ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-655

Automating Layouts of Sewers in Subdivisions

Neil Burch and **Robert Holte** and **Martin Müller** and **David O'Connell** and **Jonathan Schaeffer** [1]

Abstract. An important part of the creation of a housing subdivision is the design and layout of sewers underneath the road. This is a challenging cost optimization problem in a continuous three-dimensional space. In this paper, heuristic-search-based techniques are proposed for tackling this problem. The result is new algorithms that can quickly find near optimal solutions that offer important reductions in the cost of design and construction.

1 INTRODUCTION

The design of a housing subdivision is a complex civil engineering task. In the concept phase, the designer decides on how a tract of land will be subdivided into housing lots connected by roads. In the design phase, the core infrastructure is planned, including the analysis of the terrain, grading of the land (elevation), sewers (water, sanitary), and conduits (power, telephone). Finally, in the layout phase, all the design considerations are mapped to an implementation. The design is typically done as part of the tendering process by a civil engineering firm. If they are the successful bidder, then they do the layout.

Most of the design and layout of a subdivision is manually done, with a professional civil engineer required to verify its adequacy and compliance with regulations. One of the labor-intensive aspects is the design and layout of the sewers. This is a difficult problem for engineers. It is a three-dimensional optimization problem in a continuous search space. Typically, engineers adopt a least-effort solution, such as laying the sewers down underneath the middle of the road. These solutions are demonstrably suboptimal, adding to the subdivision developer's construction costs.

This paper poses the problem of automating the design and layout of sewers in subdivisions as an interesting AI application. Although the sewer problem is a small part of the overall process of creating a subdivision, automating this process has cost advantages. First, it will reduce the manual effort required for the design and layout phases. Second, a (near) optimal solution will reduce the construction costs.

Automation of the sewer problem has been tackled by several researchers in the engineering community. Their solutions are typically genetic algorithm or simulated annealing based. In contrast, we take a heuristic search approach.

The contributions of this paper are as follows:

1. An understanding of an important real-world engineering application. Given the difficult cost function and numerous special cases, an elegant all-encompassing solution is not possible.
2. A new algorithm for determining and placing the minimum number of manholes (ground connections to the sewer) on a single road. Dynamic programming is used to obtain an optimal global layout cost, subject to a restriction on the location of manholes in intersections.
3. The road layout, expressed as a graph, has cycles. The sewer system cannot have cycles. We present a new approach for "cutting" the graph into a tree and exploring the search space of possible spanning trees to find a high-quality solution.
4. A complete working system that can produce an anytime solution. It scales to larger networks than have been solved previously.

Although this research deals with sewer system planning, these ideas are applicable to other types of pipe networks (e.g., water distribution networks [1]).

2 SEWER SYSTEMS

A sewer system is a subterranean system used to convey waste to one or more collection points (outfalls). There are two types of sewer systems: sanitary, to convey industrial and household waste, and storm, to prevent flooding by draining surface water. In this paper, without loss of generality we limit the discussion to storm sewers.

Each segment of pipe in a sewer system is connected by a manhole. A sewer design is a list of manholes and pipes. Each manhole has a location and depth. Every pipe has an upstream and downstream manhole, a diameter, and an upstream and downstream depth.

The difficulties in designing a sewer arise from a large set of constraints. For example, the sewer must have the capacity to handle the load it is expected to experience. All roads in a subdivision must have a pipe, and all pipes must go under the road. The slope of all pipes must be over some minimum value so that they drain (we only consider systems which use gravity to properly drain) and under some maximum so that the pipes do not wear too quickly. There are only certain fixed diameters of pipes available. Pipes must be buried to a minimum depth. Segments of pipe must not exceed some fixed maximum length. We consider all of these in our work, but there are a few other parameters not included that would be needed for a production system (e.g., curved pipe segments, and constraints on the angle at which two pipes can meet).

An important issue that needs to be considered when burying pipes is the slope. Pipe slope affects both the flow velocity and the pressure within the pipes. The goal of design is to keep the flow velocity above the minimum self cleansing velocity. If the velocity of the flow is too low, deposits may build up within the pipe, obstructing the flow. However, if the velocity is high enough these deposits are prevented, and thus the system is self cleansing. Sewer systems where the slope of the pipes alone convey the sewage are known as gravity sewer systems; more complex systems may need pumps. In this paper, we only consider gravity-based systems.

The rules for the design and layout of a sewer system are governed by the regulations of the local municipality in which the housing subdivision is located. There are numerous constraints imposed by municipalities, with no standardization across jurisdictions. A production quality sewer design tool needs to handle the plethora of parameter settings.

[1] Department of Computing Science, University of Alberta, Canada, email: {rholte,jonathan}@ualberta.ca

For storm sewers, the flow of water that will go through the pipes is estimated using hydrological analysis. This is a mathematical approximation based on evaporation, precipitation, snow meltage, other types of natural water movement, and the historical record.

When creating a sewer system plan, the engineer must consider the horizontal and vertical alignments. The horizontal alignment specifies where the pipes and manholes are placed within the subdivision. Standard engineering practice defines a corridor underneath the roadway where pipes are allowed to be placed. To allow for proper maintenance of the sewer system, manholes must be placed at specified intervals along the pipe. The maximum distance differs amongst the different diameters of pipe. Manholes are also typically placed wherever the pipe changes direction. For roads with a high degree of curvature, this can result in many manholes.

Vertical alignment is concerned with the depth of sewer pipes. There are several factors that must be considered when determining these depths. First, the pipes must be buried deep enough to prevent damage from the surface, including pressure from traffic and weather. Second, there is usually a minimum required separation between the sewer and pipes/cables from other utilities. For example, underground power cables and clean water distribution systems may also lie underneath the same road corridors as the sewer system.

The optimization process for sewer system planning can be separated into the two interrelated problems of layout (horizontal alignment) and implementation (vertical alignment). The main goal is to produce the plan with the lowest cost, while meeting the specified engineering constraints defined for the project. Cost considerations typically includes the manholes (number and depth) and pipes (number, length, diameter, and depth). For more technical information consult a civil engineering text such as [6].

3 RELATED WORK

Much of the engineering literature assumes fixed pipe locations and tries to optimize the pipe diameters and slopes. "*Most pipe optimization methods have not considered the layout optimization along with the cost due to the extreme complexity involved ...*" [7]. These solvers make simplifying assumptions, such as assuming any pipe diameter is possible. By considering the diameter as a continuous variable, the problem is easier to solve by linear/non-linear programming techniques. Such solvers have been built based on dynamic programming [12] and genetic algorithms [8, 11]. An alternative is a knowledge-based blackboard architecture approach [2].

Despite the promising results reported in the literature, none of these approaches are used by the engineering community. Some of these solutions work well on small subdivisions, but do not scale to large ones. Further, the simplifying assumptions (e.g., pipe diameter) result in illegal solutions.

The graph of roads has to be converted to a tree data structure (cycles are not allowed). Cuts are made in the road graph to create a tree. When it comes to selecting where to make the cuts, many authors optimize the sewer system layout considering only the high-level connectivity [4, 9]. An intuitive approach to cut selection is to apply standard spanning tree algorithms to the multigraph. One approach is to use the cuts that yield a shortest path spanning tree rooted at the outfall [10]. Evolution-based algorithms are a popular alternative [4, 9]. The difficulty with all of the above approaches is that the cost for each element in the graph is not fixed. For example, for an edge the connectivity of the remainder of the network affects the cost. Modifying the connectivity might change the diameter of the pipe represented by the edge. To be effective, a cut selection algorithm must address these issues.

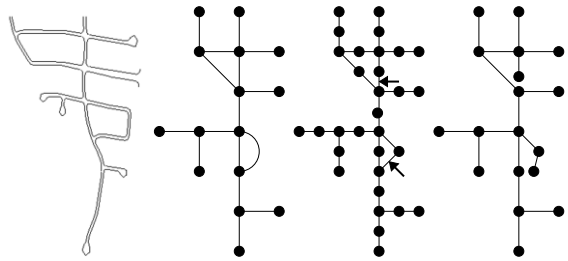

Figure 1. Small subdivision (left); multigraph (middle left); augmented graph (middle right) with two edges marked by arrows for removal to form a spanning tree; cut graph (right).

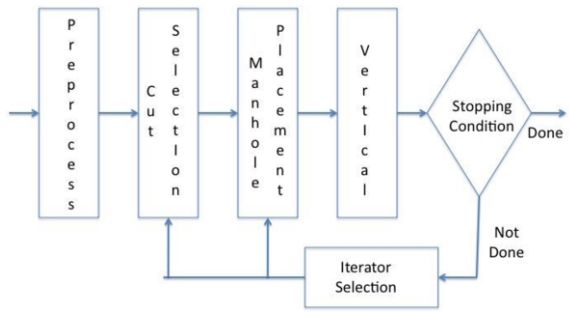

Figure 2. Sewer planner architecture.

Few researchers consider the entire sewer problem. There have been several attempts to simultaneously optimize two or more components of the problem [3, 5]. Recent work has tried multi-objective optimization [1, 7]. All of these systems were tested with either small subdivisions or a few artificial graphs. The largest problem reported solved is half that of the large subdivision reported in this paper.

Most previous research disregards the placement of manholes. A clever manhole placement scheme may result in pipelines with fewer manholes, further reducing the cost of the solution. The reduction of even a few manholes in a sewer system plan may save tens or hundreds of thousands of dollars in construction costs. Few researchers have addressed this issue. [12] presents a method for doing this, however this method requires an initial layout to be provided and is not a general solution. [3] introduces a system architecture that considers this level of optimization, but provide little detail.

4 SEWER PLACEMENT SOLVER

Consider the example subdivision shown in Figure 1 (left). This is transformed into an equivalent multigraph representation, Figure 1 (middle left), where each straight-line road segment becomes an edge and each intersection and cul de sac becomes a vertex. Cut selection algorithms are used to convert this to a cycle-free graph representing a set of pipes which cover each road in the subdivision. Figure 1 (right) is a potential cut selection result.

Our proposed solution architecture is presented in Figure 2. Given a road layout and hydrology information, the design and implementation of a sewer system can be broken down into four problems:

- Cut selection: take a network of roads, which may contain cycles, and produce a graph of roads and intersections with no cycles;
- Global layout: choose the manhole locations at intersections where multiple pipes meet;

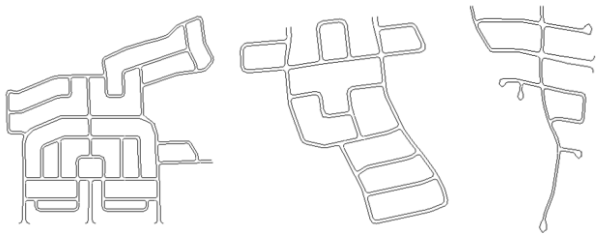

Figure 3. Test subdivisions (large, medium, and small).

- Local layout: lay down a pipeline along a single road;
- Vertical layout: resolve the third dimension by figuring out pipe diameters and depths.

With the exception of the vertical layout algorithm, which finds a local minimum, suboptimality comes from sampling and discretization. The program can increase these, allowing for improved quality solutions with increased investment of computation time.

Our system supports iterating on a solution or parts of a solution. As stated previously, the cost for each element in the graph is not fixed; for an edge the connectivity of the remainder of the network affects the cost. The vertical solver might need to get a solution, revise the edge costs, and iterate until a stable result is achieved.

All our tests used three real-world subdivisions shown in Figure 3, large (38 intersections and 55 roads), medium (18 intersections and 26 roads), and small (15 intersections and 16 roads). All three are constructed from actual curb-line survey data. They are not shown to a common scale. Note that the large subdivision has twice as many roads and intersections as the largest real-world subdivision that has been previously solved.

4.1 Cut Selection

The general problem we consider starts with nothing more than an outflow and a description of the road curb-lines. There is no information about the direction of flow, and there are almost certainly cycles. Before doing the layout we must generate a tree from the multigraph describing the roads. We cannot just remove a road from each cycle, as all roads must still be serviced. For each cycle we choose a road and introduce a cut: a small gap in the pipeline and some new manholes at the cut ends.

This cut could be placed anywhere along a road, but we only consider cutting the road right by an intersection. This only requires one additional manhole, rather than one manhole for each of two new segments of road. While this might prevent us from discovering a truly optimal solution, the cost of having an extra manhole makes this simplification seem quite reasonable.[2] The result of this process is a new graph, possibly with new vertices, which is a tree rooted at the outfall. We call this graph a cut graph, and it satisfies all the conditions necessary for running the global layout algorithm.

We found it unnecessarily complicated dealing directly with the original multigraph describing the roads. Instead we use an augmented graph, with a vertex for each intersection and road. There is an edge between road and intersection vertices if the road runs into the intersection. An example is shown in Figure 1. The small subdivision (left) has 15 intersections and 16 roads. The augmented graph (middle right) has 31 vertices representing both intersections and roads. There are 32 edges, one for each side of every road.

Generating cuts in the original multigraph is now just a matter of generating a spanning tree in the augmented graph. If a road vertex is connected to both intersection vertices in the spanning tree, the road is unmodified in the original road graph. On the other hand, if the edge to one of the intersection vertices is missing, a cut is placed on the road beside that intersection. It is not possible for both edges to be missing, as there are only two edges connected to a road vertex and all vertices will be reachable in the spanning tree. An example of this process is shown in Figure 1.

This allows us to use standard spanning tree algorithms, but it does nothing to reduce the space of possible arrangements of cuts. On the real-world test data we used, the small subdivision had only 24 possible trees, which is quite manageable. The medium sized graph had roughly ten million arrangements, and the largest had a billion. At this time, these numbers are large enough that exhaustive enumeration is not reasonable, so we randomly sample the spanning trees.

In our experiments, we tried two different sampling methods. One method was to sample uniformly at random from all spanning trees, and the other was to choose spanning trees that have a small maximum distance to the outfall. The second method finds minimum cost spanning trees where the weight of both road edges in the augmented graph is the shortest path length to the upstream intersection from the outfall in the original graph. It is not guaranteed to produce only the cut graphs with the shortest maximum pipe length, but it will produce a much lower average maximum length. The rationale for trying to find a spanning tree with a low maximum path length is that one long pipeline is likely to be more expensive than two short pipelines that have the same total length but run in parallel (roughly speaking), because the depth, and corresponding amount of dirt that must be excavated, increases over the length of the pipe.

4.2 Global Layout

Given a cycle-free pipe graph, we can consider the problem of placing the manholes within intersections (the vertices in the cut graph.) We make the simplifying assumption that we only want to minimize pipe length and number of manholes. This will minimize cost locally for any single road, but ignores global effects of flow constraint. We also ignore any constraints about the angles at which pipelines meet.

The motivating idea is to consider every possible arrangement of locations for every intersection manhole. For every arrangement, we run a local and vertical layout algorithm to get the cost. After considering every arrangement, we would know the best global layout. This is an intractable search within a non-convex continuous space, but it can be approximated by discretizing the space. We used four different subsets of the candidate manhole locations shown in Figure 4, with a spacing of 4m. The first set only contains point 1, in the center of the intersection. This provides a zero-effort baseline. The second set contains points 1 to 5, the third contains points 1 to 9, and the fourth set contains all 13 points.

Using a small set of locations makes the space finite, but the naive idea of trying all combinations of locations is still exponential in the number of intersections, with the base being determined by the number of positions per intersection. Since we assumed that all the costs are determined by local factors, we can do much better.

The network of pipes is free of cycles, and can thus be described as a tree rooted at the outfall. Combined with the costs being local, this means we can use dynamic programming to find the optimal set of locations efficiently. We visit the vertices using a postfix depth-first

[2] This may be suboptimal in the case where a road segment has a hill in the middle. Here it might be better to cut at the top of the hill.

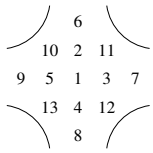

Figure 4. Global layout intersection positions.

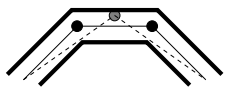

Figure 5. Reducing the number of manholes needed.

ordering, which guarantees that we will know the cost of all possible child subtrees when considering an intersection. If intersection i is a leaf, we let $cost(i, p) = 0$ for each position p of intersection i. Otherwise, $cost(i, p)$ is

$$\sum_{c \in children(i)} \min_{p_c} (cost(c, p_c) + local(i, p, c, p_c))$$

where $local(i, p, c, p_c)$ is the cost of the road from intersection i to intersection c as determined by running a local layout algorithm on that road from p to p_c.

Once the costs are known, the optimal positions for every intersection can be recovered by making a second pass through the intersections, choosing positions for the children which achieves this cost.

4.3 Local Layout

Given a road with fixed endpoints, a local layout algorithm chooses the best locations for manholes when laying a pipe down under that road. The segments of pipe can vary in length, but must remain under a maximum length and stay between the curbs of the road. There are potential savings to be found by doing a more complicated design than just running the pipeline down the center of the road. Consider the roadway in Figure 5, shown with thick lines. When the pipeline runs under the center of the road, shown by the solid line, two intermediate manholes are needed. By choosing off-centre points, as in the dashed line, only one intermediate manhole is needed. As with global layout, we use the simplifying assumption that we always want to minimize pipe length and the number of manholes.

The idea is to repeatedly find the positions that are reachable with one more pipe. For example, given one pipe of length l, we can draw an arc between the two curbs that are reachable in $\leq l$ distance and reachable from the start point. All points on the inside of this frontier are reachable using just one pipe and an extra manhole. The next frontier is generated by drawing arcs around all points within the first frontier, and taking the union of all these spaces. This process of generating new frontiers continues until the end point lies within a frontier. This is demonstrated in Figure 6 (left).

At this point we can generate the layout by tracing backwards through the frontiers. For any point outside the first frontier, there will be some point in a previous frontier which minimizes the distance from the start to the given point. This new point will be a manhole location, and we continue tracing backwards from there. This will produce an optimal solution.

Figure 6. A simple roadway with successive frontiers of reachable points (left); two strips along the roadway with dashed lines indicating allowed manhole locations (right).

```
# f is the frontier to be updated
# p is the point we are expanding from
# p_distance is the distance to reach p
UPDATE_FRONTIER( f, p, p_distance )
  for each segment s in strips
    q := farthest reachable point on s from p
    if q is a valid point
      d := DISTANCE( p, q ) + p_distance
      if s is in f
        <q', p, d'> := f[ s ]
        if q farther along s than q' or
                    q = q' and d < d'
             f[ s ] := <q, p, d>
        else f[ s ] := <q, p, d>

LOCAL_LAYOUT( start, end )
  if end->start crossing a curb
    return []

  # generate new frontiers until
  # we reach the end of the road
  create frontier f
  num_frontiers := 1
  frontiers[ num_frontiers ] := f
  finished := {}
  UPDATE_FRONTIER( f, start, 0 )
  while no point in f can reach end
    create frontier new_f
    for each segment s in f
      if s not in finished
        <p, parent, d> := f[ s ]
        UPDATE_FRONTIER( new_f, p, d )
        if p is at the end of s
          add s to finished
    num_frontiers := num_frontiers + 1
    frontiers[ num_frontiers ] := new_f
    f := new_f

  # build the final path
  s := segment in f with shortest path to end
  points := []
  for each f from num_frontiers down to 1
    f := frontiers[ f ]
    <p, parent, d> := f[ s ]
    add p to front of points
    s := segment containing parent
```

Figure 7. Local layout algorithm.

The above is not feasible for a continuous space of manhole locations, as bends or corners in the road preclude a closed form description of the frontiers. Instead, we discretize the space by dividing every straight section of the road lengthwise into strips, and only allow points along the middle of the strips. Figure 6 (right) shows a simple roadway divided into strips. A frontier is now a description of how far along a strip we can reach, for every strip. Note that running the pipe along the center of the road is equivalent to using a single strip. Figure 7 shows pseudocode for the local layout algorithm.

4.4 Vertical Layout

A complete sewer layout does not just specify the two-dimensional location of pipes and manholes. Depths must also be specified, as

well as selecting from available pipe diameters for each segment. The design must also satisfy various additional constraints, the most obvious being the capability of handling the expected load on the sewer. There exist commercial packages for automating this process (like StormCAD) that will suggest depths and pipe diameters given a description of the pipeline and the loading.

We developed a program which solves this problem for storm sewers. The cost function is messy, reflecting the real-world nature of the application. The most important factors are the number of manholes, the pipe depths (piecewise non-linear) and the pipe diameters (roughly quadratic). The solver iterates through successive mixed integer programs. Each iteration starts with the two-dimensional layout of the pipeline and the expected input to the pipeline. The output is a minimal cost set of depths and pipe diameters, given any piecewise linear approximation of a cost function.

Multiple iterations are necessary because the expected load, as determined by a commonly used prediction called the rational method, depends on hydrological information and travel times through the pipes. The times, calculated using the Manning formula, depend on the slope and diameter of the pipes, which are decided during an iteration. In the rational method, shorter times result in a larger expected load. Under-estimating the times always produces a valid design, in that the resulting sewer will handle the real load estimate given the actual times, but the pipes may be larger or steeper than necessary, which increases the cost.

The initial time estimates are set low enough to be a guaranteed underestimate, so the first proposed layout is a valid design. Actual travel times are computed for this design, and we then have times which produce a valid design as well as some new time estimates. We now proceed as follows. Using the new times, generate a new design. If this design is valid, we save the time estimates for this design, compute the new actual travel times, and continue. If the new cost is worse, we stop, and use the previous valid design. Finally, if the new layout is not valid, we generate new time estimates by taking the average of the current estimates and the times that last produced a valid layout. If no time changes by more than a minimum amount when doing this, we stop, and just use the last valid layout.

Iterating in this fashion is not guaranteed to be globally optimal, because we do not explore the entire space of flow time estimates. This code, along with a real world cost model, was used to generate the total cost estimates in the results section. The model provided a cost per manhole and a per length pipe cost over varying depths for each of 23 different pipe diameters.

While this component is similar to that in other proposed systems, ours differs by 1) using a piecewise linear approximation of the cost function, and 2) iterating starting with a lower bound.

4.5 Complete System

The information that is available initially is a collection of curb lines, each of which is a sequential set of three dimensional positions all the way along one continuous piece of curb. This gives the outline of the roads, as seen in the real-world data shown in Figure 3. The other piece of information needed is the hydrology information.

We use the process shown in Figure 8. Because local layout only considers local properties, we save time by precomputing local layout results for all positions of both intersections and all three possibilities on every road (no cut, or a cut at either end). After this, we repeatedly generate new cut graphs. For each cut graph, we run the global layout algorithm using the precomputed local layout costs. This gives a complete two-dimensional layout. At this point, one would use the topology and hydrology information to figure out how much water is expected to enter every pipe. In our experiments, we were forced to just use fixed estimates, as we did not have sufficient information available to compute this for arbitrary manhole locations. With these estimates, we can run the vertical layout algorithm to produce a complete sewer design and a dollar cost estimate.

```
SEWER_LAYOUT()
  for each road r
    for each pair u,d of upstream and
    and downstream manhole locations
      save results of LOCAL_LAYOUT(u,d) on r
      cut r by upstream manhole
      save results of LOCAL_LAYOUT(u,d) on r
      move cut beside downstream manhole
      save results of LOCAL_LAYOUT(u,d) on r
      remove in cut r
  for some desired number of cut graphs
    CUT_SELECTION()
    GLOBAL_LAYOUT( saved local layout costs )
    compute storm input for each pipe
    VERTICAL_LAYOUT()
    if new design has lowest cost so far
      save current design
  return saved design
```

Figure 8. Sewer system layout process.

5 RESULTS

It is difficult to do a fully realistic assessment of our system. There are no available test sets, other than a few artificial problems. Further, commercial data is not public—the final layout is public but the data used to compute the answer (e.g., hydrology) is not. Thus comparisons to deployed sewer systems are not possible. We are fortunate to have three real-world subdivisions for use in our testing. Even so, we do not have access to all the data needed to do a fair comparison between the computer-generated and human-generated solutions.

In the following, system performance is expressed in terms of number of manholes placed. A typical manhole costs between $20K and $50K (the cost grows nonlinearly with depth). The cost of the pipe is usually secondary.

The experiments were run on the three subdivisions shown in Figure 3: large (L), medium (M) and small (S). For all three subdivisions, we consider a number of randomly generated cut graphs and record the best result. The first two experiments consider the number of manholes used in the layout. The final experiment uses the vertical layout code to provide an estimated total cost of a layout.

The results for the first set of experiments, shown in Table 1, are generated from 100,000 random cut graphs.[3] We ran tests using 1, 5, and 20 strips with the local layout algorithm. For each of these cases, we use 1, 5, 9, or 13 intersection manhole positions for the global layout algorithm. Finally, we used both 30m and 60m for the maximum length of a pipe segment. The best results are in bold.

The most substantial improvement can be seen with the global layout algorithm. In the case of the large subdivision with shorter pipes, using 13 positions produced a layout with 16 fewer manholes than using a single manhole location. Using more strips for the local layout algorithm only reduced the manhole count by one when longer pipes were allowed. This is obviously a much more modest savings, but even a single manhole can be a significant cost.

The use of one strip and 13 positions at intersections is a close

[3] Experiments with artificially-constructed subdivisions show similar trends.

Strips	Manhole	30m pipes			60m pipes		
	Positions	L	M	S	L	M	S
1	1	210	98	62	101	47	31
1	5	202	93	62	94	46	30
1	9	196	90	60	89	42	28
1	13	195	90	60	88	42	27
5	1	207	97	62	98	45	31
5	5	199	92	62	93	44	29
5	9	192	**88**	**59**	88	**41**	28
5	13	191	**88**	**59**	**87**	**41**	27
20	1	206	97	62	98	45	30
20	5	199	92	62	93	44	29
20	9	191	**88**	**59**	88	**41**	28
20	13	**190**	**88**	**59**	**87**	**41**	**26**

Table 1. Manholes (random).

	30m pipes			60m pipes		
	L	M	S	L	M	S
1 position	210	98	62	101	47	30
5 positions	202	94	62	94	44	29
9 positions	**193**	**90**	**60**	**90**	**42**	28
13 positions	**193**	**90**	**60**	**90**	**42**	**27**

Table 2. Manholes (random; low maximum path length).

approximation to what a civil engineer might do (center of the road). Given this assumption, then the computer outperforms the human on the 30m pipes by 5, 2, and 1 manholes for the large, medium and small subdivisions, respectively, and by 1 in each case for the 60m pipes. These savings are in addition to the many hours of human labor required to obtain these numbers.

The results show diminishing returns as the number of strips and candidate manhole locations are increased. A typical run on the large subdivision took 23 minutes, suggesting that high-quality results can be obtained in a reasonable amount of time. Further, we can easily solve much larger problems (artificially constructed), but in this paper we limit ourselves to only real-world scenarios.

The second series of experiments, shown in Table 2, uses 100,000 random cut graphs with a low maximum path length. We only consider the case where we use 20 strips for local layout. The results are very similar to using random cut graphs with only a single strip for local layout. The decrease in performance when using low path length cut graphs is not entirely unexpected, as it is limiting the space of cut graphs being searched in an attempt to minimize a cost which manhole count does not reflect.

Trying to reduce the maximum pipeline length was done because we hoped that it would generate lower cost solutions. To look for this effect, we did a final series of experiments using the vertical layout code to generate a total cost. We first used a baseline of 1 strip and 1 position, running the pipe down the center of the road. We then used 20 strips for local layout, 13 positions for global layout, and a 30m maximum length for pipe segments. The solution was constrained to use a combination of 23 commerial pipe sizes. Results are shown in Table 3. Each data point represents the average of the minimum cost in millions of dollars over 10 runs with 2,000 cut graphs. For the large subdivision, a roughly $240,000 (7%) reduction is possible.

While the results for the minimum path length cut graphs are slightly negative for the medium and small subdivisions, the difference is smaller than the difference in the manhole counts multiplied by the $10,000 per manhole cost. Encouragingly, the largest subdivision result is positive despite this, suggesting that maximum pipeline length is a factor, but perhaps not as significant as hoped.

Type	L	M	S
Baseline	$3.459	$1.565	$0.983
Random cuts	$3.239	$1.459	$0.957
Random low depth cuts	$3.220	$1.462	$0.960

Table 3. Solution costs, in millions of dollars.

On a standard 2.83 GHz quad core PC, the computation times for the random cuts line in Table 3 were 120.8 (large), 26.3 (medium) and 11.5 (small) minutes. For quick prototyping, the linear relaxation of the vertical layout problem is generally around only 0.1% error, and is about 6 times faster on the largest problem.

6 CONCLUSIONS

In this paper, we demonstrated the first system for addressing the complete cycle of design and layout for a storm sewer system. The system can produce high-quality solutions that reduce the number of manholes and the cost of construction. The system is scalable to much larger subdivisions than has been attempted before.

This research is just the first step. The goal is to produce high-quality solutions for the problem of laying down both storm and sanitary sewers. With two sets of pipes, there will be conflicts as each system has additional connections (e.g., all houses have a connection to the sanitary sewers). The "easy" solution is to have one set of pipes at a lower elevation than the other, but this can dramatically increase the cost. Automating the two-pipe problem is one of the long-sought-after dreams of civil engineering.

REFERENCES

[1] M. Afshar, M. Akbari, and M. Marino, 'Simultaneous layout and size optimization of water distribution networks: Engineering approach', *Journal of Infrastructure Systems*, **11**(4), 221–230, (2005).

[2] K. Chau and C. Cheung, 'Knowledge representation on design of storm drainage system', in *Innovations in Applied Artificial Intelligence*, number 3029 in LNCS, 886–894, (2004).

[3] F. Diogo, G. Walters, E. de Sousa, and V. Graveto, 'Three-dimensional optimization of urban drainage systems', *Computer-Aided Civil and Infrastructure Engineering*, **15**, 409–42, (2000).

[4] Z. Geem, T. Kim, and J. Kim, 'Optimal layout of pipe networks using harmony search', in *International Conference on HydroScience and Engineering*, (2000).

[5] A. Hassanli and G. Dandy, 'Optimal layout and hydraulic design of branched networks using genetic algorithms', *Applied Engineering in Agriculture*, **21**(1), 55–62, (2005).

[6] H. Methods and S. Durrans, *Stormwater Conveyance Modeling and Design*, Haestad Press, 2003.

[7] T. Prasad and N. Park, 'Multiobjective genetic algorithms for design of water distribution networks', *Water Resources Planning and Management*, 73–82, (2004).

[8] D. Savic and G. Walters, 'Genetic operators and constraint handling for pipe network optimization', in *Evolutionary Computing*, number 933 in LNCS, 154–165, (1995).

[9] D. Smith and G. Walters, 'An evolutionary approach for finding optimal trees in unidirected networks', *European Journal of Operations Research*, **120**, 593–602, (2000).

[10] S. Tekeli and H. Belkaya, 'Computerized layout generation for sanitary sewers', *Water Resources and Management*, **112**(4), 500–515, (1986).

[11] K. Vairavamoorthy and M. Ali, 'Optimal design of water distribution systems using genetic algorithms', *Computer-Aided Civil and Infrastructure Engineering*, **15**, 374–382, (2000).

[12] G. Walters, 'The design of the optimal layout for a sewer network', *Engineering Optimization*, **9**, 37–50, (1985).