tune the design variables before returning the solution. This approach was found to be much more robust than Nelder-Mead, or any of three employed methods individually. The drawback to this approach was a much higher computational time during the optimization stage.

The unique representation of the pipe networks within this study required that the parametric and topological algorithms to become highly interrelated. Rather than operating solely independently, at lower levels the algorithms share information between each other in order to select the most efficient sub-routines for each step based on the current level of data available. These interactions improve the efficiency to a level that made solving complex, and minimally defined, problems possible. However, this new approach made comparing the stand alone efficiency of either the topological or parametric algorithms to the performance of previous research that focused on just one of these levels, very difficult.

With PipeSynth's robust algorithms designed to handle increasingly more complex problems, and given adequate resources, PipeSynth should be able to solve much more complex and significant problems. This automated design and optimization study proves that complex, real-world, fluid networks can not only be solved with computational synthesis, but, given adequate resources, can also be optimized beyond what any human designer can create.

There are several motivating features and capabilities that can still be added to PipeSynth. Additional cost considerations to more accurately model and assist in network design selection would be a great enhancement to any end user of this program. It is easier to construct and maintain fluid networks when the pipes are clustered and run parallel to adjacent pipes. Supporting pipes, especially larger ones, can be particularly important when heavy fluids are involved. When designed correctly, there will be an optimal tradeoff between minimizing material costs, head loss, and assembly (including bracketing) costs.

Temperature gradients could also be incorporated into PipeSynth, along with temperature dependent fluid properties, by having temperature as an additional state at each subnode. With the addition of heat transfer conduction and convection analysis, one could model systems affected by the ambient temperature.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant CMMI-0448806. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] GLS Software. *WADISO*. Web. 13 Nov. 2010. <http://www.gls.co.za/software/products/wadiso.html>.

[2] Accutech. *FLUIDFLOW3*. Web. 13 Nov. 2010. <http://www.accutech2000.com.au/Brochure.pdf>.

[3] Klarbring, Anders, et al. "Topology optimization of flow networks." Elsevier B.V., 2003.

[4] Collins, M. "Solving the Pipe Network Analysis Problem Using Optimization Techniques." Managment Science. Vol. 24, 1978.

[5] Savic, Dragan A. and Godfrey A. Walters. "Genetic Operators and Constraint Handling for Pipe Network Optimization." School of Engineering, University of Exeter. Exeter, UK.

[6] Tospornsampan, Janejira. "Optimization of a multiple reservoir system using a simulated annealing–A case study in the Mae Klongsystem, Thailand." Springer-Verlag. 2005. Paddy Water Environ.

[7] Pan, Tze-Chin and Jehng-Jung Kao. "GA-QP Model to Optimize Sewer System Design." Journal of Environmental Engineering. ASCE, 2009.

[8] Xue, Guoliang, Lillys, Theodore P. and Dougherty, David E.[ed.] Siam J. Optim. "Computing the Minimum Cost Pipe Network Interconnecting One Sink and Many Sources." Society for Industrial and Applied Mathematics. 1999. Vol. 10. pp. 22-42.

[9] Rothfarb, B. "Optimal Desgin of Offshore Natural-Gas Pipeline Systems." Informs: Institute for Operations Research, 1969.

[10] Afshar, M. H. and Marino "A parameter-free self-adapting boundary genetic search for pipe network optimization." Springer Science+Business Media, LLC, 2007.

[11] White, Frank M. *Viscous Fluid Flow*. Third Edition. NY : McGraw-Hill Education, 2006. pp. 89-111.

[12] Bhave, Pramond R. "Analysis of Flow in Water Distribution Networks." Lancaster: Technomic Publishing Company, Inc., 1991. pp. 65-85.

[13] Campbell, Matthew I. *Object-Oriented Optimization Toolbox*. Codeplex. Automated Design Lab at the University of Texas at Austin. Web. 24 Oct 2010. <http://ooot.codeplex.com>.

[14] Belegundu, Ashok D. "Optimization Concepts and Applications." Upper Saddle River: Prentice-Hall, Inc., 1999.

[15] *Helix 3D Toolkit*. CodePlex. Web. 21 Nov. 2010. <http://helixtoolkit.codeplex.com>.

[16] Ansul. *PRX Liquid Fire Supperssant*. Ansul Fire Protection, 2005. Web. 23 Nov, 2010. <http:// www.ansul.com/AnsulGetDoc.asp?FileID=8348>.
[17] *Optimal Design of Water Distribution Networks*. Pangbourn: Alpha Science International Ltd., 2003. pp. 16-33.
[18] Chanson, Hubert. *Applied Hydrodynamics*. London: Taylor & Francis Group, 2009. pp. 29-30.
[19] Larock, E. Bruce, W. Roland Jeppson, and Z. Watters. *Hydraulics of Pipeline Systems*. New York: CRC Press LLC, 2000. pp. 4-216.
[20] Nayyar, Mohinder L. *Piping Handbook*. Seventh Edition.NY: McGraw-Hill, 2000.
[21] Stephenson, David. *Pipeflow Analysis*. Elsevier Science Publishers B.V., 1984. pp. 36-39.
[22] Vreugdenhil, Cornelis B. *Computational Hydraulics*. Berlin: Springer-Verlag, 1989. pp. 1-7.
[23] Wood, W. L. *Introduction to Numerical Methods for Water Resources*. New York: Oxford University Press Inc., 1993.
[24] Campbell, Matthew I. *GraphSynth2*. Web. 14 July 2010. <http://www.graphsynth.com>.
[25] McMaster-Carr. *McMaster-Carr-Pipe Fittings*. Web. 8 Aug 2010 <http://www.mcmaster.com/#pipe-fittings/= 9v7qhe>.
[26] Eberly, David. "Distance Between Two Line Segments in 3D." Geometric Tools, LLC, 1999.
[27] Nayyar, Mohinder L. *Piping Databook*. New York: McGraw-Hill, 2002. pp. 439-484,543-556.
[28] Bates, Paul D, Stuart N. Lane, and Ferguson, Robert I. *Computational Fluid Dynamics: Applications in Enviromental Hydaulics*. Chichester: John Wiley &Sons Ltd, 2005.
[29] Fox, Robert W., Alan T. McDonald, and Philip J. Pritchard. *Introduction to Fluid Mechanics*. Hoboken: John Wiley & Sons, Inc., 2006.
[30] Gomes, Heber Pimentel. "Optimal dimensioning model of water distribution systems." Paraiba, Brazil: Water SA, July 2009, Vol. 35.
[31] Prasad, T. Devi, Sung-Hoon Hong, and Namsik Park. "Reliability Based Design of Water Distribution Networks Using Multi-Objective Genetic Algorithms." KSCE Journal of Civil Engineering, 2003, Vol. 7, p. 351-361.
[32] Wolf, Danilde and Yves Smeers. "Optimal Dimensioning of Pipe Networks with Application to Gas Transmission Networks." INFORMS, July 1996, Vol. 44, pp. 596-608.
[33] Murray, Glenn. "Rotation About an Arbitrary Axis in 3 Dimensions.". Colorado School of Mines. Web. 11 Nov 2010.<http://inside.mines.edu/~gmurray/ArbitraryAxisRotation/ArbitraryAxisRotation.html.>

