
Learning Functions to Map Network Parameters to Abstract Network Characteristics

Peter A. Hamiton

1522 Copeland Rd.
Catonsville, MD 21228-1137

PETE5@UMBC.EDU

Keywords: network layout optimization, NECP learning, controlling swarm systems, self-organizing systems

Abstract

NECP Learning, a new methodology for controlling swarm systems, can be applied to the multi-objective optimization problem of planning wireless sensor network layouts according to specific requirements. Regression algorithms are applied to the set of network configurations to develop mapping functions from the configuration parameters to the desired abstract network characteristics.

1. Introduction

Constructing the optimal organization and layout scheme of a wireless sensor network (WSN) can be a difficult task. Specific knowledge about the environment must be gathered and analyzed before planning can begin, and depending upon the application, the best locations for gathering data must be identified (Jourdan, 2006). Various network characteristics must be taken into consideration and as a result, local improvements may be made to the network layout. However, these modifications may change other network characteristics, requiring additional modifications. To avoid a never ending cycle of modifications, it would be helpful to understand how different network characteristics are related and how manipulating one will affect others.

To address this problem, I propose applying a new form of swarm system control, known as NECP Learning, as an organizational method for WSN design (Miner et al., 2008). In research conducted last year for my senior Honors Thesis, I empirically demonstrated that consistent values for several abstract WSN

characteristics, specifically network connectivity, coverage, and density, can be generated over a wide set of initial network configurations when the network sensors are treated as a swarm system (Hamilton, 2008). The consistency of the abstract property parameters supports the viability of NECP Learning in the WSN domain.

In the course of my research, I generated an extensive data set listing various network configurations and resulting abstract network characteristics, measured after layout simulations were conducted with the given network parameters. By applying various regression algorithms, I plan to construct mapping functions that translate between the network configuration parameters and each abstract characteristic. The goal of this paper is to present (1) the methodology used in generating the mapping functions, and (2) the mapping functions.

I have decided to organize the paper as follows. I will cover background information and related work, touching on the work done for my Honors Thesis as well as the concept and motivations for NECP learning in Section 2. In Section 3 I will discuss the use of regression in learning the parameter mapping functions. I will present my experimental findings in Section 4, in the form of the functions produced for the current data set and the accuracy of these functions on the network configurations. I will conclude with a brief overview of the problem and research findings in Section 5.

2. Background and Related Work

This work combines several distinct fields of Computer Science and, more specifically, Artificial Intelligence, including WSN optimization, swarm intelligence, and regression-based machine learning.

2.1. Swarm Intelligence

Swarm intelligence is a field of artificial intelligence that models the intelligent behavior observed in creature swarms by using multiagent systems (Bonabeau et al., 1999). Swarm behavior can result in unexpected yet organized emergent behaviors that make the creature swarm more robust. For example, fish swarm in schools for protection from predators.

The agents used to model a swarm are often autonomous and self-organizing, following a predefined set of low-level rules that govern individual behavior. Each rule has a specific weight factor, known as an explicit control parameter (ECP), that determines the magnitude of its influence on an agent’s actions. As each agent follows the rule set, the overall behavior of the swarm may converge to produce more complicated emergent behaviors. A traditional example of emergent behavior is flocking. Demonstrated by Reynolds, flocking is produced by following three low-level rules: avoid colliding with neighboring agents (*AvoidCollision*), move towards the center of mass of neighboring agents (*MoveToMassCenter*), and move in the direction that neighboring agents are moving (*MoveWithNeighbors*) (Reynolds, 1987).

2.2. NECP Learning

NECP Learning is a new swarm intelligence technique that seeks to represent the emergent behaviors of a swarm system as individual rules (Miner et al., 2008). In this case, it is useful to think of these abstract rules as having explicit parameters, known as non-explicit control parameters (NECPs), that describe how or to what degree the swarm demonstrates the emergent behavior. For example, the rule *FormCircle()* might have an NECP to define what the radius of the circle should be. *FormCircle(60)* would direct the swarm to form a circle of radius 60.

Since emergent behaviors are products of individual agent interactions as the agents follow the rule set, there is an intuitive connection between the low-level rules, their corresponding ECPs, and the emergent behavior, which is composed of the abstract rule and specified by its NECP. It is possible to discover a specific mapping function that translates between the ECPs and the NECP using machine learning techniques. This process is called NECP Learning. The discovery of relationships between a set of ECPs and an NECP can lead to a swarm rule hierarchy that can be used in the development of additional abstract rules. As new abstraction levels are added, original emergent behaviors become intermediary “low-level” rules in the organization of a new emergent behavior.

NECP Learning is a fairly new approach to the problem of efficient swarm control. The original work validating and demonstrating the concept focused primarily on simple geometric shape formation and ant colony models (Miner et al., 2008). The *FormCircle()* rule was one of the first rules developed to demonstrate the usefulness of top-down mapping, where the NECP is used to manipulate the ECPs. The traditional machine learning problem asks only for the NECP given the ECPs, and can be solved using almost any form of regression. However, NECP Learning is concerned with learning mapping functions that translate in both directions, bottom-up and top-down (i.e. given the ECPs, return the NECP and given the NECP, return a set of ECPs). An “inverted” regression approach is needed to create a top-down mapping. The development of such an approach is the subject of current research.

2.3. Demonstrating Applicability of NECP Learning to the WSN Domain

For my senior Honors Thesis, I analyzed the possibility of using NECP Learning to accurately predict network NECPs. I developed a network simulator in Python 2.5.1 that conducted network layout simulations. In the background, the simulator treats the network sensors as agents in a single swarm. The simulator uses the traditional definition of a WSN (Romer & Mattern, 2004). Specifically, the network is homogeneous: all sensors have the same sensing and transmission range and all sensor agents have the same set of ECPs. To simplify analysis, the network environment is modeled as a two-dimensional featureless Euclidean grid, with no obstacles in the environment aside from the other sensor agents; also, all possible grid configurations are restricted to rectangular shapes, further limiting environment complexity.

The simulator defines four basic low-level rules for sensor agents to follow:

- *AvoidCloseness* - models repulsion between swarm agents according to an inverse-squared relationship. Its NECP varies the strength of the repulsion.
- *MoveToCenter* - draws an agent towards the center of the environment to ensure it is connected to the network. Its NECP varies the speed at which an isolated agent moves to the center of the environment.
- *Slowdown* - limits the maximum speed an agent can move during the simulation. Its NECP varies the rate of deceleration.

- *StayInBorders* - prevents an agent from leaving the specified target area. Its NECP varies the strength of repulsion when the agent is in range of the target area border.

Only the *AvoidCloseness* and *MoveToCenter* ECPs were sampled with differing values during the simulation process. Both *Slowdown* and *StayInBorders* were held constant with ECP values of 0.22 and 1.0, respectively. These two rules act as swarm management rules that facilitate swarm convergence. The specific ECP values chosen for them were determined in previous work (Hamilton, 2008). These ECPs filter out the network configurations that are useful for abstract rule analysis, as opposed to those that never converge and, therefore, never yield useful NECP measurements.

Each data point generated by the simulator is composed of the area dimension (Dim.), the number of swarm agents (Agent No.), the sensing range (Range), the *AvoidCloseness* ECP (AC), the *MoveToCenter* ECP (MC), and an NECP. The NECP corresponds to one of the three abstract properties: connectivity, coverage, and density.

After conducting a preliminary analysis of the data, observable consistencies in the NECP values over multiple ECP configurations were found. This finding supports the continued study of NECP Learning in the WSN domain.

3. Method

In the course of preparing my data set for analysis, I have written several Python scripts to handle converting the data to the ARFF file format and to compute statistical information regarding data point confidence intervals.

Once the data conversion process was completed, I used the Weka Toolkit (Waikato, 2008) to conduct experiments on the dataset using several different types of regression algorithms. By comparing and contrasting multiple regression models and their accuracy over the data, I am able to determine how consistent the models are and, in general, the structure of the ideal mapping function. Ideally, the models learned will be invertible, allowing for extraction of the top-down mapping function in addition to the bottom-up mapping function.

I used the following regression algorithms in my experiments:

- Linear Regression
- Simple Linear Regression

- Least Median Squared (LMS) Linear Regression (Rousseeuw & Leroy, 1987)
- Pace Regression (Wang, 2000)
- Support Vector Regression (SVR) (Drucker et al., 1997; Shevade et al., 1999)

Given the different regression techniques, I expect SVR to work best, primarily due to its scalability to higher dimensions. Accordingly, I included both SVR data transformation schemes (normalization and standardization) in the experiment set. However, I expect it to be easier to invert some of the other regression models, especially Simple Linear Regression. I also hope to determine, in general, which attributes are irrelevant in the different generic mappings.

4. Experiments

For each experiment conducted, the corresponding regression model and associated error ratings are provided in Figures 1 to 6. Weka computes several different error statistics, including mean absolute error (MAE), root mean squared error (RMSE), relative absolute error (RAE), and root relative squared error (RRSE). For this research I look primarily at the MAE and RMSE errors, though the other two are useful for comparison. Note that in Figures 1, 3, and 5, the last table column (designated C) indicates the “free” variable that allows for additional flexibility in the model.

4.1. Connectivity

The different regression models for the connectivity NECP are provided in Figure 1, while the corresponding error rates for each are listed in Figure 2. Linear and Pace Regression generate the same regression model, though the error ratings are drastically different for RAE and RRSE. Simple Linear Regression selects the **Range** attribute as the most important and ignores the rest. Interestingly, LMS Linear Regression ignores all attributes and simply predicts the value 100.0, heavily indicating that the dataset for connectivity is far too skewed to one end of the connectivity NECP value domain. This observation is corroborated by the corresponding error rating for the LMS Linear Regression model, which shows that the model performs with comparable accuracy to both Linear Regression and SVR. Both SVR schemes yield models that are generally equivalent, having the same relative weight magnitudes for the different attributes and the same error ratings.

While the SVR models seem to be the most accurate, the indication that the dataset is deeply biased indi-

cates that new data should be generated before any further work is done for the connectivity NECP.

4.2. Coverage

Like Figures 1 and 2, Figures 3 and 4 contain the regression models and corresponding error rates. Like connectivity, Simple Linear Regression ignores all attributes but **Range**, adding a flexibility variable C to the model. LMS Linear Regression generates a mapping function that heavily discounts all attributes but the MC ECP, though the model’s accuracy is generally poor. Linear and Pace Regression again yield similar models and the two SVR models are comparable in both function weights and error ratings.

Unlike connectivity, there is not much information about the dataset itself that can be immediately derived from the various mapping functions.

4.3. Density

The density NECP models are provided in Figure 5, as are the model errors in Figure 6. In general, the accuracy of most of the density models is quite high, setting density apart from both connectivity and coverage. Interestingly, the definition of the density NECP limits its value to a small range of integers, making it possible to apply classification methods, in addition to regression algorithms, to the dataset. Though this was not done in the current work, it would be an interesting expansion of the density NECP analysis.

The only two mapping functions that stand out from the model set are those generated by Linear Regression and Simple Linear Regression. Like before, Simple Linear Regression ignores all attributes but the **Range** which, interestingly, yields the worst performance of all of the generated models. Regular Linear Regression only ignores the AC ECP (note that all of the models do as well, though by assigning the AC ECP a very low model weight).

Due to the consistency of a majority of the regression models, I believe that a generic density mapping function can be inferred, indicating a “universal” function exists for this dataset.

5. Future Work

Due to time constraints, in depth testing on new network configurations was not conducted for the more promising mapping functions. An obvious next step would be to conduct those tests on all generated mapping functions to determine, in practice, which ones are most reliable or, more specifically, which are most

accurate for a subset of the network configuration space.

After function testing and verification is completed, it would be prudent to investigate how to invert the regression models to complete the set of ECP/NECP mapping functions. For example, the following simple inversion can be done with the mapping function $T(\mathbf{x})$ generated by Simple Linear Regression for the connectivity NECP:

$$T(\mathbf{x}) = 0.33 * Range + 70.04$$

$$Range = \frac{T(\mathbf{x}) - 70.04}{0.33}$$

Note that \mathbf{x} is simply the attribute vector composed of the network configuration parameters. While this is trivial for some models, it is important for others (i.e. Support Vector Regression models).

Additionally, due to some of the characteristics of the dataset, it may be necessary to go back to the simulator and generate a more robust dataset to provide an adequate spread of ECP and NECP values.

6. Conclusions

Given a challenging multi-objective optimization problem, it is possible to apply swarm intelligence to discover general mappings among the different objective criterion. With the appropriate data, applying different regression techniques is a straightforward process and yields models that not only provide adequate mappings but also reveals further information about the dataset. While certain regression techniques generate models with similar performance, it is interesting to note how preprocessing can drastically change the resulting models.

References

- Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford-University Press.
- Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems 9, NIPS 1996*, 155–161.
- Hamilton, P. A. (2008). Applying swarm rule abstraction to a wireless sensor network domain.
- Jourdan, D. B. (2006). *Wireless sensor network planning with application to uwb localization in gps-*

Learning Algorithm	Dim.	Agent No.	Range	AC	MC	C
Linear Regression	-0.0552	0.1575	0.3294	0.0618	2048.7994	73.8003
Simple Linear Regression	0	0	0.3300	0	0	70.04
LMS Linear Regression	0	0	0	0	0	100.0
Pace Regression	-0.0552	0.1575	0.3294	0.0632	2048.7994	73.7445
SVR (Normalization)	-0.0046	0.0039	0.0064	0.0026	0.0009	0.9929
SVR (Standardization)	-0.0084	0.0053	0.0092	0.0042	0.0013	0.2881

Figure 1. Regression models generated by different regression algorithms for the connectivity NECP.

Learning Algorithm	MAE	RMSE	RAE (%)	RRSE (%)
Linear Regression	9.2036	15.3789	15.3789	106.1026
Simple Linear Regression	8.7012	16.6324	100.3111	94.7492
LMS Linear Regression	5.2585	18.3196	60.6224	104.3606
Pace Regression	9.2091	15.3805	106.1663	87.6175
SVR (Normalization)	5.2379	18.1017	60.3844	103.1194
SVR (Standardization)	5.2361	18.1512	60.3633	103.4015

Figure 2. The error rates for the connectivity mapping function for various regression algorithms.

Learning Algorithm	Dim.	Agent No.	Range	AC	MC	C
Linear Regression	-0.0859	0.261	0.5017	0.0717	1537.4515	63.2099
Simple Linear Regression	0	0	0.5	0	0	54.27
LMS Linear Regression	-0.0001	0.0002	0.0003	0.0001	-0.3770	99.976
Pace Regression	-0.0859	0.2610	0.5017	0.0717	1599.9756	63.0848
SVR (Normalization)	-0.0233	0.0174	0.0303	0.0150	0.0004	0.9694
SVR (Standardization)	-0.0466	0.0302	0.0511	0.0264	0.0013	0.3417

Figure 3. Regression models generated by different regression algorithms for the coverage NECP.

Learning Algorithm	MAE	RMSE	RAE (%)	RRSE (%)
Linear Regression	10.8109	15.8587	88.6083	78.8590
Simple Linear Regression	11.1311	18.1972	91.2334	90.4872
LMS Linear Regression	8.0918	21.6642	66.3221	107.7277
Pace Regression	10.8176	15.8605	88.6633	78.8681
SVR (Normalization)	7.8298	20.3524	64.1745	101.2045
SVR (Standardization)	7.8284	20.3762	64.1630	101.3229

Figure 4. The error rates for the coverage mapping function for various regression algorithms.

Learning Algorithm	Dim.	Agent No.	Range	AC	MC	C
Linear Regression	-0.0235	0.0667	0.1232	0	42.7083	0.8693
Simple Linear Regression	0	0	0.1200	0	0	-3.4400
LMS Linear Regression	-0.0234	0.0679	0.1287	0.0007	26.7746	0.3498
Pace Regression	-0.0235	0.0667	0.1232	0.0005	42.3338	0.8492
SVR (Normalization)	-0.4769	0.3676	0.6476	-0.0002	0.0117	0.3047
SVR (Standardization)	-0.6093	0.3925	0.6923	-0.0004	0.0116	0.0005

Figure 5. Regression models generated by different regression algorithms for the density NECP.

Learning Algorithm	MAE	RMSE	RAE (%)	RRSE (%)
Linear Regression	0.7140	0.9354	25.6441	29.1707
Simple Linear Regression	2.0061	2.4201	72.0520	75.4740
LMS Linear Regression	0.7091	0.9480	25.4699	29.5657
Pace Regression	0.7140	0.9353	25.6425	29.1689
SVR (Normalization)	0.7044	0.9492	25.2980	29.6030
SVR (Standardization)	0.7044	0.9496	25.3040	29.6152

Figure 6. The error rates for the density mapping function for various regression algorithms. Simple Linear Regression intuitively yields the worst approximation. However, the remaining algorithms generate mapping functions with comparable error levels.

denied environments. Doctoral dissertation, Massachusetts Institute of Technology.

Miner, D., Hamilton, P., & desJardins, M. (2008). Learning abstract rules for swarm systems. *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (2008)*.

Reynolds, C. (1987). Flocks, herds, and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21, 25–34.

Romer, K., & Mattern, F. (2004). The design space of wireless sensor networks. *IEEE Wireless Communications*, 54–61.

Rousseeuw, P. J., & Leroy, A. M. (1987). Robust regression and outlier detection.

Shevade, S. K., Keerthi, S. S., Bhattacharyya, C., & Murthy, K. R. K. (1999). Improvements to the smo algorithm for svm regression. *IEEE Transactions on Neural Networks*.

Waikato, U. (2008). Weka 3.6.0 documentation.

Wang, Y. (2000). A new approach to fitting linear models in high dimensional spaces.