Reliable Optimisation Control of a Reactive Polymer Composite Moulding Process Using Ant Colony Optimisation and Bootstrap Aggregated Neural Networks

Kazim-Junior Rod Mohammed and Jie Zhang

School of Chemical Engineering and Advanced Materials, Newcastle University, Newcastle upon Tyne NE1 7RU, UK

jie.zhang@newcastle.ac.uk

Abstract. This paper presents a study on the optimisation control of a reactive polymer composite moulding process using ant colony optimisation and bootstrap aggregated neural networks. In order to overcome the difficulties in developing accurate mechanistic models for reactive polymer composite moulding processes, neural network models are developed from process operation data. Bootstrap aggregated neural networks are used to enhance model prediction accuracy and reliability. Ant colony optimisation is able to cope with optimisation problems with multiple local optima and is able to find the global optimum. Ant colony optimisation is used in this study to find the optimal curing temperature profile. In order to enhance the reliability of the optimisation control policy, model prediction confidence bound offered by bootstrap aggregated neural networks is incorporated in the optimisation objective function so that unreliable predictions are penalised. The proposed method is tested on a simulated reactive polymer composite moulding process.

Keywords: *neural networks, ant colony optimisation, reactive polymer composite moulding, optimisation control.*

1 Introduction

Polymer composite materials have achieved widespread use in industry, including automobile, aviation and construction. Their manufacture involves thermoset curing which is dominated by complex dynamics and trial and error has been the only practical optimisation method to improve process operation [1]. This presents the opportunity for additional bottom-line economic benefit by application of computer based optimisation control methods, which require accurate process models.

The development of detailed mechanistic models for reactive polymer composite moulding processes is generally time consuming and effort demanding. The determination of many kinetic parameters in the mechanistic models is a difficult task. Thus, developing detailed mechanistic models would not be suitable for agile manufacturing processes where products and raw materials often change in response to market condition and customer demands. Data based models should therefore be utilised. Neural networks have been shown to be able to approximate any nonlinear functions [2] and have been applied to nonlinear process modelling [3-4].

A limitation of conventional neural networks is that they can lack generalisation capability. That is they can give very good performance on the training data but not so good, or even poor, performance on unseen data. There are several approaches for enhancing neural network generalisation capability, such as training with regularisation [5], early stopping [6], Bayesian learning [7], and combining multiple neural networks [8-10]. Among these approaches, combining multiple neural networks is the most effective one. One approach to combining multiple neural networks is through developing multiple neural networks on bootstrap re-sampling replications of the original training data [11]. Bootstrap aggregated neural networks (BAGNET) are shown to give more reliable and accurate prediction performance on unseen data than single neural networks [12]. Furthermore, model prediction confidence bounds can be calculated from the multiple neural network predictions [12]. The idea of bootstrap aggregation has also been incorporated with other nonlinear modeling tools such as supporting vector machine (SVM) [13] and Gaussian processes [14].

Conventional gradient based optimisation methods can be trapped in local minima, especially when highly nonlinear optimisation objective functions are involved. To overcome this problem, population based optimisation methods such as genetic algorithms [15], particle swarm optimisation [16], and ant colony optimisation (ACO) [17] can be used.

This paper presents a study on using ACO and BAGNET for the optimisation control of a reactive polymer composite moulding process. The paper is organised as follows. Section 2 presents a simulated reactive polymer composite moulding process and its control objective. Section 3 presents ACO for optimisation. Modelling of a reactive polymer composite moulding process using BAGNET is presented in Section 4. A reliable optimisation control method based on ACO and BAGNET is presented in Section 5 together with application results. The last section concludes this paper.

2 A Simulated Reactive Polymer Composite Moulding Process

The optimisation control strategy is tested on simulation using the mechanistic model given in reference [1]. The basic kinetic model is a combination of autocatalytic and *n*th order reaction terms with Arrhenius dependence of the rate constants:

$$\frac{da}{dt} = P_3 \exp\left(\frac{P_1}{R(T+273)}\right)(1-a)^{P_5} + P_4 \exp\left(\frac{P_2}{R(T+273)}\right)a^{P_6}(1-a)^{P_7} \qquad (1)$$

where a is the degree of cure, T is the curing temperature, R is the perfect gas content, P_1 (= -71.46) and P_2 (=-61.25) are the activation energies, P_3 (=2.54×10^4) and P_4 (=6.05×10^4) are the rates constants, P_5 (=0.492), P_6 (=1.252), and P_7 (=1.750) are the reaction orders. Two improved models incorporating glass transition temperature and a diffusion term are given in [1]. The improved model is used in this study and the model parameters can be found in [1]. The simulated reactive polymer composite moudling process is concerned with polymer composite moulding using single component epoxy resin. This type of polymer composite is commonly used in produce automobile parts [1].

The control objective is to maximise the degree of cure at the end of the batch. Only when the degree of cure is close to 1 the mould can be opened, otherwise the product could deform leading to defective products. The control policy consists of a temperature setpoint profile, **U**, throughout the batch cycle. The batch duration is divided into N equal stages and the temperature is kept constant at each stage. Thus a control profile can be represented as the following vector,

$$\mathbf{U} = [\ U(1), U(2), \ldots , U(N)\ ]^T \tag{2}$$

where $U(j)$ is the temperature setpoint at the jth stage.

3 Ant Colony Optimisation

3.1 Introduction of ant colony optimisation

The ACO meta-heuristic, based on the metaphor of foraging ants, was proposed by Dorigo and colleagues in the early 1990s [17]. It is a type of swarm intelligence, in which simple agents interact among themselves with no central control system. These and other stochastic optimisation techniques, such as evolutionary algorithms, are renowned for their ability to find global optima. This is in contrast to their traditional gradient-based counterparts, which tend to get trapped in local optima.

ACO is inspired by the behaviour of ants which are able to find the shortest path between the nest and food source. The ants are almost blind and therefore initially move about randomly. Each ant deposits pheromone while it moves, producing a trail. If another ant comes close, it can smell the pheromone and correspondingly follow the trail, depositing its own pheromone. Therefore the greater the pheromone, the more likely a close ant is to pursue the trail. Fig. 1 illustrates the selection a shorter route. When the ants approach (Fig. 1 A), the upper and lower paths are selected with equal probability (Fig. 1 B), as there are no pheromone deposits, shown as dashed lines. Assuming the ants move at the same speed, a shorter route has more pheromone at its end (Fig. 1 C), and is therefore more likely to be tracked by returning ants which approach it (Fig. 1 D). The ants quickly converge on the shortest path via this positive feedback mechanism, also termed “autocatalysis”. This behaviour is practised by the artificial ants of ACO in finding optima.

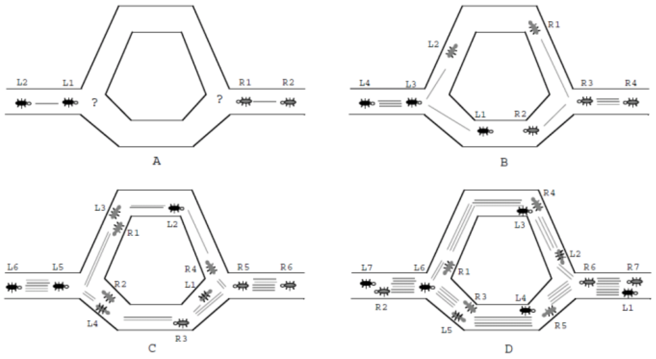

Fig. 1. How real ants select shorter path

When the food source exhausts, the ants no longer deposit pheromone on the path. The pheromone evaporates gradually, reducing the probability of successive ants following it, and hence encourages random search for new sources. This analogy is utilised in ACO for artificial ants ruling out local optima and searching for the global one.

Not many ant-inspired algorithms have been proposed for continuous problems. The first was by Bilchev and Parmee [18, 19] for local search. It was extended to global search and coined, Continuous ACO (CACO), by Woodrich and Bilchev [20]. CACO was found to be the best performing algorithm in terms of least function evaluations and was used in this study. The CACO algorithm [20, 21] is reviewed in the next subsection.

3.2 Continuous ACO

To apply ACO to continuous optimisation problems, the decision variable (x) domain has to be first divided into a specific number of, for example, R, randomly distributed regions. The algorithm employs A number of ants, of which G explore globally, while the remainder, L scout locally. The regions are initially randomly positioned within the problem space. Then global and local searches are performed iteratively, and work in tandem. The former finds promising regions, while the latter fine-searches them. This facilitates finding the global optimum.

3.2.1Global Search

The global search entails replacing the G weakest regions (with the highest objective function values) via two genetic algorithm (GA) inspired processes, termed 'Random Walk' and 'Trail Diffusion'. The new 'child' regions are obtained using information of 'parent' regions, selected from the (R-G) stronger regions (with lower objective functions). Considering that the regions are initially distributed throughout the problem space, this method iteratively eliminates less fit regions. With new children determined via stochastic operations on positions of more fit parents, convergence occurs with some diversity over the iterations.

Random Walk comprises two steps, 'Crossover' and 'Mutation'. In Crossover, the first element of the new child region's location vector is equal to that of a parent region, randomly selected from the fitter regions. Subsequent elements are each set, with a crossover probability, *CP*, to that of another random parent of the stronger regions, else it is assigned the corresponding element of the first parent. Note that if *CP* is 1, all child elements are from unique parents, while if 0, the child is the same as the first parent. Thus crossover produces new regions from a possible diversity of the fitter ones.

After inheriting stochastically from parents, the children's 'genes' are then mutated. Each element is modified, with a mutation probably, *MP*. This involves randomly adding or subtracting a step of size, Δ:

$$\Delta(I, M_s) = M_s(1 - r^{(1-I)^b}) \tag{3}$$

where M_S is the maximum step size which will keep the element within bounds, r is a random number in the interval [0, 1], I is ratio of current iteration number, n, to number of maximum iterations, $n_{\max}$, and b is a positive parameter.

Note that M_S is effectively the distance of the element from the upper bound, if Δ is being added, or from the lower bound for subtraction of Δ. I is 0 at the start, and 1 at the end; it therefore reduces the possible extent of the mutation over the iterations, and facilitates convergence of the regions towards a final solution. The parameter b influences the degree of nonlinearity of the decrease in step size over the iterations.

While in Random Walk, one parent is chosen at a time, Trail Diffusion selects a pair from the set of fitter regions. The probability, *CP*, is again used to decide when new parents are selected. The i^{th} element, $x_i(child)$, is set to one of the following options:

1. the respective element, $x_i(parent_1)$, of the first parent
2. the corresponding element, $x_i(parent_2)$, of the second parent
3. a weighted combination of the parents' respective elements:

$$x_i(child) = \alpha \cdot x_i(parent_1) + (1-\alpha) \cdot x_i(parent_2) \tag{4}$$

where α is a random number in the interval [0, 1].

The probability of option 3 is *MP*, while the remainder, (1 – *MP*), is equally shared for options 1 and 2. Thus options 1 and 2 each occurs with the probability of (1 – *MP*)/2.

After the new regions are generated, their pheromone trail values are calculated as the average of their parents'. In this, the child's trail value lies somewhere within the parents' values. Given that local ants are attracted to a region based on its pheromone deposit, using an average of the parents' values will ensure that local ants are not unfairly drawn to the child nor deterred from it.

3.2.2 Local Search

After more promising regions are found in the global search, *L* regions are probabilistically chosen for local search. In Woodrich and Bilchev's original version of the CACO [20], the regions were selected from all *R* regions. However, a modification for faster convergence by Mathur [21] is utilised, where the ants pick from the strongest S % of R regions. The probability of the j^{th} region being chosen is:

$$P_j(n) = \frac{\tau_j(n)}{\sum_k \tau_k(n)} \tag{5}$$

where τ_j is the j^{th} region's pheromone trail value, n is the current iteration number, and k designates the set of regions from which the j^{th} is chosen.

A local ant moves a short distance, d, determined from the region's age, *age*. The region's age represents the length of survival of the region. The minimum age is 0 at the start of the iterations, and at which there is a corresponding maximum distance, $d_{\max}$. At the maximum age, $age_{\max}$, the distance is at its minimum, $d_{\min}$. The maximum age corresponds to the termination of the ACO algorithm. Linear interpolation is applied to find d at current iteration, n, for the j^{th} region, $d_j(n)$:

$$d_j(n) = \frac{d_{\max} - d_{\min}}{0 - age_{\max}} age_j(n) + d_{\max} \tag{6}$$

The direction of the move is the same as that by the previous local ant at the region, if improved fitness was then found. This follows from stochastic hill-climbing, where the direction of promising results is maintained. Else, the direction is random.

Fitness is evaluated at the ant's position after the move. If it increased, the region is moved to this position, pheromone is added proportional to the increase, and the region's age is decremented. Thus promising results encourage future search in the same direction, increasing the distance of the next move. Otherwise, the age is incremented, reducing the extent and relative likelihood of later searches due to pheromone evaporation.

3.2.3 Pheromone Evaporation

Pheromone trails evaporate at the end of each iteration. The j^{th} region's value at the end of the n^{th} iteration, and hence during the $(n+1)^{\text{th}}$, $\tau_j(n+1)$, is given by:

$$\tau_j(n+1) = \rho \cdot \tau_j(n) \tag{7}$$

where ρ is the evaporation rate.

In future iterations, regions which did not receive pheromone deposits are less likely to be searched locally. It is similar to ants losing interest as food exhausts, encouraging search elsewhere. This helps escaping from local optima.

Outline of Algorithm

1. Initialise parameters

2. Randomly generate *R* regions' locations
3. Calculate fitness of regions
4. Global search:
    a. send global ants to *G* weakest regions
    b. create *G* new regions via Random Walk and Trail Diffusion:
        i. Random Walk – construct *RW* new regions by Crossover with probability, *CP*, then apply Mutation with probability, *MP*
        ii. Trail Diffusion – construct *TD* new regions, with probability, *CP*, from 2 randomly selected parents
    c. update trail values and fitness of new regions
5. Local search:
    a. send local ants to *L* of strongest *S* % of regions
    b. move each ant a short distance, size of which is proportional to region's age; direction is same as by last region's local ant, if fitness improved then, otherwise direction is random
    c. if fitness improves, move region to ant's location, update pheromone trail value proportional to fitness improvement, update fitness, and decrement age; otherwise, increment age
6. Repeat steps 4 and 5 until maximum number of iterations. Optimum is region location with maximum fitness

3.2.4 Handling constraints

Constraints can be handled using the penalty function method. The approach used by Mathur [21] was adopted for including constraints. This involved calculating the constraint violation, *v*, as:

$$v(x) = \begin{cases} \sum_i g_i(x): \text{ for } g_i(x) > 0 \\ 0: \text{ otherwise} \end{cases} \tag{8}$$

where i^{th} constraint is $g_i(x) \leq 0$.

The objective function was modified to include *v*, which is penalised by a weight, w_v:

$$J(x,v) = f(x) - w_v \cdot (v(x))^2 \tag{9}$$

where $J(x,v)$ is the new constrained objective function, $f(x)$ is the original unconstrained objective function, and w_v is gradually and linearly increased over the iterations from an initially small value.

4 Modelling Using Bootstrap Aggregated Neural Networks

4.1 Bootstrap aggregated neural networks

A diagram of bootstrap aggregated neural networks is shown in Fig. 2, where several neural network models are developed to model the same relationship. Instead of selecting a “best” single neural network model, these individual neural networks are combined together to improve model accuracy and robustness. The overall output of the aggregated neural network is a weighted combination of the individual neural network outputs. This can be represented by the following equation.

$$f(X) = \sum_{i=1}^{n} w_i f_i(X) \tag{10}$$

where $f(X)$ is the aggregated neural network predictor, $f_i(X)$ is the ith neural network output, w_i is the aggregating weight for combining the ith neural network, n is the number of neural networks, and X is a vector of neural network inputs. Proper determination of the stacking weights is essential for good modelling performance. A popular choice of stacking weights is simple averaging, i.e. the stacked neural network output is an average of the individual network outputs. Since the individual neural networks are highly correlated, appropriate stacking weights could be obtained through principal component regression (PCR) [10]. Instead of using constant stacking weights, the stacking weights can also dynamically change with the model inputs [22, 23].

Another advantage of bootstrap aggregated neural network is that model prediction confidence bounds can be calculated from individual network predictions [12]. The standard error of the ith predicted value is estimated as

$$\sigma_e(x_i) = \{\frac{1}{n-1}\sum_{b=1}^{n}[y(x_i;W^b) - y(x_i;\cdot)]^2\}^{1/2} \tag{11}$$

where $y(x_i; .) = \sum_{b=1}^{n} y(x_i;W^b)/n$ and n is the number of neural networks in an aggregated neural network. Assuming that the individual network prediction errors are normally distributed, the 95% prediction confidence bounds can be calculated as $y(x_i; .) \pm 1.96\sigma_e(x_i)$. A narrower confidence bound, i.e. smaller $\sigma_e(x_i)$, indicates that the associated model prediction is more reliable.

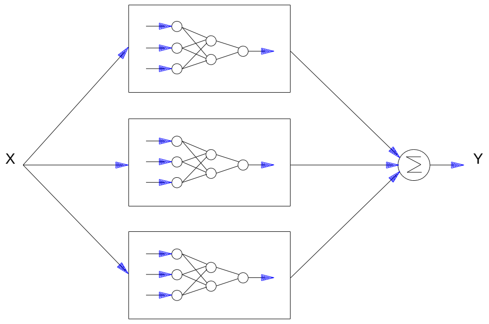

Fig. 2. A bootstrap aggregated neural network

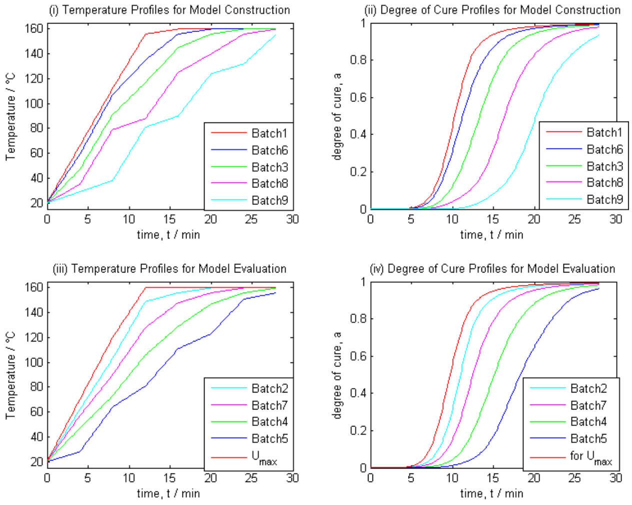

Fig. 3. Simulated process data

4.2 Modelling of the degree of cure using bootstrap aggregated neural networks

Developing data based models from limited process data is of vital importance in agile manufacturing processes. It was assumed here that valid historical data was limited – therefore 9 batches were simulated and are shown in Fig. 3. The batch time is 28 minutes and the batch duration is divided into 7 stages of 4 minutes. Of these 9 batches, batches 1, 3, 6, 8 and 9 were selected for neural network model construction, while batches 2, 4, 5, and 7 were selected as "unseen" testing data for evaluating the model. Notice that the model evaluation batches were chosen to interleave the model construction batches. The model was therefore built from a wider range of data, and validated throughout this range. From the neural network model construction data, bootstrap re-sampling with replacement [24] was used to generate 30 replications of the data. In generating a replication of this data set, a sample is randomly taken from the data set. This sample is copied and put back to the original data set. This is repeated until a replication of the data set has been generated. A neural network is developed on each bootstrap replication data set. In this study, simple averaging is used to combine the individual networks.

In order to show the advantages of BAGNET, a single neural network model was also developed. All the individual network networks are feed forward neural networks with a single hidden layer. The models represented by the neural networks are of the following form:

$$y(t) = f[y(t\text{-}1), u(t\text{-}1)] \tag{12}$$

where y is the degree of cure, u is the applied temperature, t is discrete time, $f()$ is a nonlinear function represented by the neural network.

The model can be used for calculating one-step-ahead predictions and multi-step-ahead (or long range) predictions as shown in Eq(13) and Eq(14) respectively.

$$\hat{y}(t) = f[y(t-1), u(t-1)] \tag{13}$$

$$\hat{y}(t) = f[\hat{y}(t-1), u(t-1)] \tag{14}$$

In one-step-ahead prediction, the measured degree of cure at time t-1, $y(t\text{-}1)$, is used to predict the degree of cure at time t, $\hat{y}(t)$. In multi-step-ahead prediction, the predicted degree of cure at time t-1, $\hat{y}(t-1)$, is used to predict the degree of cure at time t, $\hat{y}(t)$. For the optimisation control of polymer composite curing process, the control interest is on the final degree of cure at the end of a batch. Thus, multi-step-ahead or long range predictions are required.

The networks were trained using the Levenberg-Marquardt optimisation algorithm with regularisation and cross-validation based “early-stopping”. The number of hidden neurons used is determined though cross validation. The data for building neural network models are further partitioned into training and validation sets. A number of neural networks were developed and tested on the validation data. The network that gives the lowest sum of squared errors (SSE) on the validation data is considered to have the appropriate number of hidden neurons. Fig. 4 shows that 6 hidden neurons give the best performance. Thus all the individual networks use 6 hidden neurons.

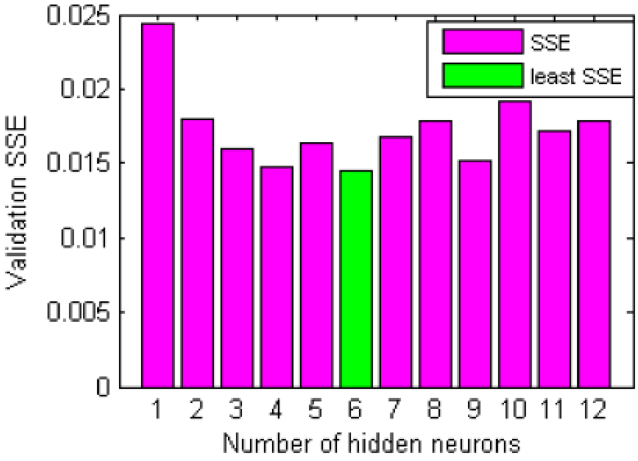

Fig. 4. SSE on the validation data

In building bootstrap aggregated neural network, 30 individual neural networks were developed and then combined. Fig. 5 shows the long range prediction SSE from bootstrap aggregated neural network with different number of networks combined. It can be clearly seen that the model prediction error reduces significantly when networks are combined. The least long-range prediction SSE on the validation subset is produced by a BAGNET with 25 networks, as shown in Fig. 5. However, SSEs for low number of neural networks are highly variable because the networks were trained just once, producing possible sub-optimal performance. As the number of neural networks increases, the variability averages out, and the SSE therefore decreases gradually. Therefore, for reliable and accurate predictions, large number of neural networks should be used. Beyond 20 networks, there is little change in SSE, while computational processing requirements are significantly increased. In this regard, a value at or just above 21 would be appropriate. A sensible value of 30 was chosen for the sake of reliable estimates of standard deviations of model predictions.

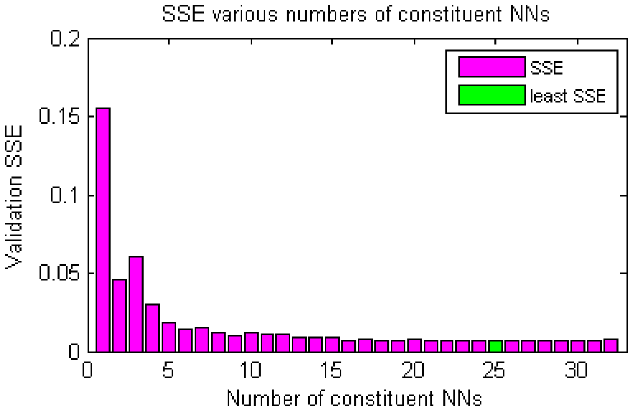

Fig. 5. Long range prediction SSE for bootstrap aggregated neural networks with various numbers of constituent networks

The mean squared errors (MSE) of long-range predictions are calculated and shown in Table 2. It can be seen from Table 1 that BAGNET gives much better performance than a single neural network.

Table 1. Performance of BAGNET and single network

Model	MSE (training)	MSE (testing)
BAGNET	0.56×10^{-3}	1.1×10^{-3}
Single network	2.5×10^{-3}	2.4×10^{-3}

5 Reliable Optimisation Control

Using a neural network dynamic model, the optimal control profile (i.e. heating profile) is calculated off-line by solving the following optimisation problem.

$$\min_{u_1,\cdots u_N,t_f} J = \beta_1[\alpha_d - \alpha(t_f)] + \beta_2 t_f + \beta_3\sigma_e(t_f)$$

s.t. product quality and operation constraints

where α_d is the desired degree of cure, t_f is the batch time, σ_e is the standard error of neural network predictions, β_1, β_2 and β_3 are weighting factors, and $u_1,\ldots, u_N$ form the control profile. Earlier studies [25, 26] show that penalising wide model prediction confidence bounds (i.e. large σ_e) leads to reliable optimal control policies.

The first term in the objective function intends to bring the final degree of cure to the desired value, which is typically 1, the maximum value. High value of degree of cure means that the product is highly solidified and the mould can be opened without affecting the product. The second term intends to reduce the batch time or curing time. Shorter batch time means that more parts can be produced for a given time, hence, improving the productivity. As the optimization is performed on a model and model plant mismatches are unavoidable, “optimal on model” may not be “optimal on plant”. The third term in the objective function penalizes wide model prediction confidence bound in order to improve the model prediction reliability under the optimisation control policy.

The CACO was applied to this polymer moulding process by using the final developed BAGNET model in the objective function. In order to prove the CACO’s functionality, a consistency check, involving 30 runs, was performed. The performance of CACO is compared with that of SQP. For SQP implementations, random initial values were employed in each run. Note that in evaluating an optimiser’s achievement of desired accuracy, and hence success, the ‘reference’ solution was taken to be the best of 100 runs on the gradient-based optimiser, with random initial values for each run.

From Table 2, the CACO was 100% reliable in optimising. In contrast, the gradient-based optimiser had success rates less than 100%. Therefore the CACO was proven to be more reliable in optimising the neural network model. However, the CACO generally requires more function evaluations in finding the solution. This is due to the method of constraints handling and to the algorithm itself, which was found by others such as Socha and Dorigo [27] to be less efficient than later continuous ACO versions. Such faster algorithms, if as reliable as the CACO on the neural network model, will be more appealing for implementation on the real process.

Table 2. Success rate of CACO and SQP

β_3	SQP	CACO
0	97	100
0.1	90	100
0.5	90	100
1	70	100

The effect of incorporating model prediction confidence in the optimisation objective function is shown in Table 3. In Table 3, a is the actual degree of cure under the optimal control policy, a_p is the neural network model predicted degree of cure under the optimal control policy, and σ_e is the standard error of model predictions. When the model prediction confidence is not considered, i.e. β_3=0, the actual degree of cure is 9931.661×10^{-4}. When the model prediction confidence is considered, the actual degree of cure is improved. Of the cases considered in Table 4, β_3=0.5 gives the highest degree of cure on the actual process. The optimal curing temperature profile for this case is:

U=[20.0, 70.0, 120.0, 159.3, 159.3, 160.0, 160.0, 160.0]

Table 3. CACO solutions using various weights β_3

β_3	a / $\times10^{-4}$	a_p / $\times10^{-4}$	σ_e / $\times10^{-3}$
0	9931.661	9871.196	4.115
0.1	9932.344	9871.195	4.114
0.5	9933.642	9871.190	4.112
1	9931.854	9871.091	4.099

6 Conclusions

Optimisation control of a reactive polymer composite moulding process using ant colony optimisation and bootstrap aggregated neural networks is studied. Bootstrap aggregated neural networks are shown to give better and more reliable prediction accuracy than single neural networks. Reliable models can be developed rapidly from process operation data by using bootstrap aggregated neural networks. Ant colony optimisation is shown to be able to find global optimum. Incorporating model prediction confidence bound in the optimisation can improve the reliability of the calculated optimal control policy. Application results on a simulated reactive polymer composite moulding process demonstrate that the proposed method is effective. The method is also applicable to the optimisation control of other batch processes.

Acknowledgments. The work is supported by the EU through the project iREMO – intelligent reactive polymer composite moulding (grant No. NMP2-SL-2009-228662). The authors thank Dr Nikos G. Pantelelis from National Technical University of Athens for providing the simulation programme.

References

1. Pantelelis, N. G.: Towards the dynamic optimisation for the cure control of thermoset-matrix composite materials. Composites Science and Technology. 65, 1254-1263 (2005)

2. Cybenko, G.: Approximation by superposition of a sigmoidal function. Math. Control Signal Systems. 2, 303-314 (1989)

3. Bhat, N. V., McAvoy T. J.: Use of neural nets for dynamical modelling and control of chemical process systems. Computers & Chemical Engineering. 14, 573-583 (1990).

4. Bulsari, A. B.: (Ed), Computer-Aided Chemical Engineering, Vol.6, Neural Networks for Chemical Engineers. Elsevier: Amsterdam, (1995)

5. Bishop, C.: Improving the generalisation properties of radial basis function neural networks. Neural Computation. 13, 579-588 (1991).

6. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press: Oxford, (1995)

7. MacKay, D. J. C.: Bayesian interpolation. Neural Computation. 4, 415-447 (1992).

8. Sridhar, D. V., Seagrave, R. C., Bartlett, E. B.: Process modelling using stacked neural networks. AIChE Journal. 42, 2529-2539 (1996).

9. Wolpert, D. H.: Stacked generalization. Neural Networks. 5, 241-259 (1992).

10. Zhang, J., Morris, A. J., Martin, E. B., Kiparissides, C.: Inferential estimation of polymer quality using stacked neural networks. Computers & Chemical Engineering. 21, s1025-s1030 (1997)

11. Breiman, L.: Bagging predictor. Machine Learning. 24, 123--140 (1996)

12. Zhang, J.: Developing robust non-linear models through bootstrap aggregated neural networks. Neurocomputing. 25, 93-113 (1999)

13. Sattlecker, M., Baker, R., Stone, N., Bessant, C.: Support vector machine ensembles for breast cancer type prediction from mid-FTIR micro-calcification spectra. Chemometrics and Intelligent Laboratory Systems. 107, 363-370 (2011)

14. Chen, T., Ren, J: Bagging for Gaussian process regression, Neurocomputing. 72, 1605-1610 (2009)

15. Goldberg, D. E.: Genetic algorithms in search, optimisation and machine learning. Addison-Wesley Publishing Company (1989)

16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In Proceedings of the1995 IEEE international conference on neural networks, Perth, Australia, vol. 6, 1942-1948 (1995)

17. Dorigo, M., Gambardella, L. M.: Ant colonies for the travelling salesman problem. Biosystems. 43, 73–81 (1997)

18. Bilchev, G., Parmee, I.C.: The ant colony metaphor for searching continuous design spaces. Lecture Notes in Computer Science. 993, 25-39 (1995)

19. Bilchev, G., Parmee, I.C.: Constrained optimisation with an ant colony search model. Procs. of the ACEDC. 145-151 (1996)

20. Wodrich, M., Bilchev, G.: Cooperative distributed search: The ants' way. Control and Cybernetics. 26(3), 413-441 (1997)

21. Mathur, M., Karale, S. B., Priye, S., Jayaraman, V. K., Kulkarni, B. D., Ant Colony Approach to Continuous Function Optimization. Industrial & Engineering Chemistry Research, 2000. 39(10): p. 3814-3822.

22. Ahmad, Z., Zhang, J.: Bayesian selective combination of multiple neural networks for improving long range predictions in nonlinear process modeling. Neural Computing & Applications. 14, 78-87 (2005)

23. Ahmad, Z., Zhang, J.: Combination of multiple neural networks using data fusion techniques for enhanced nonlinear process modeling. Computers & Chemical Engineering. 30, 295-308 (2006)

24. Efron, B.: The Jackknife, the Bootstrap and Other Resampling Plans. Society for Industrial and Applied Mathematics: Philadelphia, (1982).

25. Zhang, J.: A reliable neural network model based optimal control strategy for a batch polymerization reactor. Industrial & Engineering Chemistry Research. 43(4), 1030-1038 (2004)

26. Mukherjee, A., Zhang, J.: A reliable multi-objective control strategy for batch processes based on bootstrap aggregated neural network models. Journal of Process Control. 18, 720-734 (2008)

27. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research. 185(3), 1155-1173 (2008)