Soft Computing Approach to Production Line Design

Adil Baykasoğlu

University of Gaziantep, Department of Industrial Engineering, 27310 Gaziantep, Turkey
baykasoglu@gantep.edu.tr

ABSTRACT

In this paper one of the new generation soft computing technique that is known as Gene Expression Programming (GEP) is used to develop a meta-model for the multi-objective design of a hypothetical production line. The developed meta-model is used to optimize production line design with Multiple Objective Tabu Search algorithm (MOTS). It is found out that GEP and MOTS can be effectively used to solve production line design problems which are known as complex design problems.

Key words: Manufacturing system design, soft computing, genetic programming

1. INTRODUCTION

In today's very competitive manufacturing environment it is crucial to improve manufacturing performance in order to be able to compete. Responsiveness is already become one of the most important performance indicator. Manufacturing systems must be designed optimally by taking into account responsiveness related measures like tardiness, flow time etc in order to improve responsiveness. A manufacturing system should be robust and able to perform several objectives. This requires solving the multiple objectives manufacturing system design problem under changing production requirements. However it is very hard to define performance measures analytically for such a problem. In order to obtain analytical forms of the objective functions and in some cases constraint functions a meta-modeling approach that is based on simulation is necessary. In a meta-modeling study, simulation responses are converted into equations by using some advance statistical techniques. Classic regression models are generally used for this purpose. But the performance of these techniques is very sensitive to the type of structural equation selected and they usually give poor performance [1]. This can make the meta-modeling approach unsuccessful. Another difficulty is in the optimization of the resultant multiple objective mathematical programming models with the conventional solution algorithms, which are generally not very effective with highly non-linear functions [1]. To overcome the mentioned difficulties a meta-modeling approached that is based on soft computing algorithms is proposed in this paper. In the meta-model generation a new generation of soft computing algorithms that is known as Gen Expression Programming (GEP) is used [2]. GEP is very effective in converting simulation responses into system equations with very high correlation values. After obtaining the manufacturing system's mathematical model, the MOTS algorithm [1] is used to generate optimal manufacturing system designs. In the next section first a very brief review of GEP is given than the problem definitions along with some experimental studies are presented.

2. A BRIEF OVERVIEW OF GENE EXPRESSION PROGRAMMING

In this section, a brief overview of the GEP is given for motivation. For detailed explanations on GEP refer to Ferreira [2]. GEP is a natural development of genetic algorithms and genetic programming. GEP was invented by Ferreira [2]. Most of the genetic operators used in GAs can also be implemented in GEP with minor changes. Like genetic programming, there are mainly five components in GEP; the function set, terminal set, fitness function, control parameters and stop condition that must be determined when using GEP to solve a problem. Unlike the parse-tree representation in canonical genetic programming, GEP uses a fixed-length of character strings to represent solutions to the problems, which are afterwards expressed as parse trees (called "expression tree" in GEP) of different sizes and shapes when evaluating their fitness. Each GEP gene is composed of a list of symbols with a fixed length that can be any element from a function set like {**+, -, *, /, Sqrt**} and the terminal set like {**I, a, b, c, d**}. A typical GEP gene with the given function and terminal set can be {**+.*.Sqrt.a.*.+.+.-.c.a.d.Sqrt.c.d.2**} where "**.**" is used to separate elements for easy reading; **Sqrt** is the square-root function; **2** is a constant; and **a, b, c, d** are variable (or attribute) names. The above is typically named as Karva notation, or K-expression. A K-expression can be mapped into an expression tree (ET) following a width-first fashion. The conversation starts from the first position in the K-expression, which corresponds to the root of the ET, and reads through the string one by one. For each node (from left to right) in one layer in the ET, if it is a function with *m* (*m*>=1) arguments, then the next *m* symbols in the K-expression are attached below it as *m* child branches. Otherwise each terminal node forms a leaf of the ET. This tree expanding process continues layer by layer until all leaf nodes in the ET are composed of elements from the terminal set. For example, the above sample gene can be expressed in a mathematical form as: $a*((c-d)*(d+\sqrt{2}))+\sqrt{c+a}$.

***GEP Algorithm and Operators*:** Like GA and GP, the GEP algorithm begins with the random generation of the fixed-length chromosome of each individual for the initial population. Then, the chromosomes are expressed and the fitness of each individual is evaluated. The individuals are then selected according to fitness to reproduce with modification. The individuals of this new generation are, in their turn, subjected to the same developmental process; expression of the genomes, confrontation of the selection environment, and reproduction with modification. The process is repeated for a certain number of generations or until a solution has been found. In GEP, the individuals are often selected and copied into the next generation according to the fitness by roulette-wheel sampling with elitism, which guarantees the survival and cloning of the best individual to the next generation. Variation in the population is introduced by conducting single or several genetic operators on selected chromosomes, which include:

- *Crossover*, in which two parent genomes are randomly chosen and paired to exchange some elements between them. There are two kinds of crossover; one-point, and two-point crossover, working in the same fashion as that in GAs.
- *Mutation*, which can happen with any times at any position in a genome, as long as that the mutated individual passes the validity test. Note that like crossover, a mutation in the coding sequence of a gene usually drastically reshapes the ET.
- *Rotation*, in which two subparts of element sequence in a genome are rotated with respect to a randomly chosen point (this is similar to the inversion in GAs).

3. PROBLEM DEFINITION

The hypothetical manufacturing system consists of 5 stages. Machines in each stage are identical and are varied in the optimization and simulation process. One of the machine in each stage malfunction in 2 hours period successively. It takes 40 min. to repair the malfunctioning machine. There are 3 products in the system which arrives exponentially. The maximum number of products to be produced for the planning horizon is 1000 (for each product). TWK rule is used to assign the due dates; FIFO is used for machine selection. Products must pass from each stage and there is a buffer in front of each machine. Buffer size is set to 10 or 15. The processing times are presented in Table 1.

Table 1. Processing time in each stage

	Product 1	**Product 2**	**Product 3**
Stage 1	Uniform(20,25)	Uniform(15,20)	Uniform(15,19)
Stage 2	Uniform(15,25)	Uniform(20,25)	Uniform(10,18)
Stage 3	Uniform(12,18)	Uniform(25,30)	Uniform(15,18)
Stage 4	Uniform(25,30)	Uniform(15,20)	Uniform(15,20)
Stage 5	Uniform(14,19)	Uniform(15,25)	Uniform(15,25)

4. EXPERIMENTAL STUDY

In order to observe the system behavior, the simulation model is run with different number of machines in each stage and different buffer sizes. The number of machines in each stage is varied between 1 and 5 and the buffer size is set to 10 or 15. Simulation with 250 different combinations of variables is carried out. Tardiness and flow time measures are collected for each configuration in order to derive the corresponding equations with GEP. 210 runs are used for training and 40 runs are used for testing GEP. After training GEP, high correlations are obtained (R^2 for tardiness=0.978, R^2 for flow time=0.977). The resultant equations for tardiness and flow time are shown as the first and second objective functions in Table 2. The test results for tardiness and flow time functions are shown in Figure 1. As it can be seen from Figure 1, there is a very close match between the actual (simulation results) and the predicted (GEP results) curves. The GEP function is able to closely follow the trend in the actual data.

A stepwise regression analysis is also carried out in order to have an idea about the predictive power of the soft computing techniques in comparison to a classical statistical approach. R^2 of the prediction by the regression equation on the test data is 0.775 for tardiness and 0.775 for flow time. As it can be seen from the results, the stepwise regression analysis performed very much worse than the GEP algorithm. This was an expected result. Mainly because the production line design problem depends on many different factors, and the relations between these factors are highly non-linear and complex. The results obtained from this study have also shown that GEP is a good candidate for making predictions on the behavior of complex systems including the performance prediction of manufacturing systems.

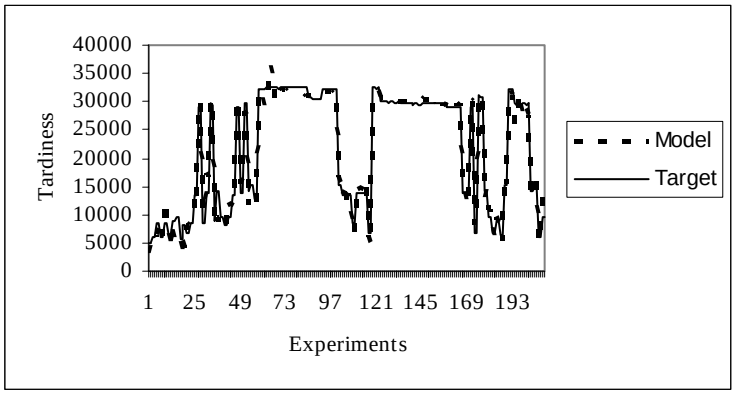
Table 2. The MOO model for the production line design problem

MIN Tardiness =(x4*((x6+(x1*x1))/tan(pow(10,x1))))+(((x1+(x6*(x6+x1)))
/tan(log(x1)))*x4)+(x3*((x1+((x4+x4)+x2))*((x3-x6)-
exp(x2))))+(tan((cos((x5/x1))+(x4+x3)))*(x1*x2))+exp((pow(10,cos((pow(10,
cos(x2))*(x3/x2))))-log(x3)))+exp((pow(10,cos(((x5/x1)/x5)))-
tan((cos(x2)*cos(x1)))))+((tan(tan(x5))-tan(tan(x2)))*(x2*x1))

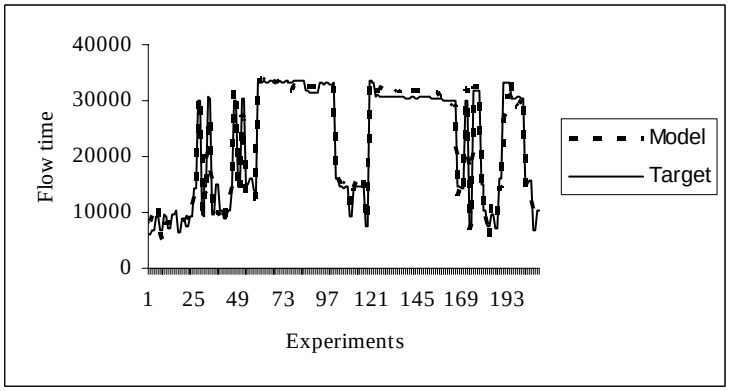
MIN Flow-time =(exp(pow(10,pow(sqrt(sqrt((x2*cos(pow(10,(((cos(x5)-x5)-
(x5*x3))/pow((x4+x3),x2)))))))),((x5/x3)-x5))))/sqrt(x4))+
(exp(pow(10,cos(exp((x4-
exp(sqrt((x4*sqrt((log(x2)+cos(cos(sin(log(log(pow(exp(x4),sqrt(x4)))))))))
))))))))/sqrt(x2))+
(x2+(x4*(((((x4*tan(pow(10,exp(cos(log10(x3))))))*x2)*x1)-
exp(x5))+x2)))+(x6*((((pow(x2,tan(pow(10,exp(cos(log10(x3))))))-x5)*x1)-
exp(x3))-x5))

MIN Total # WS= x2+x3+x4+x5+x6

Subject to: 1≤x2≤5
1≤x3≥5
1≤x4≤5
1≤x5≤5
1≤x6≤5
x1 *discrete* (10,15) & x2,x3,x4,x5,x6 *Integer*



(a)



(b)

Figure 1. The test result for the best GEP function a-Tardiness, b- Flow time

The next step in the meta-modeling study is to search for optimal production line configurations. The problem is an integer, nonlinear Multiple Objective Optimization

(MOO) problem as depicted in Table 2. There are three objectives the first one is the minimization of tardiness, the second one is the minimization of flow time and the third one is the minimization of the total number of workstations. The last objective is especially important for an economical design. There are six variables in the model the first one is a discrete variable which determines the size of the buffer, the last five are integer variables which are used to determine the number of machines in each stage. The maximum number of machines in each stage is restricted to five. The MOO problem is solved with the MOTS algorithm that was previously developed by the author. The details of this algorithm can be found in [2]. The algorithm converged into 81 Pareto optimal solutions in 3 seconds which are shown in Figure 2. The designer can select one of this efficient solutions based on his/her utility function.

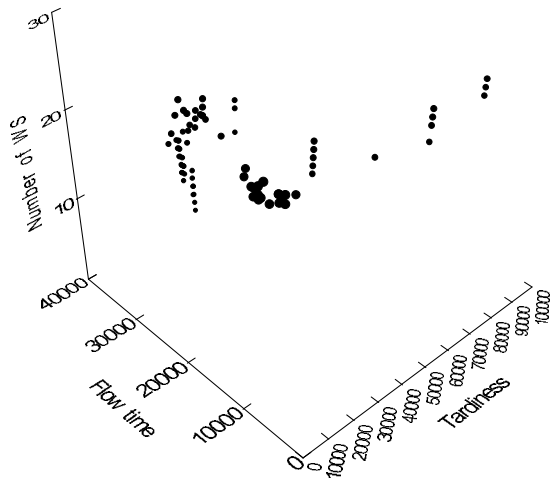


Figure 2. Pareto optimal solutions for production line design problem

5. CONCLUSION

In this paper a meta-modeling approach based on soft computing algorithms namely GEP and MOTS is proposed for the optimal design of a hypothetical production line. Responsiveness related criteria like tardiness and flow time are used to evaluate goodness of the production line through the MOTS algorithm. It has been observed that soft computing algorithms are quite effective for modeling and solving complex production line design problems. These algorithms must be included into the toolbox of the production system designers in order to obtain effective solutions.

REFERENCES

[1] **Baykasoğlu, A., Owen, S.** and **Gindy, N.** (1999) A taboo search based approach to find the Pareto optimal set in multiple objective optimisation. *Journal of Engineering Optimization*, Vol. 31, pp. 731-748.

[2] **Ferreira, C.** (2001) Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems,* Vol. 13, pp. 87-129.