Journal of Cost Analysis and Parametrics, 5:87–97, 2012

ISSN: 1941-658X print / 2160-4746 online
DOI: 10.1080/1941658X.2012.734754

ICEAA
www.iceaaonline.com

A Case Study on Target Cost Estimation Using Back-Propagation and Genetic Algorithm Trained Neural Networks

A. SALAM[1], F. M. DEFERSHA[2], N. F. BHUIYAN[1], and M. CHEN[1]

[1]Department of Mechanical & Industrial Engineering, Concordia University, Montréal, Québec, Canada
[2]School of Engineering, University of Guelph, Guelph, Ontario, Canada

Cost estimation of new products has always been difficult as only few attributes will be known. In these situations, parametric methods are commonly used using a priori determined cost function where parameters are evaluated from historical data. Neural networks, in contrast, are non-parametric, i.e., they attempt to fit curves without being provided a predetermined function. In this article, this property of neural networks is used to investigate their applicability for cost estimation of certain major aircraft subassemblies. The study is conducted in collaboration with an aerospace company located in Montreal, Canada. Two neural network models, one trained by the gradient descent algorithm and the other by genetic algorithm, are considered and compared with one another. The study, using historical data, shows an example for which the neural network model trained by genetic algorithm is robust and fits well both the training and validation data sets.

Introduction

For a manufacturing company, being able to determine reliable cost estimates for new products is essential, whether the products are developed in-house or purchased from suppliers. This is in particular crucial in aerospace companies, where some of the components and products cost several thousands or even millions of dollars. However, cost estimation of new products has always been difficult as only few attributes will be known. A case study conducted by Bounds (1998) in the U.S. showed that only 26% of the completed projects were on time and within budget. In tackling such difficult cost estimation problems, parametric methods are commonly used where the cost engineer predetermines an *a priori* cost function with parameters to be estimated from historical data. In such parametric methods, defining an appropriate cost function is always a difficult problem. Neural networks (NNs), in contrast, are non-parametric models that attempt to fit curves without being provided a predetermined function. Moreover, researchers have commended NN models for their ability to characterize complex relationships (Caputo & Pelagagge, 2008; Cavalieri et al., 2004; Tu, 1996). To achieve this, the NN has to be "trained" using historical data where the training process can be accomplished by using different techniques that include genetic algorithms (GAs).

Address correspondence to A. Salam, Department of Mechanical & Industrial Engineering, Concordia University, 1455 De Maisonneuve Blvd. West, Montréal, Québec H3G 1M8, Canada. E-mail: a_salam@encs.concordia.ca

NNs and GAs are two computational techniques that have been actively and increasingly researched in recent years. A number of researchers have proposed systems that combine both techniques to allow the evolution of neural networks; see Yao (1993) for early studies. In NNs, computation is performed through the passing of signals within a structured arrangement of connected processing units (called neurons), in response to given input signals. A NN, in addition to its connectivity details, usually includes mechanisms that specify how weights on those connections may be changed over time in response to the inputs provided to the network. In genetic algorithms, computation is performed through the creation of an initial population of individuals followed by the evaluation, synthesis, creation, and elimination of individuals over successive generations until a satisfactory solution is found. In this study, the genetic algorithm is used to find weights for a back-propagation neural network with a fixed number of layers, neurons per layer, and other basic artificial NN (ANN) parameters. The NN is to be trained and used for estimating target costs for major aircraft subassemblies for potential new airplane programs in Bombardier Aerospace.

Problem Case

Bombardier Aerospace is a global leader in the design and manufacturing of aircraft. The company is interested in quantifying target costs of various assemblies and subassemblies for potential new aircraft. One such major subassembly is landing gear. It can be divided into different subsystems, such as main landing gear (MLG), nose landing gear, among others. The case presented in this article is the estimation of target cost of MLGs. Figure 1 depicts the fully extended MLG of a Bombardier Q400 in flight. The maximum takeoff weight (MTOW), typically measured in pounds (lbs), is the heaviest weight at which the aircraft can fly while meeting all the applicable airworthiness requirements. After conducting several interviews with internal experts and analyzing the technical specifications, it seems that the MTOW of the aircraft, the weight of the MLG, and the height of the MLG are identified as the main cost drivers. The height of the MLG, measured in inches, is the vertical height of the MLG when it is fully extended as shown in Figure 1.

FIGURE 1 Fully extended MLG of a Q400 (color figure available online).

TABLE 1 Historical data

	Cost drivers			
Program	Weight (lbs)	MTOW (lbs)	Height (inches)	Cost
1	335.77	33,000	111.16	63,816
2	335.77	36,300	111.16	69,465
3	389.97	43,000	111.54	73,794
4	490.58	64,500	124.58	125,657
5	199.58	37,850	43.29	78,516
6	265.62	47,600	43.30	117,834
7	328.81	53,000	40.85	104,635
8	333.00	51,000	42.00	103,552
9	532.00	72,750	54.80	103,173
10	532.00	80,500	54.80	114,082
11	594.00	85,970	55.00	102,595
12	526.97	92,500	75.17	104,400
13	526.97	98,000	75.17	104,408

Table 1 shows the costs of 13 MLGs of existing aircraft programs along with the corresponding potential cost drivers.

It should be noted that the cost data has been masked to protect proprietary data, and the method to mask the data that has been utilized is shown in Muralidhar et al. (1999). The problem is to develop a NN model to forecast MLG cost given this limited data set. In the development process, several test cases will be considered where in each case, 3 of the 13 programs will be held for validation and the remaining 10 will be used to train the NN. Using these test cases, the genetic algorithm trained neural network will be evaluated against the conventional gradient descent algorithm trained neural network. The ability to predict costs of the validation data set will be used as a main criterion to compare the two NNs. There has been some research conducted to use back propagation (BP) NNs for estimating the cost (Hari et al., 2011; Ji & Sun, 2011; Gharehchpogh, 2011; Chou & Tia, 2010; Feng & Xin-Zheng, 2009). Moreover, there have also been studies conducted by Salam et al. (2009) and Smith and Mason (1996) comparing the estimation for regression models and NN models, finding that NNs were more suitable. However, to our knowledge, there has not been any research conducted on comparing different types of NNs for estimating costs, as is the case in this article.

Back Propagation Trained ANN

Back propagation ANNs (BP-ANN) trained by using the gradient descent algorithm are among the most popular forms of ANNs. They are able to approximate functions based on a set of sample data. In the literature, it is shown that an ANN with a single hidden layer and a non-linear activation function can approximate the decision boundaries of arbitrary complexity (Pandya & Macy, 1996). The ANN used in this study is depicted in Figure 2.

The number of neurons in the hidden layer was chosen to be five as this was found to be good in terms of generalization (avoidance of under- or over-fitting) after several experimental runs for the problem considered in this article.

A normalized input vector representing the three cost drivers (see Table 1 before normalization) is incident on the input (left) layer and distributed to the hidden (middle)

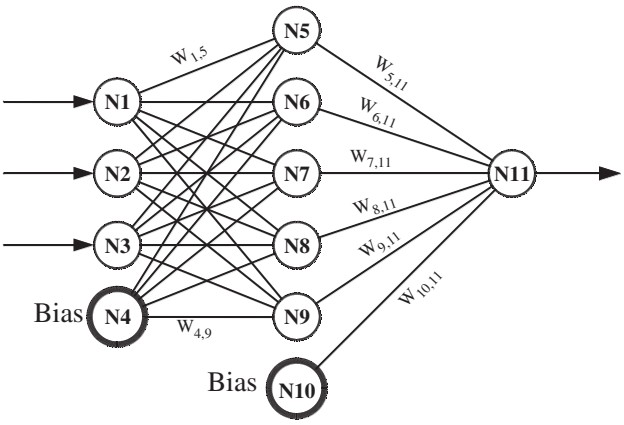

FIGURE 2 A typical back propagation neural network.

layer and finally to the output (right) layer via weighted connections. The "bias" neuron in Figure 2 always outputs unity without having an input. The bias neuron is very important, and the error-back propagation neural network without a bias neuron does not learn (Kecman, 2001). A neuron in the first layer simply spreads out its input value to all the neurons in the hidden layer. Each of the remaining neurons in the network operates by taking the sum of its weighted inputs and passing the result through a nonlinear activation function. This is shown mathematically in Equation (1). In this equation, out_j is the output of jth neuron, $w_{i,j}$ is the weight for of connection between ith source neuron and jth destination and f is a nonlinear activation function. There are several conventionally used activation functions. One of the most frequently used functions is the sigmoid function given in Equation (2). The computational simplicity of the derivative of this function simplifies the formulation of the equations needed for the training process. Moreover, this function is bounded, ensuring that certain signals remain within a range and introduces nonlinearity to the model. The term Q in this equation is referred to as the temperature of the neuron, and it determines the shape of the sigmoid. It is used to tune the network in order to improve its convergence behavior.

For the ANN to be able to predict a cost given the three cost drivers as inputs, it must first be trained using a training set. The training is a successive adjustment of the weights using the gradient descent algorithm in order to minimize a defined measure of error, which is typically the squared error (in the next section, the genetic algorithm will be presented as the training method). In this process, each time an input vector p (of the three cost drivers) from the training set is presented to the network, the difference between the desired and the actual output is computed and each weight is adjusted by an amount $\Delta w_{i,j}$, given in Equation (3). In this equation, η refers to the learning rate, $O_{p,j}$ refers to the output of the jth neuron corresponding to the input vector p, and $\delta_{p,j}$ refers to the error signal at the jth neuron $\delta_{p,j}$. The error signal is computed using Equation (4) for the single neuron in the output layer (Neuron 11) and using Equation (5) for the neurons in the hidden layer ($j = 5 \ldots 10$). Researchers, such as Huang and Huang (1991), have indicated the sensitivity of neurons in the hidden layer; however, conducting a sensitivity analysis on the neurons in the hidden layer between 3 and 30 the predictions were found to be similar for all cases. The datum t_p in Equation (4) is the desired value at the neuron in the output layer, corresponding to the pth presentation of the vector from the training set. This training algorithm is called training by back-propagation or by gradient descent algorithm. Most of the equations

presented in this section are peculiar to the neural network structure depicted in Figure 2. A more general and complete derivation of the algorithm can be found in Pandya and Macy (1996).

$$out_{i,l} = f(net_i) = f\left(\sum_j w_{j,i,l} out_{j,l-1}\right), \tag{1}$$

$$f(net_i) = \frac{1}{1 + e^{\frac{-net_i}{Q}}}, \tag{2}$$

$$\Delta w_{j,i,l} = \eta \times \delta_{p,i,l} \times O_{p,i,l}, \tag{3}$$

$$\delta_{p,1,2} = \left(t_{p,1,2} - O_{p,1,2}\right) \times O_{p,1,2} \times \left(1 - O_{p,i,2}\right), \tag{4}$$

$$\delta_{p,j,1} - O_{p,j,1} \times \left(1 - O_{p,j,1}\right) \times \delta_{p,1,2} \times w_{p,1,2}. \tag{5}$$

Genetic Algorithm Trained ANN

Genetic algorithms (GAs) introduced in the 1970s by Holland (1975), have gained increasing popularity in solving many optimization problems. It is an iterative procedure maintaining a population of structures ("chromosomes") that encode candidate solutions of a problem under consideration. Computation is performed through creation of an initial population of individuals followed by the evaluation, synthesis, creation, and elimination of individuals over successive generations until a satisfactory solution is found. Using the principle of survival of the fittest, GAs have the ability to guide their search to the most promising areas of the state space. This property can be used to find the connection weights of the back propagation neural network presented in the previous section.

Solution Representation

The chromosomal encoding of a solution is the first task in applying a genetic algorithm in any problem. In this research, a chromosomal representation of a neural network (with fixed number of layers, number of neurons per layer, sigmoid function constants) to be a string of connection weights ordered as shown in Figure 3 is considered.

The gene $W_{i,j}$ represents the connection weight between neurons i and j. This gene can assume any value between pre-specified upper bound UB and lower bound LB. These bounds on the weights can be treated as parameters of the algorithm that can be adjusted during algorithm tuning.

Input Layer Segment										Hidden Layer Segment					
N1					**N2**			**N3**			**N4**			**N5**	**N6**	**N7**	**N8**	**N9**	**N10**
$W_{1,5}$	$W_{1,6}$	$W_{1,7}$	$W_{1,8}$	$W_{1,9}$	$W_{2,5}$	⋯	$W_{2,9}$	$W_{3,5}$	⋯	$W_{3,9}$	$W_{4,5}$	⋯	$W_{4,9}$	$W_{5,11}$	$W_{6,11}$	$W_{7,11}$	$W_{8,11}$	$W_{9,11}$	$W_{10,11}$

FIGURE 3 A chromosomal representation of the neural network shown in Figure 2.

Fitness Function

The purpose of the "fitness function" is to measure the fitness of candidate solutions in the population. To calculate the fineness of an individual chromosome, we first set the values of the weights of the various connection of the neural network to those values that can be obtained from the chromosome under consideration as in Figure 3. We then use the resulting neural network to forecast target costs on all of the training data sets. Once target costs are forecasted, we calculate the sum of the squared errors where error is the difference between the desired and forecasted values. The resulting sum of squared errors is used as the fitness of the individual. Hence, the smallest this value is the more fit the individual chromosome is.

Genetic Operators

"Genetic operators" make the population evolve by creating more promising (i.e., better fit) candidate solutions to replace the less promising ones. These operators are generally categorized as selection, crossover, and mutation operators.

Selection Operator. A simple way to simulate the natural selection process in a GA is through tournament selection. In the proposed GA, we use a *k-way* tournament selection operator. In this operator, k individuals are randomly selected and the one presenting the highest fitness is declared the winner and a copy of this individual is added to the mating pool to form the next generation. Then, the k individuals in the tournament are placed back to the current population and the process is repeated. This continues until the number of individuals added to the mating pool is equal to the population size.

Crossover Operators. Once the mating pool is generated using the selection operator, the individuals in the pool are randomly paired to form parents for the next generation. Then, for each pair, the algorithm arbitrarily selects one of the available crossover operators and applies it with some degree of probability to create two child individuals by exchanging information contained in the parents. The crossover operators are (i) swap-crossover-operator-1 (SwCO-1), (ii) swap-crossover-operator-2 (SwCO-2), (iii) single-point-crossover-operator-1 (SPCO-1), and (iv) single-point-crossover-operator-2 (SPCO-2).

The crossover operator SwCO-1 arbitrarily selects a neuron (N1, N2, N3, or N4) in the input layer segment of the parent chromosomes and exchanges the weights associated to this neuron between the parent chromosomes. The SwCO-2 crossover operator exchanges the Hidden-Layer Segments of the parent chromosomes. SPCO-1 arbitral selects a crossover point in the Input-Layer-Segment of the parent chromosomes and exchanges the part of this segment lying to the left of the crossover point. SPCO-2 exchanges the part of the Hidden Layer Segment lying to the right of an arbitrarily selected crossover point on this segment. The above four crossover operators are applied with individual probabilities α_1, α_2, α_3, and α_4.

Mutation Operators. Selection and crossover do not introduce new genetic material into the population pool. This task is performed by the mutation operators acting at the gene level to alter information contained in the gene. In this research we consider a single mutation operator. This operator is applied with a small probability α_5 on a given chromosome and gets affected on each gene with another small probability α_6. Whenever this operator gets

effected on a particular gene, it steps up or down the value of this gene by a step amount θ using the equations $w_{i,j} = \min(UB, w_{i,j} + \theta)$, or $w_{i,j} = \max(LB, w_{i,j} - \theta)$, respectively. The step amount θ is calculated every time this operator is applied on a given $w_{i,j}$ with the equation $\theta = \theta_{\max} * \text{rand}(\,)$, where $\theta_{\max}$ is a parameter to be set, and *rand*( ) is the random number generator in {0, 1}.

Implementation

Once solution representation, fitness function, and genetic operators are defined, the genetic algorithm can be implemented following the general steps shown in Figure 4.

The stopping criterion used is the maximum allowable average squared error of prediction on the training data set using the "best" neural network (i.e., with the smallest error) found so far. Once the stopping criterion is reached, the algorithm is stopped and we then evaluate each neural network in the final population both on the training and validation data sets and we pick the one that outperforms the other neural networks. This is one significant advantage of the genetic algorithm-based approach over the simple gradient descent-based back propagation. At the end of its iteration, the genetic algorithm approach can provide several thousands of neural networks from which we can choose the one that provides good prediction not only on the training data set but also on the validation as well. The developed code for the GA trained NN can be found online (GA Code, 2012).

Numerical Example

In this section, we present numerical examples to compare the back propagation-trained neural network to that of the best neural network found by the genetic algorithm. As previously mentioned, the experiment is conducted on estimating the target costs for MLGs at Bombardier. Three cases were generated where subsamples of the data in Table 1 were arbitrarily picked such that ten programs were used as train data set and the remaining three were used for validation purposes. The programs used for validation were (2, 8, 12), (3, 7, 9), and (6, 7, 13) in cases 1, 2, and 3, respectively.

```
Initialize Population
Repeat
    Get a weight for the neural network from a chromosome
    Evaluate the neural network and assign a fitness to the
    chromosome
    Repeat the above two steps for each chromosome
    Perform competitive selection
    Randomly form pairs of individuals
    Apply Crossover and obtain two children from each pair
    Apply Mutation operator on each child chromosome
    Constitute the next generation from the new
    chromosomes
Until stopping criterion is reached
```

FIGURE 4 A pseudo-code for genetic algorithm.

TABLE 2 Case 1 BP-ANN vs. GA-ANN

		BP-ANN		GA-ANN	
Program	Cost	Forecast	Error %	Forecast	Error %
1	63, 816	63, 746	0.11	64, 708	1.40
3	73, 794	73, 756	0.05	75, 113	1.79
4	125, 657	125, 660	0.00	125, 132	0.42
5	78, 516	78, 665	0.19	78, 724	0.27
6	117, 834	117, 621	0.18	118, 315	0.41
7	104, 635	104, 961	0.31	105, 892	1.20
9	103, 173	105, 278	2.04	106, 056	2.79
10	114, 082	109, 176	4.30	112, 384	1.49
11	102, 595	104, 480	1.84	102, 393	0.20
13	104, 408	105, 030	0.60	102, 661	1.67
2	69, 465	66, 039	4.93	69, 472	0.01
8	103, 552	101, 768	1.72	101, 958	1.54
12	104, 400	103, 554	0.81	105, 345	0.91

TABLE 3 Case-2 BP-ANN vs. GA-ANN

		BP-ANN		GA-ANN	
Program	Cost	Forecast	Error %	Forecast	Error %
1	63, 816	63, 820	0.01	62, 400	2.22
2	69, 465	69, 299	0.24	64, 409	7.28
4	125, 657	125, 878	0.18	117, 966	6.12
5	78, 516	78, 766	0.32	82, 251	4.76
6	117, 834	117, 436	0.34	105, 175	10.74
8	103, 552	103, 944	0.38	110, 559	6.77
10	114, 082	108, 483	4.91	107, 305	5.94
11	102, 595	107, 277	4.56	104, 773	2.12
12	104, 400	105, 177	0.74	107, 047	2.54
13	104, 408	103, 989	0.40	105, 380	0.93
3	73, 794	81, 744	10.77	75, 425	2.21
7	104, 635	112, 593	7.61	111, 013	6.10
9	103, 173	109, 321	5.96	110, 109	6.72

Tables 2, 3, and 4 show both data and forecast using BP-ANN and GA-ANN for cases 1, 2, and 3, respectively. In these tables, it can be seen that the maximum error observed in all 13 data points is more or less the same in both types of neural networks. However, the GA-ANN results a lower error on the validation than the BP-ANN. A statistical test (*t*-test), which can be found in Ross (2010), comparing the means of the errors, with a 95% confidence interval also confirms that the GA model outperforms that of BP in terms of predictability. This entails that using the genetic algorithm search method enables us to find neural networks that have better generalization. Similar encouraging results were obtained in several other trials, which are not illustrated in this article.

TABLE 4 Case-3 BP-ANN vs. GA-ANN

		BP-ANN		GA-ANN	
Program	Cost	Forecast	Error %	Forecast	Error %
1	63, 816	64, 137	0.50	68, 205	6.88
2	69, 465	68, 800	0.96	72, 582	4.49
3	73, 794	74, 073	0.38	81, 354	10.25
4	125, 657	125, 394	0.21	128, 195	2.02
5	78, 516	79, 012	0.63	87, 065	10.89
8	103, 552	102, 687	0.84	106, 426	2.78
9	103, 173	107, 558	4.25	103, 751	0.56
10	114, 082	109, 218	4.26	111, 772	2.02
11	102, 595	102, 580	0.01	103, 165	0.56
12	104, 400	105, 577	1.13	104, 948	0.52
6	117, 834	105, 658	10.33	106, 122	9.94
7	104, 635	110, 544	5.65	109, 846	4.98
13	104, 408	103, 870	0.51	104, 495	0.08

Discussion and Conclusion

Neural networks and genetic algorithms are two computational techniques that have been actively and increasingly researched in recent years. A number of researchers have proposed systems that combine both techniques to allow the evolution of neural. In this article, we used genetic algorithm for training neural networks for target cost estimation. Numerical examples showed that the neural networks trained by genetic algorithm have better generalization (good prediction on the validation data). This property is essential for accurate prediction of costs on new programs with limited data, as researchers, such as Chen et al. (2000) and Fukunaga (1990), have indicated challenges on NN with limited data, however, this research indicates otherwise. Moreover, as previously mentioned, the research of Salam et al. (2009) and Smith and Mason (1996) indicate that cost estimation on NN outperforms those based on regression models. However, a limitation of the application of NN models, also highlighted by Smith and Mason (1996), which was also the case of this research applied in an industrial setting, is that management is reluctant to use NN models to their intangible phenomenon. Furthermore, even though the GA training algorithm outperforms the one of BP, it may also tend to find a local optimal solution, rather than the global one. Therefore, future research is required to test other algorithms, such as simulated annealing to better understand when NN models may perform poorly when compared to one another and to other estimation models.

References

Bounds, G. (1998). The Last Word on Project Management. *IIE Solutions*, 30, 41–43.

Caputo, A. C., & Pelagagge, P. M. (2008). Parametric and Neural Methods for Cost Estimation of Process Vessels. *International Journal of Production Economics*, 112, 934–954.

Cavalieri, S., Maccarrone, P., & Pinto, R. (2004). Parametric vs. Neural Network Models for the Estimation of Production Costs: A Case Study in the Automotive Industry. *International Journal of Production Economics*, 1, 165–177.

Chen, L. F., Liao, H. Y. M., Ko, M. T., Lin, J. C., & Yu, G. J. (2000). A New LDA-Based Face Recognition System Can Solve the Small Sample Size Problem. *Pattern Recognition*, 33, 1713–1726.

Chou, J. S., & Tia, Y. (2010). Generalized Regression Neural Nets in Estimating the High-Tech Equipment Project Cost. In *International Conference on Computer Engineering and Applications*, Bali, Indonesia (pp. 281–284).

Feng, L., & Xin-Zheng, W. (2009). The Study of the Usage of Cost-Significant Theory and Neural Network in Project Cost Estimation. In *International Workshop on Intelligent Systems and Applications*, Wuhan, China (pp. 1–4).

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*. New York: Academic Press.

GA Code. (2012). Retrieved from http://www.uoguelph.ca/~fdefersh/ga-code.rar

Gharehchopogh, F. S. (2011). Neural Networks Application in Software Cost Estimation: A Case Study. In *International Symposium on Innovations in Intelligent Systems and Applications*, Ankara, Turkey (pp. 69–73).

Hari, C. V. M. K., Sethi, T. S., Kaushal, B. S. S., & Sharma, A. (2011). CPN—A Hybrid Model for Software Cost Estimation. In *International Conference on Recent Advances in Intelligent Computational Systems*, Trivandrum, India (pp. 902–906).

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, Michigan: University of Michigan Press.

Huang, S. C., & Huang, Y.-F. (1991). Bounds on the Numbers of Neurons in Multilayer Perceptrons. *IEEE Transaction on Neural Networks*, 2, 47–55.

Ji, H., & Sun, Q. (2011). On the Cost Estimation Methods of College Engineering Based on BP Neural Network. In *International Conference on Mechanic Automation and Control Engineering*, Hohhot, China (pp. 7500–7503).

Kecman, V. (2001). *Learning and Soft Computing*. Massachusetts, England: MIT Press.

Muralidhar, K., Parsa, R., & Sarathy, R. (1999). A General Additive Data Perturbation Method for Database Security. *Management Science*, 45, 1399–1415.

Pandya, A. S., & Macy, R. B. (1996). *Pattern Recognition with Neural Networks in C++*. New York City, New York: CRC and IEEE Press.

Ross, S. (2010). *Introductory Statistics*, 3rd Edition. Waitham: Massachusetts: Academic Press.

Salam, A., Defersha, F. M., Muia, T., & Bhuiyan, N. (2009). Estimating Target Costs: A Case Study at Bombardier Aerospace. In *IIE Annual Conference and Expo 2009*, Miami, Florida.

Smith, A. E., & Mason, A. K. (1996). Cost Estimating Predictive Modeling: Regression versus Neural Network. *The Engineering Economist*, 42, 137–161.

Tu, J. (1996). Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes. *Journal of Clinical Epidemiology*, 49, 1225–1231.

Yao, X. (1993). Evolutionary Artificial Neural Networks. *International Journal of Neural Systems*, 4, 203–222.

About the Authors

Dr. Adil Salam currently manages 4 categories of indirect spend for the Canadian cross-divisional Purchasing team for the Novartis, a Pharmaceutical company headquartered in Basel, Switzerland. He received his Ph.D. degree in Mechanical Engineering (Thesis: Lean Accounting: Measuring Target Costs) from Concordia University, Montreal, Canada in 2012. He also received his Masters and Bachelor degrees in Industrial from Concordia University. His current research is in the domains of Lean Accounting, Target Costing, and Design Effort Estimation.

Dr. Fantahun M. Defersha received his PhD in Mechanical Engineering (Thesis area - Manufacturing System) in 2006 from Concordia University, Montreal, Canada. He is currently an assistant professor in the School of Engineering, University of Guelph, Guelph, Ontario, CANADA. His research interests are in Manufacturing system analysis,

flexible and cellular manufacturing systems, reconfigurable machine tools and reconfigurable manufacturing systems, part sequencing and scheduling, production and inventory planning, lean manufacturing, productivity improvement and cost analysis, artificial intelligence, and meta-heuristics.

Dr. Nadia F. Bhuiyan is an Associate Professor in the Department of Mechanical and Industrial Engineering at Concordia University (Montreal, Canada). She is also the Associate Director of the Concordia Institute of Aerospace and Design Innovation. Dr. Bhuiyan's research focuses on product development processes and lean, dealing with the design, development, production, and distribution of goods and services, with a focus on emerging tools and techniques for integrating design and manufacturing to improve process performance.

Dr. M. Chen is a Professor in the Department of Mechanical and Industrial Engineering, Concordia University. He received his B.Eng. and MS degrees in Industrial Management Engineering from Beijing University of Aeronautics and Astronautics, Beijing, China. He received his Ph.D. degree in Industrial Engineering from University of Manitoba in 1991. He was a faculty member at the University of Regina before he joined Concordia in 1999. His research is in manufacturing system and process optimization as well as in supply chain management.