Numerical Flow Optimization in Dense Wireless Sensor Networks

Masoumeh Haghpanahi, Mehdi Kalantari and Mark Shayman
Department of Electrical & Computer Engineering, University of Maryland
{masoumeh,mehkalan,shayman}@eng.umd.edu

***Abstract*— We use a vector field model to formulate flow of information at every point of a dense wireless sensor network. The magnitude of this vector field represents the density of communication activity and its direction is toward the flow of information at each point. We present a general method for flow optimization in a wireless sensor network by minimizing the *p-norm* of the information flow vector field subject to the basic flow constraints, that is the flow conservation law and boundary constraints. We have called this problem the *p-norm* flow optimization, and use Sequential Quadratic Programming to solve it. The *p-norm* flow optimization shows interesting properties for different values of p. For p close to 1, the information routes resemble the geometric shortest paths while as p gets larger, there will be more load balancing effects in the flow. In this work we focus on the numerical analysis of the *p-norm* flow problem. For this we have to approximate the continuous flow problem by a discrete problem which adds some inaccuracy to the solutions. Therefore, we propose an algorithm to continuously compensate the error in the information flow vector field and avoid error accumulation in the system. We also discuss ways with which we can support different network geometries in the numerical analysis.**

I. Introduction

Wireless sensor networks are comprised of sensors that are equipped with wireless transceivers and so are able to form a wireless network. It is envisioned that in the future, wireless sensor networks may consist of a large number of nodes, potentially on the order of many thousands [8]. There are many applications for such networks including military, environment monitoring, agriculture, transportation systems and home applications.

As the number of wireless nodes grows, careful analysis of the network behavior becomes very hard using a discrete model in space. A main shortcoming of discrete models is that as the number of wireless nodes grows very large, often times the flow models become computationally intractable.

We use a continuous space model in formulating flow of information in a wireless network with a large number of nodes. We use a vector field model to represent flow of information at every point of the network. We introduced this idea earlier in [3] and [4] with a model inspired by electrostatics. Using a quadratic cost function, we showed that the solution to the optimization problem is found by solving a set of PDEs analogous to Maxwell's equations in electrostatics. Our work was followed by Toumpis and Tassiulas [5], [6], showing that minimizing the quadratic cost function results in optimal deployment of sensors by minimizing total number of sensors required to transport the information to the intended sinks. In [7], we have proposed an optimization problem for the case with multiple commodity flows in the network. In a recent work [1], we presented a general method for flow optimization in a wireless sensor network by minimizing a *p-norm* of the information flow vector field subject to the basic flow constraints, i.e., the flow conservation and boundary conditions. This problem is called the *p-norm* Flow Optimization Problem and it is shown that as $p \to \infty$, the results achieve maximum load balancing. When p is close to 1, routes tend to pass through geometric shortest paths from a source to a sink. This reduces the average transport delay but can overload links which lie on shortest paths. By increasing the value of p from 1, the optimization tries to spread the traffic in the network. In practice, p can be selected based on a trade-off between delay and balancing the traffic over the network. We use Sequential Quadratic Programming (SQP) which is an iterative method to solve the *p-norm* flow optimization problem. In each iteration, the quadratic approximation of the cost function is found near its operating point. The solution of the new quadratic optimization problem is then added to the operating point and the same process is repeated in the next iteration with the updated operating point until satisfying a certain stopping criterion. We showed that at each iteration, the optimal solution can be found by solving an elliptic PDE with generalized Neumann boundary condition.

Although there are similarities between the elliptic PDEs mentioned above and those studied in electrostatics and fluid dynamics, they also have some basic differences which prevents us from adopting the same techniques used in those fields. We share the same divergence property as is used in electrostatics for example; however most of their problems deal with Dirichlet boundary condition which makes the analysis completely different from our case which uses Neumann boundary constraints. On the other hand, the flow must stay within a boundary in fluid dynamics and therefore the flow equation satisfies Neumann boundary condition; however fluids are not confined to be conservative or have zero curl, whereas we have shown earlier in [3] that the information flow vector field is conservative and the optimal vector field satisfies the zero curl property. Furthermore, all problems in electrostatics deal with $p = 2$, and therefore the analogy disappears as soon as $p \neq 2$ in the *p-norm* flow analysis.

This work was partially supported by AFOSR under grant F496200210217 and NSF under grants CNS-0519554 and CCF-0729129.

These differences suggest that we have to build an independent framework in order to numerically analyze the *p-norm* flow problem.

In this paper we reveal some important issues in the numerical analysis of the *p-norm* flow optimization problem. We discuss adaptive and non-adaptive methods of discretizing the continuous *p-norm* problem for numerical analysis. We show that the adaptive method results in a better accuracy with fewer discrete points and therefore less complexity in the analysis. We propose a method to support different network geometries. We also design an algorithm to compute and compensate possible errors in the vector field after each SQP iteration; this guaranties the solution to stay in the feasible set and gradually converge to the optimal solution.

The rest of the paper is organized as follows: We start with some background on the method previously used in [1] to solve the *p-norm* flow optimization problem in section II. In section III, we go over different ways to support different network geometries. Then in section IV we discuss different ways of discretizing the continuous domain of the *p-norm* optimization problem for numerical analysis. From there, we go to section V where we propose an algorithm to compensate the error in the information flow vector field after each iteration of the SQP method to prevent error accumulation and to have stable results as we run through more iterations, and finally end the discussion by analyzing a numerical example in section VI. We conclude the paper in section VII.

II. Background: p-norm Flow Optimization Problem

In this section, we give a brief overview of our previous work and go over the concept of information flow vector field as well as describing the *p-norm* flow optimization problem. More details can be found in [1] and [2].

To start, let $D(z) = (D_x, D_y)$ denote the information flow vector field with the direction of the flow of information at point z in $\mathbf{R^2}$ and with a magnitude representing the density of information rate passing per unit length of a line segment perpendicular to the direction of $D(z)$. With the above definition, for a closed contour $C \subset A$ we have:

$$\oint_C D(z) \cdot \mathrm{d}n = \int_{S(C)} \rho(z) \mathrm{d}x \mathrm{d}y \tag{1}$$

where $\mathrm{d}n$ is the outward differential normal vector at each point on C, dot represents the inner product of vectors and $S(C)$ represents the area of the closed contour C. Furthermore, $\rho(x, y)$ is defined as a scalar function representing the spatial density of rate at which information is generated in the network. This quantity is a function of location and since all the information ends at the destination, the value of $\rho(x, y)$ at the destination is defined as a Dirac delta form such that $\int_A \rho(x, y) \mathrm{d}x \mathrm{d}y = 0$.

Relation (1) together with the *Divergence Theorem* imply that

$$\nabla \cdot D(z) = \frac{\partial D_x}{\partial x} + \frac{\partial D_y}{\partial y} = \rho(z). \tag{2}$$

On the other hand, since no load is desired to exit from or enter into the boundaries of the network's geographical area, we will have the following Neumann boundary condition:

$$D_n(z) = 0, \quad z \in \partial A \tag{3}$$

where D_n denotes the normal component of D on the boundary of A, denoted by ∂A.

It is important to note that (1) and (3) do not result in a unique solution for $D(z)$ and therefore we have freedom to impose other constraints so that the solution will satisfy certain properties. We followed a similar approach in [1] and introduced the *p-norm* flow optimization problem as:

$$\begin{aligned} \text{Minimize} \quad & J(D) = \textstyle\int_A |D(z)|^p \mathrm{d}x \mathrm{d}y \\ \text{s.t.} \quad & \nabla \cdot D(z) = \rho(z) \\ & D_n(z) = 0, \ z \in \partial A \end{aligned} \tag{4}$$

for $p > 1$.

The 2-*norm* flow optimization problem was fully discussed in [2] where it was shown that the solution results in spatial spreading of the communication load in the network. Moreover, it was shown in [5] that minimizing the quadratic cost function will minimize the number of sensor nodes required to handle total communication flow of the network. The analysis of 2-*norm* flow optimization problem shows that the optimal vector field must be conservative, i.e., the optimal $D(z)$ must have zero curl and therefore can be written as the gradient of a scalar potential function: $D(z) = \nabla U(z)$. This enables the optimization problem to be solved by the following *Poisson* partial differential equation:

$$\nabla^2 U(z) = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \rho(z) \tag{5}$$

with the Neumann boundary condition mentioned above.

The more general *p-norm* flow optimization problem, $p > 1$, was discussed in [1]. At first we showed that the *p-norm* flow optimization is a convex optimization problem. It was also shown that defining $g(D_x, D_y) = |D(z)|^p$,

$$H(g) = p(D_x^2 + D_y^2)^{\frac{p-4}{2}} \begin{pmatrix} D_x & -D_y \\ D_y & D_x \end{pmatrix} \begin{pmatrix} p-1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} D_x & D_y \\ -D_y & D_x \end{pmatrix}$$

is its Hessian. In order to solve the *p-norm* flow optimization problem, we use SQP which is an iterative method where each iteration can be summarized in the following two steps:

1) Quadratic approximation: Having an operating point value for the information flow vector field at the i^{th} iteration, denoted by $D^{(i)}$, which satisfies the divergence property and the Neumann boundary condition, we find the second order Taylor approximation of the cost function near $D^{(i)}$. For this purpose, define $e = (e_x, e_y)$ as the variation of the information flow vector field near $D^{(i)}$:

$$D(z) = D^{(i)}(z) + e(z).$$

Since D must also satisfy the constraints of the optimization problem,

$$\begin{aligned} &\nabla \cdot e(z) = 0 \\ &e_n(z) = 0, \quad z \in \partial A, \end{aligned} \tag{6}$$

e_n denoting the normal component of e and finally

$$\begin{aligned} J(D) &= \int_A |D|^p \mathrm{d}x\mathrm{d}y = \int_A |D^{(i)} + \mathrm{e}|^p \mathrm{d}x\mathrm{d}y \qquad (7) \\ &\approx \int_A |D^{(i)}|^p \mathrm{d}x\mathrm{d}y \\ &+ \int_A \big(e^T \nabla g(D^{(i)}) + \frac{1}{2} e^T H(g(D^{(i)}))e\big) \mathrm{d}x\mathrm{d}y \end{aligned}$$

where $\nabla g(D^{(i)})$ and $H(g(D^{(i)}))$ represent the gradient and the Hessian of $g(D)$ evaluated at $D^{(i)}$, respectively.

2) Optimization: In this step we find e that optimizes the quadratic cost function of the previous step:

$$\begin{aligned} \text{Minimize} \quad & J_q^{(i)}(e) = \textstyle\int_A \big(e^T R + \frac{1}{2} e^T Q e\big) \mathrm{d}x\mathrm{d}y \\ \text{s.t} \quad & \nabla \cdot e(z) = 0 \\ & e_n(z) = 0, \quad z \in \partial A \end{aligned} \tag{8}$$

where $R = \nabla g(D^{(i)})$, $Q = H(g(D^{(i)}))$ and $J_q^{(i)}(e)$ represents the cost function of the SQP in the i^{th} iteration. Now as shown earlier in [1] for the optimal solution of the quadratic optimization problem (8), there exists a scalar function $\lambda(z) : A \mapsto \mathbf{R}$ that satisfies:

$$\nabla \lambda = Qe + R.$$

The above relation together with the constraints of problem (8) will lead to the following elliptic PDE:

$$\nabla \cdot (C \nabla \lambda) = \rho \tag{9}$$

in which $C(z) = Q^{-1}(z)$ and $\rho(z) = \nabla \cdot (Q^{-1}(z) R(z))$, and with the *general Neumann* boundary condition $\vec{n}^T(C\nabla\lambda) = \vec{n}^T CR$ where $\vec{n}(z)$ denotes the outward normal vector at each point $z \in \partial A$. The optimal e can now be uniquely found by letting $e = Q^{-1}(\nabla\lambda - R)$.

III. Supporting Different Geometries

One of the issues in the *p-norm* flow optimization problem is the ability to support different geometries for the network to be deployed. Numerical PDE solvers may have different ways to handle this issue. The PDE toolbox of MATLAB, for example, has predefined matrices which must be filled in a special way to support different geometries. Moreover, numerical analyzers (including MATLAB) usually have Graphical User Interfaces in which one can draw the desired geometry, and the predefined matrices will be filled indirectly. Nonetheless, as we will discuss in more detail in section V, specifying a geometry using these ways prevents us from applying algorithms which check the feasibility of the solution. Because of the discrete nature of numerical analysis, numerical solutions are prone to error. Specifically, in order to discretize the constraints of the *p-norm* flow problem, numerical analyzers approximate the divergence operator with difference equations. This, on one hand, confines us to have plaid geometries [1]. On the other hand, the approximation error of discretization obliges us to check the feasibility of the solution after each SQP iteration. This prevents error accumulation and guaranties the solution to converge to the optimal solution by running more iterations. We will show in section V that compensating the error confines us to using rectangular geometries. This section presents an alternative way with which we can support any desired geometry while using rectangular geometries.

Consider the following flow optimization problem which is more general than problem (4):

$$\begin{aligned} \text{Minimize} \quad & J(D) = \textstyle\int_A k(z)|D(z)|^p \mathrm{d}x\mathrm{d}y \\ \text{s.t.} \quad & \nabla \cdot D(z) = \rho(z) \\ & D_n(z) = 0, \ z \in \partial A \end{aligned} \tag{10}$$

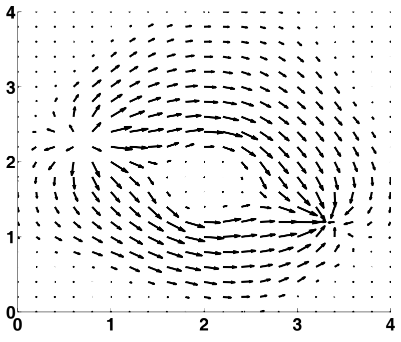
The only difference between problem (10) and (4) is in the existence of a $k(z)$ factor in the objective of the convex optimization problem. Let $k(z)$ be a positive scalar weight function on A, taking a high value in parts of the rectangular geometry where no routes are desired to pass, and a relatively small value in places where we want the actual network to be. Therefore $k(z)$ takes a small value inside the desired geometry and a relatively high[2] value in the undesired parts. In this way, the optimization problem forces the routes to stay within the desired geometry since passing through the undesired parts will drasticallly increase the cost.

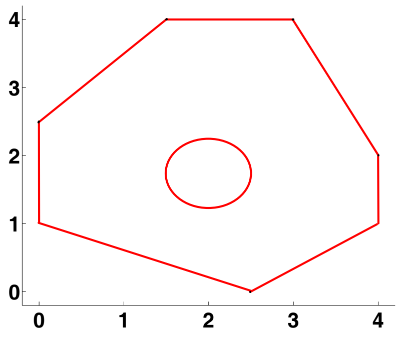
As an illustrative example, consider Fig. 1(a) which shows the result of a *2-norm* flow optimization problem in a polygon with a hole inside. As mentioned earlier, the network geometry can be defined in two ways: one way is to draw the desired geometry directly using a GUI, as shown in Fig. 1(b). The other way is to use problem (10) and suitably assign values to $k(z)$ in order to determine a network geometry within a rectangle, as shown in Fig. 1(c). In both Fig. 1(b) and 1(c), the real boundaries are specified by solid lines. The dashed lines in Fig. 1(c) are dummy boundaries which are drawn to specify our desired geometry. The area between the solid and dashed lines and inside the inner circle are parts with a high k factor. The flow diagram related to Fig. 1(b) will be drawn only within the desired geometry whereas the diagram related to Fig. 1(c) will also have small dots within the rectangle but outside the desired geometry which represent points with zero information flow. These are the points with a relatively high k factor which are forced not to relay any information through the network. This difference between the flow diagrams is not recognizable with naked eyes, therefore we did not put them separately in the paper.

Next, we consider the effect of the k factor in computing the optimal information flow vector field. We have previously shown in [3] that the set of equations for the solution of

[1]A network of uniformly spaced squares that divides a geometry into units.

[2]Theoritically, $k(z)$ must be equal to infinity in the undesired parts; however, keeping a ratio of 10^3 to 10^4 from its value in the desired parts is sufficient for the sake of numerical analysis.


(a)


(b) case I

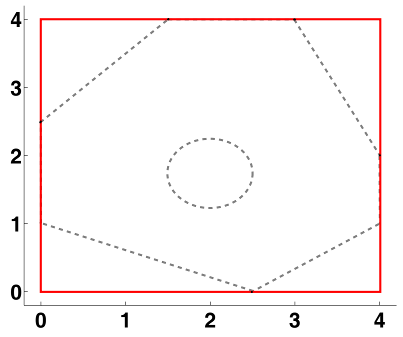

(c) case II

Fig. 1. 2-norm flow optimization problem. case I: Defining the geometry through a GUI. case II: Defining the geometry using the $k(z)$ factor.

problem (10) with $p = 2$ is analogous to Maxwell's equations in electrostatics. Specifically, $k(z)$ is analogous to the inverse of permittivity factor in a non-homogeneous media. It is also shown that this set of equations can be reduced to the following elliptic partial differential equation:

$$\nabla^2 U(x,y) - \frac{\nabla k(x,y) \cdot \nabla U(x,y)}{k(x,y)} = k(x,y)\rho(x,y)$$

or equivalently

$$\nabla \cdot (\frac{1}{k(z)} U(z)) = \rho(z)$$

where the optimal vector field $D(x,y) = \frac{1}{k(x,y)} \nabla U(x,y)$ and can be solved by numerical elliptic PDE solvers. The above vector field is the solution for $p = 2$; moreover, since it satisfies the Neumann boundary condition and the divergence property, it is in the feasible set of problem (10) and can be used as the initial value vector field needed for starting SQP iterations in finding the optimal solution of (10). The only difference here is the existence of $k(z)$ which is independent of D. This causes the Hessian and gradient of the cost function to be multiplied by $k(z)$. In other words, we have to solve the same elliptic PDE at each iteration, but with modified matrices R and Q:

$$\nabla \cdot (C \nabla \lambda) = \rho$$

where

$$\begin{aligned} C(z) &= Q_{new}^{-1}(z) = (k(z) \cdot Q(z))^{-1} \\ \rho(z) &= \nabla \cdot (Q_{new}^{-1}(z) R_{new}(z)) \\ &= \nabla \cdot (Q^{-1}(z) R(z)) \end{aligned}$$

IV. Discretization

In order to numerically analyze continuous systems of equations, one inevitable step is to discretize the domain of the problem into finitely many discrete points. The same rule applies when analyzing the *p-norm* flow optimization problem (4) for which we use Delaunay triangulation[3]algorithm to decompose the desired geometry into triangular meshes and treat triangle midpoints as discrete points. Fixing the granularity of triangles, the discrete points will have a uniform distribution over the network; this is called non-adaptive triangulation. Discrete points may also have non-uniform distribution over the network by adaptively changing the granularity of triangles which is referred to as adaptive triangulation.

[3]An optimal partitioning of the space around a set of irregular points into non-overlapping triangles and their edges. [9]

Consider a network with a single-source and a single-sink scenario. Obviously, the amount of communication activity increases as we get closer to the source or to the sink; this suggests that more precision is needed for computing the vector field near those areas. This can be satisfied by decreasing the triangulation granularity as we approach the sink and the source compared to the granularity near network boundaries. Therefore, generally it seems intuitive that we should use an adaptive triangulation with increasing number of triangles as we reach areas with larger amount of communication activity. We may also decrease the granularity over the whole geometry which will inefficiently add to the complexity of numerical analysis.

In order to show the advantage of the adaptive over the non-adaptive triangulation method, a good figure of merit would be to compute the total flux passing through a closed contour around the sink or the source. This is computed by $\oint_C D \cdot \mathrm{d}n$ where C is any closed contour around the sink or the source, and $\mathrm{d}n$ is the outward differential normal vector at each discrete point on C. Based on Equ. (1), this represents the rate at which messages exit a contour which is also equal to the total rate at which messages are generated inside it. Fig. 2 shows a rectangular network with a hole inside where no information flow should pass. This is a single-sink single-source scenario where the sink and the source are located on the right and the left side of the hole, respectively. In this figure we have used the adaptive method and computed the rate at which messages pass through different contours along the flow path from the source to the sink. As can be seen, all rates are approximately 100 which is equal to the total rate at which messages are generated at the source. The boundary condition $D_n = 0$, together with the conservative property of the vector field, force the same amount of flux to pass each cross section of the network. In Fig. 3, we have computed the flow passing a closed contour around the source with both adaptive and non-adaptive triangulation of the previous example as a function of the total number of triangles. The figure shows that the adaptive method provides a better accuracy (value closer to 100) with less number of triangles which results in less complexity of the analysis.

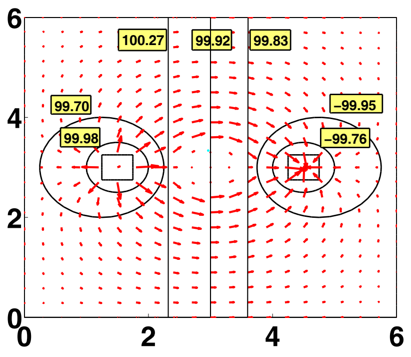


Fig. 2. Information flow passing different cross sections

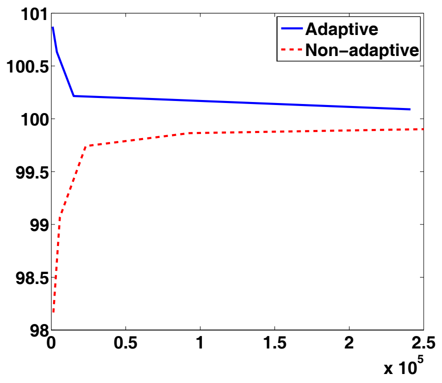


Fig. 3. Figure of merit as a function of granularity (no. of triangles)

V. Central Path

In order to numerically stabilize the solutions of the iterative method of computing the *p-norm* flow optimization problem, we have to compute and compensate any possible error in the variation of information flow vector field e after each iteration. This prevents error accumulation; otherwise, the accumulated error may cause the solutions to come out of the feasible set after a few SQP steps. Recall that in each iteration, we solve a PDE with the Neumann boundary condition $D_n(z) = 0$, $z \in \partial A$. So the boundary constraints on the vector field are always satisfied. However, due to numerical errors, the divergence property may not be fully satisfied. Therefore, in each iteration we should compute $\nabla \cdot D$ and compensate the solution based on the error. This can be done as follows:

1) For $p = 2$: As mentioned earlier, the p-norm flow optimization problem in this case is solved by the elliptic PDE $\nabla \cdot (\frac{1}{k}\nabla U) = \rho$ with a Neumann boundary condition. Finding the potential function $U(z)$, the vector field $D(z) = \frac{1}{k}\nabla U(z)$. However due to numerical inaccuracies, the solution might have some errors. To compensate the error, define a potential function $U_c(z)$ such that $\nabla \cdot (\frac{1}{k}\nabla U_c) = \rho - \nabla \cdot D$ satisfying the same Neumann boundary condition as before. This results in a unique $U_c(z)$. Now let $U_{new}(z) = U_c(z) + U(z)$. Obviously, $\nabla \cdot (\frac{1}{k}\nabla U_{new}) = \rho$, and it satisfies the Neumann boundary condition. Letting $D_{new}(z) = \frac{1}{k}\nabla U_{new}(z)$, we now have a compensated information flow vector field which satisfies the divergence property.

2) For $p \neq 2$: Having the vector field in the i^{th} iteration ($D^{(i)}$), we compute the optimal variation of the flow denoted by $e = (e_x, e_y)$, and let $D^{(i+1)} = D^{(i)} + e$. Now since $D^{(i)}$ is compensated, $\nabla \cdot D^{(i)} = \rho$, and in order to enforce the divergence property for $D^{(i+1)}$, we can add another variation vector field called e_c to $D^{(i+1)}$ such that $\nabla \cdot e_c = \nabla \cdot D^{(i)} - \nabla \cdot D^{(i+1)}$. Doing this, $D^{(i+1)}_{new} = D^{(i+1)} + e_c$ and $\nabla \cdot D^{(i+1)}_{new} = \nabla \cdot D^{(i)} = \rho$. Then we can repeat the same process for the next iteration with $D^{(i+1)}_{new}$ as the compensated initial vector field. Now letting $f = \nabla \cdot D^{(i)} - \nabla \cdot D^{(i+1)}$, we can uniquely determine e_c. We have $\nabla \cdot e_c = f$, and $\vec{n} \cdot e_c = 0$ to satisfy the boundary condition. The solution for e_c is expressed by $e_c = \frac{1}{k}\nabla U_c$; therefore, we will have the following PDE:

$$\nabla \cdot (\frac{1}{k}\nabla U_c) = f$$
$$\vec{n} \cdot \nabla U_c = 0$$

which uniquely (up to an additive constant) specifies U_c and e_c.

As can be seen from above, computing the divergence of the vector field plays an important role in compensating the error. Numerical analyzers can compute the divergence of vector fields only over plaid geometries. Therefore in order to compute the divergence and at the same time support different geometries, we have to use the $k(z)$ factor described earlier in section III.

VI. Numerical Example

As discussed earlier in [1], increasing the value of $p > 1$ in the p-norm flow optimization problem will result in increasing the spatial diversity of the information flow vector field. Our goal in this section is to analyze a *p-norm* flow optimization problem in a rather simple geometry and demonstrate the increase in the spatial diversity of the vector field as we increase the value of p. Fixing the power, we will also demonstrate the rate of convergence to the final solution as we run through different iterations.

The following analysis is for a single-source single-sink network in a rectangular geometry with a circular hole inside where the source and the sink are located in the upper left and lower right of the network, respectively. No routes are desired to pass through the hole. Therefore, we assign $k(z) = 1000$

for discrete points inside the hole, and let $k(z) = 1$ elsewhere. Fig. 4 shows the optimal information flow vector field when $p = 2$. Now choosing a different power, this result can be used as an initial value for the first SQP iteration. Choosing $p \neq 2$, SQP approximates the cost function with its second order Taylor expansion at each iteration. Therefore, it will take several iterations to converge to the optimal solution. Fig. 5 shows the maximum absolute value of the variation of information flow added in each iteration when $p = 4$. As can be seen from the figure, the variation added at each iteration is reaching zero after several iterations which makes the solution converge to a final vector field.

The effect of load balancing is shown in Fig. 6. Let S be a set that contains values of $|D|$ over the dashed circle around the sink in Fig. 4 at each iteration, and use $\max_S |D| - \min_S |D|$ as a measure of how load balances as we increase p and run more iterations. As can be seen, the difference between $|D|$ decreases at each iteration, which implies that information reaching the sink becomes more uniform in all directions as p grows larger. This further illustrates the load balancing property of the *p-norm* flow problem.

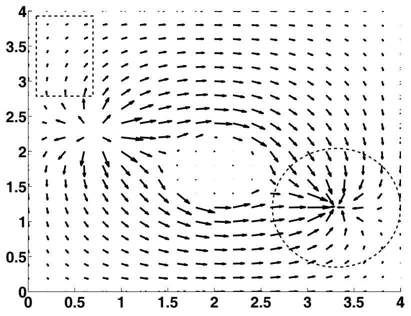


Fig. 4. p-norm flow optimization problem. power= 2, iteration= 1

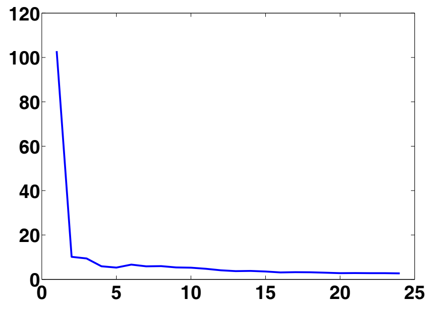


Fig. 5. Maximum absolute value of the information flow variation, i.e. $\max ||D^{(i+1)} - D^{(i)}||$ added at each iteration.

VII. Conclusions

In this paper we presented a numerical analysis of the *p-norm* flow optimization problem and discussed the different issues of numerical analysis. We presented different methods of discretizing the continuous domain of the problem into discrete points using Delaunay triangulation method. We showed that the adaptive triangulation method is superior to the non-adaptive method in reaching a certain level of accuracy with fewer triangles which leads to less complexity of the analysis. We also proposed an algorithm to compensate any possible error in the information flow vector field after each iteration to prevent error accumulation. Since the flow vector field has constraints on its divergence, this algorithm computes the divergence after each iteration and compensates the error. Numerical analyzers compute the divergence only on plaid geometries; therefore in order to support different network geometries, we presented a modified *p-norm* flow problem where we can support any desired geometry simply by assigning different values to a scalar factor $k(z)$. We ended our discussion with a numerical example in which it was shown that the variation of information flow will converge to zero as we perform more iterations. This shows that the results will converge to the desired optimal value after several iterations.

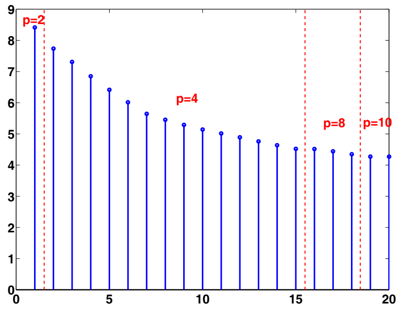


Fig. 6. $\max_S |D| - \min_S |D|$ at each iteration.

References

[1] M. Kalantari, M. Haghpanahi and M. A. Shayman. A $p-$norm Flow Optimization in Dense Wireless Sensor Networks. *IEEE INFOCOM*, 2008.

[2] M. Kalantari. Design Optimization and Security for Communication Networks. 2005.

[3] M. Kalantari and M. A. Shayman. Energy Efficient Routing in Sensor Networks. In *Conference on Information Sciences and Systems*, 2004.

[4] M. Kalantari and M. A. Shayman. Routing in Wireless Ad Hoc Networks by Analogy to Electrostatic Theory. *IEEE International Communications Conference*, 2004.

[5] S. Toumpis and L. Tassiulas. Packetostatics: Deployment of Massively Dense Sensor Networks as an Electrostatics Problem. *IEEE INFOCOM*, 2005.

[6] S. Toumpis and L. Tassiulas. Optimal Deployment of Large Wireless Sensor Networks. *IEEE Transactions on Information Theory*, $52(7) : 2935 - 2953$, 2006.

[7] M. Kalantari and M. A. Shayman. Routing in Multi-Commodity Sensor Networks Based on Partial Differential Equations. In *Proc. Conference on Information Sciences and Systems*, Princton University, NJ, March 2006.

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramanian, and E. Cayirci. A Survey on Sensor Networks. *IEEE Comun. Mag.*, vol. 40, no. 8, pp. 102-114, Aug. 2002.

[9] B. Delaunay. Sur la sphere vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793-800, 1934