SESSION

SIMULATED ANNEALING, SWARM OPTIMIZATION ALGORITHMS AND APPLICATIONS + AGENT-BASED ALGORITHMS

Chair(s)

TBA

A Novel Autoregressive Model for System State Prediction

De Z. Li[1], Fathy Ismail[1], and Wilson Wang[2]
[1]Department of Mechanical & Mechatronics Engineering, University of Waterloo, Waterloo, Ontario, Canada
[2]Department of Mechanical Engineering, Lakehead University, Thunder Bay, Ontario, Canada

Abstract - *Autoregressive (AR) model is one of the commonly used predictors for system state forecasting. Several training methods have been used to optimize AR model parameters, such as least square estimate and maximum likelihood estimate; however, both of these techniques are sensitive to noisy samples and to outliers. To address these problems, a novel AR predictor, NAR, is proposed in this work to improve the prediction accuracy and reduce the effect of noise and outliers. In NAR the model parameters of AR are trained using an adaptive least square estimate (ALSE) method. The proposed ALSE is used to learn samples characteristics more effectively. In each training epoch, the ALSE can discern the samples associated with their fitting accuracy. The samples with larger errors will be assigned a larger penalty value in the cost function; however the penalties of difficult-to-predict samples will be reduced to improve the overall prediction accuracy. The effectiveness of the developed NAR predictor is demonstrated by simulation tests. Test results show that the proposed NAR predictor can capture system dynamics effectively and track system characteristics accurately.*

Keywords: Autoregressive model, time series forecasting, least square estimate.

1 Introduction

System state prediction is a process to extract features from available data that rule the trend of the states, and forecast future states based on the extracted features. It has many important real world applications, such as electric load forecasting [1,2], financial indicator prediction [3,4], and equipment health condition monitoring [5]. A reliable health condition monitoring system can provide future states of machinery, which can be used to prevent performance degradation, malfunction and even catastrophic failure. In addition, the predictor can estimate remaining useful life of a damaged machine, so as to adaptively schedule repair operations.

There are several commonly used prediction tools such as autoregressive (AR) models, autoregressive-moving-average (ARMA) models [6], neural networks (NNs) [7-9] and particle filtering [10]. The NNs can capture data features through a training process, so as to conduct time series prediction; however, they suffer from black-box modeling mechanism. AR is more compact than ARMA, and it does not have estimation errors that arise from the moving average part in ARMA.

A boosting technique is an ensemble learning algorithm, which combines weak learners to improve the training accuracy whereby each weak learner deals with one tweaked data property (e.g., the data distribution). Boosting techniques are mainly applied in pattern classification applications [11,12]. In time series forecasting, the boosting techniques have also been employed to improve prediction accuracy [13,14]. A boosting technique can also be used as a training method to update model parameters. A novel AR (NAR) predictor is proposed in this work for system state prediction. The NAR has the generic AR model structure; however, it uses an adaptive least square estimate (ALSE) method for model parameter training. Some samples are difficult to predict because they are dominated by noise or outliers. If more effects are put to correctly predict these difficult-to-predict samples, the prediction accuracy of the already accurately predicted samples will drop, and the training process is prone to the "overfitting" problem. The difficult-to-predict samples are detected and their penalties are reduced in the proposed NAR to improve prediction performance. Another merit of the proposed NAR is that it has no parameters to be tuned. The effectiveness of the proposed NAR predictor is verified here by simulations.

The remainder of this paper is organized as follows: Section 2 presents the theoretical foundation of the proposed NAR predictor. In Section III, the effectiveness of the proposed NAR predictor is examined by simulation tests. Some concluding remarks of the study are given in Section IV.

2 The proposed novel AR predictor

The NAR predictor formulates penalties of samples at each training epoch according to their fitting errors. By assigning penalties to samples, the model parameters can be updated adaptively with respect to different samples so as to improve prediction accuracy. Those difficult-to-learn samples may mislead the training process, so their penalties will be diminished in NAR. The development of the technique is detailed below.

2.1 The NAR model structure

Consider the training data sets $z(k)$; $k = 1, 2, \ldots, K$, where K is the number of samples in the training data set.

For *s*-step-ahead prediction, the training data set can be re-arranged to have the input vector $x(i) = [z(i), z(i + 1), \ldots, z(i + d - 1)]$, and the output $y(i) = z(i + d + s - 1)$; $i = 1, 2, \ldots, N$, where $N = (K - d - s + 1)$. d is the dimension of the input vector $x(i)$.

The AR model has the form of

$$p_t(i) = \theta_1 z(i-1) + \theta_2 z(i-2) + \ldots + \theta_{r-1} z(i-r+1) \quad (1)$$

where θ_i are linear parameters; $i = 1, 2, \ldots, r - 1$. Eq. (1) can also be written in the following matrix form,

$$Y = X\theta \quad (2)$$

where

$$X = \begin{bmatrix} z(1) & z(2) & \cdots & z(r-1) \\ z(2) & z(3) & \cdots & z(r) \\ \vdots & \vdots & \ddots & \vdots \\ z(N-r+s+1) & z(N-r+s+2) & \cdots & z(N+s-1) \end{bmatrix} \quad (3)$$

$$\theta = \begin{bmatrix} \theta(1) \\ \theta(2) \\ \vdots \\ \theta(r-1) \end{bmatrix} \quad (4)$$

$$Y = \begin{bmatrix} z(r+s) \\ z(r+s+1) \\ \vdots \\ z(N) \end{bmatrix} \quad (5)$$

2.2 Parameter estimation at each training epoch

The linear parameters increment θ_t of the NAR predictor at t^{th} training epoch can be derived using the weighted least square estimate, WLSE:

$$\theta_t = (X^T W X)^{-1} X^T W Y \quad (6)$$

The weight matrix W is represented by

$$W = \begin{bmatrix} L_t(1) & \cdots & 0 & 0 \\ \vdots & L_t(2) & \cdots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & L_t(N) \end{bmatrix} \quad (7)$$

where $L_t(i)$ represent the penalties of sample i at training epoch t.

2.3 Formulation of sample penalties

In the proposed NAR, the penalties of sample i at epoch 1 can be set as $L_1(i) = \frac{1}{N}$, where N is the number of samples.

Given the penalties L_t, the update of the penalties at step t+1 will be performed by

$$L_{t+1}(i) = \frac{L_t(i)\exp\left(-\beta_t\left[\left|y_d(i) - p_t(i)\right| - w_t(i)\right]\right)}{Z_t} \quad (8)$$

$$= \frac{\exp\left(-\sum_{t=1}^{T}\left(\beta_t\left[\left|y_d(i) - p_t(i)\right| - w_t(i)\right]\right)\right)}{N\prod_{t=1}^{T} Z_t} \quad (9)$$

where y_d are the desired values; p_t are the predicted values using Eq. (1), in which θ_t derived in Eq. (6); β_t is the learning rate of the t^{th} parameter update epoch, which will be discussed in the following subsection; $w_t(i)$ is the weight regulator to mitigate the penalties of difficult-to-predict samples; $Z_t = \sum_{i=1}^{N} L_t(i)\exp\left(-\beta_t\left[\left|y_d(i) - p_t(i)\right| - w_t(i)\right]\right)$ is a normalization factor.

The NAR model parameters at step R are obtained from

$$\Theta_R = \sum_{t=1}^{R} \beta_t' \theta_t \quad (10)$$

where θ_t is the linear parameter increment. The predicted values of NAR at step R are formulated as

$$\begin{aligned} P_K &= X\Theta_K \\ &= X\sum_{t=1}^{K} \beta_t' \theta_t \\ &= \sum_{t=1}^{K} \beta_t' p_t \end{aligned} \quad (11)$$

where $\beta_t' = \frac{\beta_t}{\sum_{t=1}^{K}\beta_t}$ are the normalized learning rates.

2.4 Calculation of learning rate β_t

Let $u_t = \left|y_d(i) - p_t(i)\right| - w_t(i)$ with $u_t \in [0, M_t]$, where M_t is the maximum value of $\left|y_d(i) - p_t(i)\right|$, and $\lambda_t = \sum_{i=1}^{N} L_t(i)u_t(i)$. The upper bound of Z_t can be derived as

$$\begin{aligned} Z_t &= \sum_{i=1}^{N} L_t(i)\exp\left(-\beta_t u_t(i)\right) \\ &\le \sum_{i=1}^{N} L_t(i)\left(\frac{M_t + u_t(i)}{2M_t}\exp(-\beta_t M_t) + \frac{M_t - u_t(i)}{2M_t}\exp(\beta_t M_t)\right) \\ &= U \end{aligned} \quad (12)$$

Let $\frac{\partial U}{\partial \beta_t} = 0$, and the learning rate at step t will be,

$$\beta_t = \frac{1}{2M_t}\ln\left(\frac{M_t + \lambda_t}{M_t - \lambda_t}\right) \quad (13)$$

Substituting Eq. (13) into Eq. (12), the minimum upper bound of Z_t can be derived as

$$Z_t \le \sqrt{1-\left(\lambda_t / M_t\right)^2} \tag{14}$$

Since $M_t \ge \lambda_t \ge 0$, then $\sqrt{1-\left(\lambda_t / M_t\right)^2} \le 1$.

2.5 Upper bound of the mean absolute error

Since $\beta_t \ge 0$, $\sum_{t=1}^{T} \beta_t' = 1$, $\beta_t' \ge 0$, $\sum_{i=1}^{N} L_{T+1}(i) = 1$, $\lambda_t \ge 0$, and the final ensemble prediction $P(i) = \sum_{t=1}^{T} \beta_t' p_t(i)$, the mean absolute error (MAE) of the training data can be determined by

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \left| y_d(i) - P(i) \right| \tag{15}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left| y_d(i) - \sum_{t=1}^{T} \beta_t' p_t(i) \right| \tag{16}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left| \sum_{t=1}^{T} \beta_t' \left(y_d(i) - p_t(i) \right) \right| \tag{17}$$

$$\le \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \left| \beta_t' \left(y_d(i) - p_t(i) \right) \right| \tag{18}$$

$$\le \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \left| \beta_t \left(y_d(i) - p_t(i) \right) \right| \tag{19}$$

$$\le \frac{1}{N}\sum_{i=1}^{N} \exp\left(-\sum_{t=1}^{T} \left| \beta_t \left(y_d(i) - p_t(i) \right) \right| \right) \tag{20}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \exp\left(-\sum_{t=1}^{T} \beta_t \left| y_d(i) - p_t(i) \right| \right) \tag{21}$$

$$\le \frac{1}{N}\sum_{i=1}^{N} \exp\left(-\sum_{t=1}^{T} \beta_t \left(\left| y_d(i) - p_t(i) \right| - w_t(i) \right) \right) \tag{22}$$

$$= \sum_{i=1}^{N} L_{T+1}(i) \prod_{t=1}^{T} Z_t \tag{23}$$

$$= \prod_{t=1}^{T} Z_t \tag{24}$$

To satisfy inequality (19), $\beta_t' \le \beta_t$ or $\sum_{t=1}^{T} \beta_t \ge 1$ should be satisfied. If T is sufficiently large, then $\sum_{t=1}^{T} \beta_t \ge 1$. The inequality in Eq. (20) can be satisfied as $\sum_{t=1}^{T} \beta_t \left| y_d(i) - p_t(i) \right| \in [0, 0.567]$. This condition can be achieved by properly scaling the training data. To satisfy inequality (22), $w_t(i) \ge 0$, which will be shown in the following subsection. Therefore, the minimization of the prediction MAE is equivalent to minimizing $\prod_{t=1}^{T} Z_t$, or minimizing Z_t at each step t.

Given $\sqrt{1-\left(\lambda_t / M_t\right)^2} \le 1$, the upper bound of MAE of the training data can be derived as

$$MAE \le \prod_{t=1}^{T} Z_t \le \prod_{t=1}^{T} \sqrt{1-\left(\lambda_t / M_t\right)^2} \tag{25}$$

Therefore, as more training epochs are undertaken, the upper bound of the training MAE decreases. Consequently, the prediction accuracy can be improved by using the NAR predictor.

2.6 Weight regulator

Some samples may be noisy samples or outliers, which may misguide the training process and degrade the prediction accuracy. The following degree of difficulties can be used to detect these irregular samples:

$$\pi_t(i) = \sum_{t=1}^{t} \beta_t' \left| y_d(i) - p_t(i) \right| \tag{26}$$

By summing up the previous weighted prediction errors, the samples can be ranked according to their difficulty levels in prediction. After scaling, the weight regulator can be computed from

$$w_t(i) = \pi_t(i) \frac{M_t}{\eta_t} \tag{27}$$

where $\eta_t = \sup(\pi_t(i))$. It can be shown that $w_t(i) \ge 0$.

By applying the weight regulator to the sample penalty update as shown in Eq. (8), the difficult-to-predict samples are identified and consequently their penalties will be reduced so as to further improve prediction accuracy.

2.7 Implementation of the NAR predictor

The proposed NAR predictor is implemented using the following steps:

1)Normalize the data over a proper range (e.g. [0, 1]), so that constraints in Subsection 2.5 can be satisfied.

2)Initialize the penalties of the training data set $L_1(i) = \frac{1}{N}$; i = 1, 2, …, N.

3)Derive the parameter increment θ_t using Eq. (6) with the penalties L_t.

4)Compute the sum of weighted absolute error $\lambda_t = \sum_{i=1}^{N} L_t(i) \left(\left| y_d(i) - p_t(i) \right| - w_t(i) \right)$, where y_d are the desired values, and p_t are the predicted values using in Eq. (1) in which θ_t are derived from Eq. (6).

5)Calculate the learning rate of the t^{th} training epoch, $\beta_t = \frac{1}{2M_t} \ln\left(\frac{M_t + \lambda_t}{M_t - \lambda_t} \right)$.

6)Update the penalties of the training samples, $L_{t+1}(i) = \frac{L_t(i) \exp\left(-\beta_t \left(\left| y_d(i) - p_t(i) \right| - w_t(i) \right) \right)}{Z_t}$, where

$Z_t = \sum_{i=1}^{N} L_t(i) \exp\left(-\beta_t \left(\left|y_d(i) - p_t(i)\right| - w_t(i)\right)\right)$ is a normalization factor.

7)Repeat steps 3) to 6) at $t = 1, 2, \ldots, T$.

8)Formulate the final NAR model parameters $\Theta_T = \sum_{t=1}^{T} \beta_t' \theta_t$, where $\beta_t' = \frac{\beta_t}{\sum_{t=1}^{T} \beta_t}$ are normalized learning rates.

3 Performance evaluation

To verify the effectiveness of the proposed NAR predictor, simulation tests are conducted to evaluate its prediction accuracy. The AR model with the same structure as NAR, but trained by Kalman-filter-based maximum likelihood estimate (MLE) [15], AR-MLE, is used for comparison. To satisfy constraints as stated in Subsection 2.5, the data sets used in this section are normalized over the range of [0, 1]. Test results, however, are shown in their original scales.

The Mackey-Glass data set [16-18] is commonly used in the forecasting research field to compare the performance of predictors, due to its specific properties such as chaotic, non-periodic and non-convergence, it is given by:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (28)$$

In this simulation test, a data set is obtained from Eq. (28) with the initial conditions of τ =30, x(0) = 1.2, dt = 1 and $x(t) = 0$ for $t < 0$. 500 samples are selected for training, and 50 samples for testing. One-step-ahead forecasting is conducted in the Mackey-Glass data prediction tests. To test the noise tolerance of the two predictors, noisy samples are intentionally added to the Mackey Glass training data; the red circled samples at time step 20, 100, 135, 215, 345, are shown in Fig. 1.

The training MAE convergences of NAR(3), NAR(6), and NAR(9) are shown in Fig. 2. It is seen from Fig. 2 that the training MAEs decrease as more training epochs are used, which agrees with Eq. (25). Fig. 3(a) demonstrates the training data fitting, and Fig. 3(b) shows the prediction performance. From Fig. 3(b), it is seen that the NAR predictor outperforms AR-MLE, because NAR predictor has the provision to detect and process noisy samples to alleviate the noisy sample misleading effect.

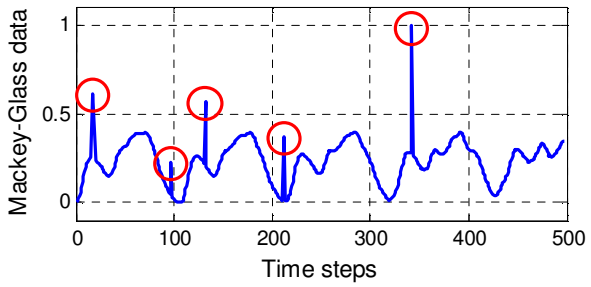


Fig. 1. The Mackey Glass training data with artificial noisy samples (red circled samples).

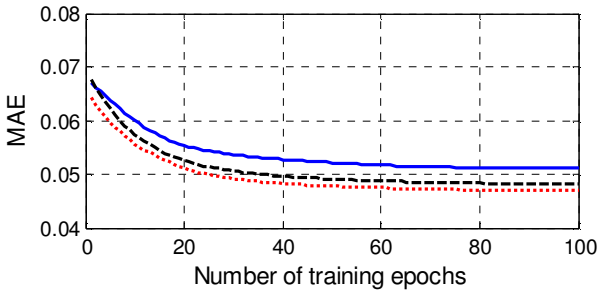


Fig. 2. The training MAE of NAR(3) (blue solid line), NAR(6) (black dashed line), and NAR(9) (red dotted line), corresponding to different number of training epochs.

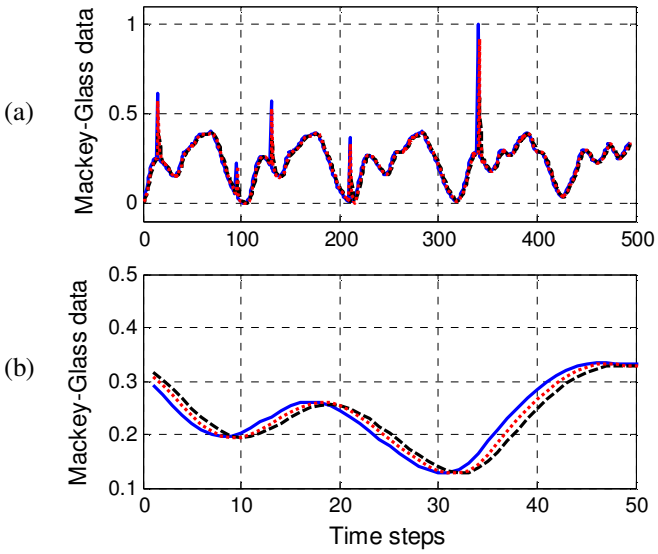


Fig. 3. The performance of (a) training data fitting, and (b) test data prediction. The blue solid line is the real data; the black dashed line represents AR(5)-MLE; the red dotted line represents NAR(5) with 100 training epochs.

The training MAEs and test MAEs of the two predictors with respect to model orders of 3, 6, and 9 are summarized in Table 1, while their corresponding training RMSEs and test RMSEs are listed in Table 2. 100 training epochs are used in this NAR. It is seen from Table 1 and Table 2 that the training errors of NAR decrease as the model order increases, because a larger model order indicates more information is fed to the predictor for processing, and the predictions become more accurate.

From Table 2, the training RMSEs of the proposed NAR are larger than those of AR-MLE, because NAR reduces the penalties of noisy samples to improve the generalization capability of the predictor. Consequently, training errors at noisy samples are large, which leads to larger NAR training RMSE.

In terms of test MAEs and test RMSEs, Tables 1 and 2 show that the NAR predictor outperforms AR-MLE at all three model orders. This better performance of NAR is attributed to its improved training technique and effective noise tolerance mechanism.

Table 1. MAEs of AR-MLE and NAR Predictors in Terms of Mackey-Glass Data.

Predictor	AR-MLE		NAR	
Model order	Training MAE	Test MAE	Training MAE	Test MAE
3	0.0670	0.0449	0.0511	0.0293
6	0.0657	0.0437	0.0482	0.0248
9	0.0632	0.0386	0.0469	0.0235

Table 2. RMSEs of AR-MLE and NAR Predictors in Terms of Mackey-Glass Data.

Predictor	AR-MLE		NAR	
Model order	Training RMSE	Test RMSE	Training RMSE	Test RMSE
3	0.1505	0.0542	0.1656	0.0352
6	0.1497	0.0534	0.1642	0.0304
9	0.1478	0.0494	0.1622	0.0297

4 Conclusions

A novel AR predictor, NAR, has been developed in this work for system state prediction. The NAR can gradually learn the training data characteristics with more training epochs, and accurately forecast the future states of a dynamic system. The noisy samples are addressed using a weight regulator to mitigate their detrimental effect on prediction. The effectiveness of the proposed NAR predictor is verified using a Mackey-Glass data set. Test results have shown that the NAR predictor can effectively capture the dynamic behavior of a test data set and track its future states accurately.

5 References

[1] H. Quan, D. Srinivasan, and A. Khosravi, "Short-term load and wind power forecasting using neural network-based prediction intervals," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 303-315, 2013.

[2] Y. Goude, R. Nedellec, and N. Kong, "Local short and middle term electricity load forecasting with semi-parametric additive models," *IEEE Transactions on Smart Grid*, vol. 5. no. 1, pp. 440-446, 2014.

[3] K. K. Ang, and C. Quek, "Stock trading using RSPOP: a novel rough set-based neuro-fuzzy approach," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1301-1315, 2006.

[4] L. Y, H. Chen, S. Wang, and K. K. Lai, "Evolving least squares support vector machines for stock market trend mining," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 87-102, 2009.

[5] W. Wang, "An enhanced diagnostic system for gear system monitoring," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 38, pp. 102-112, 2008.

[6] P. J. Brockwell and R. A. Davis, *Time Series: Theory and Methods*, New York: Springer, 2009.

[7] D. Li, W. Wang, and F. Ismail, "Enhanced fuzzy-filterd neural networks for material fatigue prognosis," *Applied Soft Computing*, vol. 13, no. 1, pp. 283-291, 2013.

[8] D. Li, W. Wang, and F. Ismail, "Fuzzy neural network technique for system state forecasting," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1484-1494, 2013.

[9] D. Li, W. Wang, and F. Ismail, "A fuzzy-filtered grey network technique for system state forecasting," *Soft Computing*, 2014, online available: http://link.springer.com/article/10.1007%2Fs00500-014-1281-1.

[10] D. Z. Li, W. Wang and F. Ismail, "A mutated particle filter technique for system state estimation and battery life prediction," *IEEE Transactions on Instrumentation and Measurement*, 2014. Online available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6740856&sortType%3Dasc_p_Sequence%26filter%3DAND(p_IS_Number%3A4407674).

[11] J. Cao, S. Kwong, R. Wang, "A noise-detection based AdaBoost algorithm for mislabeled data," *Pattern Recognition*, vol. 45, pp. 4451-4465, 2012.

[12] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine Learning*, vol. 37, no. 3, pp. 297-336, 1999.

[13] H. Drucker, "Improving regressor using boosting techniques," *Proceedings of the 14th International Conferences on Machnie Learning*, pp. 107-115, 1997.

[14] D. P. Solomatine and D. L. Shrestha, "AdaBoost.RT: a boosting algorithm for regression problems," *Proceedings of the International Joint Conference on Neural Networks*," pp. 1163-1168, 2004.

[15] C. Hevia, "Maximum likelihood estimation of an ARMA(p,q) model," *The World Bank, DECRG*, 2008.

[16] J. D. Farmer "Chaotic attractors of an infinite-dimensional dynamical system," *Physica D: Nonlinear Phenomena*, vol. 4, pp. 366-393, 1982.

[17] D. Z. Li and W. Wang, "An enhanced GA technique for system training and prognostics," *Advances in Engineering Software*, vol. 42, no. 7, pp. 452-462, 2011.

[18] W. Wang, D. Z. Li and J. Vrbanek, "An evolving neuro-fuzzy technique for system state forecasting," *Neurocomputing*, vol. 87, pp. 111-119.

Optimized Path Planning for a Mobile Real Time Location System (mRTLS) in an Emergency Department

S. Singh, M.R. Friesen, and R.D. McLeod
Electrical and Computer Engineering, University of Manitoba, Winnipeg, Manitoba, Canada
robert.mcleod@umanitoba.ca

Abstract - *Data collection methods used for localization in healthcare facilities involving human data collection often tend to generate inconsistent and ambiguous data. To address these issues, an automated ceiling mounted mobile Real-Time Location System (mRTLS) is proposed. The system utilizes mobile Radio Frequency IDentification (RFID) and is evaluated using agent based modeling. A prototypical agent model for an emergency facility is created using the AnyLogic simulation software. The model developed is based on the real hospital data to approximate patient flow in a real system. The emergency department model is outfitted with mRTLS using mobile ceiling mounted RFID readers in an emergency facility layout. Path planning and resource provisioning of mobile readers is accomplished using both multi objective genetic and simulated annealing algorithms. The genetic algorithm provides an initial placement of static readers extracted from areas of more frequently read RFID tags. The genetic algorithm solution initializes the input parameters for the simulated annealing algorithm. The simulated annealing algorithm provides the final optimized path for the deployment of mobile readers. The results generated by the model are analyzed in the simulation of the emergency facility augmented with mobile RFID readers. The applicability of the proposed model in a complex system like a healthcare facility is demonstrated through simulation.*

Keywords: RFID, Agent based modeling, genetic algorithm, simulated annealing

1 Introduction

The collection of Emergency Department (ED) data is a crucial component of patient care. The decisions based on data such as priority setting, allocation and leveraging of resources, comprehensive planning, service delivery, and performance evaluation depends on qualitative and quantitative data collection [1].

The recorded data include medical records, vital events registration, and surveillance, recording of treatments in progress, overall length of stay and discharge or admittance disposition. Data manually collected and reported related to patient care and healthcare facility activities have been found to contain errors and discrepancies. These error prone data collection techniques, and issues therein, are amenable to improvement as they can affect patient care [2],[3]. Inaccurate data can have adverse effects on patient outcomes and the efficiency of resource allocation [4]. The proposed mRTLS addresses one aspect of the overall data collection process that being patient and resource localization and tracking.

Hirshon et al. [5] analyzed the data collection methods used in healthcare facilities such as EDs. One school of thought is to increase the automation of current data collection systems. Automated data collection methods are expected to improve accuracy and reliability of the data, thus improving analysis and aiding in efficient policy making, thereby improving ED throughput and patient care outcomes.

Many scientists and researchers such as Nelson et al. [6] have explored various asset tracking means to improve throughput and resource utilization. Among the wireless technologies, RFID and RTLS [7] have been suggested as leading contenders to aid in automated data collection. Ferrer et al. [8] discussed the use of RFID tracking technology in monitoring various environments. He et al. [9] demonstrated the provisioning of real-time data collection with RFID aids in logistics and resource tracking. RFID data collected in a Texas hospital, is used to generate data for infection control, automated discharge and improve workflow, as reported by Baum [10]. This provides an indication that high quality healthcare data can be obtained using wireless automated data collection techniques that are more precise, reliable and time-saving for all involved. While commercial tracking systems are available and evolving, they are expensive solutions each with their own limitations.

The mobile RFID reader system proposed here is also not without difficulties of its own. The deployment of mobile RFID readers and the network planning to achieve two objectives is a non-linear multi objective optimization problem. Guan et al. [11] demonstrated that deployment of RFID in large scale environments requires solving a complex network planning problem. Bandyopadhyay et al [12] have developed a simulated annealing-based multi objective optimization algorithm (AMOSA) to find a solution to this problem. We also apply these methods while addressing the mobile RFID reader network planning problem.

Simulation and modeling is generally used for visualizing and assessing process flow of complex systems, and has been used in this manner by other researchers. Miller et al. [13] have used simulation experiments to demonstrate improvements to ED throughput. This supports the conjecture that simulation tools can accurately represent complex systems like EDs.

In our case, simulation and agent based modeling is useful as an inexpensive validation tool for the optimizations. Using AnyLogic simulation tools, an Agent Based Model (ABM) of an ED was developed. Using patient flow data acquired from Winnipeg Regional Health Authority (WRHA), the solution obtained by our algorithms is partially validated, without the need for expensive hardware installations.

2 Related Work

Many researchers believe RFID applications in medical institutions can help reduce medical errors and reduce labor costs [14], [15]. As such, RFID systems may also have the potential to improve the efficiency of medical services and improve patient outcomes.

Laskowski et al. [16] designed an ABM for patient tracking and assessment of error/uncertainty of low cost, fixed RFID in an ED. The limitations associated with the fixed RFID readers in the study conducted by Laskowski et al. relates to the degree of uncertainty or error that depends on the number of readers provisioned in the ED. An issue addressed here is the high cost and maintenance associated with a large number of static RFID readers deployed in the ED. Anusha et al. [17] and Xie et al. [18] are of the view that expensive deployment of fixed readers gives broad coverage, however, complete coverage might not be essential in real world, practical applications. Xie et al. [18] support the idea that efficient use of mobile readers may cover the area sufficiently, with the advantage of reduced cost of deployment due to fewer readers being required. Similarly, in order to address the issue of increasing cost and reducing patient and asset localization uncertainty in an ED, a mobile RFID/RTLS system is proposed. The proposed mobile RTLS aims at minimizing the cost by reducing the number of readers while minimizing the error in tracking the RFID tags in an ED. A significant difference here from most mobile RFID systems is that the readers in the ED would be ceiling mounted as a means of keeping the system as noninvasive as possible to the normal operations of a busy ED.

The application of evolutionary algorithms to address multi objectives and constraints optimization issues has shown promising results in research studies. The studies conducted by Guan et al. [11] and Seo et al. [19] support solving RFID network planning optimization problems by evolutionary algorithms. Guan et al. [11] applied a genetic approach to overcome complex problems such as interference of multiple readers, undesirable mutual coverage and variability in propagation environments. Guan et al. [11] are of the view that uplink signals from tag to readers should be taken into account while solving complex RFID network optimization problems. Seo et al. [19] designed a genetic algorithm based resource allocation for an RFID system to resolve the reader to reader tag collisions and interference complications. The system designed by Seo et al. [19] optimizes RFID resource allocation and the related tag recognition issues. Weijie et al. [20] analyzed RFID network features and multi objective optimization built on genetic programming. These multi objective genetic programs or algorithms typically generate the best fit layout/deployment of static readers.

3 Methodology

3.1 Agent Based Modeling

A preliminary 3-D ABM was integrated with a tracking system based upon RFID. This effort led to the idea to improve the effectiveness of the system using mobile RFID readers as an alternative to the more traditional static reader scenario. The objective of mobile RFID/mRTLS is to minimize number of readers and minimize the tracking error incurred simultaneously by finding the best path to layout the tracks or rails that the mobile readers would traverse. Actual cost of mobile RFID readers, their installation and maintenance cost is not estimated, though it is assumed that fewer readers will be significantly less costly to install. In addition, a mobile reader system is more easily upgraded as an existing track infrastructure is a more permanent fixture requiring only the upgrade of the readers themselves.

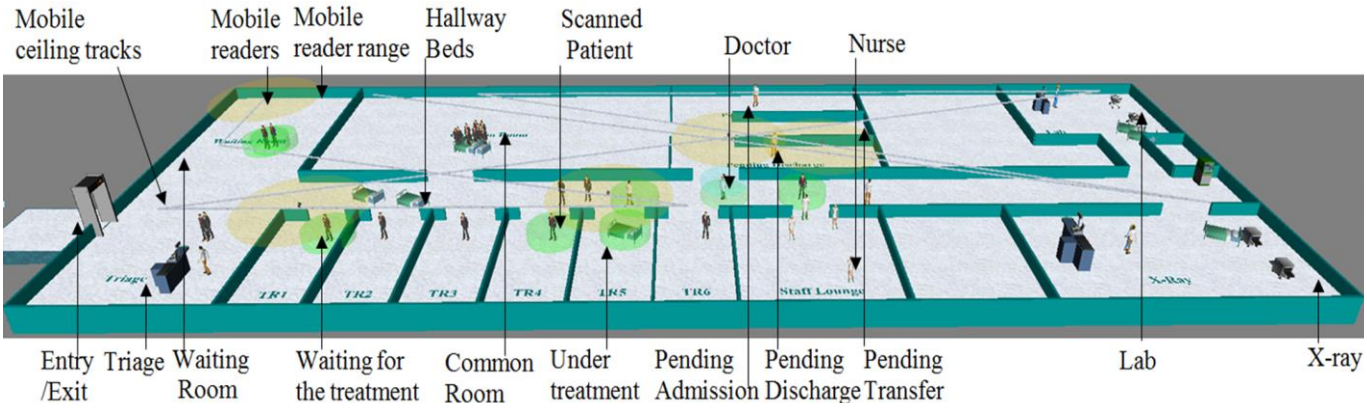


Figure 1. A screen shot of augmented RFID RTLS ABM Simulation.

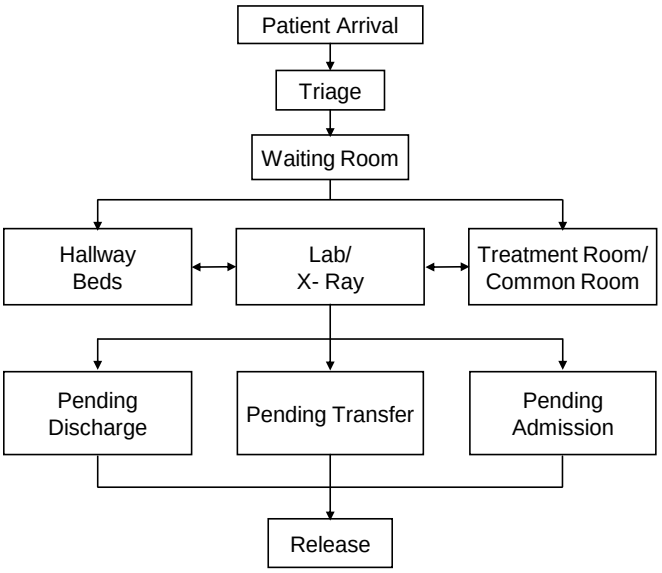


Figure 2. The flow of patients in ED simulation model.

Figure 1 provides a visual of patient movement during the course of treatment based on the real data from an ED. The data is essentially a record of start and stop of events that a patient would undergo. In addition, the data provides for an accurate model in terms of arrival rates. The flow of patients and states is approximately as shown in figure 2. The patient initially arrives in the Triage room and gets tracked by the moving mobile reader running on the ceiling tracks. The identity and timings of each individual tag is maintained by all the mobile readers in a central repository.

A standard layout is assumed for the facility, which can be modified to be applicable to any real-world ED floor plan. Installation of fixed RFID readers to obtain adequate coverage is costly primarily due to the number of readers that would be required and the space and layout constraints in the healthcare facilities. Mobile readers on the other hand act as patrolling devices to track resources (or patients) instead of waiting for the resource to come within the range of fixed readers. A mobile system would also be costly to install initially but could be reused as reader technology advances. It is conceded that the accuracy of RFID tracking systems is better in the case of sufficiently large number of fixed readers. Effectively, the level of accuracy is traded for a reduced number of readers in the case of the mobile RFID reader system. The conjecture is that as the mobile readers may still provide sufficient coverage at a significantly reduced cost.

The optimization of mobile readers is initialized by positioning a large number of static readers in the facility to cover the entire area as shown in Figure 3. This instance allows a base line for reader error to be estimated against the movement of patients and resources extracted from the ABM. Errors are essentially a non-tracking event. That is, a patient is in a known location but not read as they may be out of reader range.

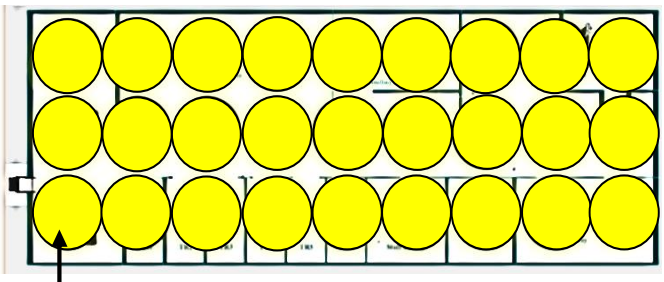


Figure 3. Initial placement of 27 static readers.

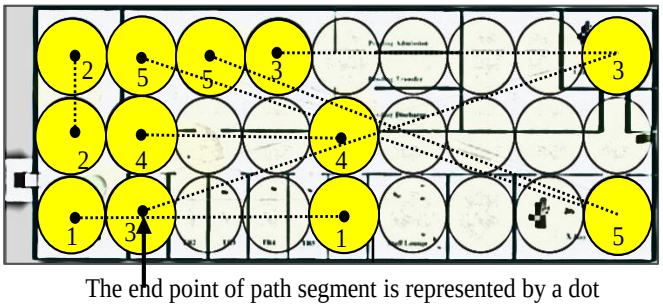


Figure 4. The parameter optimization of a static RFID reader model using a Genetic Algorithm that illustrates the high RFID read areas.

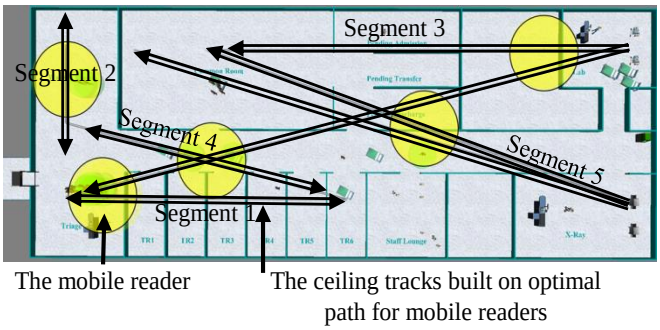


Figure 5. The dynamic parameter optimization of the mobile RFID reader model using Simulated Annealing that generates optimized paths which the mobile readers traverse.

Figures 4 and 5 illustrate the progression from the genetic algorithm static reader locations to the simulated annealing mobile reader configuration.

3.2 Multi Objective Algorithms

A multi objective genetic algorithm (GA) is designed using static parameter variations to find the high traffic areas of the agents and asset flows. Through this algorithm, an optimized layout of static readers is obtained. The optimized solution, in the form of a reader location string from the GA, shows areas of frequent traffic within the facility (figure 4). Based on the tracker or reader string (genetic sequence), the locations of most active fixed readers is determined. These

locations will be used as endpoints in the path segments followed by mobile readers.

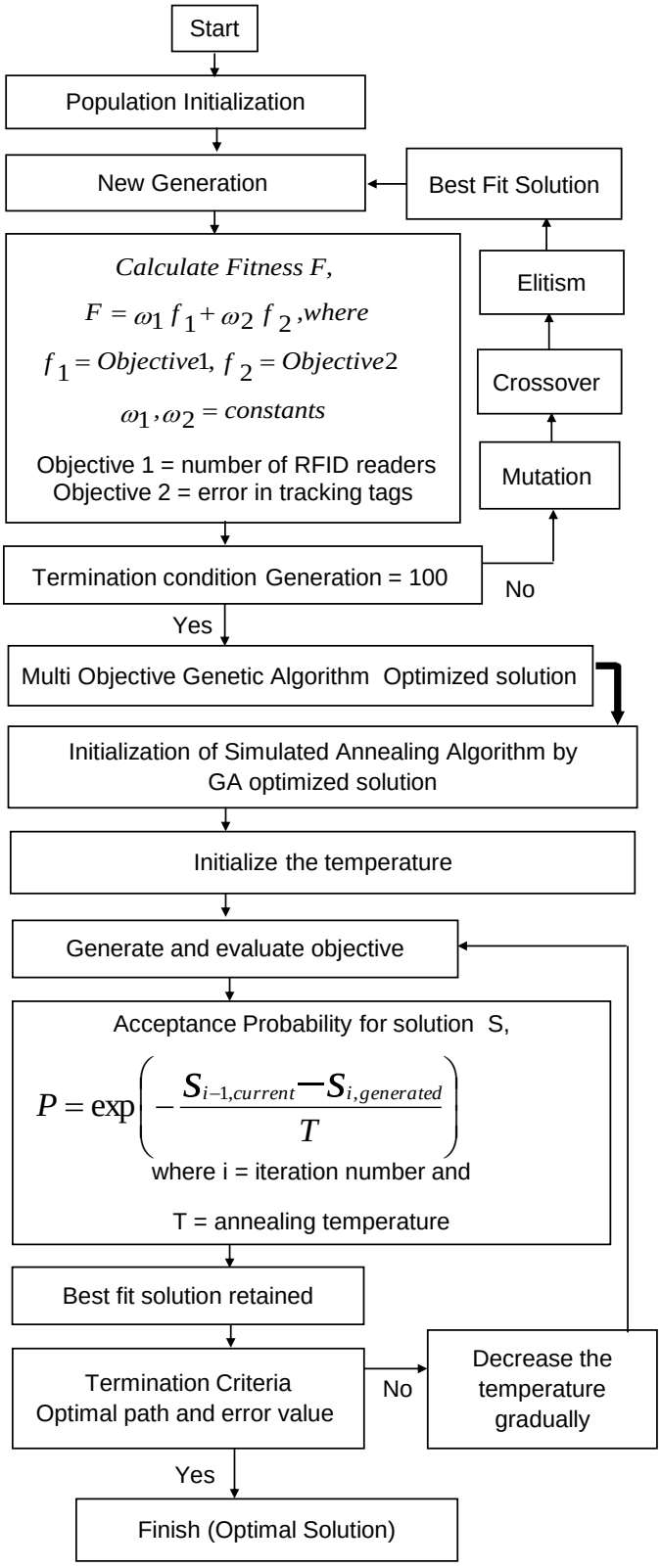


Figure 6. The optimization process (GA followed by SA).

In order to find best path covering among all the segments, a multi objective simulated annealing (SA) algorithm is used. The optimal static solution from the multi objective genetic algorithm is used as an initializing parameter to the simulated annealing algorithm that is implemented as a dynamic parameter variation experiment. The multi objective SA algorithm produces an optimized path for all the readers in the layout. The improved solution is generated that further optimizes cost by reducing the number of readers while attempting to minimal tracking errors. The flow of the multi objective optimization using GA and SA algorithms is demonstrated in Figure 6.

4 Experiments and Results

The two multi objective optimization algorithms are integrated within the AnyLogic simulation software to experimentally validate the proposed mRTLS reader system. The parameter variation experiments on the models enables "what-if " scenario verification and visual observations. The multi objective evolutionary algorithms are implemented in Java.

The optimization algorithms allow customization of their parameters. Population size, number of readers in an individual solution, rates and types of crossover and mutation operators are modifiable. The selectivity of the best solution, also known as elitism, is customizable. The implemented RFID agent model with mobile readers provides an option to change the reader range, velocity of the mobile RFID and number of readers.

The hospital data imported in the designed model can be changed and the simulated data can be imported for the purpose of analysis so the output of a large number of runs of the simulation can be statistically analyzed.

As mentioned, the best solution generated by multi objective GA is used as input for the multi objective SA algorithm to further improve optimization of the required number of readers and the best path that ensures fewer errors.

The experimental results demonstrate a decrease in tag tracking error with a concomitant reduction in the number of mobile RFID readers when compared with a greater number of static readers, even with those readers placed at high traffic locations. This is attainable by the designed error model. The readers in a high traffic area can accurately track an individual or an asset but loose the track of it as soon as it is out of range. In the error model developed, the reader that is moving effectively extends their range thereby reducing the tracking error. This is believed to be primarily due to the fixed trackers only reading tags in their proximity, while a mobile reader is able to get a better estimate of tag movement largely attributable to the time constants associated with relatively slow patient movement in an ED. This can be attributed to read hotspots being found in waiting rooms and hallways. Since the mobile trackers not only cover the hot spot locations, but also the territory between these locations, more information in regards to origin, destination and movement patterns of tags can be discerned.

Another influence of this perhaps unexpected result is the way in which error is defined within the system. Tags essentially operate as state machines, operating in either a non-error state or an error state, depending upon when the tag was last discovered by the tracking system and its movements since that time. A tag is put into the error state upon arrival at a new destination, and will remain in this state until seen by a reader. Once discovered, it will transition to the non-error state, and remain that way until moving to new destination.

Using this error model puts static readers at a disadvantage, as discussed above, due to the location of the hotspots. But it is believed that this error model more accurately reflects the ability of generating valid, useful data than others which were tested.

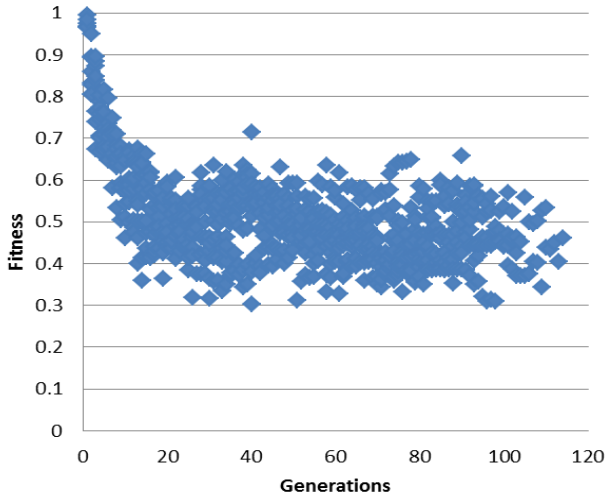


Figure 7. GA Fitness evolution over 120 generations.

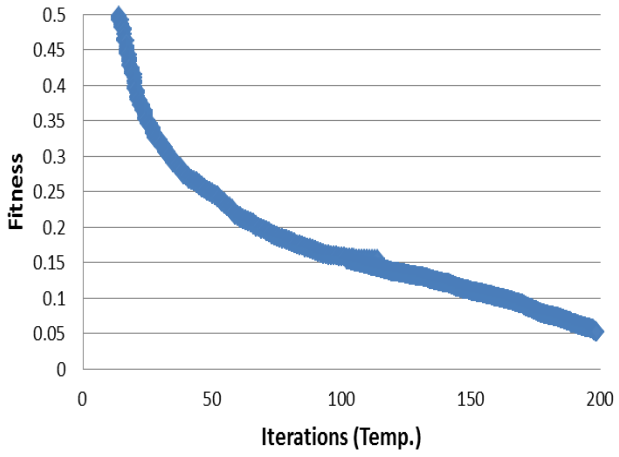


Figure 8. SA Fitness over 200 Iterations.

Figures 7 and 8 illustrate the fitness evolution over time as the GA and SA algorithms attempt to optimize the number of mobile readers and their trajectories. Table 1 provides an overview or summary of the optimized mobile mRTLS system. The fitness is directly related to the read error and number of readers.

Time (in model time units)	**Multi Objective Optimization**			
	Algorithm	***Number of Readers***	***Error*** (norm.)	***Fitness***
-	No Optimization	27	7.8	0.95
1000 (120 Gens.)	Genetic Algorithm	12	56000	0.25
1000 (200 Iters.)	Simulated Annealing Algorithm	5	6000	0.05

Table 1: Optimization of two objectives for the best mobile RFID solution.

5 Conclusion

A novel RFID RTLS is presented that uses mobile readers and delivers an optimized path for actual deployment of a minimal number of mobile readers. The RFID augmented ED model and the two proposed multi objective algorithms are used for developing path planning strategies. The two objectives achieved for the proposed agent based model are to minimize the number of RFID readers (i.e., minimize the cost) and to maximize the coverage.

The designed agent based model verifies the optimal path planning of mobile and fixed RFID surveillance system while achieving both objectives of minimizing cost and errors. The proposed model is optimized using GA and SA algorithms. The algorithms effectively optimize the RFID mobile reader network. The reduction in number of readers is from 27 in the full static reader deployment case, which is brought down to 10 static readers using the GA and further reduced to 5 mobile readers on fixed tracks or segments. It is concluded from the parametric variation experiments that the optimized solution evidently minimizes the cost by using fewer readers to achieve sufficient coverage determined based on maintaining a minimum error rate comparable to a greater number of static readers. There are obviously unavoidable tracking errors in any RFID tracking system but an mRTLS system based on mobile readers may offer a cost-effective alternative with acceptable error rates.

Future work includes deploying and testing of the proposed multi objective algorithms for mobile RFID readers navigating on the ceiling mounted tracks in a controlled or constrained or legalized reader track environment.

6 Acknowledgement

We are grateful to Trevor Strome of the Winnipeg Regional Health Authority for providing us with the data.

7 References

[1] J. Shea and J. J. Santos, "Community Needs Assessment and Data-Supported Decision Making: Keys to Building Responsive and Effective Health Centers," *Natl. Assoc. Community Heal. Centers*, no. November, 2012.

[2] C. Tsang, W. Palmer, A. Bottle, A. Majeed, and P. Aylin, "A review of patient safety measures based on routinely collected hospital data.," *Am. J. Med. Qual.*, vol. 27, no. 2, pp. 154–69, 2012.

[3] A. Nixon, "Errors in default settings of electronic medical record systems raise risks for patients," 2013. [Online]. Available: http://triblive.com/business/headlines/4654582-74/errors-patient-patients. [Accessed: 10-Mar-2014].

[4] E. J. Thomas and L. A. Petersen, "Measuring errors and adverse events in health care," *J. Gen. Intern. Med.*, vol. 18, no. 1, pp. 61–67, Jan. 2003.

[5] J. M. Hirshon, M. Warner, C. B. Irvin, R. W. Niska, D. A. Andersen, G. S. Smith, and L. F. McCaig, "Research using emergency department-related data sets: current status and future directions.," *Acad. Emerg. Med.*, vol. 16, no. 11, pp. 1103–9, Nov. 2009.

[6] J. Nelson, J. Connell, C. Isci, and J. Lenchner, "Data center asset tracking using a mobile robot," *SIGMETRICS Perform. Eval. Rev.*, pp. 339–340, 2013.

[7] M. Miliard, "RFID & RTLS can save lives | Healthcare IT News," 2013. [Online]. Available: http://www.healthcareitnews.com/news/rfid-rtls-can-save-lives. [Accessed: 10-Mar-2014].

[8] G. Ferrer, N. Dew, and U. Apte, "When is RFID right for your service?," *Int. J. Prod. Econ.*, vol. 124, no. 2, pp. 414–425, Apr. 2010.

[9] M. He, J. Morris, and T. Qin, "Quantifying the value of RFID in air cargo handling process: A simulation approach," *Proc. 2010 Winter Simul. Conf.*, pp. 1882–1889, Dec. 2010.

[10] S. Baum, "The future of healthcare? Texas hospital uses big data from RFID tags," 2013. [Online]. Available: http://medcitynews.com/2013/12/rfid-big-data-intersect-texas-hospital-pointing-future-healthcare/. [Accessed: 10-Mar-2014].

[11] Q. Guan, Y. Liu, Y. Yang, and W. Yu, "Genetic approach for network planning in the RFID systems," *Intell. Syst. Des. Appl. 2006. ISDA '06. Sixth Int. Conf.*, pp. 567–572, 2006.

[12] S. Bandyopadhyay, S. Member, S. Saha, S. Member, U. Maulik, and K. Deb, "A Simulated Annealing-Based Multiobjective Optimization Algorithm : AMOSA," *Evol. Comput. IEEE Trans.*, vol. 12, no. 3, pp. 269–283, 2008.

[13] M. J. Miller, D. M. Ferrin, M. Ashby, E. Highland, and K. P. White, "Using RFID Technologies to Capture Simulation Data in a Hospital Emergency Department," *Proceedings of the 2006 Winter Simulation Conference L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.*, pp. 1365–1371, 2006.

[14] E. McGrady, S. Conger, S. Blanke, and B. J. L. Landry, "Emerging technologies in healthcare: navigating risks, evaluating rewards.," *J. Healthc. Manag.*, vol. 55, no. 5, pp. 353–64; discussion 364–5.

[15] J. Cho, S. Cobbs, E. Curtiss, K. Overton, and M. Redner, "USE OF RFID IN HEALTHCARE INDUSTRY," *Acad. Bus. Res. Inst.*, 2013.

[16] M. Laskowski, B. Demianyk, M.R. Friesen, and R.D. McLeod, "Uncertainties Inherent in RFID Tracking Systems in an Emergency Department," in *2010 IEEE Workshop on Health Care Management (WHCM), ,* 2010, pp. 1–6.

[17] S. Anusha and S. Iyer, "RFIDcover - A Coverage Planning Tool for RFID Networks with Mobile Readers," in *Embedded and Ubiquitous Computing – EUC 2005 Workshops*, 2005, pp. 1047–1057.

[18] L. Xie, Q. Li, X. Chen, S. Lu, and D. Chen, "Continuous scanning with mobile reader in RFID systems: an experimental study," *MobiHoc*, pp. 11–20, 2013.

[19] H. Seo and C. Lee, "A new GA-based resource allocation scheme for a reader-to-reader interference problem in RFID systems," *Commun. (ICC), 2010 IEEE Int. Conf.*, pp. 1–5, May 2010.

[20] P. Weijie and L. Shaobo, "Multi-objective optimization of RFID network based on genetic programming," *Netw. Based Genet. Program. Inf. Technol. J. 10*, vol. 10, no. 12, 2011.

Direction of Arrival Estimation Using Particle Swarm Optimization-based SPECC

In-Sik Choi
Department of Electronic Engineering, Hannam University
133 Ojung-dong, Daeduk-Gu, Daejeon 306-791, Republic of Korea

Abstract

This paper proposes a novel direction of arrival (DOA) estimation scheme using particle swarm optimization (PSO)-based SPECC (Signal Parameter Extraction via Component Cancellation). The proposed algorithm is a PSO-based optimization method and extracts the amplitudes and incident angles of signal sources impinging on a sensor array in a step-by-step procedure. On the other hand, other algorithms extract those parameters at the same time. Proposed algorithm is very fast, robust to noise and has a high resolution in DOA estimation. Simulation results using artificially created data show the accuracy in the angle estimation and robustness to noise.

Key words: particle swarm optimization, direction of arrival, signal parameter extraction via component cancellation

1. Introduction

Estimation of signal direction of arrival (DOA) from the received data by array antenna is a critical issue in radar, sonar and communication systems. A variety of techniques for DOA estimation have been proposed. The well-known methods are the maximum likelihood (ML) technique [1], the multiple signal classification (MUSIC) [2], the root-MUSIC [3], the Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) [4], the genetic algorithm (GA)-based method [5], the maximum likelihood estimation using particle swarm optimization (PSO) [6] and the evolutionary programming (EP)-based signal parameter extraction via component cancellation (SPECC) [7]. Each algorithm has a strength and weakness relative to each other.

In this paper, a novel DOA estimation algorithm, PSO-based SPECC is proposed. The previously developed algorithms, such as ML, MUSIC, root-MUSIC, ESPRIT, GA-based method and ML using PSO extract the parameters (amplitudes and DOAs) of all source signals at the same time. Whereas, in the EP-based SPECC and the PSO-based SPECC, the parameters of each source signal out of multiple signals impinging on a sensor array are extracted in a step-by-step procedure. For the optimization of cost function, the PSO is used instead of EP. Recently, a PSO algorithm has been applied to electromagnetic optimization problems [8 − 10], because the basic algorithm of PSO is very simple and easy to implement.

Our algorithm is fast, robust to noise and accurate. In the simulation using artificially created data, we conduct some tests to verify the robustness to noise, and accuracy of the proposed PSO-based SPECC algorithm.

2. Particle Swarm Optimization

Traditionally, the gradient-based methods are used for complex function optimization. But, the gradient-based methods have a problem of local minima because of the characteristic of local search behavior. For this reason, the global optimization algorithms, such as the genetic algorithms (GA), the evolutionary strategies (ES), the evolutionary programming (EP), and particle swarm optimization (PSO) have emerged as efficient and robust search methods.

PSO is an optimization technique that offers attractive benefits to the user, including the fact that the algorithm can be easily understood and implemented. Compared to evolutionary optimization methods, PSO is based on the simulation of the social behavior of bird flocks and fish schools [8]. In PSO, individual particles in a swarm represent potential solutions, which move through the problem search space finding an optimal or good solution. In Fig. 1, the flowchart of typical PSO routine is shown in detail.

First, we define an *N*-dimensional vector space and a population of N_p particles is assumed to evolve in this vector space. Each particle is assigned the following position and velocity vectors, respectively:

$$\mathbf{x}_i(t) = [x_i^1(t) \;\; x_i^2(t) \;\; \cdots \;\; x_i^N(t)] \quad (1)$$

$$\mathbf{v}_i(t) = [v_i^1(t) \;\; v_i^2(t) \;\; \cdots \;\; v_i^N(t)] \quad (2)$$

where $i = 1,2,\cdots,N_p$.

We also postulate two more variables $\mathbf{x}_{pbest}$, the local particle best, and $\mathbf{x}_{gbest}$, the global best. The key idea of the classical PSO algorithm is how to predict the best update for the position in the next iteration. The particle dynamics utilizes its own experience and that of its neighbors. The idea is to change the velocity component in a manner that the increments are proportional to the difference between the current position of the particle and the local and global best, respectively. The particle dynamics are accomplished by the following two equations:

$$\mathbf{v}_i(t) = \varphi\mathbf{v}_i(t-1) + \rho_1\left(\mathbf{x}_{pbest} - \mathbf{x}_i(t)\right) + \rho_2\left(\mathbf{x}_{gbest} - \mathbf{x}_i(t)\right) \quad (3)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \quad (4)$$

where $i = 1,2,\cdots,N_p$; $\rho_1 = r_1c_1$ and $\rho_2 = r_2c_2$; c_1 and c_2 are the cognitive and social factors, respectively. r_1 and r_2 are two statistically independent random variables uniformly distributed between 0 and 1; φ is the inertia factor. In this paper, we select c_1 and c_2 are 1.5 and φ = 0.5.

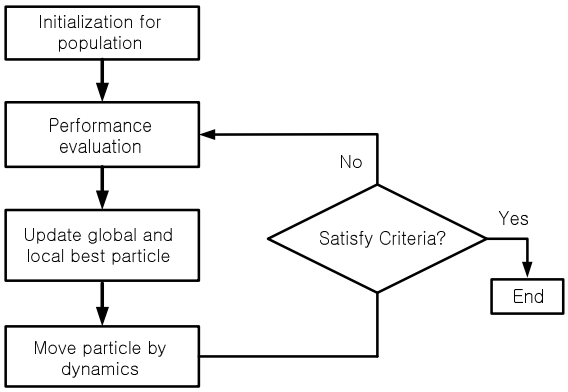


Fig. 1. Flowchart of PSO

3. PSO-based SPECC

The proposed PSO-based SPECC algorithm extracts the desired number of signal sources in order from the highest energy to the lowest energy. Therefore, the individual's dimension becomes very low compared to the GA-based method [5] and PSO-based method [6] which extract parameters of all signal sources simultaneously. In the methods of [5] and [6], each individual is composed of DOAs, amplitudes, and relative phases of whole signal sources, whereas in our algorithm, each individual is composed of only one signal source parameters. As the dimension of the individual vector increases, premature convergence to local minimum is more likely to occur and the convergence time is much longer.

In this paper, we first consider a linear array with *P* sensor elements as in Fig. 2. Then, the *M* narrow-band signals are assumed to be received by the linear array antenna.

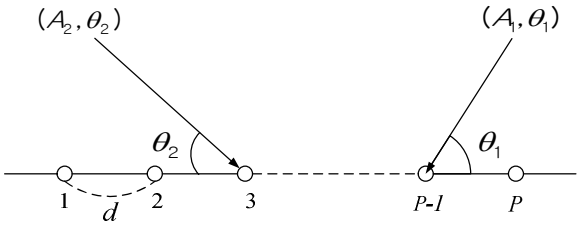


Fig. 2. Geometry of linear array antenna with two signal sources

If the first array element is assumed as reference point, the complex signals received by the *p*th element can be expressed by

$$y_p(k) = \sum_{m=1}^{M} A_m(k) \cdot \exp\left[j\frac{2\pi}{\lambda}(p-1)d\cos\,\theta_m\right] + n_p(k) \quad (5)$$

where λ is the signal central wavelength, d is the distance between the array elements, θ_m is DOA of the *m*th source signal, $A_m(k)$ is the complex amplitude of the *m*th source signal at *k*th time sample, and $n_p(k)$ is the additive noise of *p*th array element at *k*th time sample. We obtain the average received signal of *p*th array element using *K* time samples to reduce the noise by the equation (6)

$$y_p^{ave} = \frac{1}{K}\sum_{k=1}^{K} y_p(k) \quad (6)$$

The detailed PSO-based SPECC algorithm is as follows:

STEP 1. (Initialization) Set *m*=1, where *m* is the index of iteration to extract the *m*th signal source, and define the received average signal at the *p*th array element as $y_p^{ave,m}$ for *p=1,2,...,P*, where *P* is the number of array elements.

STEP 2. (Parameter extraction) Obtain the complex coefficients A_m and θ_m that minimize the following cost function F_m of the *m*th iteration using the PSO subroutine:

$$F_m = \sum_{p=1}^{P}\left|y_p^{ave,m} - A_m \cdot exp\;\left[j\frac{2\pi}{\lambda}(p-1)d\cos\,\theta_m\right]\right|^2 \quad (7)$$

In the PSO subroutine, the particle position vector is composed of the real and imaginary parts of A_m and θ_m. Terminate the PSO subroutine when the available execution time has passed because we do not know the minimum value of F_m.

STEP 3. (Component cancellation) Subtract the components of the determined signal sources in Step 2 from $y_p^{ave,m}$ and obtain $y_p^{ave,m+1}$, $p=1,2,...,P$, as follows:

$$y_p^{ave,m+1} = y_p^{ave,m} - A_m \cdot exp\left[j\frac{2\pi}{\lambda}(p-1)d\cos\theta_m\right] \quad (8)$$

STEP 4. (Termination check) Let $m=m+1$. Return to Step 2, unless the desired M components are extracted.

As explained in [7], the general SPECC algorithm recursively estimates a DOA and amplitude of each source signal. During each iteration, the highest energy source in the remained average signal ($y_p^{ave,m}$) is determined, and its DOA and amplitude are regarded as parameters of each source signal. After determining one signal source, the SPECC algorithm subtracts the determined signal components from the complex remained signal ($y_p^{ave,m}$) and obtain $y_p^{ave,m+1}$ which is the remained average signal used for the next iteration. This procedure is repeated until the residual energy is below the predefined threshold related to noise level or the iteration index (m) reaches the predefined (or estimated) value M. In this paper, we assume that M is known or pre-estimated. Typically, for an estimation of the total signal sources, AIC [11] and MDL [12] could be used, but they have a high computational cost and may fail in noisy environments. Unlike the MUSIC or root-MUSIC, the false estimation of the number of signal sources (M) does not affect the accuracy of extracted parameter values in our algorithm. Therefore, our algorithm can even be used for a sufficiently large M case and observe the magnitude of extracted source signal. If the signal magnitude becomes relatively small, we can consider this signal as noise and stop the SPECC algorithm.

4. Simulation results

To verify the performance of the PSO-based SPECC algorithm, we use the artificially created data. We consider two source signals which are located in 45^o and 135^o. The amplitudes and DOAs of two signal sources are shown in Table I. The number of array element P = 10 and inter-element distance $d=\lambda/2$.

Table I. Amplitudes and DOAs of two signal sources

number	DOA [degree]	Amplitude
1	45	1.0+j0.0
2	135	1.0+j0.0

Fig. 3 shows the extracted parameters of the proposed algorithm in the noise-free condition. The number of time samples (K) is assumed as 1. You can see that the two signals sources at 45^o and 135^o are accurately extracted in the proposed algorithm. Estimated DOAs (45.02^o & 134.21^o) and its amplitudes (0.9919+j0.0212 & 0.9762-j0.1412) are very accurate.

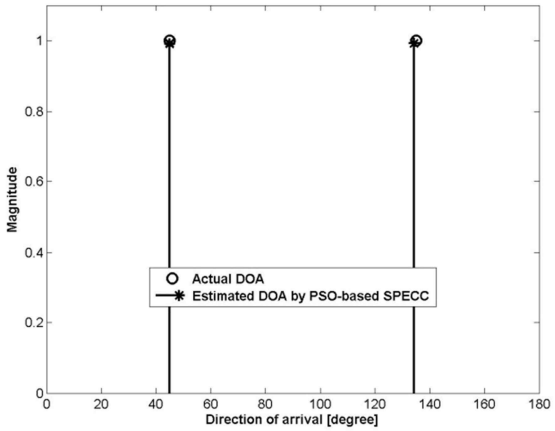


Fig. 3. Proposed algorithm, noise-free, M=2, K=1

In next simulation, we added zero-mean white Gaussian noise to the received signals to test robustness to noise.

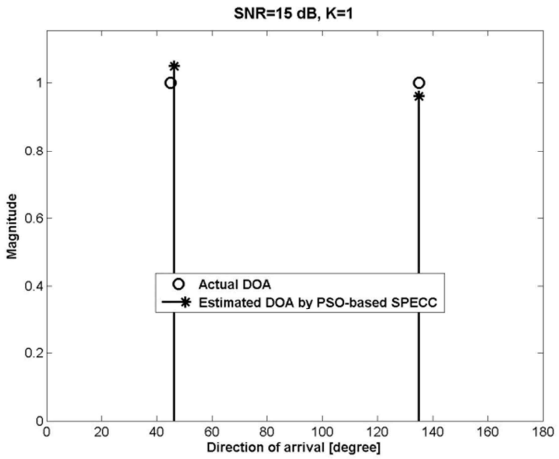


Fig. 4. Proposed algorithm, SNR=15 dB, M=2, K=1

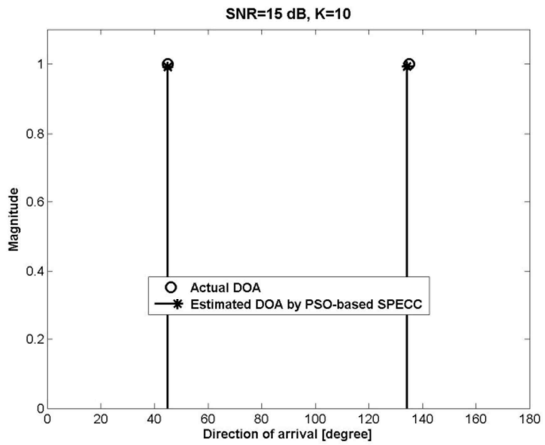


Fig. 5. Proposed algorithm, SNR=15 dB, M=2, K=10

Fig. 4 and Fig. 5 show the extracted DOAs of two source signals using PSO-based SPECC algorithm with a SNR of 15 dB when the number of time samples (K) is one and ten, respectively. As the number of time samples (K) is increased, the average signal (y_p^{ave}) has a lower noise level. Therefore, we can obtain more accurate DOAs.

5. Conclusion

We proposed a novel high-resolution DOA estimation method which is called PSO-based SPECC algorithm in this paper. The proposed algorithm has the high-resolution, robustness to noise, and good accuracy. Our algorithm also converges to the optimum value very fast compared to the conventional method which extracts all DOA parameters at the same time. In the simulation results, we verified these characteristics using the artificially created noise-free and noisy data. Future works include the performance comparison of our method with the well-known methods such as root-MUSIC or EP-based SPECC, etc.

6. Acknowledgment

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2012R1A1A4A01009094).

7. References

[1] Schweppe, F. C.: Sensor array data processing for multiple signal sources. IEEE Trans. on Inform. Theory 14, 294−305 (1968)

[2] Schmidt, R. O.: Multiple Emitter Location and Signal Parameter Estimation. IEEE Trans. on Antennas and Propagation 34, 276−280 (1985)

[3] Rao, B. D.: Performance analysis of root-MUSIC. IEEE Trans. On Acoustics, Speech, and Signal Processing 37, 1939−1949 (1989)

[4] Roy, R., Paulraj, A. and Kailath T.: ESPRIT - Estimation of Signal Parameters via Rotational Invariance Techniques. IEEE Trans. On Acoustics, Speech, and Signal Processing 37, 984−995 (1989)

[5] Karamalis, P., Marousis, A., Kanatas, A., Constantinou, P.: Direction of arrival estimation using genetic algorithm. 2001 IEEE Vehicle Technology Conference (VTC2001), pp. 162−166 (2001)

[6] Zeng, J., He, Z., Liu B.,: Maximum likelihood DOA estimation using particle swarm optimization algorithm, 2006 CIE International conference on Radar, pp. 1-4 (2006)

[7] Choi, I.-S., Rhee, I.-K., Lee, Y.-H.: Signal Parameter Extraction via Component Cancellation Using Evolutionary Programming. 2007 International Conference on Future Generation Communication and Networking (FGCN 2007) 2, 458-462 (2007)

[8] Boeringer, D. W. and Werner, D. H.: Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis, IEEE Trans. on Antennas and Propagation 52, 771-779 (2004)

[9] Robinson, J., Sinton, S., and Rahmat-Samii, Y.: Particle swarm, genetic algorithm, and their hybrids: Optimization of a profiled corrugated horm antenna, IEEE Antennas Propagation Soc. Int. Symp. Dig. 1, 314-317 (2002)

[10] Malik, N. N. N. A., Esa, M., Yusof, S. K. S., and Marimuthu, J.: Suppression of antenna's radiation sidelobes using particle swarm optimization, PIERS Proceedings, 18-21 (2009)

[11] Wax, M. and Kailath, T.: Detection of signals by information theoretic criteria. IEEE Trans. On Acoustics, Speech, and Signal Processing 33, 387-392 (1985)

[12] Wax M. and Ziskind, I.: Detection of the number of coherent signals by the MDL principle. IEEE Trans. On Acoustics, Speech, and Signal Processing 37, 1190-1196 (1989)

Enhancement of Power Quality through Particle Swarm Optimization

L.Ravi Srinivas[1], B.Mahesh Babu[2], S.S.Tulasi Ram[3]
[1]Department of EEE, Gudlavalleru Engineering College, JNTUK, Kakinada, A.P, India
[2]Department of EEE, Gudlavalleru Engineering College, JNTUK, Kakinada, A.P, India
[3]Department of EEE, JNTUH, Hyderabad, AP, India

Abstract: *The term Electric Power quality (PQ) has recently acquired intensified interest due to the wide spread use of large number of complicated industrial processes. The necessity of using intelligent tools in power quality analysis is increasing day by day to enhance power quality in the utility. This paper deals with analysis and comparison of PI based Particle Swarm Optimization (PSO) with conventional PI controller for harmonic reduction in the source current. Shunt Active power Filter (SAPF) is one of the controllers that can be used to suppress the source current harmonics. Hysteresis Band Current Controller (HBCC) is used to control the switching of Voltage Source Inverter. The Instantaneous Active and Reactive power (PQ) Theory is used to generate the reference compensating currents. Simulations are carried out in MATLAB/SIMULINK environment using Simpower system toolbox.*

Keywords: Power Quality (PQ), Shunt Active Power Filter (SAPF), Instantaneous Active and Reactive Power (PQ) Theory, Hysteresis Band Current Controller (HBCC), Particle Swarm Optimization (PSO), Total Harmonic Distortion (THD).

1 Introduction

The advent and increasing usage of a variety of non linear loads, large number of complicated industrial processes and resonance in utility power system can cause a large amount of harmonics injection into power system. Shunt Active power filter (APF) based on power semiconductor technology is currently considered as the most competitive equipment for mitigation of power harmonics. Since the concept, "Active power filter", was first developed by L. Gyugyi[l] in 197Os. Even now the research progress on several issues about active power filters like configuration, measurement, control and installation. The control strategy for active power filter was concluded by H. Akagi [2] as three modes, load current detection supply current detection and voltage detection, for different applications. The international standards recommendation and requirements for harmonic control in electrical power systems imposed some harmonic limits [3].

Since 1970's when the seeds for development of power electronic equipment were sown, the active power filters (APF's) have been one of the most attractive solutions in suppressing harmonic content to enhance power quality which in turn ensure the better power distribution system. In view of its connection to the power system, there are two types of (APFs) as Series Active Power Filter and Shunt Active Power Filter [4]. The Research & Development shows that the series active power filter is preferable to compensate the harmonics in supply voltage, while parallel active power filter is more suitable to compensate the harmonics in the supply current. The Active power filters

(APFs) firstly developed by Akagi, provide better performances of harmonic elimination [2],[5]. Fig.1 shows the block diagram representation of SAPF.

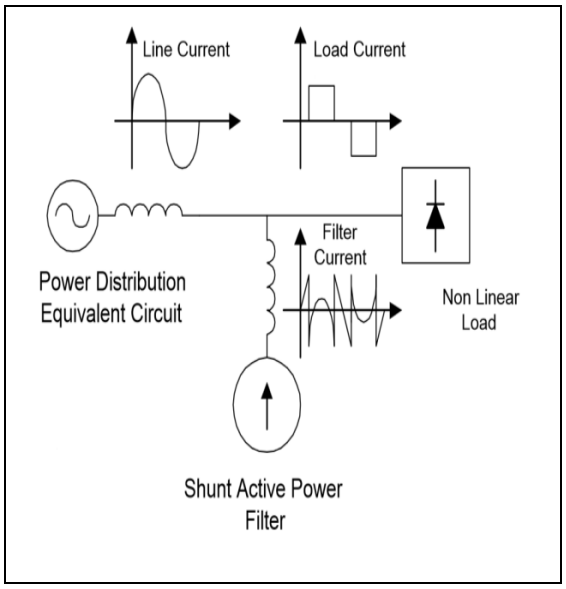


Fig.1: Compensation characteristics of a Shunt Active Power Filter.

The present paper focuses on identification of harmonic currents with Instantaneous active and reactive power theory (p–q theory) in time domain. Proportional plus Integral controller is used to control the loop. The research work describes the optimization of Kp, Ki parameters of PI controller using Particle Swarm Optimization (PSO) to maintain the constant DC capacitor voltage. Then finally Hysteresis Band Current Controller (HBCC) is used to control the switching of Voltage Source Inverter.

2 System Configurations

A 3-Ø, 415V, 50Hz supplied Shunt Active Filter is developed in MATLAB/SIMULINK using Simpower System toolbox.

2.1 Design Parameters of Shunt Active Filter

A shunt active filter consists of three phase, 6 pulse voltage source inverter. The line-line voltage is considered as 415V. The designs for dc bus capacitor, dc bus voltage are as follows

2.1.1 DC Capacitor Voltage

The minimum dc bus voltage should be greater than twice of the peak of the phase voltage of the system. The dc bus voltage is calculated as

$$v_{dc} = \frac{2\sqrt{2}}{\sqrt{3}} \frac{v_{LL}}{m} \quad (1)$$

Where, m is the modulation index =1 and V_{LL} is the ac line voltage of three phase source.

2.1.2 DC Bus Capacitor

The value of dc bus capacitor (Cdc) is given by

$$\frac{1}{2} C_{dc} [(v_{dc}^{2}) - (v_{dc1}^{2})] = 3V(aI)t \quad (2)$$

Where, Vdc is the reference dc voltage and Vdc1 is the minimum voltage level of dc bus, a is the over loading factor taken as 1.2, V is the phase voltage, I is the phase current and t is time by which the dc bus voltage is to be recovered [6].

3 Control Strategy

3.1 Instantaneous Power Theory

Instantaneous active and reactive power theory (p–q theory) can be used to identify the reference harmonic currents. The first step of this method is transforming the three phase (a, b, c) voltages and currents to two-phase (α, β) using the direct conversion. The principle adjustment of this method is to extract the fundamental component using Butterworth filters. The voltages and currents at the points of connections are sensed from nonlinear load can be converted by the components into:

$$\begin{bmatrix} v_\alpha \\ v_\beta \\ v_0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} v_{sa} \\ v_{sb} \\ v_{sc} \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \\ i_0 \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} i_{sa} \\ i_{sb} \\ i_{sc} \end{bmatrix} \quad (4)$$

The instantaneous active and reactive power can be expressed by the following system:

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} v_\alpha & v_\beta \\ -v_\beta & v_\alpha \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} \quad (5)$$

The instantaneous active and reactive power can be decomposed into AC and DC parts. The DC part resulted from the fundamental current and voltage and the AC part resulted from the harmonics [7-8].

$$\begin{aligned} p &= \bar{p} + \tilde{p} \\ q &= \bar{q} + \tilde{q} \end{aligned} \quad (6)$$

Where $\bar{p}, \bar{q}$: DC average value of the instantaneous real and reactive power respectively.

$\tilde{p}, \tilde{q}$: AC value of the instantaneous real and reactive power respectively

The reference currents are calculated by the following expression:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{1}{v_\alpha^{\,2} + v_\beta^{\,2}} \begin{bmatrix} v_\alpha & -v_\beta \\ v_\beta & v_\alpha \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} \quad (7)$$

With

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \frac{1}{\Delta}\begin{bmatrix} v_\alpha & -v_\beta \\ v_\beta & v_\alpha \end{bmatrix}\begin{bmatrix} \bar{p} \\ 0 \end{bmatrix} + \frac{1}{\Delta}\begin{bmatrix} v_\alpha & -v_\beta \\ v_\beta & v_\alpha \end{bmatrix}\begin{bmatrix} 0 \\ \bar{q} \end{bmatrix} + \frac{1}{\Delta}\begin{bmatrix} v_\alpha & -v_\beta \\ v_\beta & v_\alpha \end{bmatrix}\begin{bmatrix} \tilde{p} \\ \tilde{q} \end{bmatrix} \quad (8)$$

Where $\Delta = v_\alpha^{\,2} + v_\beta^{\,2}$

The reference currents based on the instantaneous active and reactive power are determined according to the flowing equation:

$$\begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\beta \end{bmatrix} = \frac{1}{v_\alpha^{\,2} + v_\beta^{\,2}} \begin{bmatrix} v_\alpha & -v_\beta \\ v_\beta & v_\alpha \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \tilde{q} \end{bmatrix} \quad (9)$$

Finally, the calculation of the reference harmonic current is determined as follows:

$$\begin{bmatrix} i_a^* \\ i_b^* \\ i_c^* \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} \tilde{i}_\alpha \\ \tilde{i}_\phi \end{bmatrix} \quad (10)$$

The block diagram of (p–q theory), is shown in Fig.2.

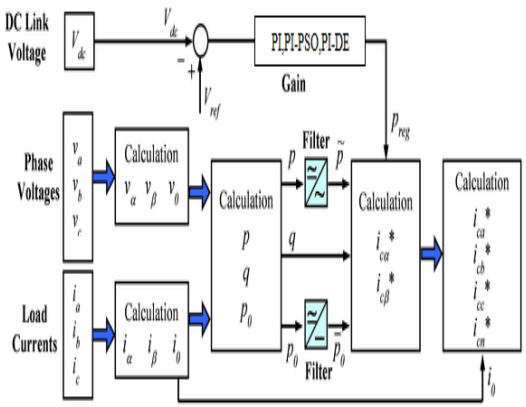


Fig.2: Instantaneous Active and Reactive Power Theory for Reference Current Generation.

3.2 Hysteresis Band Current Technique

The instantaneous value of the error between the reference harmonic currents (i^*) generated from pq theory and the actual injection harmonic currents (i_{inj}) generate the gate pulses [9]. The hysteresis control law is given as in Fig.3.

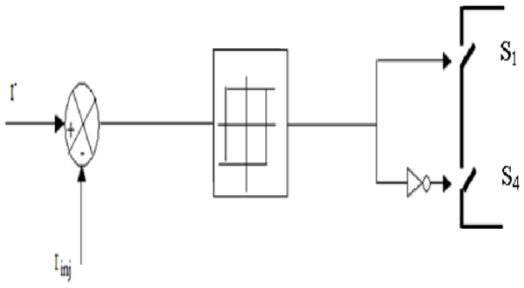


Fig. 3. Hysteresis Band Current Control for generating pulses for the controlled VSI.

The output of hysteresis band current control (S_{14},S_{36},S_{25}) are the gate pulses of six switches[8] :

$$S_{14} = \begin{cases} 0 & if\ S_1\ is\ closed\ \ and\ \ S_4\ \ is\ open \\ 1 & if\ S_4\ is\ closed\ \ and\ \ S_1\ \ is\ open \end{cases}$$

$$S_{36} = \begin{cases} 0 & if\ S_3\ is\ closed\ \ and\ \ S_6\ \ is\ open \\ 1 & if\ S_6\ is\ closed\ \ and\ \ S_3\ \ is\ open \end{cases}$$

$$S_{25} = \begin{cases} 0 & if\ S_2\ is\ closed\ \ and\ \ S_5\ \ is\ open \\ 1 & if\ S_5\ is\ closed\ \ and\ \ S_2\ \ is\ open \end{cases}$$

3.3 PI Controller

The PI controller consists of proportional term (Kp) and integral term (Ki). Proportional value determines the reaction to the error; the Integral determines the reaction based on the sum of recent errors. The reference currents for the control of active filter are generated according to the equation (11).The output of PI controller at the dc bus voltage of active filter is considered as the current (i_{loss}) for meeting the injection requirements.

$$i_{loss(n)} = i_{loss(n-1)} + k_p |v_{sn} - v_{s(n-1)}| + k_i v_{sn} \quad (11)$$

Where,($v_{sn} - v_{s(n-1)}$) is the error between the reference(Vdc*) and sensed (Vdc) dc voltage at the nth sampling instant. Kp and Ki are the proportional and the integral gains of the dc bus voltage PI controller. The placement of PI controller is shown in Fig.2 [6].

3.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique inspired by social behavior of bird flocking or fish schooling. PSO learns from the scenario and uses it to solve the optimization problems [9-10]. The moments of the birds are reflected as and we call it as moments of "particle". All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. This paper employs the objective function as minimization of Total Harmonic Distortion (THD). The fitness function is defined as follow:

$$F = f_{THD} \quad (12)$$

The optimization parameters are proportional gain (Kp) and integral gain (Ki), the transfer function of PI controller is defined by:

$$G_c(s) = k_p + \frac{k_i}{s} \quad (13)$$

The gains Kp and Ki of PI controller are generated by the PSO algorithm for a given plant. As shown in Fig.4.

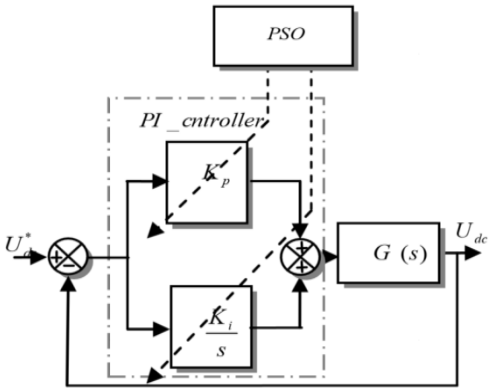


Fig.4. PI -PSO Control System.

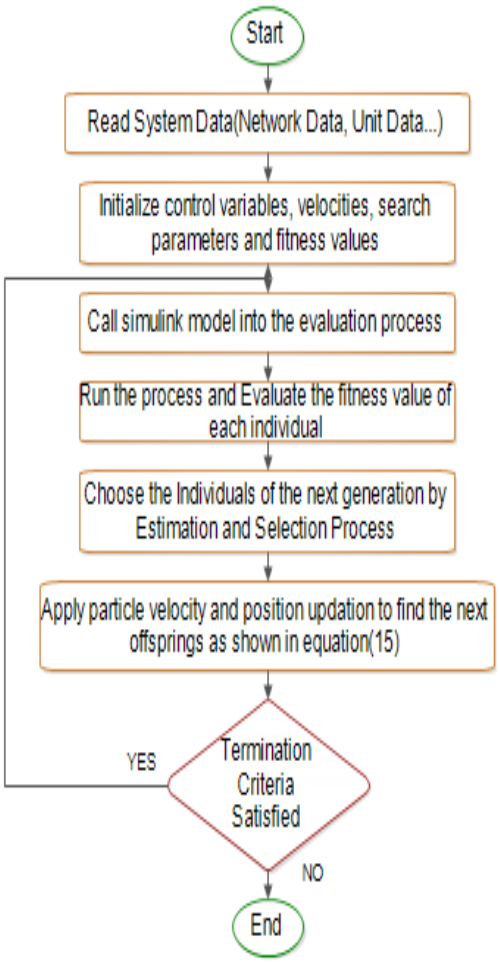


Fig.5. Flowchart of PSO algorithm.

The output of the PI controller u(t) is given by:

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt \quad (14)$$

For the taken plant, the problem of designing a PI controller is to adjust the

parameters Kp and Ki for getting a desired performance of the considered system.

The position of particle move rule is shown as follows:

$$V_s(t+1) = wV_s(t) + c_1 r_1(p_{best} - x_s(t)) + c_2 r_2(G_{best} - x_s(t))$$
$$X_s(t+1) = X_s(t) + V_s(t+1) \quad (15)$$

where $V_s(t)$ represents the velocity vector of particle s in t time; $X_s(t)$ represents the position vector of particle s in t time; $_{best}$ is the personal best position of particle s, G_{best} is the best position of the particle found at present; w represents inertia weight; c1, c2 are two acceleration constants, called cognitive and social parameters respectively; and r1 and r2 are two random functions in the range [0, 1]. The flow chart of general PSO is shown in Fig.5.

4 Simulation Results

This section describes the comparison of total harmonic distortion of source current by employing shunt active power filter controlled by PI controller and Partical Swarm Optimization. Three phase non-linear load (Diode rectifier) is considered for simulation. Without filtering the source current THD is 26.78% which is shown in Fig.6. Instantaneous power theory is used to estimate the reference compensating current.

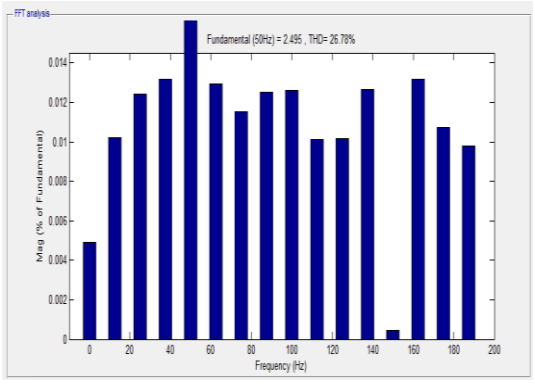


Fig.6. THD representations without Shunt Active filter.

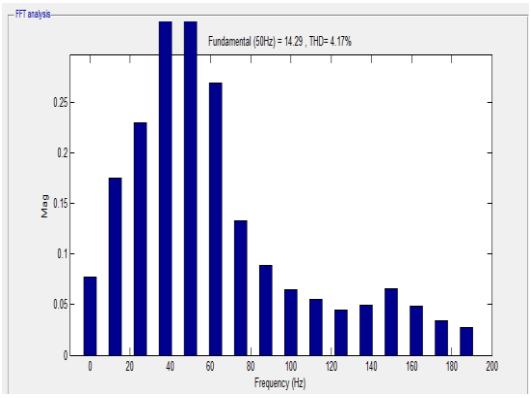


Fig.7. THD representations with Shunt Active filter controlled by PI controller.

With SAPF PI controller t the illustration of THD in source current which is reduced to 4.17% by making Kp=1 and Ki=1 shown in Fig.7.

With SAPF PI-PSO optimized controller the illustration of THD in source current which is reduced to 1.69% making Kp and Ki ranging from (0,180), c1=2 and c2=2 is shown in Fig. 10. The optimized values of Kp and Ki occurs at 15.572 and 180 respectively making the velocities run per 20 divisions between maximum and minimum velocities. Ten populations is considered for evaluation.

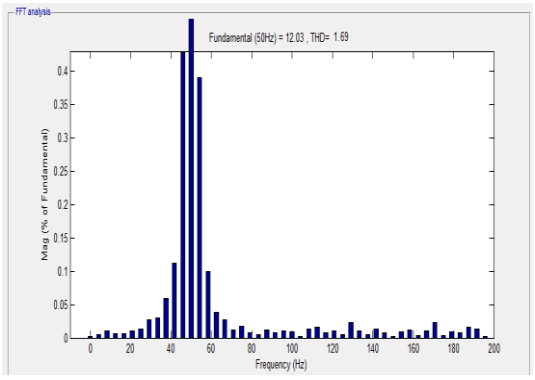


Fig.8. THD representation with Shunt Active filter controlled by PI-PSO controller

With SAPF PI-PSO optimized controller the illustration of convergence of THD in source current and the Dc link voltages is shown in Fig.9 and Fig.10. The PI-PSO optimization has been run for 0 to 100 iterations.

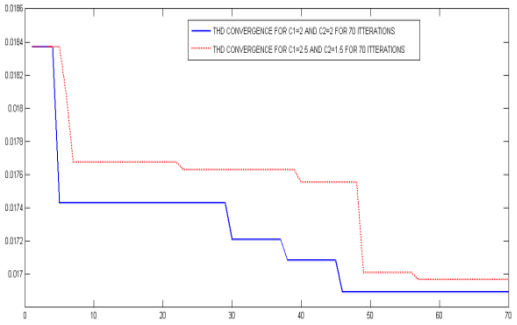


Fig.9. THD Convergence of PI-PSO Algorithm.

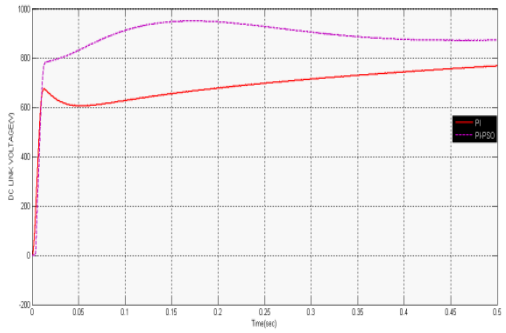


Fig.10. DC Link voltage representation with Shunt Active filter controlled by PI, PI-PSO.

Table.1. THD of Source Current (Phase A)

Parameter	THD
Without Filter	26.78%
With Conventioal PI	4.17%
With Fuzzy logic[6]	3.59%
With Neural Network's[6]	3.07%
With optimized PI-PSO	1.69%

Table.2. THD of Source Current (Phase A)

Parameter		THD
C1	C2	(%)
2	2	1.69
2.5	1.5	1.71
3	1	1.74
1	3	1.81

Table 1 discuss the comparative analysis of the total harmonic distortion of PSO algorithm with the prior artificial intelligent techniques where as the Table 2 discuss the total harmonic distortion of self evaluation of PSO algorithm varying the parameters c1 and c2 (equation 15).

5 Conclusions

This paper presents the comparative performance analysis of the THD using the shunt active filter based on artificial intelligent techniques. Comparative analysis of PI controller and Particle Swarm Optimization showed that Particle Swarm Optimization has been proved to be better in terms of harmonic reduction and compensating the reactive power. The dc bus voltage has been maintained constant equal to the reference voltage by PI, Particle Swarm Optimization controllers. It has been found that these robust and nonlinear controls prove to be better than conventional control.

6 References

[l] L. Gyugyi and E. C. Strycula,"Active AC power filter. Proceedings in IEEE IAS Annual Meeting", pp. 529–529, 1976.

[2] H. Akagi, "New trends in active filter for power conditioning", IEEE Transaction on Industry Application, Vol. 32, 1996.

[3] "IEEE recommended practices and requirements for harmonic control in electrical power systems", IEEE Std 519-1992, 12 April, 1993.

[4] M. EI-Habrouk, M. K. Darwish, and P. Mehta, "Active power filters: A review". Proceedings in IEE on Electric Power Applications, Vol. 147, pp. 403-412, 2000.

[5] F. Z. Peng, H. Akagi, and A. Nabae, "Compensation characteristics of the combined system of shunt passive and series active filters", IEEE Transaction on Industry Applications, Vol. 29, pp. 144-152, 1993.

[6] Sakshi Bangia, P.R.Sharama, Maneesha Garg, "Comparison of Artificial Intelligence Techniques for The Enhancement of Power Quality", International Conference on Power, Energy and Control (ICPEC), 2013.

[7] A. Emadi, A. Nasiri, S. B. Bekiarov, "Uninterruptible power supplies and active filters" Illinois Institute of Technology Electrical

and Computer Engineering Department Chicago, IL, 2005.

[8] Aziz Boukadoum, Tahar Bahi, Abla Bouguerne, Youcef Soufi, Sofiane Oudina, "Hysteresis Band Current and Fuzzy Logic Control for Active Power Filter", Eighth International Conference and Exhibition on Ecological Vehicles and Renewable Energies (EVER), 2013.

[9] Brahim Berbaoui, Chellali Benachaiba and Rachid Dehini, " Design of optimal PI controller for Shunt Active Power Filter using Particle Swarm Optimization", Quatrième Conférence Internationale surle Génie Electrique CIGE'10, Université de-Bechar, Algérie, 2010.

[10] Kennedy J, Eberhart R, "particle swarm optimization", In: Proceedings of IEEE international conference on neural networks, pp 1942-1948, 1995.

SESSION

EVOLUTIONAY ALGORITHMS AND APPLICATIONS + ANN + FORCASTNIG AND TIME SERIES ANALYSIS

Chair(s)

TBA

Forgetting Beneficial Knowledge in Decomposition-Based Reinforcement Learning Using Evolutionary Computation

Sean Mondesire[1] **and R. Paul Wiegand**[2]
[1]College of Engineering and Computer Science, University of Central Florida, Orlando, Florida, USA
[2]Institute for Simulation and Training, University of Central Florida, Orlando, Florida, USA

Abstract— *This work demonstrates that critical information can easily and prematurely be removed from a decomposition-based reinforcement learning system. One possible effect to the forgotten knowledge is the complete loss in ability to solve a previously learned problem when the system is given a new problem to optimize. In artificial neural networks, this is called catastrophic forgetting and has been shown to cripple performance. We study this phenomenon to understand its effects on problem performance and to investigate suspected consequences experienced by other decomposition-based approaches. Furthermore, using an abstract decomposition-based reinforcement learning paradigm with a simple evolutionary algorithm, we analyze the role stability-plasticity imbalance has in the premature loss of critical knowledge.*

Keywords: Stability-Plasticity Dilemma, Decomposition-Based Reinforcement Learning, Evolutionary Computation

1. Introduction

Using evolutionary computation, we demonstrate the unintentional loss of beneficial knowledge in decomposition-based reinforcement learning. This premature removal of critical information has had disastrous effects on performance and learning rates in other learning genres. By providing this demonstration, we highlight the challenge to those who use decomposition-based reinforcement learning, identify potential causes, and provide detailed analysis of the effects of forgetting. Additionally, the reinforcement learning and evolutionary computation communities gain a deeper understanding of why this type of forgetting can be so devastating to learning effectiveness. Finally, this demonstration is needed to uncover potential performance-hindering pitfalls future learning systems can prepare for.

Reinforcement learning (*RL*) is a machine learning area that solves problems based on maximizing the expected cumulative reward of a system's actions [1]. *Decomposition-based reinforcement learning* (*DBRL*) is a RL method that optimizes a learning system's action-selection by solving a series of simpler subproblems to solve a complex problem. Hierarchical abstract machines [2], MAXQ [3], and layered learning [4] are examples of DBRL techniques used to train computer-based agents to solve complex problems.

One challenge of DBRL is retaining the ability to solve previously learned problems (stability) while learning to solve new ones (plasticity). This balancing act is called the *stability-plasticity dilemma* [5]. Systems that favor stability may have difficulty in adapting to learn how to solve new problems quickly; on the other hand, a preference for plasticity can result in the rapid loss of solutions to previously learned problems. It is this imbalance that favors plasticity and the loss of *beneficial knowledge*, critical information that improves performance of a solution, that we are focused on in this work.

Existing literature reveals a preference for plasticity in artificial neural networks can severely hinder performance of a learning system [6], [7]. The problem arises when a network is optimized on a set of inputs to solve one problem, then optimizes its activation weights on a new problem's input. The second set of inputs causes the network to override weights necessary to solve first problem. This imbalance for plasticity can be extremely detrimental to earlier learned problems to a point where performance of the first problem is completely lost. This occurrence is called *catastrophic forgetting* [8].

Previous work in HBRL using evolutionary computation methods have suggested and demonstrated that catastrophic forgetting can occur in the learning approach. For example, in a direct policy search used to optimize autonomous agent policies in a predator-prey scenario, experiments have shown that previously learned proficiency of critical skills can drop significantly when an agent learned a new skill [9]. Because each skill was vital in solving the problem the agents were learning, overall performance and the learning rate suffered. Though the work did not explicitly focus on forgetting phenomenon, the work's experiment results suggest that a stability-plasticity imbalance was the cause of the knowledge loss.

In a second work, catastrophic forgetting in HBRL using an evolutionary algorithm was analyzed while verifying forgetting diagnostic and measurement metrics [10]. The experiments demonstrated that the optimization of new problems drove out critical, older knowledge. This caused previously learned solutions of older problems to be lost and older problem performance to drastically decrease. The work used contrived Boolean-logic problem that purposely contained conflicting subproblems, ensuring older beneficial

knowledge would be replaced by new, conflicting information.

In this work, we demonstrate the implicit premature loss of beneficial knowledge in DBRL using simple, well-crafted methods and problems. This work is different from others because it explicitly studies the loss of beneficial knowledge while learning. Additionally, it uses standard optimization problems that are not contrived to demonstrate forgetting in this area of RL.

To achieve the demonstration goal, we use the abstract DBRL paradigm of *layered learning* with a *(1+1) Evolutionary Algorithm* (*[1+1] EA*) to solve two Boolean-logic problems. Layered learning was selected because of its qualities shared with many other RL techniques, including robot shaping and transfer and incremental learning. Because of these shared characteristics, demonstrating catastrophic forgetting in layered learning may lead to important stability-plasticity findings in related approaches. A simple (1+1) EA is used because it reduces complexity to the employed learning algorithm and makes experiment results easier to interpret. By accomplishing our goal, we will provide a better understanding of what causes the imbalance and its the effects.

2. Related Work

Catastrophic forgetting is caused by the premature removal of critical information relied on for problem solving from a system. The challenge is trying to understand what causes the premature knowledge loss.

Several methods have had success in trying to delay the loss of important knowledge or regain forgotten information. These methods include the use of deletion strategies, complementary learning systems, and rehearsal techniques, to name a few.

Deletion strategies are mechanisms used to determine when a pair is removed from a policy. Some strategies include random, frequency, temporal, spatial, and utility-based removal.

Rehearsal techniques repeat the optimization of a problem to reintroduce forgotten knowledge, reinforce solutions previously learned, or further improve the current problem solving ability [11], [12]. By repeating the optimization of a problem, a system can revert policy changes to mappings that increased performance of a solution.

Complementary learning systems have shown measurable success in preventing complete proficiency loss in neural networks. The system applies dual-memory storage regions to a network originally with one, in a manner that is similar to short- and long-term memory in natural systems. Examples of these two-phased learning systems include Hattori's use of a hippocampus-neocortical configuration with a Hopfield network [13], *pseudorehearsal* [11], *pseudo-recurrent networks* [14].

These discussed works demonstrate that there is a forgetting problem in other learning systems and provide mitigation techniques; unfortunately, the same amount of attention has not been given to DBRL. Two works that explore forgetting in this type of learning are in [9], [10]. Both works combine layered learning, an abstract DBRL paradigm, with an evolutionary algorithm to facility learning of complex problems.

The first work studies the effectiveness of a DBRL when applied to a direct policy search to train autonomous *non-playable characters* (*NPCs*) in a predator-prey based video game [9]. The study decomposes a predator behavior into three subproblems and sequentially trains the NPC policies on each. The result shows that as the performance criterion changes during the optimization of each subproblem, performance of an older subproblem decreases. The work concludes that the learner's sub-optimal performance, which is compared to a monolithic learner, is caused by the decreases in subproblem performance.

The second work introduces two performance-based forgetting metrics [10]. The metrics, *direct* (*DFM*) and *maximized* (*MFM*) forgetting metrics, are proposed to classify and measure a type of forgetting that is occurs in an RL system. To validate and verify the metrics, experiments use layered learning with a (1+1) EA to optimize a bit-string over the *leading ones trailing zeros* (*LOTZ*) problem. LOTZ and the employed decomposition of subproblems are contrived to purposely drive out older information from the system by having conflicting subproblems optimize on between consecutive layers. Experiment results show performance of the LOTZ problem suffers because critical subproblem performance decreases. The work concludes that under these conditions, the metrics are validated and that a demonstration of forgetting in the abstract DBRL system took place.

The research in this paper is an extension of these two works ([9], [10]) and explicitly demonstrates the loss of beneficial knowledge in an DBRL system. To do so, we use two completely different, less contrived Boolean-logic problems to show the forgetting pitfalls and challenges of an RL system. Furthermore, we purposely avoid all mitigation techniques to study what happens when nothing is explicitly in place to safeguard the learning system.

3. Methodology

To demonstrate forgetting, we use layered learning in combination with an evolutionary algorithm to solve two Boolean-logic problems. The objective is to show that as the learner optimizes a solution for a new problem, performance of an older, previously learned problem significantly decreases. It is suspected that several factors will cause beneficial knowledge to be driven out of the system, including subproblem conflict and the greedy nature of the employed algorithm. This investigation will determine

the causes of catastrophic forgetting under these simple experiment conditions.

Layered learning is an abstract machine learning paradigm that solves a complex problem by incrementally optimizing performance of simpler sub-problems [15], [4]. The paradigm has been used to train real and simulated autonomous robots and computer-based systems to solve a variety of complex problems, such as performing motor-skills and playing robotic soccer [16], [17], [18]. The paradigm learns by establishing and then sequentially iterating through a set of *layers*. Each layer organizes a learning plan by indicating the subproblem to optimize, the training scenario and conditions for that optimization, and the halting condition to signal when to proceed to the next layer, each predefined by the developer. The paradigm is abstract because it provides a general process of how to solve a complex problem using decomposition. Therefore, layered learning is not a machine learning algorithm but instead uses existing learning algorithms to facility learning.

We combine layered learning with a *(1+1) evolutionary algorithm* (*[1+1] EA*) to provide our demonstration. The (1+1) EA is a simple EA that uses one genetic operator to modify the sole individual in the population [19], [20]. Because the employed learning algorithm only has one operator and individual, the learning process is less complicated, allows for easier interpretation of what occurs during problem optimization, and focuses on the decomposition instead of the underlying algorithm.

This learning system optimizes a 128-bit long bit-string to solve two Boolean-logic problems: OneMax and Spin Glass. These problems were selected because for their simplicity and difference in linearity. OneMax is a linear function that has become a standard optimization problem in machine learning. Spin Glass is a natural non-linear problem that is not designed to demonstrate forgetting. By using these completely different problems, we demonstrate that the unintentional loss of beneficial knowledge can occur under conditions on opposite sides of the linearity spectrum.

Finally, we recognize that these two Boolean-logic problems are optimization problems. We use these simple problems and bit-string representation to symbolize more complex and common RL problems and representations. For instance, the bit-string may abstractly represent a policy in a RL's direct policy search. Here, each bit represents a state-action mapping (a *pair*) an agent can perform at any decision point, while the bit-string is the *policy* (the collection of pairs). Optimization of the bit-string on a Boolean-logic problem could then correspond to the optimization of an autonomous agent's policy on a RL multi-agent system problem.

3.1 OneMax

OneMax is a standard linear test problem that rewards the bit string for the number of 1 bits it contains, where the all-one string is optimal. The OneMax performance function is defined in Equation 1.

$$x \in \{0,1\}^n, OneMax(x) = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (1)$$

3.2 Spin Glass

In the Spin Glass problem the bit string is rewarded for every occurrence where consecutive bits and the first and last bits differ [21]. The Spin Glass performance function is defined in Equation 1, where $\oplus$ represents exclusive-or (xor). This problem represents natural problems with interacting, non-linear relationships between individual decisions.

$$x \in \{0,1\}^n, SpinGlass(x) = \frac{1}{n}(\sum_{i=1}^{n-1}(x_i \oplus x_{i+1}) + (x_n \oplus x_1)) \quad (2)$$

Table 1: Task and Subproblem Descriptions

Subproblem	Maximization Objective Description
OneMax	One bits in the entire string.
FirstHalfOneMax	Ones in the first half of the string.
SecondHalfOneMax	Ones in the second half of the string.
SpinGlass	Different consecutive bits and first and last bits.
MiddleOneThirdSpinGlass	Spin Glass over the middle 1/3rd bits.
MiddleTwoThirdSpinGlass	Spin Glass over the middle 2/3rd bits.
FirstThirdSpinGlass	Spin Glass over the left-most 1/3rd bits.
FirstTwoThirdSpinGlass	Spin Glass over the left-most 2/3rd bits.

Each Boolean-logic problem is decomposed into subproblems, displayed in Table 1. Each problem has two *test cases* that determine the halting condition of a layer. The *find optimal* trigger to move to the next layer is not activated until the subproblem's performance has reached optimality. The second is *fixed duration*, which runs a fixed number of mutations and evaluations (*time-steps*) in a layer before proceeding to the next. Within a test case are several test plans. A *test plan* is the decomposition used to solve the overall complex problem the system is optimizing. In a layer, the (1+1) EA continuously optimizes the bit-string on a test plan's subproblem until the test case's halting condition is satisfied.

The learning process is as follows: in the initial layer, a randomly generated bit-string is passed into a layer. A clone of the bit-string is then made and is mutated. The mutation operator modifies the clone by traversing over each bit, toggling a bit's value at a 1/*n* probability. Using the subproblem's performance evaluation function, the clone's performance is compared to that of the parent. If the clone's subproblem score is greater than the parent's, the clone replaces the parent in the population. This process of cloning, mutating, and evaluating repeats until the test case's halting

condition is satisfied. Once a layer is halted, the bit-string at the end of one layer is the initial bit-string for the next.

Test cases with one subproblem/layer act as the control for these experiments as it monolithically learns the complex Boolean-logic problem without decomposition. This control is important to our demonstration because monolithic learning using this (1+1) EA does not experience forgetting of a complex problem because the sole individual is only replaced if a clone outperforms the parent. In other words, performance is guaranteed to never decrease.

Finally, the fixed duration limits the number of time-steps in a test plan to be 90% of the averaged required time-steps to produce the optimal bit-string in the corresponding monolithic *find optimal* test plan.

4. Experiment Results

All test plans for both Boolean-logic problems were evaluated for 100 trials to have a large enough sample to draw significant conclusions. First we discuss our catastrophic forgetting demonstration using OneMax.

4.1 OneMax

Both of the OneMax Experiment test cases contained five test plans, defined in Table 2.

Table 2: OneMax Test Plans with Subproblem Sequence

Test Plan #	Layer 1	Layer 2	Layer 3
Plan0	OneMax	-	-
Plan1	FirstHalfOneMax	OneMax	-
Plan2	SecondHalfOneMax	OneMax	-
Plan3	FirstHalfOneMax	SecondHalfOneMax	OneMax
Plan4	SecondHalfOneMax	FirstHalfOneMax	OneMax

Table 3 contains the averages and standard deviations over the 100 trials of the total number of time-steps required to maximize each layer's subproblem performance for each of the test plans. The monolithic plan (plan 0) required the least amount of time-steps to transform the randomly generated bit string into the all ones-string, requiring an average of 1,451.11 time-steps. Plans 1 and 2 required the next least amount of time-steps, 2,066.4 and 2,166.45, respectively. The plans with three layers of decomposition, plans 3 and 4, used the most time to generate the optimal bit strings for all of their subproblems, requiring 2,928.55 and 2,937.43 average time-steps to iterate through all of their layers.

A multi-sample comparison using a left-tailed Z-tests with the Bonferroni adjustment for multi-way comparisons confirms that the monolithic test plan outperformed all of the decomposition plans. From the Z-scores in Table 3 and the adjusted α value of 0.01 (pre-adjustment α = .05), each of the null hypotheses stating monolithic time-step averages are greater than or equal to those of plans 1 through 4 were rejected. The monolithic test plan's average number of time-steps to optimize the bit string over the OneMax problem is significantly less than the decomposition-based test plans.

Because each decomposition was dissimilar and found subproblem optimality at different rates, each test plan transitioned to new layers at different time-steps: plan 1 at 754, plan 2 at 832, plan 3 at 821 and 1,584, and plan 4 at 769 and 1,631.

Because a (1+1) EA has a greedy selection mechanism, the monolithic plan has no negative performance changes throughout the learning process. More saliently, it reaches optimality in all subproblem performances the fastest. Plans 1 and 2 reached OneMax optimality the second fastest, but both test plans experienced a slight decrease in their first layer's subproblem performance after a transition to the OneMax layer. The plans with three layers, plans 3 and 4, experienced the largest first layer's subproblem performance decrease of all of the plans. In plan 3's case, the average FirstHalfOneMax performance was optimized, yielding a perfect value of 1 at the end of layer 1. At the completion of layer 2, the same subproblem's average performance plummeted to 0.56, before optimizing again to a subproblem value of 1. Plan 4 had the same outcome, resulting in an average SecondHalfOneMax performance decrease from 1 at the end of the first layer to .59 at the end of the second layer.

Table 3: Average Time-Steps to Find Optimal OneMax Bit String

Test Plan	Time-Steps	STDV	Z-Score
Plan0	1,451.11	456.81	-
Plan1	2,066.4	460.35	-9.49
Plan2	2,166.45	536.29	-10.15
Plan3	2,928.55	574.57	-20.13
Plan4	2,937.43	582.16	-20.09

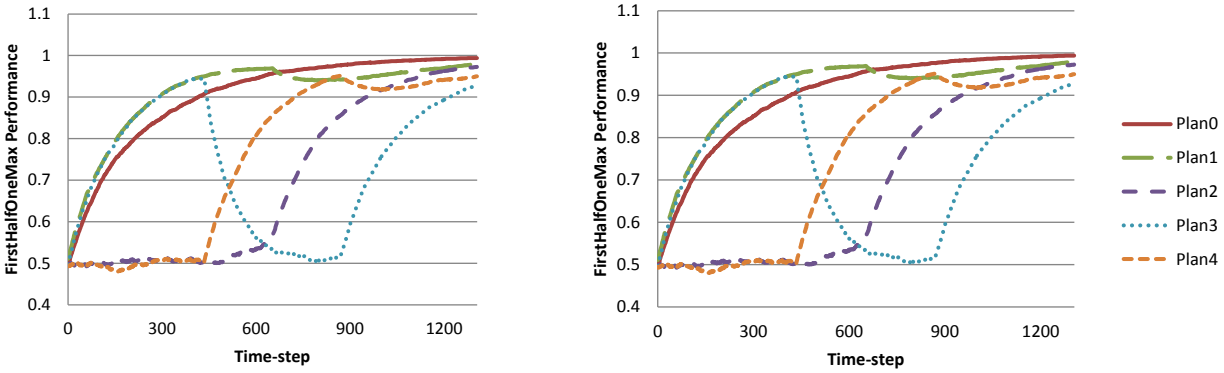
OneMax test case 2 continuously optimized the bit string on a subproblem for a fixed 1,306 time-steps to iterate through their layers. Each test plan's layer was then given the ceiling of the 1,306 divided by the number of layers in that particular test plan.

Table 4 displays the average final OneMax performances, their standard deviations, and Z-scores with the fixed duration halting condition. Contrasting from the first test case, the primary measure is subproblem performance instead of time-steps to find the optimal. Therefore, the closer the performance value is to 1, the more preferable the outcome.

Table 4: Average Final OneMax Performances with Fixed Layer Duration

Test Plan	OneMax Perf.	STDV	Z-Score
Plan0	0.9937	0.0065	-
Plan1	0.9723	0.013	14.73
Plan2	0.9732	0.01354	13.57
Plan3	0.9397	0.01901	26.87
Plan4	0.9382	0.01634	31.51

From the OneMax performance averages, again the monolithic test plan outperforms the others by producing a higher



(a) FirstHalfOneMax Performance

(b) SecondHalfOneMax Performance

Fig. 1: OneMax Subproblem Performance with Fixed Layer Duration

performing bit string at the end of the 1,306 time-steps. Bit strings developed by plans 1 and 2 were the next best performing and test plans 3 and 4 were the worst average performers. The Z-statistics tests confirm that the monolithic test plan significantly outperformed the decomposition approaches. In the Z-test, the claim was that the monolithic test plan's average OneMax performance over the 100 trials was greater than each of the decomposition-based test plans.

Following the first test case, performance of the first layer's subproblem decreases during the second layer in test case 2. In test plans 3 and 4, the layer transition from layer 2 to 3 also resulted in a small second layer subproblem performance decrease even though the first layer's subproblem increased. This observation is interesting because the third layer's subproblem depends on the first two layers' subproblem for optimality and a decrease was not expected. The explanation for the second layer's subproblem decrease is while the bit string is optimizing for the OneMax problem, over 90% of the bits for the second layer's subproblem are already set to 1. At the same time, close to 50% of the bits in the first layer's subproblem are still 0. These two differences in set bits means there was a higher probability that the mutation operator would correctly toggle zero bits affected in the first layer's subproblem simply because there were more zero bits. Combine these probabilities with the guarantee that the (1+1) EA only accepts mutations that progress the current layer's subproblem performance, it is understandable that the second subproblem's performance could decrease while the first subproblem's performance rises during the last layer.

Figures 1 (a) and (b) graph subproblem performances over the 1,306 time-steps. These figures show that when layer 2 is activated in plans 3 and 4, their first learned subproblem's performance suddenly drops then increases when layer 3 is activated.

These experiments demonstrate that in even very simple RL problems where decisions are linearly separable from a performance point of view, layered learning can experience a stability-plasticity imbalance.

4.2 Spin Glass

The test plans in the Spin Glass experiments used the subproblem sequences listed in Table 5.

In the first Spin Glass test case, the monolithic plan 0 required the least number of time-steps to generate the optimal bit string when compared to the two decomposition plans. Plan 0 required an average of 317,719.60 time-steps to satisfy the halting condition of its only layer. Plan 1 required 34,320.81 more time-steps and plan 2 required 15,804.66 more time-steps than plan 0. The monolithic plan's average number of time-steps to generate the optimal bit string was significantly lower than those required by the two decomposition plans. Table 6 displays the average time-steps, their standard deviations, and Z-scores comparing plans 1 and 2's average time-steps to the monolithic plan.

Table 5: Spin Glass Test Plans with Subproblem Sequence

Test Plan #	Layer 1	Layer 2	Layer 3
Plan0	SpinGlass	-	-
Plan1	Middle OneThird SpinGlass	Middle TwoThird SpinGlass	SpinGlass
Plan2	FirstThird SpinGlass	First TwoThird SpinGlass	SpinGlass

The second test case limited the number of time-steps to 285,947 derived from 90% of the first test case's monolithic average time-steps to generate the optimal string. From these trials, the monolithic test plan produced a Spin Glass average performance of 0.995 at the end of the limited layer duration. From the decompositions that segmented the subproblems in thirds, plan 1 averaged a Spin Glass performance of 0.821 and plan 2 averaged 0.989. The Z-tests comparing the monolithic and decomposition-based test plans confirm

that the monolithic approach produced a significantly higher average performance than plans 1 and 2. With a critical value of 1.96, Table 7 displays the average Spin Glass performance after all of the layer iterations have completed, their standard deviations, and Z-scores.

Table 8 displays the average subproblem performances for each test plan at the conclusion of each layer. One noteworthy observation is that the first subproblem of each decomposition-based test plan was optimized although the number of bit string manipulations was limited to 90% of the monolithic test plan's result in the first test case; the plan 0's final performance averaged very close to optimal, 0.995.

From Table 8, subproblem performance always decreased in the decomposition-based test plans after a transition from a subproblem's layer to another layer. For instance, plan 1's transition from layer 1 to layer 2 resulted in the performance of the MiddleOneThirdSpinGlass subproblem to decrease. The subproblem decreases were not as large as those in the OneMax experiment. The small decreases in performance is attributed to the problem decompositions being lower-ordered versions of the Spin Glass problem, where each layer's subproblem directly benefits from positive changes gained from earlier layers.

Table 6: Average Time-steps to Find Optimal Spin Glass Bit String

Test Plan	Time-steps	STDV	Z-Score
Plan0	317,719.69	295,666.7228	-
Plan1	352,040.5	318,712.1596	-0.79
Plan2	333,524.35	323,185.469	-0.36

Table 7: Average Spin Glass Performance with Fixed Layer Duration

Test Plan	SpinGlass Perf.	STDV	Z-Score
Plan0	0.9952	0.0072	-
Plan1	0.8209	0.0078	163.71
Plan2	0.9891	0.0081	5.61

Table 8: Average Subproblem Performances per Layer's Completion

Plan #	Subproblem	Layer 1	Layer 2	Layer 3
Plan0	SpinGlass	0.9952	-	-
Plan1	Middle OneThird SpinGlass	1	0.9843	0.9829
Plan1	Middle TwoThird SpinGlass	0.751	1	0.9933
Plan1	SpinGlass	0.6614	0.8209	0.9891
Plan2	FirstThird SpinGlass	1	0.981	0.9824
Plan2	FirstTwoThird SpinGlass	0.7498	0.9995	0.9936
Plan2	SpinGlass	0.6633	0.8202	0.9891

These experiments demonstrate that the stability-plasticity imbalance created by layered learning application highly non-linear problems may be nuanced and complex.

5. Conclusion

From these experiments, we have demonstrated that the loss of beneficial knowledge can take place in a DBRL system using a simple (1+1) EA on both linear and challenging non-linear problems. Because each test case for both of the tested Boolean-logic experiments always resulted in a decrease in the first optimized subproblem, we conclude that this system favors plasticity, causing an imbalance. Because of this imbalance, two main negative effects were experienced. First, it took longer for the decomposition-based learner to reach optimality compared to our control, the monolithic learner. The reason for the required extended learning duration is because the learner had to spend effort in the final layer to regain beneficial knowledge that was previously lost. The second negative effect was that overall performance suffered. The decomposition-based approach underperformed, averaging problem performances significantly below the monolithic learner.

Both of the negative effects are attributed to the premature removal of beneficial knowledge (in this case, bits), forgotten to optimize on newer subproblems. The attribution was uncovered by the realization that the first subproblem's performance decreased in the second layer for each test cases with three layers. Therefore, we believe it is the DBRL's transitions to new subproblems that is the main culprit for performance loss. In addition, the greedy behavior of the (1+1) EA and the problem decomposition both have played roles in driving out critical performance knowledge prematurely.

The implications of DBRL systems not heeding to this analysis' warnings can lead to the severe performance degradation experienced with neural networks from catastrophic forgetting. We have demonstrated using two simple, non-contrived problems that performance is affected with the premature removal of beneficial knowledge; a more complex system may suffer even more performance sub-optimality if stability and plasticity are not balanced or mitigation methods are not developed and deployed.

6. Acknowledgment

We thank the Florida Education Fund, the McKnight Fellowship, and the Advanced Research Computing Center (ARCC) in the Institute for Simulation and Training at the University of Central Florida. Access to STOKES HPCC has been instrumental to the research performed in this work.

References

[1] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[2] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," in *Advances in Neural Information Processing Systems 10*. MIT Press, 1998, pp. 1043–1049.
[3] T. G. Dietterich, "The maxq method for hierarchical reinforcement learning," in *In Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998, pp. 118–126.
[4] P. Stone and M. Veloso, "Layered learning," in *Machine Learning: ECML 2000 (Proceedings of the Eleventh European Conference on Machine Learning)*, R. L. de Mántaras and E. Plaza, Eds. Barcelona,Catalonia,Spain: Springer Verlag, May/June 2000, pp. 369–381.
[5] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory," 1994.
[6] Y. Jin and B. Sendhoff, "Alleviating catastrophic forgetting via multi-objective learning," in *International Joint Conference on Neural Networks*. IEEE Press, 2006, pp. 6367–6374.
[7] F. M. Richardson and M. S. Thomas, "Critical periods and catastrophic interference effects in the development of self-organizing feature maps," *Developmental Science*, vol. 11, no. 3, pp. 371–389, 2008. [Online]. Available: http://dx.doi.org/10.1111/j.1467-7687.2008.00682.x
[8] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning probleml," *The psychology of learning and motivation*, vol. 24, pp. 109–165, 1989.
[9] S. Mondesire and R. P. Wiegand, "Evolving a non-playable character team with layered learning," in *IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM)*, 2011, pp. 52–59.
[10] ——, "Forgetting classification and measurement for decomposition-based reinforcement learning," in *Proceedings of The 15th International Conference on Artificial Intelligence (ICAI'13)*, 2013.
[11] A. Robins, "Catastrophic forgetting, rehearsal and pseudorehearsal," *Connection Science*, vol. 7, pp. 123–146, 1995.
[12] A. Robins and S. Mccallum, "Catastrophic forgetting and the pseudorehearsal solution in hopfield type networks," *Connection Science*, vol. 10, pp. 121–135, 1998.
[13] M. Hattori, "Avoiding catastrophic forgetting by a dual-network memory model using a chaotic neural network," 2009.
[14] R. M. French, "Pseudo-recurrent connectionist networks: An approach to the "sensitivity-stability" dilemma," *Connection Science*, vol. 9, pp. 353–379, 1997.
[15] P. Stone and M. Veloso, "A layered approach to learning client behaviors in the robocup soccer server," *Applied Artificial Intelligence*, vol. 12, pp. 165–188, 1998. [Online]. Available: http://nn.cs.utexas.edu/pub-view.php?PubID=126580
[16] S. Whiteson, N. Kohl, R. Miikkulainen, and P. Stone, "Evolving keepaway soccer players through task decomposition," *Machine Learning*, vol. 59, no. 1, pp. 5–30, May 2005.
[17] A. Cherubini, F. Giannone, and L. Iocchi, "Robocup 2007: Robot soccer world cup xi," U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, ch. Layered Learning for a Soccer Legged Robot Helped with a 3D Simulator, pp. 385–392.
[18] P. Fidelman and P. Stone, "Layered learning on a physical robot," 2005.
[19] P. A. Borisovsky and A. V. Eremeev, "A study on performance of the (1+1)-evolutionary algorithm," in *FOUNDATIONS OF GENETIC ALGORITHMS, 7*. Morgan Kaufmann, 2003, pp. 271–287.
[20] I. Wegener and C. Witt, "On the behavior of the (1+1) evolutionary algorithm on quadratic pseudo-boolean functions," 2000.
[21] M. Pelikan, M. Pelikan, D. E. Goldberg, and D. E. Goldberg, "Hierarchical boa solves ising spin glasses and maxsat," in *In Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2003), number 2724 in LNCS*. Springer, 2003, pp. 1271–1282.

EVOLMUSIC – A PREFERENCE LEARNING ACCOMPANIST

Shu Zhang
Institute for AI, University of Georgia
261 Arch St.
Athens, GA-30601, US
1-706-207-2237
zhangshu@uga.edu

Roi Ceren
Dept. of CS, University of Georgia
415 Boyd GSRC
Athens, GA-30602, US
1-909-851-3505
ceren@cs.uga.edu

Khaled Rasheed
Institute for AI, University of Georgia
Boyd GSRC, Room 111
Athens, GA-30602, US
1-706-542-0358
khaled@cs.uga.edu

Abstract

This project is an extension of a previous work, "Evac: An Evolutionary Accompanist" (Zhang & Bailey, 2013), in which a computer improvisational accompaniment system, Evac, was built and described. Despite the fact that the EvolMusic and Evac both involve genetic algorithms (GA), they are designed to address different issues and hence possess different characteristics.

The two main differences are as follows. First, Evac uses an implicitly interactive genetic algorithm, which has a hardcoded mathematical fitness function, while EvolMusic allows direct control from the user over the accompaniment, using machine learning techniques to learn a fitness function from the user's preferences. Second, Evac runs in real time, as the accompaniment is generated and played online, while EvolMusic records the user's musical input, generates four different accompaniments for the users to vote on, adjusts the GA's fitness function, and then generates new accompaniments which can be further used to learn the user's preferences.

Keywords
Evolutionary computing, genetic algorithm, computer music accompanist, machine learning, hill climbing;

I. BACKGROUND

The increasing popularity of interactive music systems in computer music have led to extensive research. In his book "Interactive Music Systems – Machine Listening and Composing", Robert Rowe defined interactive computer music systems as computer systems whose output is modified in response to their musical input. In the book "Composing Interactive Music: Techniques and Ideas Using Max", Todd Winkler defined interactive music as "a music composition or improvisation where software interprets a live performance to affect music generated or modified by computers". (Winkler, 2001) The systems affect the output music by altering musical parameters such as pitch, tempo, rhythm, orchestration, and so on. Modeling of human hearing, understanding, and response are involved in interactive systems. (Rowe, 1993)

According to Winkler, there are generally five steps in the interactive process: human performer or musical instruments input, computer listening, software interpretation of the computer listener and detecting useful data for composition, computer composition based on the interpretation from last step, and, finally, sound output by hardware.

In the literature, real-time music accompaniment applications have adopted different approaches (Dannenberg, 1984) ranging from conventional taped accompaniment (one that is unable to synchronize the user's input with the accompaniment the computer plays back) to interactive improvisation involving algorithms, such as evolutionary computing algorithms. Roger B. Dannenberg at Carnegie-Mellon University published a series of papers in which he proposed and developed computer music systems that can listen to a live monophonic performer, align his or her input to a stored

music score, and then play back pre-composed accompaniment simultaneously.

In the paper "An On-Line Algorithm for Real-Time Accompaniment", Dannenberg proposed his computer accompaniment system for the first time. He noted that there are three sub-processes involved: taking input from a live soloist and processing it, aligning the processed input with a stored musical score, and determining the timing signal used to trigger the correct accompaniment from a pre-composed score. Since the live performer is not immune to making mistakes while performing, and the computer's input detection is not perfect, the accompaniment system must maintain a certain level of tolerance in matching the input with the stored score.

Specifically, the way the system was implemented was based on modeling the user's input and the stored score as two sequences (or streams) of events. Then the problem was boiled down to finding the "best" match between these two sequences. There are many ways to seek the 'best' match, and the one Dannenberg adopted was rather intuitive – to find the 'longest common subsequences' of the two sequences (Dannenberg, 1984). The concept of a "virtual clock" was used to generate the timing information to play the accompaniment based on the matching. Two experimental systems were constructed: one with an AGO keyboard for input and the other one with trumpet input.

There are several limitations on these systems to consider. It was only able to adjust tempos on a small scale rather than attempt to adjust tempos in a certain musical manner. Additionally, it was only able to adjust the accompaniment based on the tempo information, ignoring all the other cues such as articulation, loudness and so on. Lastly, it assumed that the input is an ordered set of events, thus it did not allow matching to sets of unordered or simultaneous events. Despite the limitations, the system was able to follow and accompany a live performer and had a desirable level of tolerance for different tempos, extra notes and omitted notes.

To overcome the shortcomings inherent in a mathematical approach to music composing or music understanding, we can adopt machine learning techniques. Machine learning can use multi-dimensional training data and take into account a large number of low-level features such as pitch and tempo, thus building effective classifiers which are able to undertake higher-level music processing.

In the paper "A Machine Learning Approach to Music Style Recognition" (Dannenberg, 1997), three types of machine learning classification techniques, namely naïve Bayesian classification, linear classification, and neural networks, were used to classify an improvisation as certain music styles. (Rubine, 1991) For each type of classification, two sets of experiments were carried out. The first experiment was to classify the piece of music into four styles: "Frantic", "lyrical", "syncopated", or "pointillistic". In the second sets of experiments, four additional musical styles were added for consideration: "quote", "blue", "low", and "high". Training data were labeled for use in supervised training, and 13 lower-level features were identified based on the MIDI data. Cross-validation was used for the training. Confidence measures were used to reduce false positives.

Results showed that all three types of classifiers can classify music with a high accuracy rate, ranging from 77% to 99%. The experiment with four target features showed higher accuracy rate. Neural networks took hours to train while the other two types of classifiers took minutes (Bishop, 1995). The technique developed in the paper was believed to have applications in the field of music composition, understanding and performance.

Evaluation is another important step to consider in computer music generation, though it was omitted for EvolMusic. In the paper "Critical Issues in Evaluating Freely Improvising Interactive Music Systems" (Linson 2012), several approaches to evaluating interactive computer music systems were reviewed, with

their strengths and weaknesses analyzed and compared.

Specifically, it discussed the advantages and disadvantages of computer based quantitative evaluation (Pressing, 1987), and also concluded that human expertise, which conducts qualitative evaluation, is more appropriate and, in some circumstances, indispensable. (Pearce and Wiggins, 2001) The paper pointed out that quantitative evaluation has advantages in evaluating music in which the melody and/or the rhythms are constrained by a set of 'well-defined style-based rules'.

On the other hand, for music like freely improvised music, the quantitative evaluation loses its advantages because freely improvised music differs in evaluation criteria (Lehmann and Kopiez, 2010). In other words, definable musical rules are not enough to describe or to judge freely improvised music because of the spontaneous characteristics and real-time negotiating interactions of this music style. (MacDonald, et al, 2011) Other shortcomings of quantitative analysis include its inability to decide which musical features from a performance have the most significance in the analysis process. Also, it is not good at detecting large scale structures within a music piece, nor does it take into account the listening context or the subjectivity of the audience (Clarke, 2005).

In contrast, human experts don't have as many inherent problems in these aspects (Smith, 1997), although it is conventionally thought to lack scientific rigor. For example, human experts are able to avoid measuring merely the compliance to musical rules.

That said, the paper still noted that no evaluation methods should be used to replace the others for all the music systems, and it is important to find an appropriate and flexible evaluation method for a specific music system. (Stowell, 2009)\ For example we can collect listeners' opinions through a survey if we have a population of listeners available; or we can focus on the underlying software to find more results in answer to our research question.s (Collins, 2008) In conclusion, the author notes that there has been no "well-established evaluation method that is widely recognized in the literature" for interactive computer music systems.

For this paper, no formalized evaluation system was built due to the limitation of time and resources. In spite of that, one can keep playing and listening to the responsive real-time accompaniment, maybe in different styles, and have some idea of the quality of the resulting music pieces.

II. USER INTERFACE OF EVOLMUSIC

The user interface of EvolMusic is simple and clear. It has nine buttons in total and the layout is intuitive: one bigger button to let the user input music, four buttons assigned to play each of the four generated accompaniments, and four other buttons to let the user vote for the respective accompaniments. On startup, the interface shows the greeting words at the message area, the top right of window, and only the button for inputting music is clickable, with all the other eight buttons greyed out, as shown in Fig. 1.

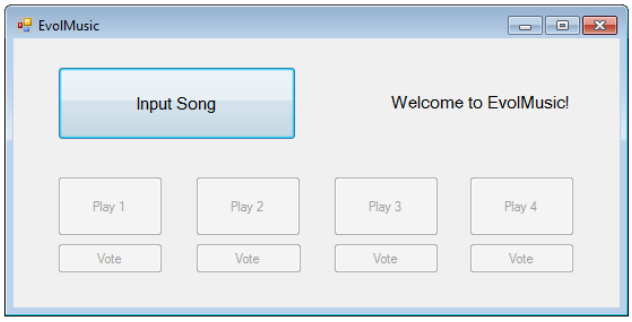


Figure 1: Startup window of EvolMusic

Once the user starts inputting music by typing on the keyboard, EvolMusic starts counting notes the user has input, and correspondingly the number of notes left for the user to input is shown in the message area of the window. At this stage, all the buttons on the window are greyed out, as shown in Fig. 2, to suggest the user to complete the music input before any further action.

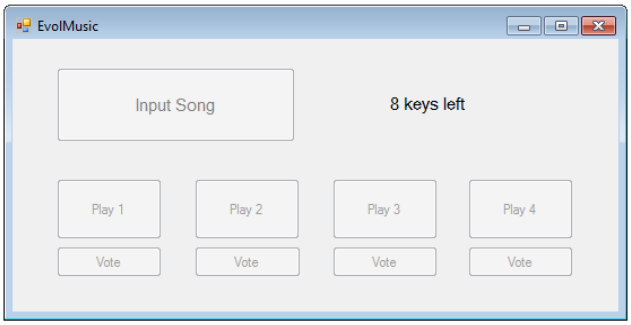


Figure 2: EvolMusic interface when user inputs music

After the user has played 12 notes, the window shows the message "Generating songs!" (as shown in figure 3) while EvolMusic works on generating four pieces of accompaniment. All the buttons are still greyed out at this point.

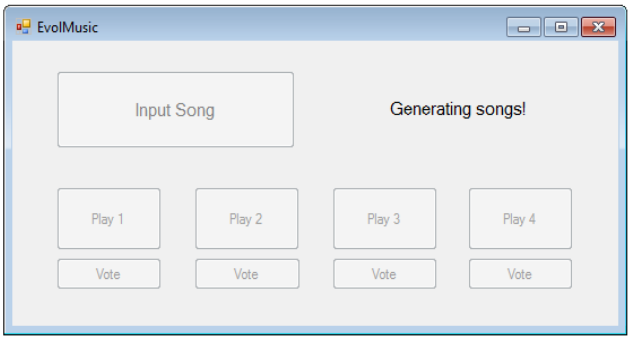


Figure 3: EvolMusic interface when new accompaniments being evolved

Once EvolMusic has completed generating all accompaniments, all the buttons become clickable to indicate the accompaniments, along with recorded user's input music, are ready to play and be voted on (Fig. 4).

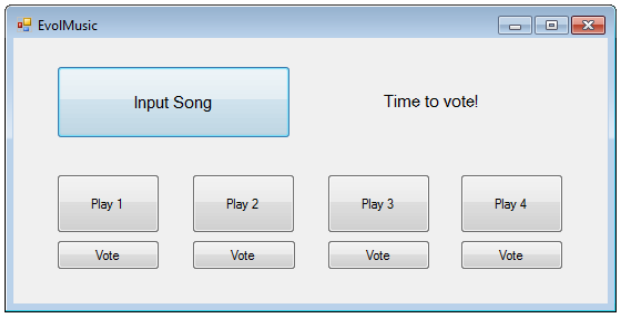


Figure 4: EvolMusic interface when it's ready to let user play the generated accompaniments and vote

V. SYSTEM DESIGN

EvolMusic was developed based on Evac (Shu & Bailey, 2013), and the main frame of the GA remains unchanged. The main part that has been changed is the fitness function. In the following two sections, first we will discuss in short about the overall configuration for the GA being used here (for more detail descriptions please refer to the paper "Evac: An Evolutionary Accompanist"), and will follow with more details about the hill climbing technique used in fitness function learning.

III. Genetic Algorithms

As with Evac, the GA individual representation used in EvolMusic was an array of 16 single notes. Each single note was an 11-bit integer encoding of several values (Fig. 5). Tournament selection was used as the parent selection strategy. A generational survival strategy with elitism was used. Uniform crossover with 90% probability and uniform mutation with mutation rate of 20% per gene were used. The population size was chosen to be 100, which was small enough to prevent excessive computing time, and large enough to maintain a certain level of diversity in the GA individuals.

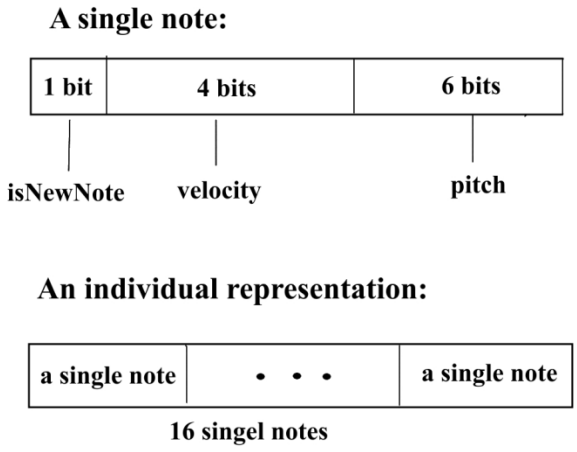


Figure 5: Representation for a single music note and an individual

VI. Fitness Function Learning

In Evac, the fitness of an individual in the GA was achieved by transforming a similarity function. The similarity function was computed by comparing each individual against the user's input for the previous 16 single notes. Depending on the harmonic "distance", a mathematical value was assigned to each of the 12 different musical intervals that Evac allows the user to input. The similarity value of an individual was the sum of the similarity values calculated for each of the 16 single notes within that individual, and the fitness value for that individual was determined by passing its similarity value through a quadratic function. The quadratic function was set so that individuals with excessively low or high similarities received lower fitness values, whereas those with mid-to-high range similarity values received higher fitness values.

However, with EvolMusic, the quadratic weighting function was omitted, and the "similarity pairing values" were adjustable based on human evaluation (user voting) instead of being hardcoded. It should be pointed out that since the similarity pairing values directly reflected the user's preference, the sum of the similarity paring values does not need to be passed through any weighting function; rather, the sum of the similarity pairing values can serve directly as fitness values for an individual. To reduce the size of the search space for the fitness function, instead of dealing with similarity values for all 12 intervals, the intervals were hand-grouped into four sets: high, medium-high, medium-low, and low similarity (as shown in table 1).

Table 1: Interval categories

Interval : Musical	Similarity value
0 : Unison	High
1 : Minor second	Low
2 : Major second	Low
3 : Minor third	Medium-high
4 : Major third	Medium-high
5 : Perfect fourth	High
6 : Tritone	Low
7 : Perfect fifth	High
8 : Minor sixth	Medium-high
9 : Major sixth	Medium-high
10 : Minor seventh	Medium-low
11 : Major seventh	Medium-low
12 : Octave	High

During the hill climbing technique to learn similarity values, there were four alternate model manipulations: higher similarity values between notes, lower similarity values between notes, increased relative distance between notes, and similarity values between notes that were unchanged. Specifically, the way the first three models tune the similarity values are shown in table 2. γ was set to 0.9 originally, and tuned over time, to accumulatively slow down the manipulation speed. The similarity values were controlled so that they were always between 0 and 1. As can been seen from the mathematical functions "Model_up" and "Model_down" do not adjust similarity values linearly – the size of the adjustments gets smaller as the similarity values grow (Model_up) or shrink (Model_down). While Model_up and Model_down "squash" the similarity values, the third model spreads all the similarity values, where mind in the equation was either the difference between 1 and the maximum similarity value, or the minimum similarity value, depending on which one is smaller.

Table 2: Implementation of the three model manipulations

	Implementation functions
Model_up	$similarityValue_k = similarityValue_k + \gamma * 0.5 * (1 - similarityValue_k)$
Model_down	$similarityValue_k = similarityValue_k - \gamma * 0.5 * similarityValue_k$
Model_spread	$similarityValue_k = 0.5 * similarityValue_k \pm 0.5 * \gamma * min_d * (\frac{\lvert similarityValue_k - \overline{similarityValue}\rvert}{similarityValue_{max} - \overline{similarityValue}})$

VII. RESULTS

First of all, the UI design of EvolMusic made it very handy and intuitive to use. Second, the choice of input 12 notes at a time was successful – it allowed the user to input music that was long enough for a human being to sense and judge, and short enough to avoid human fatigue of listening and evaluating, as well as to prevent memory loss when the user compares and votes. Third, the computing time of generating new accompaniments were not unbearably long – it generally took less than 3 seconds.

More important, EvolMusic achieved its main goal successfully – to be able to generate different accompaniments and learn human preferences based on user feedback. The four accompaniments were all reasonably good – which was to some extent a merit inherited from Evac. The user was able to sense the difference between these accompaniments and make a choice, at least with multiple trials of listening and comparing, which was well supported by EvolMusic.

However, the fact that there were only four models of adjusting similarity values made the learning a bit crude. There were two reasons behind using only four models: first, more manipulating models means more computing resources will be consumed, and second, too many accompaniment choices will add more burden and confusion to the user, since he or she would have to listen to and compare all the alternatives.

VIII. CONCLUSION

Evac is excellent for what it is - a first approximation that demonstrates the power of implicit interactivity. Evac demonstrates satisfying performance on musical improvisation with the user. Evac's ability to run in real time allows the user to experience the same kind of exploration that happens in real life improvisation scenarios with other musicians.

Evac is able to follow the music that the user plays, but Evac is more than a simple reharmonizer. It never copies the user, nor does it repeat its own previous melody. When the user stops playing (while keeping the program running), Evac will also slow down gradually, but it will never completely stop playing – it will play few notes every now and then, as if it is asking and waiting for the user to respond. Once the user starts playing again, Evac will re-start the cooperation with the user, with very short amount of time needed at the beginning to adapt to the user's music style.

IX. FUTURE WORK

Despite the positive results of EvolMusic, there are a few things that might be improved. First of all, EvolMusic was not able to incorporate the Evac's real time performance. After the user votes for his or her favorite accompaniment, and EvolMusic digests that information and generates new accompaniments, the user would have to repeat the same procedure again – listening to four alternative accompaniments and voting. In other words, EvolMusic was only able to learn the user's preference, but not to put that knowledge to use in accompaniment. This problem will

have to be solved if we want EvolMusic to be practically useful.

Second, though with careful listening, a user can find the differences between the alternative accompaniments, but making the differences more obvious would dramatically improve the selection process.

Third, as with Evac, EvolMusic is a prototype that serves to convey the author's initial idea. As a result, these two pieces of software are not to their highest potential. Many details need to be improved in order to make them more successful. For example, a button to tune the volume of the accompaniment relative to the user's input music seems necessary for the software UI, and permitting the user to choose the number of notes in the input, as well as more timbre choices, is desirable. Finally, more than one accompaniment instrument should be considered to enhance the performance of the software.

REFERENCES

[1] Roger B. Dannenberg 1984. An On-Line Algorithm for Real-Time Accompaniment. International Computer Conference, 193-198

[2] Roger B. Dannenberg, Thom, Belinda, & Watson, David 1997. A Machine Learning Approach to Musical Style Recognition. Computer Science Department. Paper 504 http://repository.cmu.edu/compsci/504

[3] Bishop, C. M. 1995. Neural Networks for Pattern Recognition. Clarendon Press.

[4] Rowe, R , 1993. Interactive Music Systems. MIT Press.

[5] Rubine, D. 1991. The automatic recognition of gestures. Tech. rep., School of Computer Science, Carnegie Mellon University

[6] Adam Linson, Chris Dobbyn, & Robin Laney, 2012. Critical Issues in Evaluating Freely Improvising Interactive Music Systems. International Conference on Computational Creativity.

[7] Clarke, E. 2005. Creativity in Performance. Musicae Scientiae 9:1, 157-182

[8] Clarke, E. 2005. Ways of Listening: An Ecological Approach to the Perception of Musical Meaning. Oxford University Press

[9] Collin, N. 2008. The Analysis of Generative Music Programs. Organised Sound, No. 3: 237-248

[10] Lehmann, A., R. Kopiez, 2010.The Difficulty of Discerning between Composed and Improvised Music. Musicae Scientiae 14: 113-129

[11] MacDonald, R., G. Wilson, & D. Miell. 2011. Improvisation as a Creative Process within Contemporary Music. Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance and Perception, D. Hargeaves, D. Miell, and R. Macdonald, Eds. Oxford University Press

[12] Pressing, J. 1987 The Micro- and Macrostructural Design of Improvised Music. Music Perception 5(2): 133-172

[13] Pearce, M. T., & G. A. Wiggins, 2001. Towards a Framework for the Evaluation of Machine Compositions. Proc. of the AISB'01 Sym. On Artificial Intelligence and Creativity in the Arts and Sciences. Brighton: 22-32

[14] Stowell, D., A. Robertson, N. Bryan-Kinns, &M. D. Plumbley. 2009. Evaluation of Live Human Computer Music Making: Quantitative and Qualitative Approaches. Int. J. of Human Computer Studies 67, No. 11: 960-975.

[15] Smith, H. &R. T. Dean. 1997. Improvisation Hypermedia and the Arts since 1945. Routledge.

[16] Shu Zhang & C. Thomas Bailey. 2013 Evac: An Evolutionary Accompanist

Evolution of event and delay controlled neuronal network for locomotion

Maria Shirshova[1] and Mikhail Burtsev[1]
[1]National Research Center "Kurchatov Institute", 1 Kurchatov sqr., Moscow, Russia

Abstract - *Fine neurocontrol of complex dynamical systems such as locomotion of robot requires feedback for the estimation of deviations from the target behavior. Artificial neural networks is a powerful tool for locomotion control but traditional architectures lack explicit mechanisms for the evaluation of actions results. To address this issue we propose a novel model of formal neuron capable of simple assessment of the timing as well as results of its' own activity. This event and delay controlled (ECD) model of neuron was used to evolve neurocontrollers for robot locomotion. Suggested ECD-model had significantly better performance in our simulations compared to traditional neural networks architectures.*

Keywords: Artificial neural networks, neurocontrol, goal estimation, neuroevolution, locomotion

1 Introduction

The problem of robot locomotion control can be considered as a task of steering a complex dynamical system. In a general case, solution of this problem requires definition of a function that maps states of the system and environment to control signals. It is well known that artificial neural networks are able to extract essential features from input values and perform non-linear function approximation. These properties make different architectures of neural networks popular for control of dynamical systems [1,2].

Majority of neurocontroller applications in robotics are strongly tired to the specifics of the problem to be solved either by constraining the network topology or tuning the learning rule or the both. The problem of robot locomotion control is a good illustration of the current situation in the field. A range of analytical and heuristic solutions have been developed for the task of movement patterns generation. The oldest approach is an analytical one that utilizes an understanding of basic properties of the motion constrained by an anatomy of the robot. One of the examples here is the Zero Moment Point method for humanoid robots, where the motion is planned with taking into account an equilibrium position of the whole body [3].

Another set of approaches relies on the networks of connected oscillators defined by the systems of differential equations, for example Hopf oscillator [4] or spiking neurons [5], where parameters of equations are adjusted with supervised learning techniques or with evolutionary algorithms. Controllers of this type show good continuous and periodic dynamics, thanks to the dynamical properties of oscillators taken as a basis for the system. Similar to these methods is an approach where fully-connected neural network of leaky-integrator neurons with fixed topology is tuned by evolution to generate smooth periodic dynamics for robot joints [6,7].

The other area of neurocontrol research is focused on the development of controllers that integrate information about the current state of the robot, sensory values and memory. Here a variety of predefined neural network architectures with fixed topology and neuron types are trained with different techniques of supervised learning or optimized with an evolutionary algorithm. In the work [8] Jordan's sequential network, which is capable of learning sequences of patterns under supervised learning, was modified and used to generate several bipedal trajectories. Fixed topology network of spiking neurons was tuned with an evolutionary algorithm and demonstrated the capability to control a stable biped walking in [9]. Berns et al. [10] presented the modular, hierarchical control architecture for locomotion where single components were the neural networks trained on the set of samples for different functions such as "leg control", "leg coordination" etc. Associative memory in CMAC neural networks was able to control the walk of a biped robot after the supervised learning [11].

Another set of methods is related to the algorithms able to grow a neural network topology. These methods are in general more flexible and less tied to the particular control task. In [12] some modification of the popular neuroevolutionary algorithm NEAT was successfully applied for the evolution of controllers for 3D biped locomotion. The NEATfields with the multi-level network structure was suggested in [13]. Here the network consists of so called fields at the highest level and each field consists of elements that are neural networks of arbitrary topology. The high level topology as well as elementary networks evolves according to the modified NEAT algorithm.

Control of locomotion is a goal-directed behavior. As a result controller should be able to estimate mismatch between the current and the goal states to adapt control signal. Usually in application of neural networks to the task of robot locomotion control the goal-directedness is not addressed explicitly. Formally, the problem of mismatch estimation can be solved by standard recurrent neural networks but there are no efficient methods or general micro-circuit architectures to do this. To make progress in this direction we suggest minimal modifications of the formal neuron model to incorporate dynamic feedback. As a proof of the principle we

studied performance of our model in the task of robot locomotion.

2 The model

2.1 Background

Minimal goal-directed behavior consists of recognition of context or problem state followed by an action to solve the problem and subsequent estimation of discrepancy between resulting and goal states to make decision about continuing or terminating the action. This scheme of behavior organization is known as Test – Operate – Test – Exit (T.O.T.E.) principle in cognitive sciences [14] and as "afferent synthesis" – "program of actions" – "acceptor of actions' results" in the neurophysiological theory of Functional Systems [15,16]. In the field of neurorobotics similar logic is developing in the framework of the Dynamic Field Theory for organization of sequential behaviors [17].

In this paper we propose to implement behavioral process described above at the level of individual neuron. As a result we expect that the network consisting of such neurons should demonstrate capability of distributed feedback control and might be easier to apply for the control of dynamical systems.

In our model the neuron has the following functionality:

- recognition of the "problem" (context) input and subsequent activation of the output (i.e. execution of action);
- recognition of the "target" input and subsequent deactivation of the output (i.e. termination of action);
- timing the period of action execution and deactivation of the output if the period exceeds expected time for the action completion.

This functionality can be implemented by the micro-circuit of three neurons (fig. 1). The first "problem" neuron detects situations when the action should be executed. The second "action" neuron is activated by the "problem" neuron and has two self-referent collateral connections. The first collateral connection is positive and sufficient for sustained activity of the "action" neuron once it was kick-started by the "problem" neuron. The second collateral connection is negative and delayed by the time expected for the successful action completion. Thus the negative feedback shuts down the "action" neuron if it is active more than usual. The third "goal" neuron detects situations when the action succeeded, i.e. the target state is present. There is negative connection from "goal" to "action" neuron able to turn off the latter. Thus the micro-circuit has two set of inputs – activating ("problem") and deactivating ("goal"), and one output ("action"). To simplify design of the network we consider this micro-circuit as an elementary unit and call it *event and delay controlled* (EDC) neuron.

EDC-neuron can also be seen as an information processing element that encodes information about occurrence of events on its' inputs. If EDC-neuron has activation weights set up for the recognition of an event *A* and deactivation weights for an event *B* as well as some delay τ then the output activity of the neuron will indicate that the transition *A*→*B* might be in a progress. In the case if the delay component of the model is omitted ($\tau = \infty$) then the resulting event controlled (EC) neuron will generate information about transition *A*→*B* without regard to its duration. And finally if the event component is absent (no deactivating weights) then the DC-neuron will represent a memory that an event *A* has been occurred no longer than period τ ago. In our simulations we studied all mentioned above modifications (EC, DC and EDC) of our model.

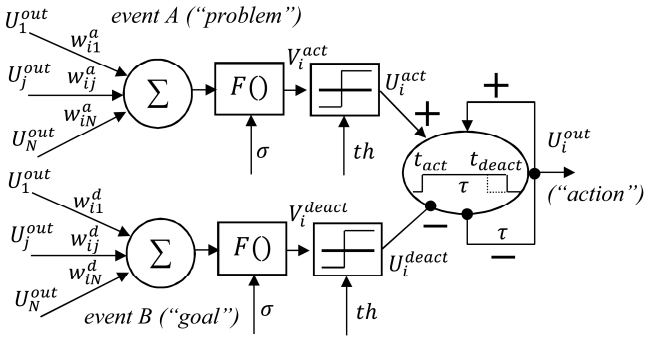


Figure 1: Diagram of the EDC-neuron (look for the details in the text).

2.2 EDC-neuron

Now we define the EDC-neuron more formally.

The value of the activating or deactivating signal of the i^{th} EDC-neuron is determined by the nonlinear activation function:

$$V_i = F\left(\sum_{j \in I_i} w_{ij} U_j^{out}\right) + \sigma, \quad (1)$$

where $F(x) = \frac{1}{1+e^{-k(x-x_0)}}$ is a nonlinear function, σ is a random value, U_j^{out} is an activity of the j^{th} neuron and w_{ij} is a strength of connection between i^{th} and j^{th} neurons. Then a threshold is applied to the value of activating or deactivating signal:

$$U_i = \begin{cases} V_i, & V_i > th \\ 0, & V_i \le th \end{cases}. \quad (2)$$

The resulting output activity of EDC-neuron is defined by the formula:

$$U_i^{out}(t) = \\ = \begin{cases} U_i^{act}\ (t_{act}), & (t_{act} \le t < t_{deact}) \wedge (t - t_{act} \le \tau) \\ 0, & (U_i^{act} = 0) \vee (t - t_{act} \ge \tau) \end{cases}, \quad (3)$$

where t_{act} and t_{deact} are the moments when the activating U_i^{act} and deactivating U_i^{deact} signals exceed the threshold,

and τ is a delay for the self-deactivation. This formula means that EDC-neuron is active with the output value equal to the activating signal between the activating and deactivating events.

Given the full model of the EDC-neuron DC and EC neurons can be defined by "removing" of deactivating connections and setting the delay period to infinity ($\tau = \infty$) respectively. So, in the case of EC-neuron the output is:

$$U_i^{out}(t) = \begin{cases} U_i^{act} \ (t_{act}), & t_{act} \le t < t_{deact} \\ 0, & U_i^{act} = 0 \end{cases}. \quad (4)$$

And in the case of DC-neuron the output is:

$$U_i^{out}(t) = \begin{cases} U_i^{act} \ (t_{act}), & t - t_{act} \le \tau \\ 0, & (U_i^{act} = 0) \vee (t - t_{act} \ge \tau) \end{cases}. \quad (5)$$

2.3 The Robot

For our experiments we used the model of humanoid robot ASTI (fig. 2) in the robotic simulator V-REP [18]. ASTI has 18 joints with each joint controlled by two neurons of the output layer of the network. The sign of the difference between output values of these neurons determines the direction of a joint rotation. On each cycle of calculation the value of the joint angle increases or decreases on the fixed value which is equal to the one tenth of the full range of variation of the angle. Input neurons of the network receive the information only about values of current joints angles, because we assumed that information about dynamics of the robot can be calculated by the network itself. The duration of the robot «life» in the virtual environment was equal to the 20 seconds.

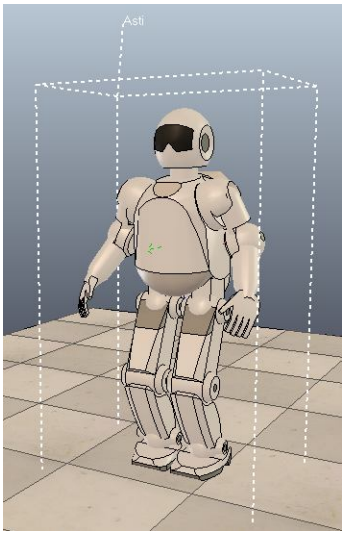


Figure 2: Simulated robot Asti [19].

2.4 Evolutionary algorithm

The neurocontroller for robot was optimized by the evolutionary algorithm [20] designed for the evolution of network topology by duplication of neurons together with attached connections (Fig. 3). In this algorithm after duplication of the neuron its' input connections retain values of weights but the weights of the output connections are divided in half for the both parent and descendant neurons. Thus, postsynaptic neurons receive the same value of signal, as they received before the duplication. So right after the duplication, duplicated and descendant neurons implement together the same function as the function implemented before by the parent neuron only, but in subsequent evolution they can evolve independently. To balance the growth of the search space we also added *neuron deletion* mutation that removes neuron with all its output and input connections. For more precise tuning of connectivity *delete connection* and *add connection* mutation operators were implemented.

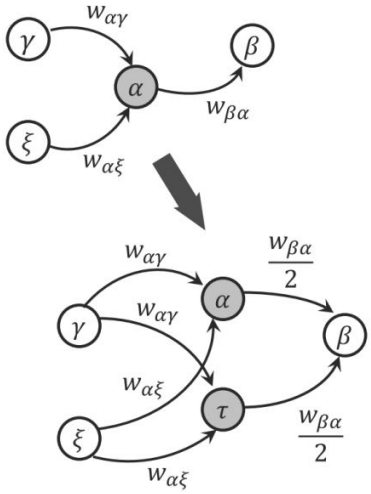


Figure 3: Schematic illustration of the *neuron duplication* mutation operator. The child neuron τ inherits all synaptic weights from the parent neuron α. $w_{\alpha\beta}$ denotes synaptic weight of connection between neurons α and β. (Figure from [20])

The selection of parents for the next generation was decided by the fitness proportional selection, also known as roulette wheel selection, where the probability of each individual in population to become a parent is proportional to the ratio of fitness of this individual to the total fitness of all agents in the population. We also used elitist strategy for selection when some part of the bests agents in the population remain in the next generation without any change. This combination of selection methods showed the best results in pilot experiments.

Initial structure of the neural network consist of three layers: an input layer containing neurons receiving information about environment; an intermediate layer with the same number of neurons as in the input layer; and an output layer which produces signals for the control of robot locomotion. Each neuron from the intermediate layer has

connections to all neurons from the input layer, and respectively each neuron from the output layer has connections to all neurons from the intermediate layer. The initial values of the network weights were set in the range of zero and one.

Fitness of the neurocontroller was defined as the distance covered by the robot for the fixed period of time.

3 Results

A series of simulation runs were performed to estimate performance for the different modifications of the EDC-neuron model: with a delay (DC-neuron) or an event (EC-neuron) control only, and a full model.

Parameters of the evolutionary algorithm were the same for all simulations (see Appendix).

To select the baseline for the EDC-neuron model performance we have studied different predefined architectures of neurocontrollers for the initial network. The first was just common network based on standard formal neurons (due to evolving topology recurrent connections were possible). The second network had mutual inhibition between neurons in the pairs controlling joints. In the third architecture each neuron of the output layer had self-referent connection. We expected that these modifications of the basic network architecture could provide smoother changes of the angles of the robot joints and thus better opportunities for the robot locomotion. As these architectures were planned as a baseline for the further research we were interested in an average expected value of the resulting fitness. The comparison of the baseline network architectures is shown on the Figure 4. We found no significant difference between considered modifications, and for these reasons further experiments were compared with the basic (no inhibitory or self-referent connections in the output layer) three-layer network architecture.

Fitness evaluation in our simulation was rather expensive computationally thus typical run for 3000 generations required about twenty-one hours. This forced us to limit the number of evolutionary runs for each parameter settings to five.

In a series of evolutionary runs for the same variant of the model some runs usually demonstrated very low fitness (Fig. 5), in fact there were no solution found at all. Best neurocontrollers evolved in these runs possessed no stable locomotion and produced discontinuous movements. On the other hand for the every set of parameters there were some runs that demonstrated continuous locomotion and significant displacement of the robot from the starting position. In these cases evolution was able to solve the task with more or less success.

Given this separation of low and high fitness runs it was decided to use two measures for comparison of results for the different modifications of the neurocontroller: 1) averaging of the best fitness in the final population over all runs for the given model variant and 2) averaging over runs with final best fitness higher than the average fitness over all runs for the given model variant. It is reasonable to assume, that the second measure better reflects the potential of the model modification to produce effective controllers and it is more suited for the estimation of the network elements role.

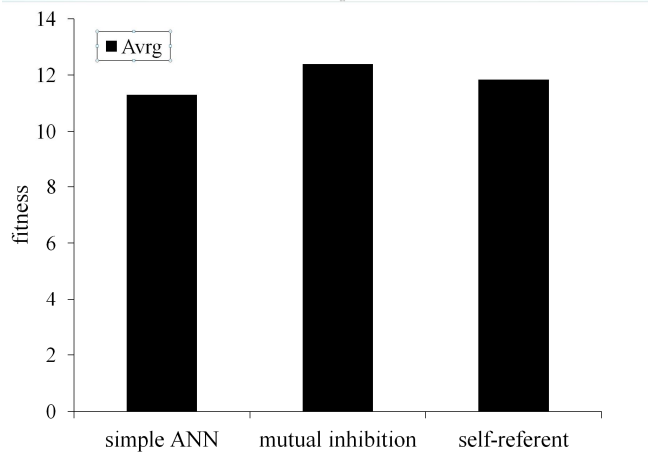


Figure 4: Comparison of the average best fitness (averaging over five runs) for different initial architectures of the networks of formal neurons: 1) simple ANN; 2) with mutual inhibition between neurons in the pairs controlling joints; 3) with self-referent connection for each neuron of the output layer.

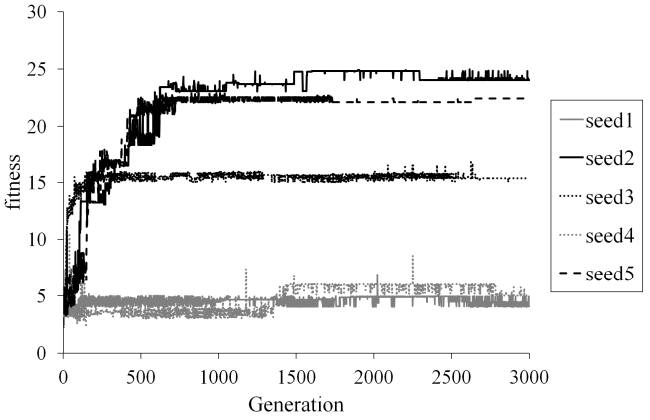


Figure 5: Evolutionary dynamics of the best fitness for the series of runs of the DC-neuron model with mutating delays, which illustrates the typical behavior of the fitness.

The next series of experiments was performed to study delay controlled (DC) neurons with the several variants of delay setting:

1) the same delay for all network elements ($\tau = 160$ ms) and no mutation of delays during evolution;

2) the normal distribution of delays with the average $\tau = 160$ ms and variance $D_\tau = 8$ ms for the neurons of controllers in initial population and no mutation of delays during evolution;

3) the normal distribution of delays with the average $\tau = 160$ ms and variance $D_\tau = 8$ ms for the neurons of controllers in initial population and mutation with the

probability $P_\tau = 0.1$ per neuron with the amplitude drawn from the normal distribution with a mean $\tau = 0$ ms and a variance $D_\tau = 8$ ms.

The purpose of these experiments was to study the effect of diversity and possibilities to adapt delay periods in DC-neuron model on the best fitness. Results for these experiments are shown on the summary figure 6. It makes sense to predict that for the delay controlled neurons the final fitness should increase with a number of options to adapt delay values. However, contrary to our expectation the best performance was demonstrated by the network of DC-neurons with initially the same and not mutating memory periods (fig. 6). On the other hand, maximal average best fitness was achieved in the evolution of the network with initially random but not mutating delay periods (fig. 6).

It should be noted that the difference in an average best fitness between all and high performance runs is smaller for the DC-neurons with random none mutating delays compared to the modification with equal none mutating delays. So in the first case variation of the evolved performance is lower than in the second. The network of DC-neurons with the initially random and mutating neurons demonstrated results similar to the network with random but none mutating delays.

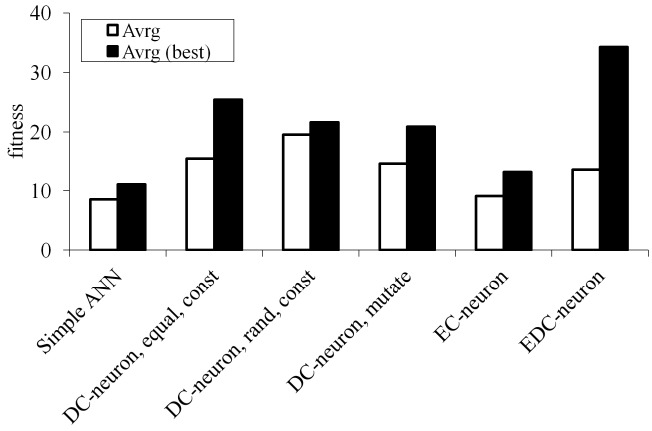


Figure 6: Comparison of the average of best fitness over all runs (white bars) and the average of the best fitness over successful runs (filled bars) for all types of networks: 1) simple ANN; 2) DC-neurons with the same and not mutating delays; 3) DC-neurons with the initial normally distributed and not mutating delays; 4) DC-neurons with mutating delays; 5) EC-neurons; 6) EDC-neurons.

Finally we simulated evolution for the networks consisting of event controlled (EC-neurons) and combined event and delay controlled (EDC-neurons) neurons. As one can see from the figure 6 the networks consisted of EDC-neurons have demonstrated the significant advantage over networks of EC-neurons. Moreover network of EDC-neurons produced the best controller over all simulations performed.

In our study the topology of neural networks is evolvable and, hence, the network size can change during evolution. Average number of connections in the final generation was similar for simple and DC-neuron networks with values about 1000 (fig. 7A). Runs with EC-neurons resulted in about 2000 and with EDC-neurons in about 5500 connections. Addition of deactivating weights to the EC-neuron model easily explains twofold increase in the number of evolved connections. Number of weights for the EDC-model after correction for the addition of deactivating connections is still on the average 2 times higher than observed for the other models. Further analysis demonstrated that fitness of the best agent in the last generation for all modification of the model except EDC-model is not correlated with the number of connections but for the EDC-model there is positive correlation (fig. 7B).

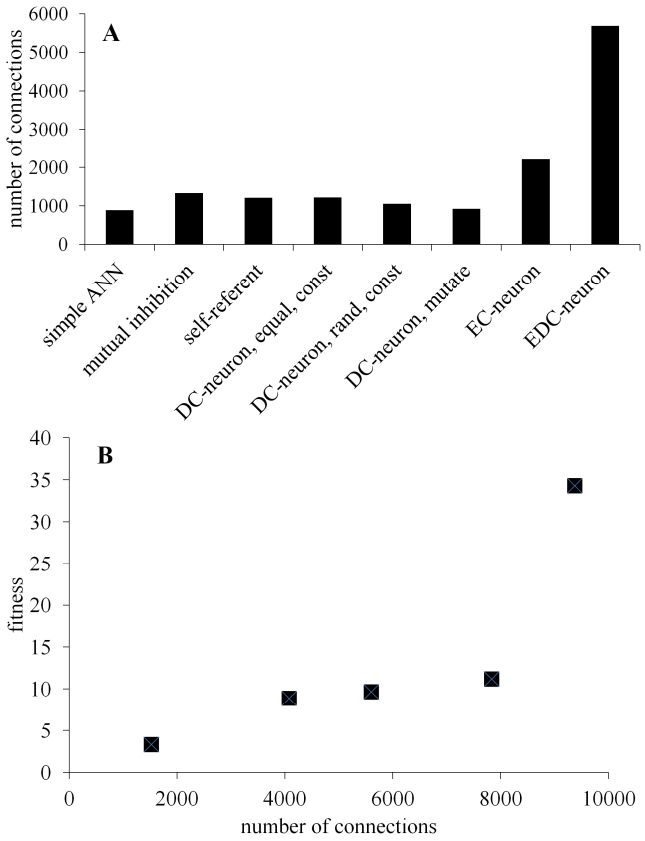


Figure 7: (A) The average number of connections (over five runs for each variant of model) at the end of the evolution. (B) Resulting best fitness and the number of connections for the EDC-neuron networks. Five points corresponds to the five runs with the different initial values of random number generator.

4 Conclusions

To address the issue of appropriate feedback in neurocontrollers we propose an original model of the formal neuron. The output of the neuron is activated in the problem context and this activity persists until the target input observed or expected duration of action exceeded. We suggest that a network of such event and delay controlled neurons (EDC-neurons) should be more suitable for neurocontrollers then traditional architectures due to its'

ability of distributed estimation of the success of goal-directed behavior.

Performance of neurocontrollers with the full EDC-model and a number of its' modifications was studied in the task of locomotion control.

Comparing the resulting fitness for all types of the networks in our study we can conclude that the different variants of the networks consisted of the delay controlled DC-neurons have demonstrated significantly better results than the network consisted of simple formal neurons as for averaging over all or over successful runs only. The networks consisted of the event controlled EC-neurons have demonstrated only the slight advantage over the networks of simple neurons, that is probably related to the fact that addition of the deactivating connections increases the search space and the complexity of the fitness landscape. The EDC-model evolved the best performance thus confirming our modelling assumptions and solid perspectives for the further development and applications.

5 Appendix

All simulations were performed with the following parameters of the evolutionary algorithm:

- number of generations $N_{gen} = 3000$,
- population size $N_{desc} = 100$,
- fraction of the ancestor population retained in the next generation $S = 0.5$.
- probability of a neuron duplication $P_{dup} = 0.005$ (for each neuron),
- probability of a neuron deletion $P_{del} = 0.005$ (for each neuron),
- probability of a connection addition $P_{add} = 0.005$ (for each connection),
- probability of a connection deletion $P_{rem} = 0.005$ (for each connection),
- probability of weights mutation $P_w = 0.1$,
- probability of a delay mutation $P_\tau = 0.1$,
- variance of weights mutation $D_w = 0.1$,
- variance of a delay mutation $D_\tau = 8\,ms$ (one time-step).

6 References

[1] Omatu, Sigeru, Marzuki Khalid, and Rubiyah Yusof. *Neuro-control and its applications*. London: Springer, 1996.

[2] Siciliano, Bruno, and Oussama Khatib, eds. *Springer handbook of robotics*. Springer, 2008.

[3] Wright, Jonathan, and Ivan Jordanov. "Intelligent approaches in locomotion."*Neural Networks (IJCNN), The 2012 International Joint Conference on*. IEEE, 2012

[4] Righetti, Ludovic, and Auke Jan Ijspeert. "Programmable central pattern generators: an application to biped locomotion control." *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006.

[5] Russell, Alexander, Garrick Orchard, and Ralph Etienne-Cummings. "Configuring of spiking central pattern generator networks for bipedal walking using genetic algorthms." *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*. IEEE, 2007.

[6] Reil, Torsten, and Phil Husbands. "Evolution of central pattern generators for bipedal walking in a real-time physics environment." *Evolutionary Computation, IEEE Transactions on* 6.2 (2002): 159-168.

[7] Jakobi, Nick. "Running across the reality gap: Octopod locomotion evolved in a minimal simulation." *Evolutionary Robotics*. Springer Berlin Heidelberg, 1998.

[8] Srinivasan, S., Robert E. Gander, and Hugh C. Wood. "A movement pattern generator model using artificial neural networks." *Biomedical Engineering, IEEE Transactions on* 39.7 (1992): 716-722.

[9] Wiklendt, Lukasz, S. T. Chalup, and Maria M. Seron. "Simulated 3d biped walking with an evolution-strategy tuned spiking neural network." *Neural Network World,* 19.2 (2009): 235.

[10] Berns, Karsten, Rüdiger Dillmann, and Stefan Piekenbrock. "Neural networks for the control of a six-legged walking machine." *Robotics and autonomous systems*14.2 (1995): 233-244.

[11] Sabourin, Christophe, Olivier Bruneau, and Gabriel Buche. "Control strategy for the robust dynamic walk of a biped robot." *The International Journal of Robotics Research* 25.9 (2006): 843-860.

[12] Allen, Brian, and Petros Faloutsos. "Complex networks of simple neurons for bipedal locomotion." *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009.

[13] Inden, Benjamin, et al. "Exploiting inherent regularity in control of multilegged robot locomotion by evolving neural fields." *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*. IEEE, 2011.

[14] Miller, G. A, Galanter, E., and Pribram, K. A (1960). *Plans and the structure of behavior*. Holt, Rhinehart, & Winston, New York, NY.

[15] Anokhin, P. K. (1935). *Problems of centre and periphery in the physiology of nervous activity*. Gorki: Gosizdat (in Russian).

[16] Anokhin, P. K. (1974). *Biology and Neurophysiology of the Conditioned Reflex and Its Role in Adaptive Behavio*r. Pergamon, Oxford.

[17] Sandamirskaya, Y. and Schöner, G. (2010). An embodied account of serial order: How instabilities drive sequence generation. *Neural Networks*, 23(10):1164–1179.

[18] Rohmer E., Signgh S. P. N. and Freese M. (2013) V-REP: a Versatile and Scalable Robot Simulation Framework, In *Proceeding of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013.

[19] Randell, Lyall (aka "Crazy Eyes"). Asti CAD model.

[20] Lakhman, K. and Burtsev, M. (2013). Neuroevolution results in emergence of short-term memory in multi-goal environment. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference, GECCO '13*, pages 703–710, ACM, New York, NY

SESSION

GENETIC ALGORITHMS AND APPLICATIONS + GENETIC PROGRAMMING + DYNAMIC OPTIMIZATION

Chair(s)

TBA

The Parallelization of Two Constrained Optimization Problems Using Evolutionary Algorithms

Michael Scherger
Department of Computer Science
Texas Christian University
Fort Worth, TX, 76129, USA
Email: m.scherger@tcu.edu

Roja Ramani Molupoju
Department of Computing Science
Texas A&M University – Corpus Christi
Corpus Christi, TX, 78412, USA
rmolupoju@tamucc.edu

Abstract – *(GEM'14) This research discusses the parallelization of two constrained optimization problems: Rosenbrock's function and the Benacer-Tau function. The general nonlinear programming (NLP) problem requires finding the optimum point (minimum or maximum) of a function of n real variables subjected to some given constraints. There are numerous situations in science and engineering where the optimum is bounded adding complexity to the optimization problem. An approach used in this research by [31] uses evolves two populations. One population is evolved inside the feasible domain of the design space and the second population is evolved outside this feasible domain. In this research a parallelized version of this algorithm was implemented. The results of this research show that this parallel technique is both effective and accurate.*

Keywords: Evolutionary algorithms, constrained optimization problems, parallel algorithms.

1 Introduction

Many search and optimization problems in science and engineering involve a number of constraints that the optimal solution must satisfy. A constrained optimization problem is usually written as a nonlinear programming (NLP) problem of the following type. The problem seeks to minimize $F(x_1, \ldots, x_i, \ldots, x_n)$ $(1 \le i \le n)$ subject to side constraints $x_{i\,min} \le x_i \le x_{i\,max}$ $(1 \le i \le n)$ and inequality and equality constraints $g_j(x_1, \ldots, x_n) \le 0$ where $(1 \le j \le m)$ and $h_j(x_1, \ldots, x_n) = 0$ where $(1 \le j \le p)$. For these types of NLP problems, there are n variables (that is, X is a vector of size n), m inequality constraints, and p equality constraints. The function $F(X)$ is the objective function, $g_j(X)$ is the j^{th} inequality constraint, and $h_k(X)$ is the k^{th} equality constraints. The i^{th} variable varies in the range $x_{imin} : x_{imax}$. Equality constraints could be eliminated with some of the variables [26]. However, only NLP problems that are subject to inequality constraints were considered in this research.

There are many evolutionary algorithms that were introduced by researchers by which the general NLP problem could be solved. One of the main differences between various approaches lay in constraint-handling the techniques employed as follows:

- Various implementations of the penalty function method
- Specialized representations and operators
- Repair algorithms
- Separation of objectives and constraints (behavioral memory, superiority of feasible points, multi-objective optimization techniques)
- hybrid algorithms etc.

Precisely searching the boundary area between the feasible and the infeasible regions is most critical for any function to find the global optimum. The inability of evolutionary systems to search precisely the boundary area is the main reason for difficulty in locating the global optimum. Some of the constraints are active at the global optimum. For this reason, in may constrained optimization problems, it is more difficult to locate the global optimum [23]. A two-population evolutionary computation approach was proposed in paper [31] two populations of individuals are evolved: one of feasible and the other of infeasible by which the search pressure upon the boundaries of the feasible space can be increased. Feasible individuals (also called females) are the ones that are evolved inside (including the boundaries of the feasible space), while the infeasible individuals (also called males) are evolved outside the feasible space. These two populations are subject to mutation and asexual crossover. Female-male crossover ensures the search pressure on the boundaries of the feasible space. This crossover can be performed between two or more feasible and infeasible individuals, and with, or without, favoring the better-fit females or the better-ranked males.

When two populations, one of feasible and one of infeasible individuals, are distinctively evolved, these two populations interact systematically (rather than occasionally) for the purpose of exploring the boundaries of the feasible space. Since the criterion based on which individuals are assigned to the two populations does not

change during evolution, the behavior of the two populations is easier to understand.

The two-population evolutionary algorithm is described in the section 4, was proposed by [31]. This algorithm was initially tested for functions with two variables; this project works with the algorithm for functions with multiple variables and is tested. This algorithm is implemented using C++ programming using Message Passing Interface (MPI). This is a parallel implementation of the algorithm and use MPI on a local cluster of workstations. It is run for two benchmark problems from literature: Rosenbrock's function and the Benacer-Tau function. Evaluation of the effectiveness of this technique is provided in sections 5 and 6.

2 Background and Related Research

The origins of evolutionary algorithms can be traced to at least the 1950's (e.g., Fraser, 1957; Box, 1957). For the sake of brevity, main methodologies that have emerged in the last few decades: "evolutionary programming" (Fogel et al., 1966), "evolution strategies" (Rechenberg, 1973), and "genetic algorithms" (Holland, 1975) used in this research.

Genetic algorithms (GAs), developed by Holland (1975), have traditionally used a more domain independent representation, namely, bit-strings. However, many recent applications of GAs have focused on other representations, such as graphs (neural networks), Lisp expressions, ordered lists, and real-valued vectors.

In general, GAs are highly simplified and abstract computational models inspired by natural selection [14]. The differences in "fitness calculation" in GAs and nature clearly illustrate this. Standard GAs apply an *a priori* defined fitness function (e.g. the function one wants to optimize) to an individual. They typically use an "all at once" calculation: individuals are evaluated immediately after their creation (i.e. birth). Fitness calculation in nature is substantially different [1]. It consists of a continuous series of tests during an individual's life. Furthermore, these tests are not strictly pre-defined. They originate from a complex environment. This environment is not only influenced by the animal's own actions but also by the other individuals as well as other processes occurring in the world (e.g. climatologically or geophysical changes). Summarizing one can say that - in contrast with GAs - nature uses a far more partial but continuous fitness evaluation in order to adapt to a complex world.

One of the key problems for using GAs in practical applications is to design the fitness function, particularly when we do not know where the global optimum is located [19] [22] [32]. A comparative estimate of how good as a solution turns out to be enough in most case (e.g. the largest value has to be closer to the global maximum if we are trying to maximize the objective function), but if we are dealing with constrained problems, we have to find a way of estimating also how close in an infeasible solution from the feasible region [33]. This is not an easy task, since most real-world problems have complex linear and non-linear constraints, and several approaches have been proposed in the past to handle them [14]. From those, the penalty function seems to be yet the most popular technique for engineering problems, but the intrinsic difficulties to define good penalty values makes harder the optimization process using a GA [6] [7] [19].

3 Parallelization Technique

Since evolutionary algorithms are characterized by their repeated fitness evaluation of the individuals in the population, it is natural to view them as parallel algorithms. In generational evolutionary algorithms, substantial savings in elapsed time can often be obtained by performing fitness evaluations in parallel. In the simplest form of parallelism, a manager process performs all the function of the evolutionary algorithm except evaluation of individuals, which are performed in parallel by worker processes operating on separate processors. The master process waits for all workers to return the evaluated individuals before varying on with the next generation.

The algorithm in this research is implemented using MPI in which there are n processes that work simultaneously using inter-process communication. This performs well when a complex function is being used and there is more number of iterations that are to be performed.

Manager-Worker paradigm is used to implement the algorithm. In this, there is one Manager and several workers and is depicted in Figure 1. The Manager generates uniform random points and sends these points to the workers; the workers process these points (check the feasibility and compute the value of the function at that point if it is a feasible point) and send back the processed results. The manager would then differentiate them into female and male based on their feasibility value (female if it is zero and male if not). Then the manager ranks the female individuals based on their fitness and the male individuals based on the number of constraints they violate. Generations of new individuals and mutating female and male individuals would also happen. The female-male pairs are formed with the females choosing their males in the rank decreasing order. Then the crossover is performed and all the steps are until the best female is constant for a given number of individuals.

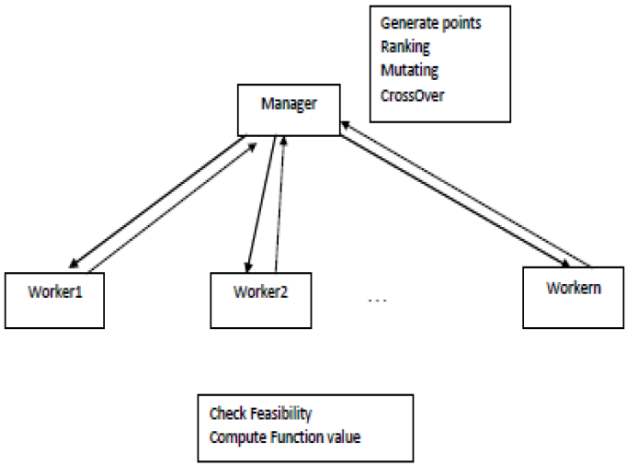


Figure 1: Manager-Worker paradigm for this research.

4 Test Functions

In this project, a technique based on the concept of co-evolution is used to create two populations that interact with each other in such a way that the objective function is minimized. The approach has been tested with two single-objective optimization problems with linear and non-linear inequality constraints and its results are compared with those produced by other GA-based and mathematical programming approaches.

This research is tested using two benchmark problems from literature: Rosenbrock's function and the Benacer-Tau function. This section provides the description of these two constrained optimization problems.

4.1 Rosenbrock's Function (in two variables)

In mathematical optimization, the Rosenbrock function is a non-convex function used as a performance test problem for optimization algorithms introduced by Rosenbrock (1960). It is also known as "Rosenbrock"s valley" or "Rosenbrock's banana function." The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum however is difficult.

The original Rosenbrock function was unconstrained. The constraints for this function were introduced by [31]. A plot of Rosenbrock's function of 2 variables is shown in 2D in Figure 2 and in 3D in figure 3. The function is to minimize:

$$F(x_1, x_2) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$$

subject to:

$$-2 \le x_1 \le 2 \; and \; -1 \le x_2 \le 3$$
$$g_1(x_1, x_2) < 0 \; and \; g_2(x_1, x_2) \le 0$$

with g_1 and g_2 cubic and linear functions as follows:

$$g_1(x_1, x_2) = (x_1 - 1)^3 - x_2 + 1$$
$$g_2(x_1, x_1) = x_1 + x_2 - 2$$

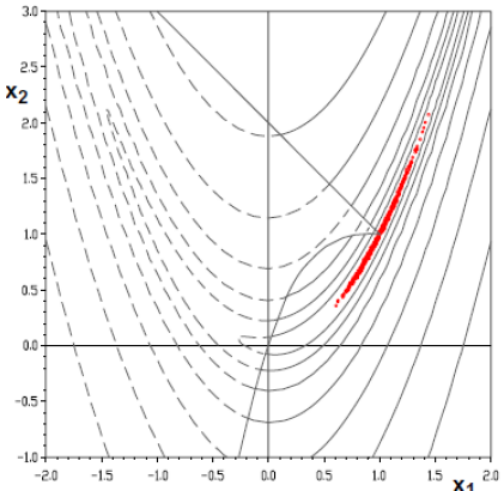


Figure 2: 2D Rosenbrock's function in two variables.

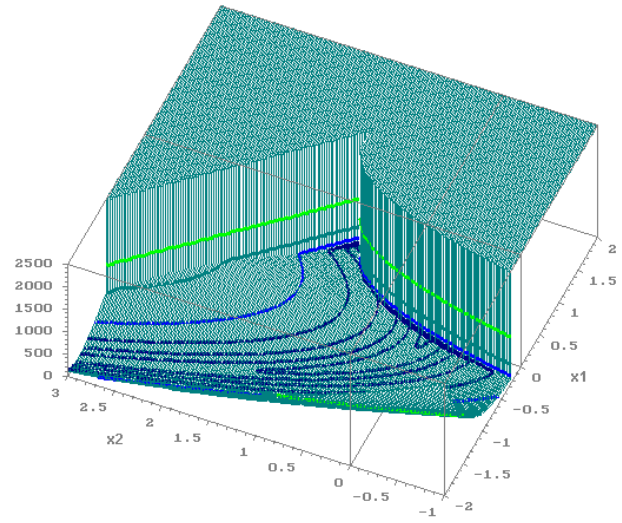


Figure 3: 3D Rosenbrock's function in two variables.

4.2 Benacer-Tao Function (in two variables)

Benacer and Tao introduced this function in 1986. The function can be stated as to minimize:

$$f(x_1, x_1) = x_1 - \frac{1}{2}x_2 - \frac{2}{3}x_1^2 + \frac{1}{2}x_2^2$$

subject to:

$$g(x_1, x_2) = 2x_1 - x_2 - 3 \le 0$$
$$x_1 \ge 0 \; and \; x_1 \ge 0$$

A 3D plot of the Benacer-Tao function is shown in Figure 4.

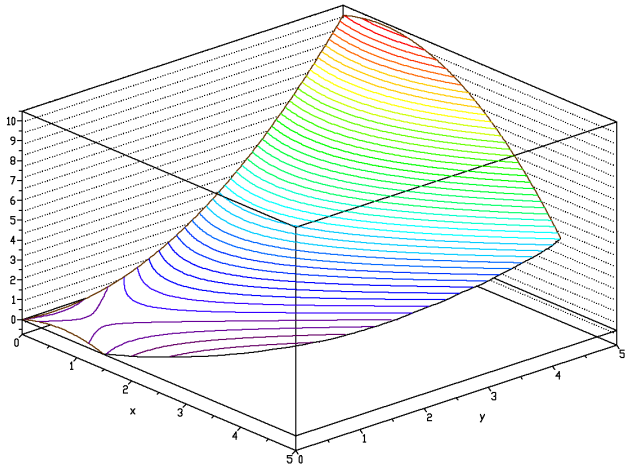


Figure 4: Plot of the Benacer-Tao function.

5 Summary of Results

The testing and analysis of the results for the Rosenbrock and Benacer-Tao functions is presented in this section. The program was run several times varying the number of processes and varying the female and male population sizes.

5.1 Rosenbrock's Function

As an example, Figure 5 presents a plot of the initial female and male populations overlaid on the plot of the 2D Rosenbrock's function.

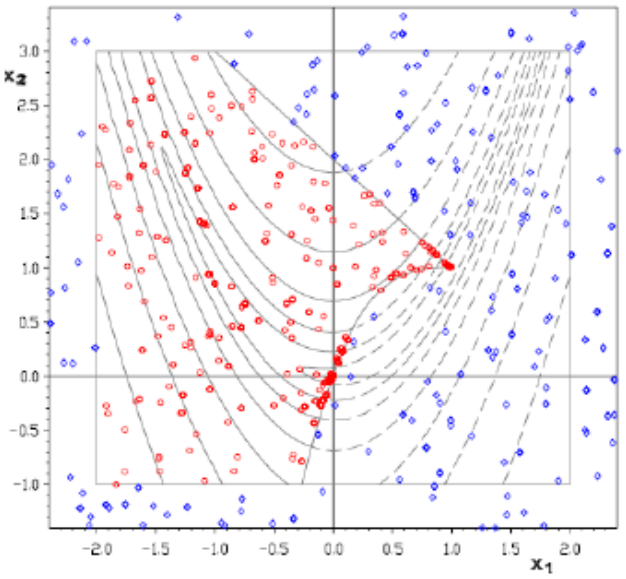


Figure 5: Sample male and female populations for Rosenbrock's function.

It can be clearly shown that the female population is evolved inside the feasible space and the male population is evolved outside the feasible space. Figure 6 presents a table in which the number of individuals of the female and male populations is fixed, but the number of processes varies between 1 and 8. An average of 50 runs with a particular number of processes and population size. The final function value begins to approach the optimum value of 0 at point (1,1) yet the number of function calls increases. Because the number of processes is increasing at a fixed population, the algorithm and function times decrease.

No.of processes	No. of female individuals	No. of male individuals	No. of iterations	Function Value	Total Time		No. of function calls	Optimum Point
					Algorithm time	Function time		
1	20	15	1000	0.268495	2.211688	0.3465099	2470	(0.979631,1.01145)
2	20	15	1000	0.029471	1.351219	0.461643	2601	(0.992211,1.00163)
3	20	15	1000	0.217387	1.224079	0.536698	2438	(0.98401, 1.01485)
4	20	15	1000	0.046619	1.11176	0.535684	2432	(0.99153, 1.00471)
5	20	15	1000	0.014651	0.99926	0.393892	2576	(0.99452,1.00116)
6	20	15	1000	0.041042	0.996646	0.495492	2696	(0.991972,1.00425)
7	20	15	1000	0.009716	0.971562	0.502946	2407	(0.996228,1.00232)
8	20	15	1000	0.007312	0.949841	0.403412	2765	(0.998651,1.00101)

Figure 6: Results of Rosenbrock's function for different number of processes.

Figure 7 presents a plot of points where the two populations converged at various points (x_1, x_2). It is clearly shown that most runs converge on the point (1,1).

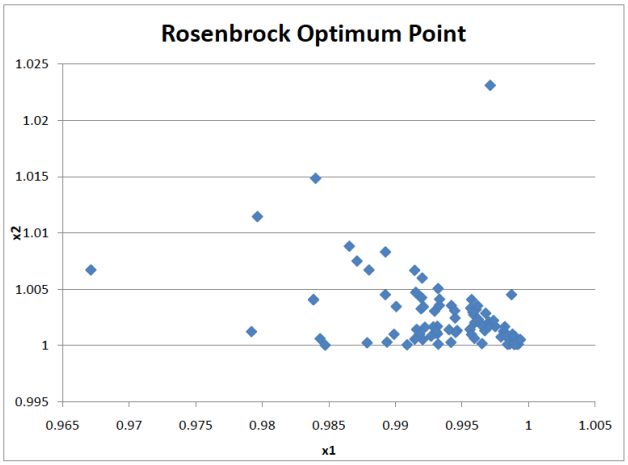


Figure 7: Plot showing the optimum computed values for Rosenbrock's function.

Figure 8 shows a plot of the total time for the algorithm, total function time, and total time (wall clock time). It can be seen from this plot that as the number of processes increases, the computation time decreases. This is not true for all values. This is because the number of an increased number of function calls that make the computation time rise. If there is an increase in the number of function calls, then time for computing the

function value increases and thus increases the time for the worker's response to the manager.

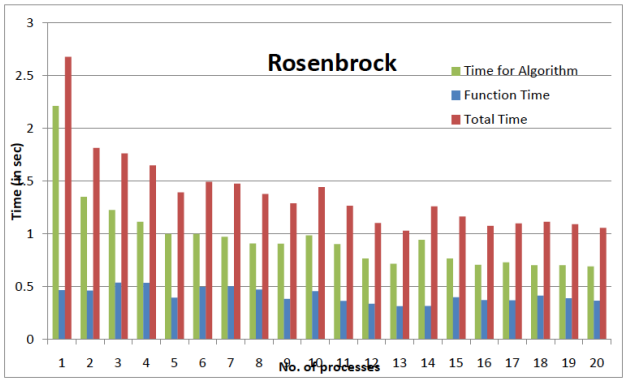


Figure 8: Single graph showing all computation times for different number of processes for Rosenbrock's function.

5.2 Benacer-Tao Function

Figure 9 presents a table of an average of 50 runs with a particular number of processes and population sizes was performed.

No. of processes	No. of female points	No. of male points	No. of iterations	Function Value	Total Time	No. of function calls	Optimum Point
1	20	15	1000	-0.74875	0.449998	5470	(2.25818, 1.5175)
2	20	15	1000	-0.74894	0.38384	6601	(2.24248, 1.48597)
3	20	15	1000	-0.74999	0.228379	5438	(2.25188, 1.50376)
4	20	15	1000	-0.74930	0.299412	6432	(2.27155, 1.54318)
5	20	15	1000	-0.74924	0.239412	5576	(2.22644, 1.45289)
6	20	15	1000	-0.74949	0.225073	5696	(2.2387, 1.47775)
7	20	15	1000	-0.74855	0.217929	5407	(2.25951, 1.52032)
8	20	15	1000	-0.74913	0.31041	6595	(2.23128, 1.49712)

Figure 9: Results of the Benacer-Tao function for different number of processes.

Figure 10 shows the optimum point for the Benacer-Tau function. The theoretical value for the optimum minimum for the function is (2.25, 1.5) at which the function value is -0.75. The best optimum point found in this research was (2.4999, 1.4999) at which the function value is -0.75.

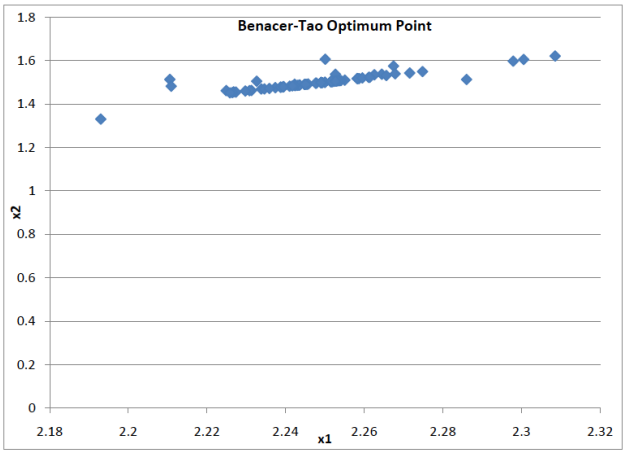


Figure 10: Plot showing the optimum computed values for the Benacer-Tau function.

Figure 11 shows a plot of function time, algorithm time, and total time overlaid on the same graph. From the graph, it can be seen that as the number of processes increases, the time decreases. The graph is not completely decreasing as the total time is dependant on the number of function computation time calculated at the worker processes

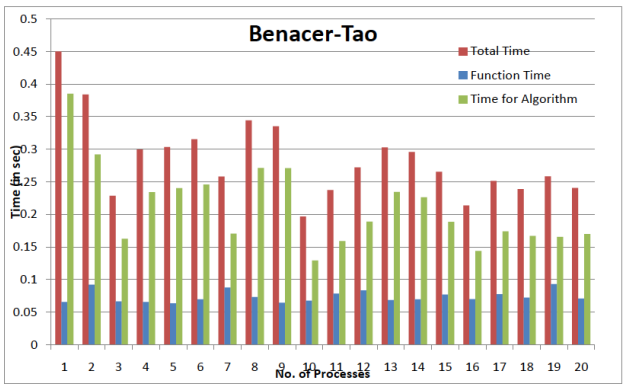


Figure 11: Single graph showing all computation times for different number of processes for the Benacer-Tau function.

6 Conclusions and Future Work

Solving constraint nonlinear programming problems using a co-evolutionary algorithm was implemented. The two-population evolutionary algorithm proposed by [31] was used in which two distinct populations are evolved. The interaction between these populations induces an exploratory pressure on the boundaries of the feasible space. Mutating the female and male populations induced search capabilities inside the feasible space and parallel to

the feasible-infeasible boundary. Figure 12 presents a table of a summary of the results in this research.

Optimization Problem	Theoretical Value	This Algorithm	Best Solution	Constraint Values
Rosenbrock	Value : 0 Point : (1,1)	Average : 0.01886 (0.99463,1.00301) Worst : 0.510719 (0.96712,1.00671)	Value : 0.000312 Point : (0.99938,1.00052)	G1 = -0.00052 G2 = -0.0001
Benacer-Tao	Value : -0.75 Point : (2.25,1.5)	Average : -0.74358 (2.24809,1.50043) Worst : -0.73751 (2.22677,1.45573)	Value : -0.75 Point : (2.24999,1.49998)	G1 = -2.22045 e-16

Figure 12: Summary of results for this research.

There is no penalty factor that is involved in constraint handling; hence a clear distinction can be made between the feasible and infeasible individuals. Parallel Implementation of the Co-evolutionary Genetic Algorithm has been a long awaited development for the evolutionary algorithms as mentioned in [7] [19] [31]. In this project, the algorithm has been implemented using MPI, which uses multiple processes that run in parallel, by which the computation time is reduced. This implementation worked well with several test problems that were previously solved using GA-based and mathematical programming techniques, producing in most of the cases results better than those previously reported. The technique is able to achieve such good results with relatively small populations and using a relatively low number of generations.

Functions of multiple variables used in various disciplines of Engineering can be solved for the global optimum using this research. This research provides sound results in terms of the computation time and the number of generations that need to be evolved for a more accurate value. The project is tested using the two benchmark problems from the literature. A detailed comparison of the results obtained from the project with that of those from the literature is presented.

7 References

[1] Adenike A. Adewuya, "New Methods in Genetic Search with Real-Values Chromosomes", Thesis, Mississippi State University, 1993.

[2] T. Back, D. Fogel and Z. Michalewicz, Handbook of Evolutionary Computation, The Institute of Physics Publishing, 2000.

[3] Benacer R and Pham Dinh Tao, "Global Maximization of a Non-definite Quadratic Function over a Convex Polyhedron", Fermat Days 85: Mathematics for Optimization, Amsterdam, Holland, 1986, pp. 65-76.

[4] Carlos R. Garcia-Alonso, Leonor M. Pérez-Naranjo, Juan C. Fernandez-Caballero, "Multiobjective evolutionary algorithms to identify highly auto correlated areas: the case of spatial distribution in financially compromised farms" *Annals of Operations Research*, Jan 2011.

[5] Carlos A. Coello Coello, "Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art," Computer Methods in Applied Mechanics and Engineering, Vol. 191, No. 11-12, pp. 1245-1287, 2002.

[6] Carlos A. Coello Coello, "Constraint-handling using an evolutionary multiobjective optimization technique", *Civil Engineering and Environmental Systems*, 17:319-346, 2000.

[7] Carlos A. Coello Coello, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems", May 1999.

[8] Carlos A. Coello Coello, "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design", PhD thesis, Department of Computer Science, Tulane University, New Orleans, LA, Apr 1996.

[9] Pei-Chann Chang and Shih-Hshin Chen. The development of a sub-population genetic algorithm II (SPGA II) for multiobjective combinatorial problems, *Applied Soft Computing*, Vol. 9, No. 1, pp. 173-181, January 2009.

[10] C. Y. Cheong, K. C. Tan and B. Veeravalli. A multi-objective evolutionary algorithm for examination timetabling, *Journal of Scheduling*, Vol. 12, No. 3, pp. 121--145, April 2009 .

[11] David W. Coit, Alice E. Smith, David M. Tate, "Adaptive Penalty Methods for Genetic Optimization of Constrained Combinatorial Problems", *Informs Journal On Computing*, Vol. 8, No. 2, Spring 1996, pp. 173-182, DOI: 10.1287/ijoc.8.2.173.

[12] A.E. Eiben, "Multiparent Recombination in Evolutionary Computing," in A.Ghosh and S. Tsutsui, editors, Advances in Evolutionary Computing, Natural Computing Series, Springer, pp. 175-192, 2002.

[13] Eshelman Larry J & Schaffer David J, "Real-coded Genetic Algorithms and Interval-Schemata", In Whitley, D. L. (Ed.), Foundations of Genetic Algorithms 2 (pp. 187-202). California: Morgan Kaufmann, 1993.

[14] Mistuo Gen and Runwei Cheng, "Genetic Algorithms and Engineering Design", John Weily & Sons, Inc, New York, 1997.

[15] Grefenstette J. J., "Predictive models using fitness distributions of genetic operators", In *Foundations of Genetic Algorithms 3*, D. Whitley (Ed.), San Mateo, CA: Morgan Kaufmann.

[16] David M Himmelblau, "Applied Nonlinear Programming", McGraw Hill, New York, 1972.

[17] Homaifar A, Lai S H Y and Qi X, "Constrained Optimization via Genetic Algorithms, Simulation", 62(4):242-254, 1994.

[18] Antony Iorio, Xiaodong Li, "Parameter Control within a Co-operative Co-evolutionary Genetic Algorithm", May 2002.

[19] Kenneth De Jong, William Spears, "On the State of Evolutionary Computation", September, 1993.

[20] [Kenneth Dee Jong A., "Are genetic algorithms function optimizers?" *Proceedings of the Second International Conference on Parallel Problem Solving from Nature*, 1992.

[21] Kenneth De Jong, William Spears, Thomas Back, David Fogel, Hugo de Garis, "An Overview of Evolutionary Computation", April, 1993.

[22] Kim J. H and Myung H, "Evolutionary programming techniques for constrained optimization problems," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 2, pp. 129–140, 1997.

[23] Michalewicz Z and Attia N, "Evolutionary Optimization of Constrained Problems", In Proceedings of the 3rd Annual Conference on Evolutionary Programming, pages 98-108, World Scientific, 1994.

[24] Michalewicz Z and Schoenauer M, "Evolutionary Algorithms for Constrained Parameter Optimization Problems," Evolutionary Computation, Vol. 4, No. 1, pp. 1-32, 1996.

[25] Papalambros P and Li H L, "Notes on the Operational utility of Monotonicity in Optimization", ASME Journal of Mechanisms, Transmissions and Automation in Design, Vol.105, 1983, pp. 174-180.

[26] Papalambros P and Wilde D J, "Principles of Optimal Design – Modeling and Computation", Cambridge Univ. Press, New York, 2000.

[27] Paredis Jan, "Coevolutionary Algorithms", *Proceedings of the Seventh International Conference on Genetic Algorithms* ed TBaeck (San Mateo, CA: Morgan Kaufmann) pp 393-399, 1997.

[28] Ragsdell K M and Phillips D T, "Optimal Design of a class of Welded Structures Using Geometric Programming", ASME Journal of Engineering for Industry, Vol. 98, pp. 1021-1025, 1997.

[29] Rosenbrock, H. H., "An automatic method for finding the greatest or least value of a function", *The Computer Journal* 3: 175–184, doi:10.1093/comjnl/3.3.175, ISSN 0010-4620, MR0136042.

[30] Simionescu P.A., Dozier G.V. and Wainwright R.L. "A Two-Population Evolutionary Algorithm for Constrained Optimization Problems," IEEE World Congress on Computational Intelligence, Vancouver, Canada, July 16-21, 2006.

[31] Simionescu P.A., Beale D.G. and Dozier G.V. "Constrained Optimization Problem Solving Using Estimation of Distribution Algorithms," IEEE World Congress on Evolutionary Computation, Portland, OR, June 20-23, 2004.

[32] Min-Jea Tahk and Byung-Chan Sun, "Co evolutionary Augmented Lagrangian Methods for Constrained Optimization", IEEE Transactions On Evolutionary Computation, Vol. 4, No. 2, July 2000.

[33] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, Qingfu Zhang, " Multiobjective evolutionary algorithms: A survey of the state-of-the-art," *Swarm and and Evolutionary Computation*", March 2011.

Improving the Multi-Restart Local Search Algorithm by Permutation Matrices and Sorted Completion Times for the Flow Shop Scheduling Problem

J. C. Seck-Tuoh-Mora[1], **L. Garcia-Lechuga**[2], **J. Medina-Marin** [1], **N. Hernandez-Romero** [1], **and E. S. Hernandez-Gress**[1]

[1]Área Academica de Ingeniería, Universidad Autónoma del Estado de Hidalgo, Pachuca, Hidalgo, Mexico

[2]Área de Electromecánica Industrial, Universidad Tecnológica de Tulancingo, Tulancingo, Hidalgo, Mexico

Abstract— *Iterated local search (ILS) is a metaheuristic used successfully to solve the flow shop scheduling problem. In particular, the multi-restart ILS (MRSILS) is an easily implementable algorithm which obtains good results. In this paper, we modify the MRSILS algorithm in two ways. First, small changes in the initial solution are generated by permutation matrices in order to improve it before using the MRSILS. Second, a minor variation is made in the strategy of the MRSILS. Sorted completion times are taken to select the job to be inserted in new positions to obtain a better scheduling. The original MRSILS and both modifications are evaluated with well-known benchmark instances. The experiments show that the new modifications produce slightly better results than the original one, especially for a large number of jobs.*

Keywords: Flow shop scheduling problem, iterated local search, permutations, work time costs

1. Introduction

The flow shop scheduling is a classic combinatorial problem established by Johnson in [1]. In this problem, n jobs must be processed on m machines, where each machine is required to process the set of all jobs in the same order. The most common objective in this problem is to minimize the completion time of the last job, or makespan.

Many evolutionary methods have been proposed for makespan minimization. Some of them are based on the modification of continuous evolutionary algorithms. For instance, recent modifications of particle swarm optimization methods have been proposed in [2], [3], [4], and a bee colony algorithm has been presented in [5].

These are only a few examples of the great number of papers devoted to adapt evolutionary and metaheuristic algorithms to minimize the makespan in the flow shop scheduling problem. Very often these papers present complex algorithms in order to construct factible and better solutions. The basic operations in these algorithms, however, are a mixture and/or extensions of job insertions and swaps.

Past research shows that simpler algorithms are able to obtain effective results without need of complex strategies. In particular, the iterated local search method (ILS) has been applied in the minimization of the total flow time for the flow shop scheduling problem [6]. This method has been recently improved using multi-restart points (MRSILS), showing better results than many more intricate and sophisticated metaheuristics [7]. This method is only based in insertions of jobs; without considering swaps.

This paper takes the MRSILS algorithm proposed by Dong et al. in [7] in order to minimize the makespan of the flow shop scheduling problem. Then, two modifications are presented. The first one consists of applying a matrix which defines job permutations to improve a given initial solution. This improved solution is passed to the MRSILS algorithm to obtain a final result. This modified algorithm is called MRSILS_PM.

The second modification is a slightly change in the strategy of the MRSILS, where the job to be inserted is selected sorting the differences of completion job times. For this reason, this modification is called MRSILS_SD.

The three algorithms have been evaluated taking the benchmark instances proposed in [8] and [9], which can be downloaded as the data file *flowshop1* from the web site OR Library.[1] The results show that MRSILS_PM and MRSILS_SD have obtained slightly better results than the original MRSILS in minimizing the execution time and/or the makespan.

2. Flow shop scheduling problem

The flow shop scheduling problem consists of the assignment of a set of n jobs $W = \{J_1, \ldots, J_n\}$, each job J_i is composed of a set of operations $J_i = \{O_{i1}, \ldots, O_{im}\}$ processed by a set of machines $M = \{M_1 \ldots M_m\}$. Every operation O_{ij} has associated a value T_{ij} that represents the processing time of operation O_{ij}; for $1 \leq i \leq n$ and $1 \leq j \leq m$. Each machine can process only one job at a time. It is assumed that the machine sequence is identical for all jobs, and the job sequence is the same for all machines.

[1]http://people.brunel.ac.uk/˜mastjjb/jeb/orlib/flowshop info.html. Consulted in January 27th, 2014.

Therefore, a job schedule is represented as a permutation $\pi = \{\pi_1, \ldots, \pi_n\}$ of W. The completion time of the operation O_{ij} is indicated by C_{ij}. The objective of the flow shop scheduling problem is the minimization of the makespan C_{max}, or the completion of the last operation in π.

The model of the flow shop scheduling problem can be defined in the following way:

$$\begin{aligned} C_{\pi_1 1} &= T_{\pi_1 1} \\ C_{\pi_1 j} &= C_{\pi_1 j-1} + T_{\pi_1 j} \\ C_{\pi_i 1} &= C_{\pi_{i-1} 1} + T_{\pi_i 1} \\ C_{\pi_i j} &= \max\{C_{\pi_{i-1} j}, C_{\pi_i j-1}\} + T_{\pi_i j} \end{aligned} \quad (1)$$

where $i \in \{2, \ldots, n\}$ and $j \in \{2, \ldots, m\}$. Thus $C_{max} = C_{nm}$.

3. MRSILS algortihm

The framework of the MRSILS algorithm is very simple [7]. First, a well-known heuristic is applied in order to generate a *good* initial solution. This paper applies the NEH method described in [10], which constructs a solution π in negligible time.

The MRSILS algorithm improves the solution by inserting a selected job into different positions of π to generate a new solution π'. If a better solution is obtained, then π is replaced by π'.

Let π^* be the best solution found by the algorithm. In order to avoid that π^* be trapped in a local optimum, the search space is extended by generating restart solutions from a set Π (or pool) of solutions. This pool keeps the best q solutions calculated so far. If the MRSILS algorithm does not improve the solution after n iterations, the local optimum π is added to Π if it is not already there. If Π has more than q solutions, the worst solution of the pool is deleted.

When Π is not full, the restart solution is generated from π^*; otherwise, a solution π' is randomly chosen from Π. The selected solution is perturbed by inserting one randomly chosen job into another randomly chosen position. In this way, a new restart solution π is defined.

Figure 1 shows the pseudocode of the MRSILS algorithm taking the parameters specified in [7].

```
1: Set q = 20, lim = 1000
2: Set cnt = 0, flag = 0, Π = ∅
3: Generate π* using NEH and set π = π*
4: for i = 1 to lim do
5:     for j = 1 to n do
6:         Find k such that π_k = π*_j
7:         Insert π_k into other n − 1 positions in π,
           let π' be the best solution from the
           n − 1 generated permutations
8:         if π' is better than π then
               Set π = π', cnt = 0
9:         else
               Set cnt = cnt + 1
10:        if π is better than π* then
               Set π* = π, flag = 1
11:        if cnt = n then
12:            if flag = 1 then
                   Set Π = ∅, flag = 0
13:            if π ∉ Π then
                   Add π into Π
14:            if |Π| > q then
                   Delete the worst π' from Π
15:            if |Π| < q then
                   Perturb π* to obtain a new π
16:            else
                   Select π' at random from Π
                   and perturb it to obtain a new π
17:            if π is better than π* then
                   Set π* = π
18:            Set cnt = 0
19: return π*
```

Fig. 1: Pseudocode of MRSILS.

4. Permutation matrix

The MRSILS algorithm is only based on insertions. It can be considered as an algorithm which performs small changes in every solution in order to achieve an improvement. So the MRSILS algorithm is suitable to refine solutions.

The application of swaps and permutations of jobs, however, are useful to improve an initial solution at the beginning of the optimization process. These operations may produce the exploration of a bigger space search in order to obtain a better schedule which can be refined after with the MRSILS.

These swaps and permutations can be specified in a matrix M, where it has as many rows as n (number of jobs) and r columns. Here, r is the number of different solutions to be obtained for the swaps and permutations defined by M.

For a job schedule $\pi = \{\pi_1 \ldots \pi_n\}$, every column j in M defines a new permutation π'. The entry m_{ij} indicates the new place of the job π_i. If $m_{ij} = i$, then $\pi'_i = \pi_i$; that is, the job i in π conserves the same position in π'. Otherwise, for $m_{ij} = z \neq i$, $\pi'_i = \pi_z$. The job at position z in π changes into position i in π'. Therefore, every column j in M is a permutation of $Z_n = \{1, \ldots, n\}$. Since M consists only of permutations of Z_n, it is easily defined and can be applied quickly to π to produce a new schedule.

In order to define the matrix M, we are taking into account four parameters:

$$\begin{aligned} r &\rightarrow \text{number of columns} \\ \alpha &\rightarrow \text{proportion of columns which define large changes} \\ \gamma &\rightarrow \text{proportion of permuted jobs for columns defining large changes} \\ \delta &\rightarrow \text{proportion of permuted jobs for columns defining small changes} \end{aligned} \quad (2)$$

In Eq. 2, r can be defined as a multiple of q, the size of the pool Π in the MRSILS algorithm. In our experiments,

the best performance has been obtained with $r = 10 * q$, $\alpha = 0.55$, $\gamma = 0.6$ and $\delta = 0.1$.

```
1: Set r = q * 10, α = 0.55, γ = 0.6 δ = 0.1
2: Set lim = 1000
3: Generate π* using NEH
4: for i = 1 to lim do
5:     Set π = π* and P = ∅
6:     for j = 1 to r do
7:         Set a column vector c = Z_n^T
8:         Generate a random number s ∈ [0, 1]
9:         if s < α then
               Permute round(γ * n) elements in c
10:        else
               Permute round(δ * n) elements in c
11:        Add the column c to M
12:    for j = 1 to r do
13:        Obtain π' by permuting π according to
           column j in M
14:        Add π' into P
15:    Select the best solution π' from P.
16:    if π' is better than π* then
17:        Set π* = π'
18: return π*
```

Fig. 2: Pseudocode of PM.

```
1: Set q = 20, lim_a = 500, lim_b = 500
2: Set r = q * 10, α = 0.55, γ = 0.6 δ = 0.1
3: Set cnt = 0, flag = 0, Π = ∅
4: Generate π* using algorithm PM with lim_a
   iterations.
5: Improve π* using algorithm MRSILS with
   lim_b iterations.
6: return π*
```

Fig. 3: Pseudocode of MRSILS_PM.

The algorithm that defines and applies the permutation matrix (PM) is presented in Fig. 2. This algorithm takes first an improved solution from the NEH heuristic, where Z_n^T is the transpose of Z_n and round(a) is the closest integer of a real number a. Figure 3 describes the integration of the MRSILS and PM algorithms (MRSILS_PM). This algorithm takes first an improved solution from PM using small and large random permutations. Finally, the best solution is taken from PM and it is refined (or exploited) by the MRSILS algorithm. Notice that half of the iterations are done by PM and the final half are realized by MRSILS.

5. Strategy using sorted completion times

The second modification proposed in this paper is a slight change in the selection of the job to be inserted in the line 6 of the MRSILS algorithm (Fig. 1). In the original algorithm, the job is selected finding the position k in π containing the job j in the current best solution π^*. In the proposed modification, first the completion times of every job in π^* are taken. These completion times define a vector D as follows:

$$D = \{C_{im} : i = 1, \ldots, n\} \tag{3}$$

Equation 3 can be obtained when the makespan is calculated with Eq. 1, therefore, this calculus does not imply an extra numerical work. Then, the differences E between completion times is calculated as follows:

$$E = \{D_{j+1} - D_j : j = 1, \ldots, n-1\} \tag{4}$$

Notice that $|E| = n-1$ and all the values of E are positive because $D_j < D_{j+1}$ for every j in Eq. 4.

Once calculated Eq. 4, E is sorted in a descending way, where I is the index vector associated with E. That is, the first element of I is the index of the highest element in E; the second element of I is the index of the second highest element in E, and so on.

Take the first index μ in I. This index represents the pair of jobs π_μ and $\pi_{\mu+1}$ with the greatest completion time difference. A big difference would indicate that these jobs are not well-ordered in π^*, and a best solution could be obtained if one of the jobs is inserted in a different position.

Consequently, each π_μ and $\pi_{\mu+1}$ are inserted in the other positions, similar to the process done in the MRSILS algorithm. If some insertion improves the current solution, D and E are also updated. This process is iteratively repeated taking the i-th element of I in every step. This modification is called MRSILS_SD. In short, MRSILS_SD takes as criterion the descending sort of completion time differences to select the jobs to be inserted, in order to improve the current schedule. Figure 4 presents the pseudocode of the MRSILS_SD algorithm.

6. Experimental results

The original MRSILS algorithm and the two modifications (MRSILS_PM and MRSILS_SD) have been evaluated taking 29 flow shop scheduling instances commonly used in the literature (car1,. . ., car8; and rec01, rec03,. . ., rec43) [8] [9].

The parameters selected to compare the three algorithms are presented in Table 1. For each case, 50 independent runs have been calculated for every flow shop scheduling instance. Table 2 presents the data of each problem: name (N), size (n, m) indicating n jobs and m machines, and their optimum value (O). In order to compare the results, Table 2

1: Set $q = 20$, $lim = 1000$
2: Set $cnt = 0$, $flag = 0$, $\Pi = \emptyset$
3: Set $D = \emptyset$, $E = \emptyset$, $I = \emptyset$
4: Generate π^* using NEH and set $\pi = \pi^*$
5: **for** $i = 1$ to lim **do**
6: **for** $j = 1$ to $n - 1$ **do**
7: Set $D = \{C_{im} : i = 1, \ldots, n\}$
8: Set $E = \{D_{j+1} - D_j : j = 1, \ldots, n - 1\}$
9: Set I as the index vector associated with E sorted in a descending way
10: Take index $\mu = I_j$ and insert π_μ and $\pi_{\mu+1}$ into the other positions in π, let π' be the best solution from the $2n - 2$ generated permutations
11: Follow algorithm MRSILS from line 8.
12: **return** π^*

Fig. 4: Pseudocode of MRSILS_SD.

Table 1: Parameters to compare the three algorithms.

Parameter	Value
Pool size (q)	20
Number of iterations (lim)	1000
Number of iterations for the PM part (lim_a) in MRSILS_PM	500
Number of iterations for the MRSILS part (lim_b) in MRSILS_PM	500
Number of permutations (r) in PM	200
Proportion of columns (α) in PM defining large changes	0.55
Proportion of permuted jobs (γ) in PM for columns defining large changes	0.6
Proportion of permuted jobs (δ) in PM for columns defining small changes	0.1

also shows the results obtained by each algorithm: minimum (M), mean value (MV) and mean execution time (MET, in seconds). The best results obtained by any of the algorithms are presented in bold. Notice that more than one algorithm may reach the best value in every instance.

Table 4 shows a summary of the results obtained by the algorithms. This table presents the number of times in which each algorithm has obtained the best results.

From Tables 2 and 4, we can see that the three algorithms optimize easily the problems car1 to car8. In these ones, the original MRSILS has the minimum MET. For the problems rec01 to rec41, the three algorithms had similar results to find the optima, and MRSILS and MRSILS_SD have obtained almost the same results in finding the best MV. We can see, however, that MRSILS_SD had a better performance than the other algorithms to find the best M. In particular, MRSILS_SD has outperformed the other algorithms to find a minimum makespan for the greater instances of the rec problems (from rec19 to rec 41). Nevertheless, MRSILS_SD is a more time-consuming algorithm than the other ones. This is notorious as well in Table 2 for the greater instances of the rec problems. The MET parameter of MRSILS_SD is almost 75% greater than the same parameter of MRSILS in the last three problems.

Finally, it is clear that MRSILS_PM is the fastest algorithm for the rec problems. Actually, MRSILS_PM is executed in almost 50% less time that the original MRSILS for the last three problems in Table 2.

For each of the 12 hardest problems (rec19 to rec41), another set of 50 independent runs have been executed for every algorithm, but taking now $lim = 4000$. This implies that $lim_a = lim_b = 2000$ in the MRSILS_PM algorithm. Table 3 displays the results of this experiment and Table 5 presents the result summary. In the second experiment, we can see again that MRSILS_PM is faster than the other two algorithms. All the algorithms improved their results, especially the algorithm MRSILS_PM which improves the performance of MRSILS in the *Optimum* and *Best M* parameters. We can notice that MRSILS_SD outperforms the other algorithms in the second experiment as well, mainly in the last three problems.

7. Concluding remarks

The MRSILS is a very simple algorithm capable to obtain very good results in the optimization of the flow shop scheduling problem. On the contrary to more complex methods of optimization, the MRSILS algorithm is only based in insertions, starting from a single initial solution. In this manuscript, two modifications of the MRSILS have been proposed in order to improve its performance.

The first one (MRSILS_PM) consists of executing random permutations in the initial solution. One part of these permutations take into account a minimal part of the jobs, and the other part permutes a greater number of jobs. The experimental results show that this process is faster in a rough 50% than the original MRSILS algorithm for the bigger problems presented in this paper. Besides, MRSILS_PM obtains similar results to MRSILS for the *Best M* parameter.

The second modification (MRSILS_SD) is a small change in the strategy of the original algorithm. It selects the jobs to be inserted according to the difference of completion times between contiguous works. We suppose that bigger differences may indicate bad placed jobs that need to be reallocated in order to improve the solution. The experimental results show that MRSILS_SD outperforms slightly the other algorithms. This performance is not for free, the sorting operation involved in the step 9 of the algorithm (Fig. 4) increases the execution time in a rough 75% compared to the original MRSILS, for the bigger cases.

Future work could consider the implementation of an algorithm where the first part applies the matrix permutation and the second one implements the strategy of differences between completion times.

Table 2: Results of the three algorithms for $l = 1000$.

N	S	O	MRSILS			MRSILS_PM			MRSILS_SD		
	(n, m)		M	MV	MET	M	MV	MET	M	MV	MET
car1	$(11, 5)$	7038	**7038**	**7038**	**2.29**	**7038**	**7038**	3.52	**7038**	**7038**	3.1
car2	$(13, 4)$	7166	**7166**	**7166**	**3.07**	**7166**	**7166**	3.89	**7166**	**7166**	4.29
car3	$(12, 5)$	7312	**7312**	**7312**	**2.65**	**7312**	7315.24	3.71	**7312**	**7312**	3.71
car4	$(14, 4)$	8003	**8003**	**8003**	**3.52**	**8003**	**8003**	4.14	**8003**	**8003**	4.97
car5	$(10, 6)$	7720	**7720**	**7720**	**1.91**	**7720**	**7720**	3.32	**7720**	**7720**	2.55
car6	$(8, 9)$	8505	**8505**	**8505**	**1.31**	**8505**	**8505**	3.06	**8505**	**8505**	1.66
car7	$(7, 7)$	6590	**6590**	**6590**	**1.01**	**6590**	**6590**	2.81	**6590**	**6590**	1.23
car8	$(8, 8)$	8366	**8366**	**8366**	**1.29**	**8366**	**8366**	3.01	**8366**	**8366**	1.64
rec01	$(20, 5)$	1247	**1247**	**1248.32**	7.5	**1247**	1248.64	**6.4**	**1247**	1248.68	11.3
rec03	$(20, 5)$	1109	**1109**	**1109**	7.52	**1109**	**1109**	**6.4**	**1109**	**1109**	11.3
rec05	$(20, 5)$	1242	1245	1245	7.50	**1242**	**1244.88**	**6.45**	**1242**	1244.94	11.43
rec07	$(20, 10)$	1566	**1566**	**1566.04**	8.91	**1566**	1567.44	**7.4**	**1566**	1566.36	13.65
rec09	$(20, 10)$	1537	**1537**	**1537**	8.79	**1537**	**1537**	**7.33**	**1537**	**1537**	13.66
rec11	$(20, 10)$	1431	**1431**	**1431**	8.89	**1431**	**1431**	**7.39**	**1431**	**1431**	13.58
rec13	$(20, 15)$	1930	**1930**	1933.48	10.28	**1930**	1934.7	**8.43**	**1930**	**1932.82**	16.04
rec15	$(20, 15)$	1950	**1950**	**1950.24**	10.24	**1950**	1952.64	**8.36**	**1950**	1950.36	15.98
rec17	$(20, 15)$	1902	**1902**	**1902**	10.07	**1902**	1903.18	**8.35**	**1902**	1902.28	16.07
rec19	$(30, 10)$	2093	**2099**	2104.44	22.65	**2099**	2102.92	**14.75**	**2099**	**2102.84**	37.03
rec21	$(30, 10)$	2017	**2020**	2044.04	27.81	2021	2045.08	**17.29**	**2020**	**2040.78**	38.35
rec23	$(30, 10)$	2011	2014	2019.74	28.1	2014	2020.18	**17.43**	**2013**	**2019.24**	39.03
rec25	$(30, 15)$	2513	**2513**	**2523.3**	32.49	**2513**	2532.24	**20.06**	**2513**	2529.26	47.44
rec27	$(30, 15)$	2373	2378	2389.34	32.62	2378	2392.34	**20.14**	**2377**	**2389.08**	47.84
rec29	$(30, 15)$	2287	**2287**	**2299.96**	32.81	2290	2304.56	**20.21**	**2287**	2301.02	47.35
rec31	$(50, 10)$	3045	**3053**	3056.94	95.38	**3053**	3061	**51.73**	**3053**	**3056.86**	143.27
rec33	$(50, 10)$	3114	**3114**	**3119.06**	93.16	**3114**	3122.04	**50.46**	**3114**	3126.46	138.31
rec35	$(50, 10)$	3277	**3277**	**3277**	94.47	**3277**	**3277**	**51.16**	**3277**	**3277**	140.64
rec37	$(75, 20)$	4951	5007	**5039.18**	395.04	5022	5051.24	**205.15**	**5000**	5042.78	681.12
rec39	$(75, 20)$	5087	5120	**5134.84**	402.62	5124	5142.86	**208.38**	**5114**	5135.9	702.96
rec41	$(75, 20)$	4960	5018	5049.84	400.62	5026	5058.7	**206.99**	**5017**	**5048.12**	696.53

Table 3: Results of the three algorithms for $l = 4000$.

N	S	O	MRSILS			MRSILS_PM			MRSILS_SD		
	(n, m)		M	MV	MET	M	MV	MET	M	MV	MET
rec19	$(30, 10)$	2093	2096	2100.76	110	**2093**	**2100.08**	**68.15**	2099	2100.2	153.02
rec21	$(30, 10)$	2017	**2020**	2038.8	111	**2020**	2038.62	**68.44**	**2020**	**2036.18**	153.3
rec23	$(30, 10)$	2011	2013	**2017.08**	111.2	**2011**	2017.86	**68.85**	2013	2018.12	154.95
rec25	$(30, 15)$	2513	**2513**	**2518.46**	127.61	**2513**	2520.48	**78.93**	**2513**	2520.72	187.62
rec27	$(30, 15)$	2373	2376	2382.04	128.96	2376	2384.68	**79.56**	**2373**	**2381.7**	189.75
rec29	$(30, 15)$	2287	**2287**	**2292.42**	129.14	**2287**	2293.88	**79.72**	**2287**	2292.88	190.1
rec31	$(50, 10)$	3045	3053	3053.5	377.22	**3048**	3054.22	**204.99**	**3048**	**3053.22**	560.71
rec33	$(50, 10)$	3114	**3114**	3114.68	371.17	**3114**	3115.96	**201.07**	**3114**	**3114.16**	545.77
rec35	$(50, 10)$	3277	**3277**	**3277**	376.84	**3277**	**3277**	**204.05**	**3277**	**3277**	558.32
rec37	$(75, 20)$	4951	4979	**5014.06**	1558.13	4993	5022.88	**809.45**	**4964**	5016.14	2642.87
rec39	$(75, 20)$	5087	5120	5128.74	1595.96	5120	5130.14	**827.02**	**5114**	**5126.92**	2712.61
rec41	$(75, 20)$	4960	4993	5026.7	1575.54	5000	5033.02	**822.53**	**4991**	**5023.64**	2651.83

Table 4: Best results obtained by the algorithms for $l = 1000$.

Algorithm	Optimum	Best M	Best MV	Best MET
MRSILS	20	23	21	8
MRSILS_PM	20	22	12	21
MRSILS_SD	21	29	19	0

Table 5: Best results obtained by the algorithms for $l = 400$.

Algorithm	Optimum	Best M	Best MV	Best MET
MRSILS	4	5	5	0
MRSILS_PM	6	8	2	12
MRSILS_SD	5	10	7	0

Other possibility is to investigate other criteria in order to select the jobs to be inserted in the different places to improve the current schedule. For instances, the idle times between contiguous jobs or machines.

References

[1] S. R. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Nav. Res. Log.*, vol. 1, no. 1, pp. 61–68, March 1954.

[2] C. Zhang and J. Sun, "An alternate two phases particle swarm optimization algorithm for flow shop scheduling problem," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5162–5167, April 2009.

[3] H. Liu, L. Gao, and Q. Pan, "A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4348–4360, April 2011.

[4] Z. Lian, X. Gu, and B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos Soliton. Fact.*, vol. 35, no. 5, pp. 851–861, March 2008.

[5] M. F. Tasgetiren, Q. Pan, P. Suganthan, and A. H.-L. Chen, "A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops," *Inform. Sciences*, vol. 181, no. 16, pp. 3459–

3475, August 2011.
[6] X. Dong, H. Huang, and P. Chen, "An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1664–1669, May 2009.
[7] X. Dong, P. Chen, H. Huang, and M. Nowak, "A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time," *Comput. Oper. Res.*, vol. 40, no. 2, pp. 627–632, February 2013.
[8] J. Carlier, "Ordonnancements a contraintes disjonctives," *R. A. I. R. O. Oper. Res.*, vol. 12, no. 4, pp. 333–350, February 1978.
[9] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, January 1995.
[10] M. Nawaz, E. E. E. Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

Design of a PID control gains for a robotic manipulator under parametric uncertainties by using DE-SQP algorithm

Miguel G. Villarreal-Cervantes[1]
[1]**Postgraduate Department, Instituto Politécnico Nacional, CIDETEC, Mexico, D. F., Mexico.**

Abstract—*As the demand of robotic manipulator with an adequate performance under the effect parametric uncertainties is increased, the fixed PID controller requires an new approach to handle with such uncertainties. In this paper the formal formulation of a robust fixed PID control design is proposed as a constrained dynamic optimization problem. Hybrid algorithm is used to solve the problem. The empirical results show the importance of using heuristic approaches in real world optimization problem.*

Keywords: Differential evolution, heuristic algorithms, SQP algorithm, fixed PID control.

1. Introduction

The proportional-integral-derivative (PID) controller is widely used in industrial applications in spite of advanced control techniques [1]. The popularity and widespread use of PID controller is attributed to its structural simplicity, performance characteristics and the application of a broad class of dynamic systems.

Several works deal with the linear fixed PID controller under parametric uncertainties called robust fixed gain PID controller. Those tuning strategies are mainly based on optimization approaches. Newton-Raphson algorithm is proposed in [2] to find the proportional-integral (PI) control gains satisfying H_∞ specifications. The initial condition of the Newton-Raphson algorithm is a crucial factor to find good solutions. In [3], the H_∞ static output feedback control technique is proposed to transform a robust PID control method in order to find the optimal PID controller gains in a multi-machine power system. An iterative linear matrix inequalities technique is used to solve the optimization problem. Another approach to deal with the robust PID controller is proposed in [4]. This is based on probabilistic collocation method (PCM) with stochastic parametric uncertainties. Nonlinear programming method is used to compute the optimal PID parameters. In that work the PCM method is compared with the Monte Carlo (MC) Method. The PCM method showed similar performance with less computational effort than the MC method. In [5] presents a robust PID tuning method based on the plant step response experimentally obtained, such that, the transfer function of the plant is not required.

In order to deal with the highly non-convexity H_∞ problem, the higher order linear systems and with the convergence to suboptimal solution in the robust fixed PID controller problem, meta-heuristic algorithms have been used to solve the problem. In this aspect, augmented Lagrangian particle swarm optimization is proposed in [6]. In [7], a heuristic Kalman algorithm (HKA) is proposed to solve unconstrained optimization problem with penalty function and H_∞ performance function. Nevertheless, the algorithm used to solve the optimization problem depends on the problem at hand [8]. There are two main approaches to be considered in the selection of the algorithm and they depend on how the search direction is computed: 1) The gradient based algorithms [9], [10] and 2) the meta-heuristic based algorithms [11].

The gradient based algorithms (GBA) such as sequential quadratic programming (SQP) algorithm, depend on the initial condition and the convexity of the optimization problem [12]. The drawbacks in the GBA are the convergence to local solution near the initial one when the optimization problem is multi-modal and it can not be used in discontinuous optimization problem. Hence, gradient based algorithms do not ensure global optimum and they have limited application.

On the other hand, meta-heuristic based algorithms (MHBA) have been widely used to solve real world optimization problems, [13], [14], [15]. Differential evolution (DE) [16] is an MHBA which can be used to solve continuous, discontinuous, linear, nonlinear, dynamic and static optimization problems. DE performs mutation based on the distribution of the solutions in the current population such that, the search direction depends on the location and selection of individuals. The simplicity, easy of implementation, reliability and high performance made the DE algorithm the most used algorithm in real word optimization problem and one of the algorithms used to compare the performance of a proposal algorithm. Nevertheless, some modifications to the original DE schemes have been done to enhance the explorative and exploitative performance. In [17], the modified versions of the DE are classified into two classes: i) DE integrating an extra component and ii) Modified structures of DE. The first class of DE is of the interest in this paper.

There are several DE variants but the most popular is called DE/rand/1/bin where just one difference (of two randomly chosen individuals added to another solution) is calculated. In this paper, the exploitative performance of the DE/rand/1/bin is enhance by adding an extra component

in the DE search. The exploitative pressure is done by combining a SQP approach.

In the works mentioned above, the robust fixed gain PID controller approaches are stated to linear systems. In this paper a new robust fixed gain PID design approach to handle nonlinear systems is proposed. The importance of using an evolutionary and gradient based algorithm is discussed.

The rest of the paper is organized as follows: The robust fixed gain PID design approach is formulated in Section 2. In Section 3 the DE algorithm is shown. The result and discussion are given in Section 4. Finally, in Section 5 the conclusions are drawn.

2. Robust PID control design problem statement

The purpose of the robust PID control design is to obtain the gains where the performance of the system remains inside the design specifications in spite of unknown parameter variations. So, the robust integrated design is formulated as a nonlinear mono-objective dynamic optimization problem where the control parameters $p \in R^{n_c}$ are optimized by minimizing the robust performance index $\Psi \in R$ (1), subject to the dynamic model of the feedback system and the sensitivity vector of the states (2), static and dynamics inequality constraints (3), static and dynamics equality constraints (4) and lower and upper limits in the input vector u (5). The state vector is represented by $x \in R^n$ with x_0 is the initial state vector for the nonlinear differential equation $\dot{x} = f(x, p, \xi, u, t)$ (dynamic model of the system), $S_\xi = \frac{\partial x}{\partial \xi} \in R^n$ is the sensitivity vector of the state vector x with respect to the unknown parameters ξ, t is the time variable and $u = f_u(x, p, \xi, t) \in R^m$ is the input vector of the dynamic system. The unknown parameters $\xi \in R$ can vary from their nominal one $\bar{\xi}$ accordingly to $\xi = \bar{\xi} + \Delta\xi$. $g(x, p, t)$ and $h(x, p, t)$ are static/dynamic inequality or equality constraint vectors, respectively.The function L is a nonlinear one of class C^1. In this paper the function L is called goal function and the term $\frac{\overline{\partial L}}{\partial \xi}$ i.e., the sensitivity of L with respect to the unknown parameters $\xi \in R$, is called sensitivity of the goal function.

$$\underset{p}{Min}\ \Psi = \underset{p}{Min}\ \left.\frac{\partial \bar{J}}{\partial \xi}^2\right|_{\xi=\bar{\xi}} \qquad (1)$$

$$\bar{J} = \int_0^{t_f} L^2(x, p, \xi, u, t)dt$$

subject to:

1.- Dynamic model of the system and the sensitivity vector of the state, with $x(t_0) = x_0$, $S_\xi(t_0) = 0$ and $u = f_u(x, p, \xi, t)$.

$$\left.\begin{bmatrix} \dot{x} \\ \dot{S}_\xi \end{bmatrix} = \begin{bmatrix} f(x, p, \xi, u, t) \\ \frac{\partial f}{\partial x} S_\xi + \frac{\partial f}{\partial \xi} \end{bmatrix}\right|_{\xi=\bar{\xi}} \qquad (2)$$

2.- Static and dynamic inequality constraint vector:

$$g_j(x, p, t) < 0, \text{ for } j = 1, \ldots, n_{gs} \qquad (3)$$

3.- Static and dynamic equality constraint vector:

$$h_k(x, p, t) = 0, \text{ for } k = 1, \ldots, n_{hs} \qquad (4)$$

4.- Lower and upper limits in the input vector u:

$$u_{\min} \leq u \leq u_{\max} \qquad (5)$$

The gradient $\frac{\partial \bar{J}}{\partial \xi}$ is computed as in (6), where $\frac{\overline{\partial L}}{\partial \xi} = \left(\frac{\partial L}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial L}{\partial u}\left(\frac{\partial u}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial u}{\partial \xi}\right) + \frac{\partial L}{\partial \xi}\right)$.

$$\frac{\partial \bar{J}}{\partial \xi} = 2\int_0^{t_f} L\frac{\overline{\partial L}}{\partial \xi}dt \qquad (6)$$

The sensitivity vector S_ξ is obtained as [18]:

$$\dot{S}_\xi = \frac{\partial f}{\partial x}S_\xi + \frac{\partial f}{\partial u}\left(\frac{\partial u}{\partial x}S_\xi + \frac{\partial u}{\partial \xi}\right) + \frac{\partial f}{\partial \xi} \qquad (7)$$

It is important to remark that the minimization of the robust performance function $\Psi = 4L^2\left(\frac{\partial L}{\partial \xi}\right)^2$ implies the minimization of both terms, the goal function L and the sensitivity of the goal function $\frac{\overline{\partial L}}{\partial \xi}$.

In the next subsections, the robotic system, the design variables, the robust performance function and constraints for the robust PID control design of a particular problem are detailed.

2.1 Robotic system

The robust PID control design approach is applied to the $3R$ manipulator with a parallelogram five-bar mechanism. It is considered the mass of the end-effector (mass of the fifh link m_5 in Fig. 1) as the unknown parameters, i.e., $\xi = m_5$. We assume that a payload can be added to the tip of the end-effector, changing the mass of the fifh link.

The $3R$ manipulator with a parallelogram five-bar mechanism presents three degree of freedom in the joint space which provide the ability to move the tip of the end-effector in the plane $X - Z$ with an orientation $\bar{\phi}$ with respect to the axis X of the inertial coordinate system $X - Z$. The parallelogram five-bar mechanism, included into the $3R$ robot, achieves a higher precision and a higher stiffness than a $3R$ robot without the parallelogram five-bar mechanism [19]. The $3R$ manipulator is shown in Fig. **??**, where q_i, $\dot{q}_i$, $\ddot{q}_i$, m_i, l_i, l_{c_i}, $I_i \ \forall\ i = 1, 2, ..., 4$ are the joint position, joint velocity, joint acceleration, mass, length, mass center length, inertia of the $i-th$ link length, $(\hat{x}, \hat{z})$ and $\hat{\phi}$ are the Cartesian coordinate and the angular position of the manipulator's end-effector, respectively.

Let $x = [x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9]^T = [q_1, q_2, q_3, \dot{q}_1, \dot{q}_2, \dot{q}_3, \int_0^\tau e_1(\tau)d\tau, \int_0^\tau e_2(\tau)d\tau, \int_0^\tau e_3(\tau)d\tau]^T \in R^9$

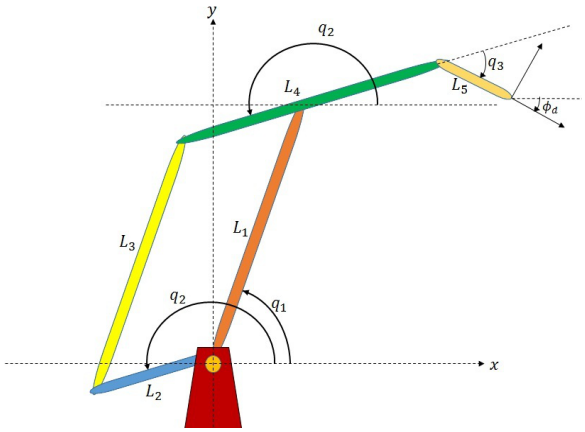


Fig. 1: Schematic diagram of the $3R$ manipulator with a parallelogram five-bar mechanism.

the state variable vector and $u = [u_1,\ u_2,\ u_3]^T \in R^3$ the input vector, the dynamic model of the $3R$ manipulator in the state space x is shown in (8), where $M \in R^{3\times3}$ is the inertia matrix, $C[x_3,\ x_4,\ x_5]^T \in R^3$ is the centrifugal and Coriolis force vector, $G \in R^3$ is the gravity vector, $e_i = \bar{x}_i - x_i$, $\dot{e}_i = \bar{x}_{i+3} - x_{i+3}\ \forall\ i = 1, 2, 3$ is the joint position and velocity error, respectively and the desired state vector is represented by $\bar{x}_i\ \forall\ i = 1,\ 2,\ ...,\ 6$.

$$\dot{x} = \begin{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T \\ M^{-1}\left(u - C\begin{bmatrix} x_3 & x_4 & x_5 \end{bmatrix}^T - G\right) \\ \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix}^T \end{bmatrix} \tag{8}$$

$$\frac{dx}{dt} = f\left(x, p, \xi, u\right) \tag{9}$$

As it is assumed PID controllers, the last three equations in (8) must be included into the robot dynamics. The PID controller is shown in (10), where k_{p_i}, k_{i_i}, k_{d_i} are the proportional, integral and derivative gains, respectively.

$$u_i = k_{p_i} e_i + k_{d_i} \dot{e}_i + k_{i_i} \int_0^t e_i(\tau) d\tau \tag{10}$$

2.2 Design variables

In this paper the design variables are the gains of the PID controller. Hence, the design variable vector is shown in (11).

$$p = [k_{p_1},\ k_{i_1},\ k_{d_1},\ k_{p_2},\ k_{i_2},\ k_{d_2},\ k_{p_3},\ k_{i_3},\ k_{d_3}]^T \in R^9 \tag{11}$$

2.3 Robust performance function

The design objective is to provide the gains of the PID control where the end-effector of the $3R$ manipulator follows a desired trajectory and orientation in the Cartesian Space. In addition, the trajectory tracking must be as insensitive as possible to variations at the end-effector's payload. As the end-effector trajectory can be map to the joint space of the $3R$ manipulator by using the inverse kinematics [20], and the inverse kinematics does not depend of the control design variables (PID gains), then the joint position tracking error is chosen as the goal function (i.e., $L_i = e_i$). Considering the mass of the end-effector as the unknown parameters $\xi = m_5$ (mass of the fifh link m_5 in Fig. xxx), the robust performance function Ψ is showed in (12). The unknown parameters can vary from the nominal one accordingly to $\xi = \bar{\xi} + \Delta\xi$, where the nominal unknown parameter $\bar{\xi}$ is chosen accordingly to the mass of the end-effector without payload and its variations $\Delta\xi$ is the mass of the added payload.

$$\Psi = \sum_{i=1}^{3} \frac{\partial \bar{J}_i}{\partial \xi}^2 \Bigg|_{\xi=\bar{\xi}} \tag{12}$$
$$\bar{J}_i = \int_0^{t_f} e_i^2 dt$$

When the robust performance function (12) is optimized, the variations of the joint position error due to the changes at the unknown parameters $\xi = m_p$, as well as the joint position error $e_i = \bar{x}_i - x_i$, are simultaneously optimized within the same performance function, as it is observed in (6).

The desired end-effector position $(\bar{\hat{x}}, \bar{\hat{z}})$ and its orientation $\bar{\hat{\phi}}$ are described by the hypocycloid trajectory and sinusoidal trajectory, given by (13)-(15), where t is the time variable.

$$\bar{\hat{x}} = 0.2 + 0.08181\cos(1.2566t) + 0.01818\cos(5.6548t) \tag{13}$$

$$\bar{\hat{z}} = 0.1 + 0.08181\sin(1.2566t) - 0.01818\sin(5.6548t) \tag{14}$$

$$\bar{\hat{\phi}} = 0.4363\sin(2.0943t) \tag{15}$$

In order to compute the desired state vector $\bar{x}_i$ from the end-effector desired trajectory (13)-(15), the inverse kinematic of the $3R$ manipulator must be obtained. The inverse kinematics is show in (16)-(18).

$$\bar{x}_1 = A\tan 2\left(\frac{\bar{\hat{z}}_m}{\bar{\hat{x}}_m}\right) - \gamma \tag{16}$$

$$\bar{x}_2 = \bar{x}_1 + q_2' + \pi \tag{17}$$

$$\bar{x}_3 = \bar{\hat{\phi}} - \bar{x}_2 + \pi \tag{18}$$

where
$\gamma = A\tan 2\left(\frac{l_4\sin(q_2')}{l_1 + l_4\cos(q_2')}\right)$

$$q_2' = A\tan 2\left(\frac{-\sqrt{4l_1^2l_4^2-\left(\tilde{x}_m^2+\tilde{z}_m^2-l_1^2-l_4^2\right)^2}}{\tilde{x}_m^2+\tilde{z}_m^2-l_1^2-l_4^2}\right)$$

2.4 Constraints

The dynamic behavior of the $3R$ manipulator with a parallelogram five-bar mechanism with its PID control system is included as a dynamic constraint. Those are stated in (9) and (10).

In real application, the input continuous torque applied to the manipulator joints has physical limits. If such limits are not considered, the PID control system may become unstable in real application which often incur a physical damage of the manipulator. Hence, the bounds of the input toque vector u, is considered as a dynamic constraint (19), where $u_{BOUND} = 3Nm$ is the input toque bound.

$$g_i(t) : |u_i(t)| - u_{BOUND} \leq 0 \ \forall i = 1, 2, 3 \qquad (19)$$

Other important constraint is the maximum response of the actuators at each sampling time. This constraint is stated in (20), (21), $\forall i = 1, 2, 3$. Without including that constraint, the robotic manipulator can reach singular configurations. Singular configurations can provide uncontrollable movements. Then, if that constraint is not fulfilled, the dynamic behavior of the closed-loop system must be stopped, i.e., the performance function is not evaluated and it is assigned a maximum value.

$$g_{i+3}(t) : |q_i(t + \Delta t) - q_i(t)| - 0.05 < 0 \qquad (20)$$
$$g_{i+6}(t) : |\dot{q}_i(t + \Delta t) - \dot{q}_i(t)| - 11.98 < 0 \qquad (21)$$

2.5 Optimization problem formulation

The optimization problem for the robust PID control design of the $3R$ manipulator consists on finding the optimal design parameter vector p^* (11) such that both the sensitivity of the joint position error and the joint position error (12) are simultaneously minimized subject to dynamic constraint (23) of the nonlinear differential equation that describes the dynamic behavior of the system (8) with the control signal u (10), the dynamic inequality constraint vector $g(t) \in R^9$ (24) that includes the bounds in the control signal (19) and the bounds in the maximum response of the actuators at each sampling time (20)-(21).

$$\underset{p \in R^9}{Min}\ \Psi \qquad (22)$$

Subject to:

$$\dot{x} = f\left(x, p, \xi, u\right)) \qquad (23)$$
$$g(x, p, t) < 0 \qquad (24)$$

```
1   BEGIN
2     G = 0
3     Initialization stage
4     Do
5       For i = 1 to NP Do
6         Mutation and crossover stages
7         Selection stage
8       End For
9       G = G + 1
10    While (G ≤ G_Max)
11  END
```

Fig. 2: DE algorithm.

3. Differential evolution

The DE algorithm is composed by four stages (see Fig. 2): Initialization, mutation, crossover and selection [16]. The main differences among the DE variants are in the mutation and crossover states (rows 9-17 in Fig. 2). The general convention used to name the DE variant is DE/x/y/z. DE means Differential Evolution, x represents the way that the vector is perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exponential or binomial). The most common DE variants are DE/rand/1/bin, DE/rand/1/exp, DE/best/1/bin, DE/best/1/exp, DE/current-to-rand/1, DE/current-to-best/1, DE/current-to-rand/1/bin and DE/rand/2/dir. In this paper DE/rand/1/bin is used.

In the initialization stage, the initial population design vector $x^i_{j,G=0}$ is randomly generated for all population, where $G = 0$ indicates the initial generation which the vector belongs, $i \in [1, .., NP]$ is a population index, $j \in [1, ..., D]$ is the design parameter index. This stage is shown in (25) where $rand_j(0, 1) \in [0, 1)$ is a uniformly distributed random number generator and $b_{j,U}$, $b_{j,L}$ indicate the upper and lower limit for the $j-th$ design parameter. For each initial design vector, its objective function is computed.

```
For i = 1 to NP Do
  For j = 1 to D Do
    x^i_{j,G=0} = rand_j(0,1)(b_{j,U} − b_{j,L}) + b_{j,L}
  End For
  Evaluate J(x⃗^i_{G=0})
End For
```
(25)

Once the initial population vector is initialized, the mutation stage creates a mutant vector $\vec{v}^i_G$ by the mutation and recombination of the individuals of the population. In the crossover stage, trial vector $\vec{u}^i_G$ is generated when the target vector $x^i_{j,G}$ is crossed with the mutant vector $\vec{v}^i_G$. The mutation and the crossover stages depend on the DE variant. The mutation and crossover variants of the DE/rand/1/bin is shown as in 26.

$$u_j^i = \begin{cases} \text{if rand}_j(0,1) < CR \text{ or } j = j_{rand} \\ \quad v_j^i = x_j^{r_3} + F(x_j^{r_1} - x_j^{r_2}) \\ \text{otherwise} \\ \quad x_j^i \end{cases} \tag{26}$$

The scale factor $F \in (0,1)$ in the mutation process, is a positive real number that controls the influence of the selected individuals in order to generate the mutant vector. The indexes r_1, r_2 and r_3 are randomly chosen from the range $[1, NP]$. Those indexes can not have the same value.

The crossover probability $CR \in [0,1]$ controls the influence of the parent vector in the generation of the offspring (higher values mean less influence of the parent vector, hence higher influence of the mutant vector).

Finally, the last stage involves a selection process between the trial vector $\vec{u}_G^i$ and the target vector $x_{j,G}^i$. First, the performance function is evaluated with the trial vector and then an elitism procedure is done where the best of them pass to the next generation. In (27), the selection stage is shown.

$$\begin{array}{l} \text{Evaluate } J(\vec{u}_G^i) \\ \textbf{If } (J(\vec{u}_G^i) < J(\vec{x}_G^i)) \textbf{ Then} \\ \quad \vec{x}_{G+1}^i = \vec{u}_G^i \\ \quad J(\vec{x}_{G+1}^i) = J(\vec{u}_G^i) \\ \textbf{Else} \\ \quad \vec{x}_{G+1}^i = \vec{x}_G^i \\ \quad J(\vec{x}_{G+1}^i) = J(\vec{x}_G^i) \\ \textbf{End If} \end{array} \tag{27}$$

4. Results and discussion

In this work, the DE/rand/1/bin is programmed in Matlab Release 7.9 on a Windows platform. Computational experiments were performed on a PC with a 2.8 GHz Core $i7$ Duo processor and 16 GB of RAM. Five independent runs are carried out. The parameters of the DE algorithm are proposed as follows: the population size NP consists of 100 individuals. The scaling factor F and the crossover constant CR are randomly generated in the interval $0.3 \leq F \leq 0.9$ at each generation, and in the interval $0.8 \leq CR < 1$ at each optimization process. The stop criterion is when the number of generations is fulfilled $G_{Max} = 400$.

4.1 Performance of the DE and SQP algorithm

In Table 1 the performance for five independent runs of the DE algorithm are shown. The term $mean(J)$ and $\sigma(J)$ are the mean and the standard deviation of the performance function of the population in the $400th$ generation. The best performance function value found in the last generation is place in the column J^{*DE}. J_{Eval} and $J_{NotEval}$ are the number of times that the performance function is evaluated or not evaluated, respectively. The number of times that the performance function is not evaluated is when the constraint (21) is not fulfilled. The third column $Time[hr]$ is the convergence time.

It is observed in Table 1 that all runs converge to a similar performance function (see column J^{*DE}) and all population in the last generation converge to a similar value, as indicate in the column $mean(J)$ and $\sigma(J)$. This indicates that the explorative performance of the DE/rand/1/bin algorithm makes the convergence of the algorithm is towards the same performance function value. On the other hand, there are several times that the performance function is not evaluated (see column $J_{NotEval}$). This reduces the convergence time to get the solution, hence the importance of including the constraint (21) in the optimization problem.

In order to enhance the exploitative of the individuals in the DE algorithm, the best individual in different generations is used as the initial condition of the SQP algorithm. The stop criterion is when the number of iteration $ITR = 100$ is fulfilled. In Table 2 the performance of the SQP algorithm is shown. The column J_{IC} indicate the performance function value with the initial condition and the column J^{*SQP} is the performance function value when the number of iteration is satisfied. It is observed that the performance function value converges to suboptimal solutions near the initial ones, i.e., the SQP algorithm is highly sensitivity to the initial condition. This indicates that the optimization problem is a multimodal one. As expected, the convergence time of the SQP algorithm is less than the ones of the differential evolution. The SQP algorithm is also test with random initial condition but it does not converge to a solution (it is not an easy task to find good initial condition). For that reason the results with random initial condition are not shown.

4.2 Optimal design

The design performance with the design parameters resulting from the SQP algorithm is compared with the design performance with the design parameters given by a trial & error procedure. In Table 3 the design parameters obtained by the SQP algorithm and by the trial & error procedure are shown.

In Fig. 3 and Fig. 4, the trajectory tracking with the robust fixed PID control design and the trial & error PID design are shown with no load and with a load of $0.1247kg$. With no load, both designs track the desired trajectory. Nevertheless, with load, the robust fixed PID control design tracks the desired trajectory, while the trial & error PID design a trajectory tracking error is observed.

In Fig. 5 and Fig. 6, the control signals with both designs are displayed. When there is not load, both designs satisfy the input torque bounds given by the dynamic constraint (19). With load, the robust fixed PID control design fulfills the input torque bounds while the trial & error PID design does not satisfy.

The trajectory tracking performance of robust fixed PID control design is better than the one of trial & error PID design when the robotic manipulator handle different loads

Run	$mean(J)$	$\sigma(J)$	J^{*DE}	J_{Eval}	$J_{NotEval}$	$Time[hr]$
1	$5.46e-13$	$3.76e-16$	$5.45e-13$	20044	20056	35.63
2	$5.45e-13$	$2.92e-16$	$5.44e-13$	27481	12619	48.61
3	$5.46e-13$	$3.95e-16$	$5.45e-13$	20268	19832	35.18
4	$5.45e-13$	$4.49e-16$	$5.44e-13$	23750	16350	40.13
5	$5.45e-13$	$2.72e-16$	$5.44e-13$	29845	10255	51.97

Table 1: Experimental results of the DE/rand/1/bin algorithm.

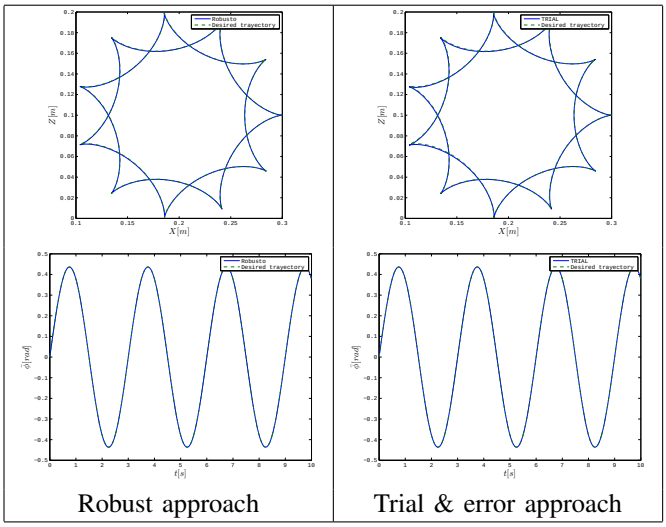


Fig. 3: Trajectory tracking of the end-effector of the robotic manipulator without load.

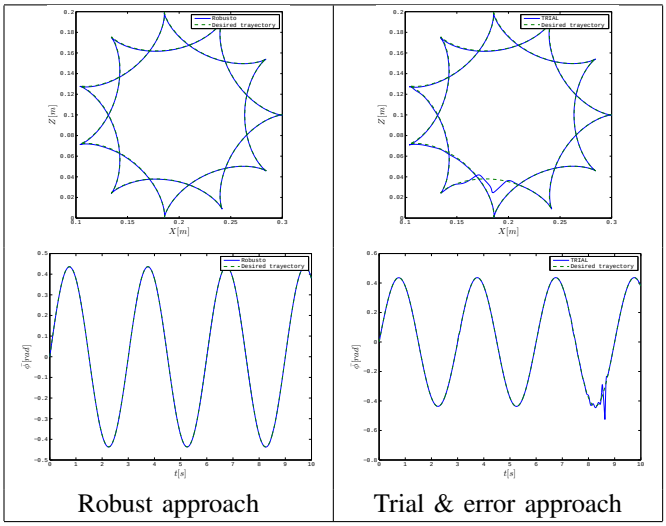


Fig. 4: Trajectory tracking of the end-effector of the robotic manipulator with load of $0.1247Kg$.

Table 2: SQP algorithm with different initial condition given by different generations in the DE algorithm.

$Generation$	J_{IC}	J^{*SQP}	$Time[hr]$
100	$28.52e-13$	$10.8e-13$	2.96
200	$11.07e-13$	$8.39e-13$	3.11
300	$5.50e-13$	$5.50e-13$	12.10
400	$5.47e-13$	$5.44e-13$	11.98

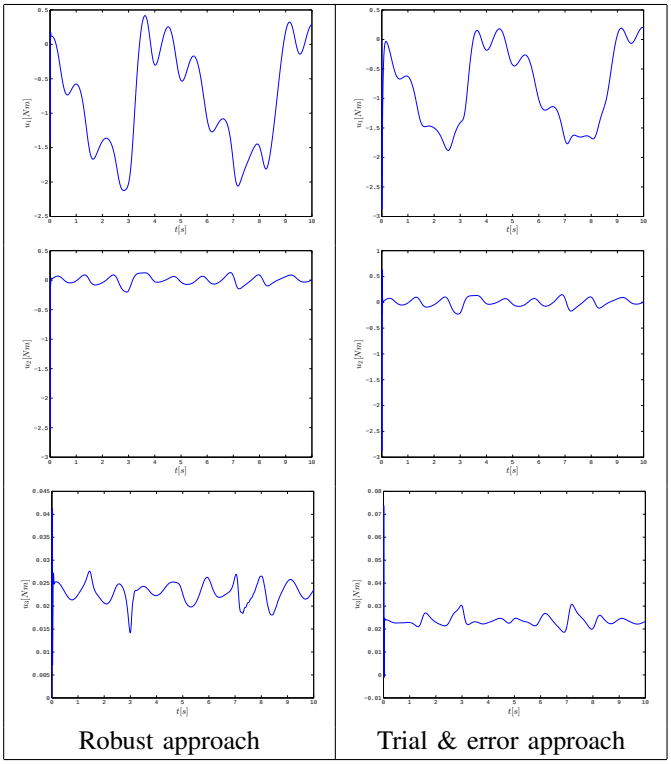


Fig. 5: Control signal for the trajectory tracking without load.

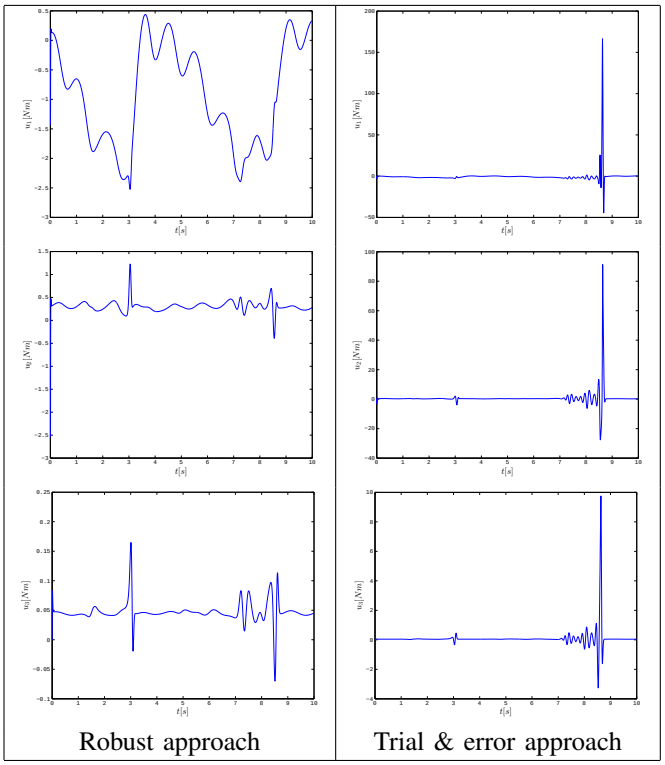
satisfying the input torque bounds. Hence, the proposed robust fixed PID control design is validated.

5. Conclusion

In this paper the formulation of the robust fixed PID design for a robotic manipulator is proposed as an optimization problem. The optimization problem is solved by combining the DE and SQP algorithm. The empirical results show that the combination of an evolutionary algorithm with a gradient one make a good exploration and exploitation

Table 3: Design parameters provided by the SQP algorithm.

$PIDdesignapproach$	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
Robust fixed PID control	999.9709	9.7566	9.8934	999.6667	9.9437	8.8279	45.1166	9.9890	0.0138
Trial & error	500	42	20	3000	4	10	44	50	0.005



Robust approach | Trial & error approach

Fig. 6: Control signal for the trajectory tracking with load of $0.1247Kg$.

of the individuals in the search space. The evolutionary algorithm is highly useful when the SQP algorithm presents high sensitivity to the initial condition such that the searching of the initial condition is not an easy task.

From an engineering design point of view, a heuristic approach must be considered first and then a gradient approach in order to finely search in the complete space and hence finding the best design.

The trajectory tracking performance of robust fixed PID control design is better than the one of trial & error PID design when the robotic manipulator handle different loads satisfying the input torque bounds. Hence, the proposed robust fixed PID control design is numerically validated.

Acknowledgment

The author acknowledges support from COFAA and SIP of the Instituto Politécnico Nacional through project no. SIP-20140926. and the support from the Consejo Nacional de Ciencia y Tecnología (CONACyT) through project no. 182298.

References

[1] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2001.

[2] K. J. Ástr "Design of {PI} controllers based on non-convex optimization," *Automatica*, vol. 34, no. 5, pp. 585–601, 1998.

[3] H. Bevrani, T. Hiyama, and H. Bevrani, "Robust {PID} based power system stabiliser: Design and real-time implementation," *International Journal of Electrical Power & Energy Systems*, vol. 33, no. 2, pp. 179–188, 2011.

[4] P. Luu, T. Duong, and M. Lee, "Robust {PID} controller design for processes with stochastic parametric uncertainties," *Journal of Process Control*, vol. 22, no. 9, pp. 1559–1566, 2012.

[5] J.-C. Jeng, W.-L. Tseng, and M.-S. Chiu, "A one-step tuning method for {PID} controllers with robustness specification using plant step-response data," *Chemical Engineering Research and Design*, vol. 92, no. 3, pp. 545–558, 2014.

[6] T.-H. Kim, I. Maruta, and T. Sugie, "Robust {PID} controller tuning based on the constrained particle swarm optimization," *Automatica*, vol. 44, no. 4, pp. 1104–1110, 2008.

[7] R. Toscano and P. Lyonnet, "Robust {PID} controller tuning based on the heuristic kalman algorithm," *Automatica*, vol. 45, no. 9, pp. 2099–2106, 2009.

[8] E. Mezura-Montes, J. Velzquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," *Genetic And Evolutionary Computation Conference*, pp. 485–492, 2006.

[9] S. S. Rao, *Engineering Optimization*. John Wiley & Sons, 1996.

[10] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Wiley, 2006.

[11] C. A. Coello-Coello, G. B. Lamont, and D. A. Van-Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007.

[12] M. Villarreal-Cervantes, C. A. Cruz-Villar, and J. Alvarez-Gallegos, "Synergetic structurecontrol design via a hybrid gradient-evolutionary algorithm," *Optimization and Engineering*, pp. 1–29, 2014.

[13] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Alvarez-Gallegos, and E. A. Portilla-Flores, "Differential evolution techniques for the structure-control design of a five-bar parallel robot," *Engineering Optimization*, vol. 42, no. 6, pp. 535–565, 2010.

[14] E. A. Portilla-Flores, E. Mezura-Montes, J. Alvarez-Gallegos, C. A. Coello-Coello, C. A. Cruz-Villar, and M. G. Villarreal-Cervantes, "Parametric reconfiguration improvement in non-iterative concurrent mechatronic design using an evolutionary-based approach," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 757 – 771, 2011.

[15] J. Cabrera, F. Nadal, J. Muoz, and A. Simon, "Multiobjective constrained optimal synthesis of planar mechanisms using a new evolutionary algorithm," *Mechanism and Machine Theory*, vol. 42, 2007.

[16] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*. Springer, December 2005.

[17] N. Ferrante and T. Ville, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 61–106, 2010.

[18] P. Frank, *Introduction to system sensitivity theory*. Academic press, 1978.

[19] T. Lung-Wen, *Robot analysis: The mechanics of serial and parallel manipulators*. John Wiley & Sons, February 1999.

[20] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. John Wiley & Sons, 2004.

Optimal Selection of Milling Parameters for Maximum Profit Rate by Genetic Algorithms

Libao An[1], Hong Zhang[2], and Peiqing Yang[3]
[1]College of Mechanical Engineering, Hebei United University, Tangshan, Hebei, China
[2]College of Life Sciences, Hebei United University, Tangshan, Hebei, China
[3]Fushun Mechanical Equipment Manufacturing Co., Ltd. Fushun, Liaoning, China

Abstract - *Fabricating machine parts and components with low cost, less time, and high quality is the major objective in the metal cutting industry. Machining parameter optimization plays an important role in achieving this goal. It is an essential part in digital manufacturing with little or no involvement from humans. In this paper, the parameter optimization problem for face-milling operations is studied with the purpose to maximize the product profit rate. Various practical conditions and restricts from the cutting operation are considered as constraints. The mathematical optimization model was solved using a traditional genetic algorithm for optimal machining parameter values. An example from the literature is presented to illustrate the model and solution method. Higher profit rate values were obtained by the proposed approach compared to the previous research.*

Keywords: Milling Parameter, Optimization, Maximum Profit Rate, Genetic Algorithms

1 Introduction

Producing products with low cost, less time, and high quality is continuously a central concern for manufacturing enterprises. Machining parameter optimization plays an important role in reducing machining time and production cost in metal cutting industry. It is an essential part of a computer aided process planning system in digital manufacturing with little or no involvement from human beings. Therefore, it is critical for machining process optimization and management. Machining parameter optimization usually involves the optimal selection of cutting speed, feed rate, depth of cut and the number of passes for a machining process. In most cases in practice, machining parameters have been selected from machining database or handbooks. The cutting regimes given in such a way are actually not the optimal values [1]. Machining parameter optimization problems have been intensively studied for chasing maximum production rate/minimum production time [2-4] or minimum production cost [5-7]. The maximum profit rate criterion involves maximizing the return on the operation in a given time interval. It was first put forward as "the maximum profit" in 1964 [8] and has been named "the maximum profit rate" since 1966 [9]. It is the criterion to be recommended when there is insufficient capacity for a specific time interval for a machine shop. Maximum profit rate is an appropriate criterion for selecting optimum machining parameters rather than the conventional criteria of maximum production rate or minimum production cost.

In this paper, the unit profit rate is optimized for face-milling operations. A variety of realistic machining conditions and quality specifications are considered as constraints. The problem is solved by a traditional genetic algorithm (GA). An example is given to illustrate the model and solution procedure.

2 Milling process model

2.1 Objective function

For a metal cutting process of a machine part, the profit rate *PR* (\$/min) can be defined as [10],

$$PR = \frac{SP - UC - C_{mat}}{UT} \tag{1}$$

where *SP* denotes the sale price of the product (\$); C_{mat} represents the cost of raw material (\$); *UC* and *UT* are respectively unit production cost and unit machining time. Unit machining time *UT* (min) is comprised of actual machining time t_m, machine idle time t_l, and tool replacement time t_r. Dividing a face-milling process into one finish pass and *n* rough passes, actual machining time, $t_m = \frac{\pi D L_{ts}}{1000 V_s f_s Z} + \sum_{i=1}^{n} \frac{\pi D L_{tr}}{1000 V_{ri} f_{ri} Z}$, where *D* (mm) is the diameter of the workpiece; L_{ts} (mm) is the finish cutting travel length and $L_{ts} = L + 0.5(D - \sqrt{D^2 - B^2}) + 3$; L_{tr} (mm) is the rough cutting travel length and $L_{tr} = L + D + 3$; *L* (mm) and *B* (mm) are respectively the length and width of the workpiece; *Z* is the tooth number of the cutting tool; V_s (m/min) and f_s (mm/tooth) are respectively cutting speed and feed rate for the finish pass; V_{ri} (m/min) and f_{ri} (mm/tooth) are respectively cutting speed and feed rate for the *i*-th rough

pass. Machine idle time is defined as $t_l = t_p + t_i$ [5], where t_p (min) is preparation time, and t_i (min) is idle tool motion time. Therefore, $t_l = t_p + n(h_1 L_{tr} + h_2) + (h_1 L_{ts} + h_2)$, where h_1 (min/mm) is tool travel time and h_2 (min) is tool approach/depart time. Tool replacement time t_r can be given by $t_r = t_e Z \frac{t_m}{T}$, where t_e (min) is tool exchange time, T (min) is tool life. Then

$$UT = t_m + t_l + t_r = t_s + \sum_{i=1}^{n} t_{ri} + t_p \quad (2)$$

where

$$t_s = (1 + \frac{t_e Z}{T}) \frac{\pi D L_{ts}}{1000 V_s f_s Z} + (h_1 L_{ts} + h_2) \quad (3)$$

$$t_{ri} = (1 + \frac{t_e Z}{T}) \frac{\pi D L_{tr}}{1000 V_{ri} f_{ri} Z} + (h_1 L_{tr} + h_2) \quad (4)$$

Similarly, unit production cost UC ($/piece) is comprised of actual machining cost MC, machine idle cost IC, tool replacement cost RC, and tool cost TC. MC is based on actual machining time t_m (min) and labor cost, k_o ($/min), including overhead, then $MC = k_o t_m$. The machine idle cost is defined as $IC = k_o t_l$. Tool replacement cost RC and tool cost TC can be respectively given by $RC = k_o t_r$ and $TC = k_t Z \frac{t_m}{T}$, where k_t ($) is tool cost. Then

$$UC = MC + IC + RC + TC = UC_s + \sum_{i=1}^{n} UC_{ri} + k_o t_p \quad (5)$$

where

$$UC_s = \left(k_o + \frac{k_t Z}{T} + \frac{k_o t_e Z}{T} \right) \frac{\pi D L_{ts}}{1000 V_s f_s Z} + k_o (h_1 L_{ts} + h_2) \quad (6)$$

$$UC_{ri} = \left(k_o + \frac{k_t Z}{T} + \frac{k_o t_e Z}{T} \right) \frac{\pi D L_{tr}}{1000 V_{ri} f_{ri} Z} + k_o (h_1 L_{tr} + h_2) \quad (7)$$

2.2 Constraints

For given cutting conditions, there exist reasonable ranges of cutting speed V, feed rate f, and depth of cut d, for either a finish or rough pass:

$$V_{\min} \le V_s \le V_{\max}, \ f_{\min} \le f_s \le f_{\max}, \ d_{s,\min} \le d_s \le d_{s,\max} \quad (8)$$

$$V_{\min} \le V_{ri} \le V_{\max}, \ f_{\min} \le f_{ri} \le f_{\max}, \ d_{r,\min} \le d_r \le d_{r,\max} \quad (9)$$

Tool life in face-milling can be given by

$$T = \frac{C_v K_v D^{q_v}}{V_s d_s^{x_v} f_s^{y_v} B^{s_v} Z^{p_v}} = \frac{C_v K_v D^{q_v}}{V_{ri} d_{ri}^{x_v} f_{ri}^{y_v} B^{s_v} Z^{p_v}} \quad (10)$$

where C_v, K_v, q_v, p_v, x_v, y_v, and s_v are constants; T is tool life (mm) and we assume the tool lives are identical in finish and rough machining operations and require the same tool replacement time. Surface finish requirements can by given by

$$f_s \le \sqrt{r_e R_{s,\max} / 0.0321} \quad (11)$$

$$f_{ri} \le \sqrt{r_e R_{r,\max} / 0.0321} \quad (12)$$

where r_e is cutter nose radius (mm); $R_{s,\max}$ and $R_{r,\max}$ are surface roughness requirements (mm) for finish machining and rough machining, respectively. Cutting force constraints can be written as

$$F = \frac{C_f K_f B^{s_f} Z^{p_f} d_s^{x_f} f_s^{y_f}}{D^{q_f}} \le F_{\max} \quad (13)$$

$$F = \frac{C_f K_f B^{s_f} Z^{p_f} d_{ri}^{x_f} f_{ri}^{y_f}}{D^{q_f}} \le F_{\max} \quad (14)$$

where C_f, K_f, x_f, y_f, p_f, q_f and s_f are constants; $F_{\max}$ is maximum available cutting force (kgf). Cutting power can be derived by multiplying cutting force and cutting speed,

$$P = \frac{F V_s}{6120\eta} = \frac{C_f K_f B^{s_f} Z^{p_f} V_s d_s^{x_f} f_s^{y_f}}{6120\eta D^{q_f}} \le P_{\max} \quad (15)$$

$$P = \frac{F V_{ri}}{6120\eta} = \frac{C_f K_f B^{s_f} Z^{p_f} V_{ri} d_{ri}^{x_f} f_{ri}^{y_f}}{6120\eta D^{q_f}} \le P_{\max} \quad (16)$$

where η is the efficiency of the machine tool and $P_{\max}$ is the maximum power (kW). The total depth of cut d_t can be expressed as

$$d_t = d_s + \sum_{i=1}^{n} d_{ri} \quad (17)$$

V_s, f_s, d_s, V_{ri}, f_{ri}, d_{ri} and n are decision variables of the model.

3 Solution method

Genetic Algorithms (GAs) is a particular class of evolutionary algorithms that make use of techniques motivated by evolutionary biology such as selection,

mutation, and crossover. When maximizing profit rate to select the optimal machining parameters using a traditional GA in this work, decision variables V_s, f_s, d_s, V_{ri}, f_{ri}, d_{ri} are represented by binary numbers and these numbers are aligned in a long binary string which is called a chromosome. The population size in this research is 100. Crossover is the operation to exchange some part of two chromosomes to generate new offspring (crossover rate is 80% in this paper). This operation is important for exploring the whole search space rapidly. Mutation operation randomly alters each bit of a binary string after crossover with a small probability (mutation rate is 0.05 in this work) to provide a small uncertainty to the new chromosome. In the paper, 20% chromosomes with best fitness values are kept within the population to avoid losing the best strings, and the rest chromosomes apply to a crossover and mutation operation during each reproduction cycle. The same population size is maintained during the evolution process. After crossover and mutation, a new generation forms and the machining parameter values are calculated. After a certain number of generations (2000 iterations in this research), the GA should converge to the best chromosome, which represents the optimal or near-optimal solutions to the problem.

We assume that the unwanted material should be cut off with one finish pass and n rough passes ($n \geq 1$). Therefore, the total number of cutting passes is $N=n+1$. The total depth of cut considered in this paper is 2.0mm$\leq d_t \leq$8.0mm.

When n=1, there are 5 decision variables: V_s, V_r, f_s, f_r, d_s ($d_r = d_t - d_s$). When n=2, there are 8 decision variables: V_s, V_{r1}, V_{r2}, f_s, f_{r1}, f_{r2}, d_s, d_{r2} ($d_{r1} = d_t - d_s - d_{r2}$). The decision variables for other n values can be listed in the same way.

Based on a given value of the total depth of cut and the feasible ranges of rough and finish passes, possible numbers of total passes can be calculated. The algorithm performs computation for each case and compares the results for an optimal pass number.

Chromosomes in a population evolve based on their fitness values. In this paper, the fitness is chosen to be equal to the objective function value *PR*,

$$F_i = PR_i,\ i=1,2,3\ldots100. \tag{18}$$

The total fitness for a population is

$$F = \sum_{i=1}^{100} F_i \tag{19}$$

Selection probability for each chromosome is

$$p_i = \frac{F_i}{F},\ i=1,2,3\ldots100. \tag{20}$$

Cumulative probability for each chromosome is

$$q_i = \sum_{j=1}^{i} p_j,\ i=1,2,3\ldots100. \tag{21}$$

A chromosome with a higher fitness value has a higher probability of being selected to survive. In order to avoid losing the best strings, 20% chromosomes with best fitness values in a population are selected to directly enter the new population, and crossover is conducted to the rest 80% chromosomes. Mutation rate is a probability to alter one gene (one bit of a chromosome). The mutation rate should be very low and is set as 0.05 in this paper.

4 Example problem and solutions

The face-milling example given in Table 1 [11] is considered in this paper. Cemented carbide cutting tools are used to machine a gray cast iron workpiece (190HB). The same example was used to illustrate an optimal solution approach for minimum total unit production cost, also using Genetic Algorithms [12].

Table 1. Data for the given example

L = 240mm, D = 160mm, r_e = 1mm, B = 100mm, Z = 16
k_o = 0.5\$/min, k_t = 2.5\$, t_e = 1.5min, t_p = 0.75min, SP = 25\$, C_{mat} = 0.5\$
$h_1 = 7\times10^{-4}$min/mm, h_2 = 0.3min
V_{max} = 300m/min, V_{min} = 50m/min, f_{max} = 0.6mm/tooth, f_{min} = 0.1mm/tooth, $d_{s,max}$ = 2mm, $d_{s,min}$ = 0.5mm, $d_{r,max}$ = 4mm, $d_{r,min}$ = 1mm
T = 240min, $R_{s,max}$ = 0.0025mm, $R_{r,max}$ = 0.025mm, F_{max} = 815.77kgf, P_{max} = 10kW, η = 0.8
C_v = 445, l = 0.32, x_v = 0.15, y_v = 0.35, p_v = 0, q_v = 0.2, s_v = 0.2, K_v = 1.0
C_f = 54.5, x_f = 0.9, y_f = 0.74, s_f = 1.0, p_f = 1.0, q_f = 1.0, K_f = 1.0

Substituting the data into the objective function, we have

Maximize:

$$PR = \frac{23.83394 - 9.07346 V_s^{-1} f_s^{-1} - \sum_{i=1}^{n} (0.24119 + 5.86623 V_{ri}^{-1} f_{ri}^{-1})}{13.9267 V_s^{-1} f_s^{-1} + \sum_{i=1}^{n} (0.4824 + 9.5499 V_{ri}^{-1} f_{ri}^{-1}) + 1.3321} \tag{22}$$

Subject to

$$50 \le V_s \le 300$$

$$50 \le V_{ri} \le 300$$

$$0.5 \le d_s \le 2.0$$

$$1.0 \le d_{ri} \le 4.0$$

$$0.1 \le f_s \le 0.2791$$

$$0.1 \le f_{ri} \le 0.6$$

$$V_s f_s^{0.35} d_s^{0.15} \le 84.6285$$

$$V_{ri} f_{ri}^{0.35} d_{ri}^{0.15} \le 84.6285$$

$$f_s^{0.74} d_s^{0.9} \le 1.4968$$

$$f_{ri}^{0.74} d_{ri}^{0.9} \le 1.4968$$

$$V_s f_s^{0.74} d_s^{0.9} \le 89.8349$$

$$V_{ri} f_{ri}^{0.74} d_{ri}^{0.9} \le 89.8349$$

$$d_t = d_s + \sum_{i=1}^{n} d_{ri}$$

Table 2 shows the optimal solutions by the GA with MATLAB programming for d_t = 2.0, 2.5, 4.0, 6.0 and 8.0mm. The optimal values of unit profit rate for each d_t represent the average values after 20 repeat calculations.

Table 2. Optimal solutions by the GA

d_t (mm)	2.0	2.5	4.0	6.0	8.0
PR ($/min)	11.1605	11.2034	11.1962	8.6084	8.5970

The results in Table 2 show that two rough passes and one finish pass are required when the total depth of cut is d_t = 8.0mm, with the profit rate of 8.5970$/min. According to [13], three-objective optimization was performed for face-milling operations when d_t = 8.0mm, and the depths of cut were 3.6mm and 2.4mm for the rough passes and 2.0mm for the finish pass respectively, with a total profit rate of 7.25$/min. The proposed optimization method in this paper increases profit by 18.58%. The proposed method also presents better results than other methods in the literature [12,14,15] in which profit rate was optimized. The profit rates calculated based on the reported results in the three references are respectively 6.22$/min, 6.78$/min, and 6.66$/min.

The effect of crossover and mutation rates on the profit rate was also studied and the results are given in Table 3 for d_t = 6.0 mm. From the table, we can see that the maximum profit rate is 8.6390$/min with the crossover rate of 0.80 and the mutation rate of 0.04.

Table 3. Unit profit rate for various GA factors (d_t =6mm)

Mutation Rate	Crossover Rate		
	0.75	0.80	0.85
0.03	8.6018	8.6287	8.5498
0.04	8.6355	8.6390	8.6261
0.05	8.5464	8.6084	8.5814

5 Conclusions

The optimization of machining parameters for face-milling operations was studied in this paper. Unit profit rate was optimized by a traditional genetic algorithm. The solutions to our model are superior to those from other optimization methods in the literature. The method presented in this paper can also be used in other machining operations such as grinding and drilling and some non-traditional machining processes. In addition, other objectives such as surface quality and tool life can also be optimized using the proposed method. These may form our future work in the area of machining parameter optimization.

6 References

[1] Franci Cus, Uros Zuperl. "Approach to Optimization of Cutting Conditions by using Artificial Neural Networks"; Journal of Materials Processing Technology, Vol. 173, Issue 3, 281-290, Apr 2006.

[2] R. Venkata Rao, P. J. Pawar. "Parameter Optimization of a Multi-pass Milling Process using Non-traditional Optimization Algorithms"; Applied Soft Computing, Vol. 10, Issue 2, 445-456, Mar 2010.

[3] Wassila Bouzid. "Cutting Parameter Optimization to Minimize Production Time in High Speed Turning"; Journal Materials Processing Technology, Vol. 161, Issue 3, 388-395, Apr 2005.

[4] V. Tandon, H. El-Mounayri, H. Kishawy. "NC End Milling Optimization using Evolutionary Computation"; International Journal of Machine Tool and Manufacture, Vol. 42, Issue 5, 595-605, Apr 2002.

[5] Y. C. Shin, Y. S. Joo. "Optimization of Machining Conditions with Practical Constrains"; International Journal of Production Research, Vol. 30, Issue 12, 2907-2919, Jan 1992.

[6] A. Jeang, Huan-Chung Li, Yi-Chi Wang. "A Computational Simulation Approach for Optimising Process Parameters in Cutting Operations"; International Journal of Computer Integrated Manufacturing, Vol. 23, Issue 4, 325-340, Apr 2010.

[7] K. Vijayakumar, G. Prabhaharan, P. Asokan, R. Saravanan. "Optimization of Multi-pass Turning Operations using Ant Colony System"; International Journal of Machine Tool and Manufacture, Vol. 43, Issue 15, 1633-1639, Dec 2003.

[8] K. Okushima, K. Hitomi. "A Study of Economical Machining: An Analysis of the Maximum-profit Cutting Speed"; International Journal of Production Research, Vol. 3, Issue 1, 73-78, 1964.

[9] E. J. A. Armarego, J. K. Russell. "Maximum Profit Rate as a Criterion for the Selection of Machining Conditions"; International Journal of Machine Tool Design and Research, Vol. 6, Issue 1, 15-23, Mar 1966.

[10] N. R. Abburi, U. S. Dixit. "Multi-objective Optimization of Multipass Turning Processes"; International Journal of Advanced Manufacturing Technology, Vol. 32, Issue 9-10, 902-910, Apr 2007.

[11] N. Nefedov, K. Osopov. "Typical Examples and Problems in Metal Cutting and Tool Design". Mir Publishers, Moscow, 1987.

[12] M. S. Shunmugam, S. V. Bhaskara Reddy, T. T. Narendran. "Selection of Optimal Conditions in Multi-pass Face-milling using a Genetic Algorithm"; International Journal of Machine Tool and Manufacture, Vol. 40, Issue 3, 401-414, Feb 2000.

[13] Wen-an Yang, Yu Guo, Wenhe Liao. "Multi-objective Optimization of Multi-pass Face Milling using Particle Swarm Intelligence"; The International Journal of Advanced Manufacturing Techhology, Vol. 56, Issue 5-8, 429-443, Sep 2011.

[14] Sourabh Saha. "Genetic Algorithm based Optimization and Post Optimality Analysis of Multi-pass Face Milling"; The Computing Research Repository, arXiv:0902.0763[cs.CE], Vol. 902, 0763, 2009.

[15] C. A. C. António, C. F. Castro, J. P. Davim. "Optimisation of Multi-pass Cutting Parameters in Face-milling based on Genetic Search"; The International Journal of Advanced Manufacturing Techhology, Vol. 44, Issue 11-12, 1106-1115, Oct 2009.