A Fitness Differential Adaptive Parameter Controlled Evolutionary Algorithm with Application to the Design Structure Matrix

Kai Cheng, John Lancaster

A Fitness Differential Adaptive Parameter Controlled Evolutionary Algorithm with Application to the Design Structure Matrix

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0930.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	02-Mar-2007
Complete List of Authors:	Cheng, Kai; Brunel University, Advanced Manufacturing and Enterprise Engineering Lancaster, John; Brunel University, Advanced Manufacturing and Enterprise Engineering
Keywords:	EVOLUTIONARY ALGORITHMS, SCHEDULING, OPTIMIZATION
Keywords (user):	

A Fitness Differential Adaptive Parameter Controlled Evolutionary Algorithm with Application to the Design Structure Matrix

JOHN LANCASTER and KAI CHENG*

Word Count: 4015

This paper investigates a methodology for adaptation of the mutation factor within an Evolutionary Algorithm by means of measuring the improvement differential between successive generations. When no improvement is obtained in an Evolutionary Algorithm and it has not located the global optimum, it is an indication that the algorithm may have become trapped within a local minimum or maximum. Mutation is a tool within the algorithm that is designed to assist in escaping from these local extremes. It is therefore the premise of this paper that if the preset value for mutation probability is proving insufficient to release the algorithm from entrapment in a local minima or maxima, then a temporary increase in this mutation probability may assist in freeing the algorithm and therefore increasing its chances of ultimately converging on a global optimum.

In order to determine when to implement the increase in mutation probability our algorithm measures the fitness improvement between successive generations in the algorithm. When no improvement is detected for a number of successive generations the probability is increased.

The Design Structure Matrix (DSM), a scheduling tool, that has previously been optimized via the application of Evolutionary Algorithms has been used as a practical implementation of differential adaptation to investigate it's effectiveness in solving real

world problems. Solutions provided by Todd (1997) are used to benchmark the algorithms effectiveness.

Keywords: Differential Adaptation, Evolutionary Algorithms, Design Structure Matrix.

***To whom correspondence should be addressed**

Professor Kai Cheng
Chair in Manufacturing Systems
Head, Advanced Manufacturing & Enterprise Engineering (AMEE)
School of Engineering and Design
Brunel University
Middlesex UB8 3PH
UK
Email: kai.cheng@brunel.ac.uk

1. Introduction.

Within an Evolutionary Algorithm the mutation operator is utilized to generate diversity within the algorithms search. This diversity is required to prevent premature convergence on local optima (minima or maxima). As can be seen from Figure 1 below Evolutionary Algorithms can easily converge prematurely on these local optima by obtaining local improvement.

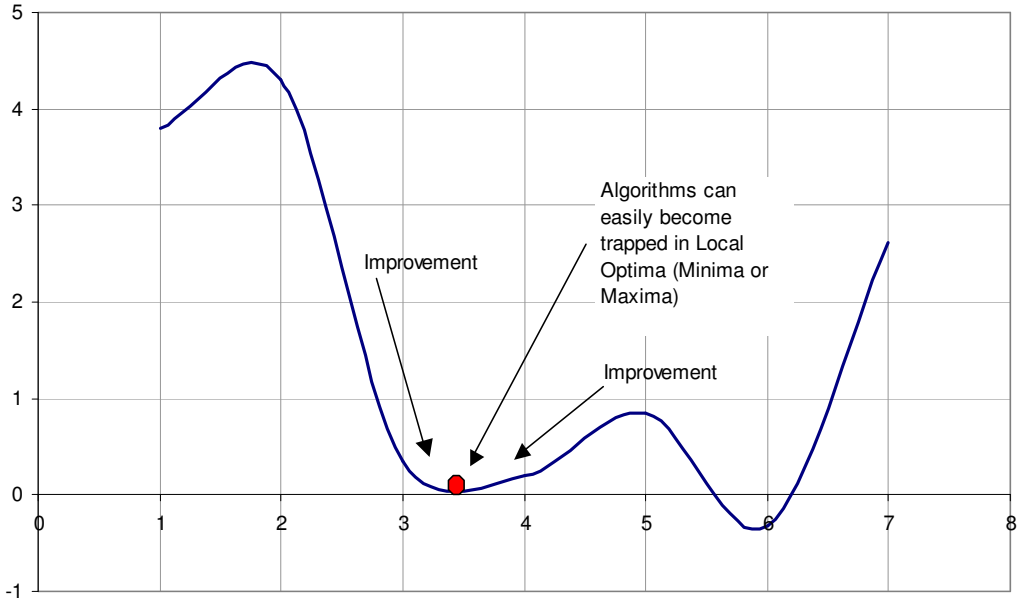


Figure 1 – Trapping at Local Optima (minimising algorithm).

Mutation is effected by randomly selecting genes within the chromosomes and changing their values, the probability of selecting a chromosome to undergo mutation is normally an input variable provided at run time. This mutation probability needs to provide sufficient variety to the algorithm to allow the search space to be thoroughly investigated for global optima whilst being limited sufficiently to allow the algorithm to converge on such optima once these have been detected. These two conflicting requirements need to

be carefully balanced in order to ensure optimization takes place. A number of studies have been undertaken to determine the optimum probability settings for these variables, for various applications, techniques such as Taguchi's design of experiments (DOE) have been employed for this purpose (Younes and Rahli, 2006).

Whilst studies into optimizing these variables prior to execution will ultimately improve the performance of the algorithm, they do not allow for the dynamic state of the algorithm during processing. Ideally the mutation probability needs to adapt its value according to it's position in the search space i.e. when trapped in a local optima it should increase in order to widen the algorithms search but when not trapped the mutation probability should be low enough to allow the algorithm to converge towards a possible global optimum. In order to achieve this, the technique described in this paper as 'fitness differential adaptive parameter control' is employed.

Previous research which has applied evolutionary algorithms to optimization of the Design Structure Matrix (DSM) is reviewed in section 2. Evolutionary algorithm parameter settings in general are discussed in section 3. The structure and functionality of the fitness differential adaptive parameter control evolutionary algorithm is then discussed in detail in section 4.

The Design Structure Matrix has been selected as the application for the algorithm due to other research being conducted by the authors into the use of evolutionary algorithms for schedule optimization. The Design Structure Matrix is described in section 5.

The test problems, the tests and the achieved results are discussed in section 6 and conclusions and suggestions for further work discussed in sections 7 & 8 respectively.

2. Review of previous research applying Evolutionary Algorithms to the DSM.

Rogers (1994, 1996) implemented a genetic algorithm into NASA's DSM tool 'DeMAID' (Design managers aid to intelligent decomposition) to optimise the sequence of activities in the DSM in order to minimise the impact of iteration, which requires the DSM to be moved as close as possible to becoming lower-triangular (see section 5 for further discussion). Satisfying this objective has the effect of minimising the overall duration of the project (make span) and therefore reduces the overall time dependant cost. DeMAID applies duration and cost to the individual tasks but minimization of these two characteristics is not used as an optimization objective, they are merely applied to the iteration minimal matrix.

Todd (1997) considered the maximisation of concurrency as well as the minimisation of iteration. As Todd details, the maximization of concurrency within the DSM is effected by moving as many of the links as close to either the left hand side of the matrix, or alternatively to the bottom edge of the matrix. On first consideration this may seem compatible with the lower triangularisation required by minimization of iteration and indeed the two could be mutually achieved, however the network logic will often deny satisfying both objectives and having a high percentage of links aligned with the left edge of the matrix may cause the small number of remaining links high into the upper triangle obstructing the objective of minimum iteration. Todd's algorithm utilized Enhanced Edge

Recombination (EERX) crossover (Starkweather, 1991) in conjunction with 2-city[1] adjacent swap mutation. Todd's experimentation had shown that the EERX crossover had proved most successful in combinatorial problems such as the Traveling Salesman Problem and he therefore chose to apply this to the DSM.

Whitfield et al (2003) performed extensive research into the application of various cross-over and mutation operators to the DSM. They found that Todd had been incorrect to assume that an operator, which performs well for one combinatorial problem, is best suited for all combinatorial problems. Whitfield et al reported that the combination of EERX and 2-point adjacent swap mutation to be among the worst combination of operators and in fact revealed the Independent position crossover (IPX) in conjunction with the Shift Operator mutation (SOM) (Murata and Ishibuchi, 1994) to be the best combination suited to this application.

Zhuang and Yassine (2004) utilised a Genetic Algorithm to optimise the RCPSP problem using the Dependency Structure Matrix. Zhaung and Yassine applied two crossover techniques:

- Leu and Yang's (1999) Union Crossover 3 (UX3) operator - this operator performs crossover whilst maintaining conformance to precedence relationships.
- Goldberg's (1989) One-point Crossover Operator.

[1] The 'City' terminology is derived from the operator's previous application to the Traveling Salesman Problem.

Their experimentation yielded very poor results for the UX3 operator compared to the one-point operator, they concluded that this is due to the one-point crossover being able to maintain larger portions of good schema across generations.

Another important feature of Zhuang and Yassine's research was the stochastic calculation of feedback within the DSM. Probability values were randomly applied to the feedback values within the algorithm in order to calculate the likely duration of the project. 31 random trials were conducted in order to evaluate the range of possible durations.

The body of existing research in this area is relatively small compared to the work that has been conducted in applying evolutionary algorithms to traditional scheduling networks.

3. Parameter Settings.

A lot of work has been invested in the study of optimal settings for the operating parameters for Evolutionary algorithms. Parameter setting can be executed in two main modes prior to run and during run.

Ursem (2003) and Eiben *et al* (1999) both provide taxonomies for Parameter setting, which follow the same basic structure with some minor terminology differences, we provide the taxonomy as per Eiben *et al* (1999) in Figure 2 below:

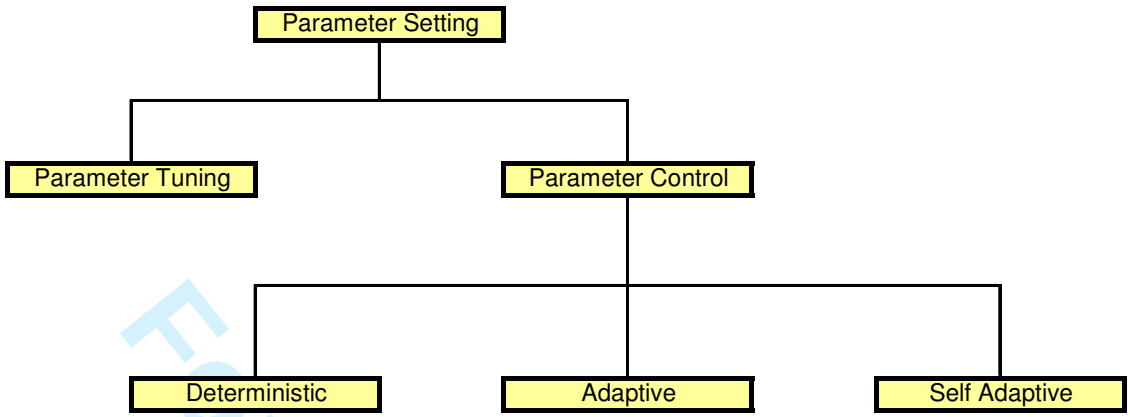


Figure 2 – Taxonomy of Parameter Setting.

Parameter tuning is concerned with refining the setting parameters prior to run time. The parameters remain constant throughout the execution of the algorithm. Many methods have been applied to tuning these parameters including Taguchi methods (Younes and Rahli, 2006). Thierens (2002) demonstrated the use of adaptive mutation control, employing two methods of controlling the mutation factor by testing the effects of increased and decreased mutation rates and then modifying the mutation probability accordingly.

Parameter Control is concerned with the modification of parameters during the run time of the algorithm there are a number of methods by which this can be achieved:

- Deterministic.
- Adaptive.
- Self-Adaptive.

These three classifications specify the method by which the algorithm receives instruction to alter the value of a parameter.

Deterministic control involves the modification of the algorithm according to a pre-selected schedule or function, that is, no feedback is received from the values produced by the algorithm during its run-time. As this method receives no-feedback it is not able to adapt according to the current state of optimization. Our aim is to produce an algorithm that detects and escapes from trapping in local optima, so this method will not be suitable.

Adaptive Control is achieved by modifying parameters based on the values yielded by the algorithm during its run time. Adaptive control reacts to feedback from the algorithm and is the method of control we have selected for the algorithm presented in this paper.

Self-Adaptive Control is obtained by extending the chromosome by additional genes. These genes are evolved during the execution of the algorithm along with the rest of the chromosome. Through this method the best settings for parameters can be evolved during run time. The nature of this method of control is that of progressive refinement, we aim to produce an algorithm that reacts quickly to the trapping and temporarily modifies its behavior to suit, so again this method is not suitable to our research. Sewell *et al.* (2006) utilized self-adaptation in their 'rank-scaled mutation rate' genetic algorithm. This algorithm, applied to the traveling salesman problem, adapted the mutation probability of each chromosome dependant on the individual's fitness. Sewell *et al.* concluded that their algorithm performed competitively in problems where many local optima were present.

Two further classifications of adaptive parameter control should be discussed here, the first is concerned with the source of control, that is which algorithm generated data is used to drive the parameter changes. This could be any number of measures; in the algorithm presented in this paper we are concerned with preventing premature trapping of the algorithm within local optima. A characteristic of such a trapped algorithm is that its fitness will not improve whilst it is trapped in the local optimium, therefore for this algorithm we have chosen to use a measure of fitness improvement over a number of successive generations to be the driving measure.

Last but not least we need to define what aspect, or parameter of the algorithm is being adapted. In order to prevent trapping diversification of search is required, this is most effectively achieved via increased mutation rate and therefore mutation rate has been identified as the object of adaptation in this algorithm.

Due to this classification system we have termed the algorithm utilised here a Fitness Differential Adaptive Parameter Control Evolutionary Algorithm (FDAPCEA). The structure of this algorithm is discussed in the following section.

4. Fitness Differential Adaptive Parameter Control Evolutionary Algorithm (FDAPCEA) for the DSM.

Fitness Differential adaptation involves monitoring the improvement of the best solution from one generation to the next. In this algorithm the mutation factor is modified when the algorithm yields no improvement for a number of consecutive generations. The model of the FDAPCEA is otherwise quite typical. The flow diagram is given in Figure 3:

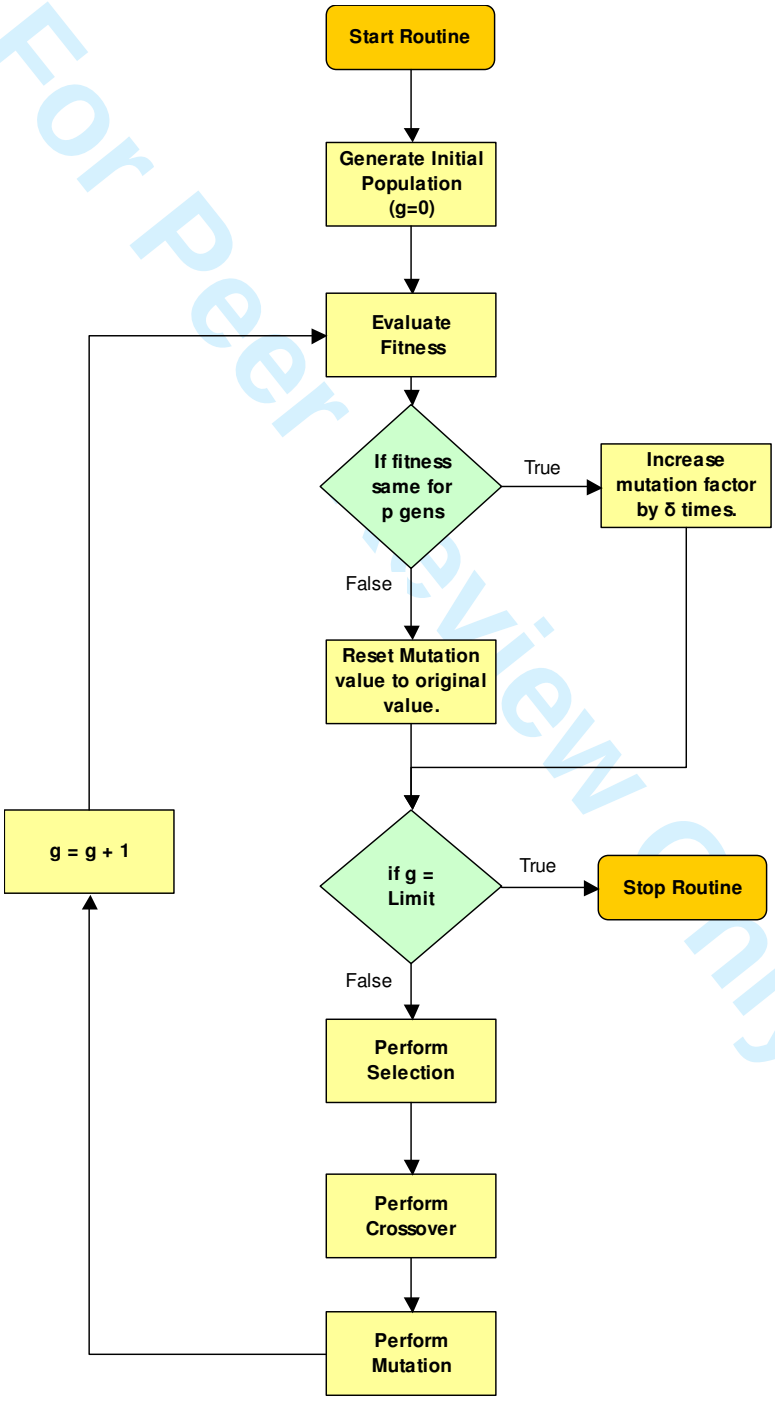


Figure 3 – The FDAPCEA flow diagram.

The best fitness for each generation is stored in a vector, after the pth generation, the previous p generations fitness values are inspected and compared, if no improvement is detected across the p generations, the mutation probability is increased by a factor δ. This causes a large amount of mutation, increasing the spread of the search. If a better solution is found, the best fitness will have increased and the mutation factor will return to normal. If no improvement is found the mutation factor will remain at the increased level, widening the search again for the following generation.

This process aids the algorithm to escape from local minima and is employed only when the algorithm detects the possibility that it has become, or is likely to become trapped.

Two variables have been identified in the above discussion; p the number of generations for which the algorithm will allow no improvement before applying increased mutation and δ the factor by which the mutation probability is increased after the period p with no improvement. For purposes of this discussion, p is termed the differential period and δ the differential factor.

Due to the combinatorial nature of the problem the algorithm uses 'real' encoded chromosomes, the operators are therefore also of the real encoded type.

The individual components of the algorithm are further detailed below:

4.1 Fitness Measurement.

As the Fitness measurement is application specific, discussion of this component is postponed to section 5.1 after discussion of the DSM in general.

4.2 Selection.

The algorithm uses Roulette selection as described by Goldberg (1989).

4.3 Crossover.

Two types of crossover operator have been used in this work; two-point centre crossover and independent position crossover.

4.3.1 Two-point centre crossover.

In the two-point centre crossover operator (Murata, 1997), two Random points are selected on the first parent chromosome. Genes falling inside these two points are transferred directly to the child chromosome. The remaining genes from the first parent are transferred to the child chromosome in the order they occur in the second parent. This is shown diagrammatically in Figure 4.

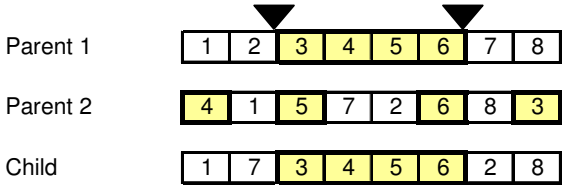


Figure 4 – Two-point centre crossover.

This process is then repeated working from the second parent to produce a second child. For the solutions to the problems discussed in section 6, the crossover factor was set to 0.7.

4.3.2 Independent Position Crossover.

The second method of crossover employed is Independent position crossover. This method of crossover applies a probability of 0.5 to each gene of being transferred directly from the first parent to the child. The values that have then not been transferred to the child due to this process are then added in the order they occur in the second parent. Figure 5 below shows this process graphically.

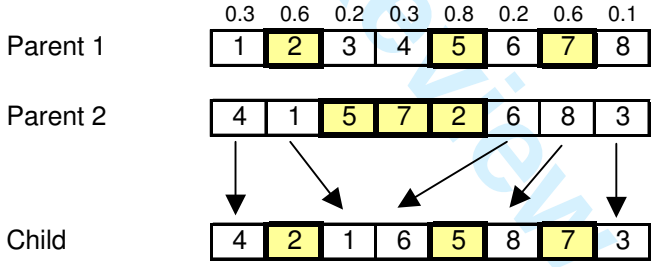


Figure 5 – Independent Position Crossover.

The first row of figures above the first parent in figure 4.3 are random variables generated for each gene in order to determine whether they are eligible for transfer to the child chromosome. As can be seen all the values greater than 0.5 have been transferred directly to the child (2, 5 & 7) the balance of the genes (1, 3, 4, 6 & 8) have been transferred to the child in the order they occur in the second parent.

4.4 Mutation.

For the solutions to the problems discussed in section 6, the mutation factor was set to 0.07.

5. The Design Structure Matrix.

The Design Structure Matrix (DSM) is a scheduling tool, which caters for iteration between tasks. As its name suggests the DSM is formed as a square matrix (number of columns equals number of rows) with the task being listed along both the horizontal and vertical axis, the task itself being represented by the respective block on the diagonal. This diagonal listing of tasks divides the matrix into two triangular portions, the lower triangle being used for the mapping of forward feeding task links and the upper triangle for backward feeding (iterative) task links. This is shown diagrammatically in the Figure 6below:

DESIGN STRUCTURE MATRIX

Task Description	Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Task 1	1	1														
Task 2	2	1	2													
Task 3	3		1	4					FEEDBACK							
Task 4	4		1		6											
Task 5	5		1			7										
Task 6	6			1			13	1		1					1	
Task 7	7						1	24								
Task 8	8							1	45	1						
Task 9	9				1					63						
Task 10	10									1	27	1		1	1	
Task 11	11		FEED FORWARD								1	14				
Task 12	12											1	7	1		
Task 13	13				1	1			1				1	6		
Task 14	14													1	5	
Task 15	15														1	1

Figure 6 – The Design Structure Matrix.

The larger the iterative loops present in the schedule the greater its duration is likely to be and the more assumptions that need to be made during design. It is therefore desirable to optimize the sequence of execution in order to minimize iteration. Within the DSM minimizing iteration equates to moving the DSM as close as possible to being lower triangular, that is, all the links sitting in the feed forward (lower) portion of the matrix.

This can be clearly seen as a combinatorial problem, a class of problems that have successfully had Evolutionary techniques applied to them (for example the Traveling Salesman Problem). Indeed a number of researchers have applied Evolutionary algorithms successfully to the DSM as discussed in section 2 above.

5.1 Fitness Measurement in the DSM.

As already discussed the objective of this algorithm is to minimize iteration, this is characterized by moving the matrix as close as possible to being lower triangular, that is the feed back links either need to be within the lower triangle or failing this as close as possible to the diagonal. The measure of fitness can therefore be determined by summing the distance from the diagonal of all the feedback links i.e. links in the upper triangle. The measure of Total fitness is therefore given by:

$$\sum_{i=1}^{n} w_i \cdot \left(x_i - y_i \right)$$

Letting n be the number of activities in the upper triangle, w be the feedback value (in this case always 1[2]) and x and y being the position in the sequence of the predecessor and successor respectively i.e. the distance from the diagonal.

6. Results obtained with FDAPCEA.

6.1 Standard Problems used for comparison.

Project scheduling problem (PSP) libraries such as PSPLIB (Kolisch and Sprecher, 1996), normally utilized for benchmarking of PSP, do not provide problems with iterative links; therefore to provide a benchmark for this algorithm the problems considered by Todd (1997) are utilized. Todd uses three problems:

- KUSIAK '91 – A twelve-activity schedule.
- STEWARD '81 – A twenty-activity schedule.
- AUSTIN '96 – A fifty-one-activity schedule – In this case the original DSM was not provided, Todd therefore sought further improvement of the solution offered by Austin. For comparative purposes the same approach has been taken here.

The original authors offered solutions to each of their respective problems. These solutions used methods other than evolutionary techniques. The best values obtained for these problems before Todd's (1997) work are given in Table 1 below:

[2] This paper is limited to the study of Binary type DSMs. DSMs are also utilized with numerical feedback values these being referred to as Numerical DSMs (NDSM).

Problem	Best solution
KUSIAK '91	7
STEWARD '81	93
AUSTIN '96	320

Table 1 – Best non-evolutionary solutions.

6.2 Results obtained with the FDAPCEA.

	2	3	11	1	7	6	10	12	9	8	5	4
2	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
11	1	1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	1	0	0	0	0
10	1	1	1	0	0	1	0	1	0	0	0	0
12	0	0	1	1	0	0	1	0	1	0	0	0
9	0	1	0	0	0	1	1	0	0	0	0	0
8	0	0	1	1	0	0	0	0	1	0	0	1
5	0	0	1	0	0	1	0	0	0	1	0	0
4	0	1	0	0	0	1	0	1	0	0	1	0

Total Fitness (Iteration) = 6

Figure 7 – Resultant DSM (KUSIAK '91).

The algorithm yielded a number of different solutions with a total fitness of 6. Figure 7 shows the solution to the KUSIAK '91 problem and Figure 8 indicates that the solution is typically arrived out without any significant periods being 'trapped' in local minima. The best solution to this DSM yielded by Todd was also 6.

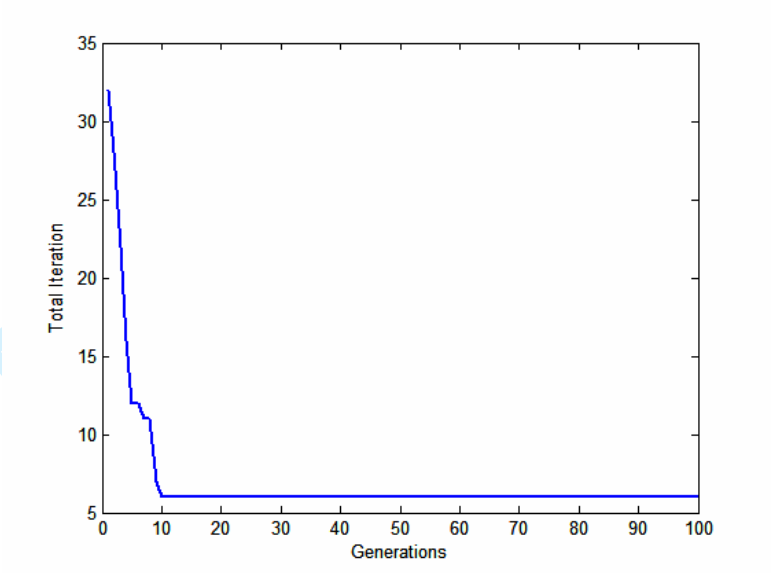


Figure 8 – Algorithm improvement over successive generations (KUSIAK '91).

	2	19	5	16	6	7	8	18	9	11	10	17	3	4	1	14	20	15	12	13
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
6	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	1	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0
17	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	1	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0
13	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0

Total Fitness (Iteration) = 24

Figure 9 – Resultant DSM (STEWARD '81).

Figure 9 shows the solution to the STEWARD ’81 problem while figure 10 shows a typical improvement curve achieved for this problem. Figure 10 shows that after generation 20 there exist a number of plateaus in the improvement graph where the algorithm is potentially ‘trapped’ for a number of generations before finding further improvement, for the run shown in Figure 10 below the differential adaptation factor was set to 4 generations, the plateaus in the improvement graph below appear to be typically around 4 generations in length or greater indicating that the sudden increase in mutation rate could be responsible for a number of these stepped improvements.

The best solution produced by the algorithm for this DSM was a total fitness of 24; this result is also equal to the best result reported by Todd.

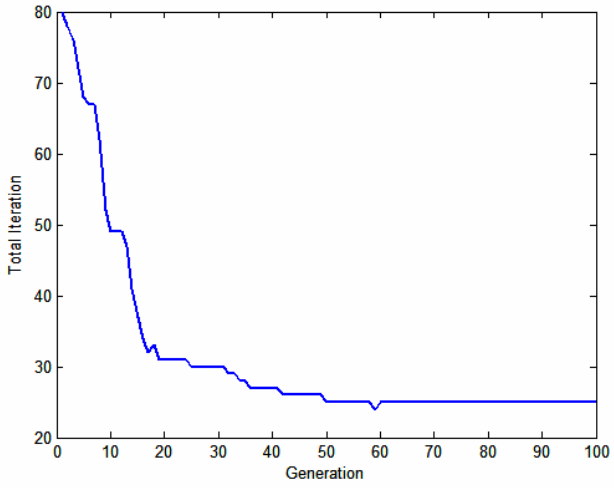
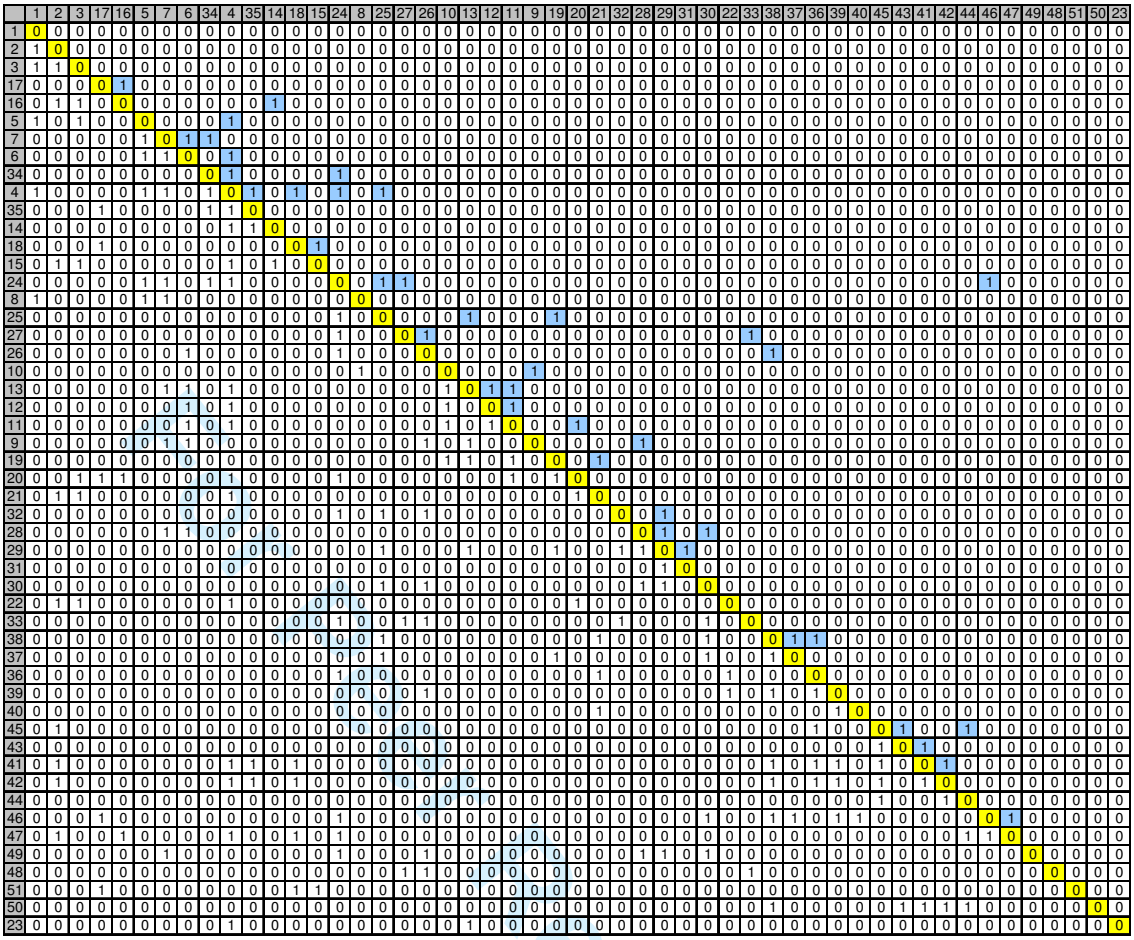


Figure 10 – Algorithm improvement over successive generations (STEWARD ’81).

The resultant DSM for the problem of AUSTIN ’96 is given below in Figure 11.



Total Fitness (Iteration) = 157

Figure 11 – Resultant DSM (AUSTIN '96).

The result produce for the AUSTIN '96 DSM was a total fitness (iteration) of 157 an improvement of 1 over Todd's single objective algorithm result of 158.

The Improvement graph for the algorithm for this solution is shown below in Figure 12.

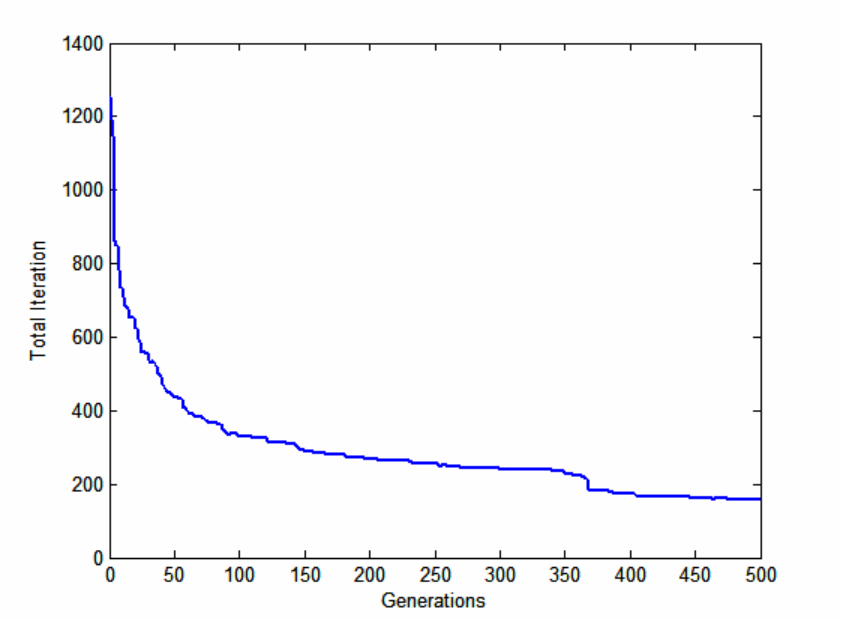


Figure 12 – Algorithm improvement over successive generations (AUSTIN '96).

The improvement curve shows that constant improvement has been achieved right to the last 25 generations. It should be noted that a function of the algorithm is that the differential adaptation is not applied during the last 10% of the generations on each run, in order to allow convergence. In this run the differential adaptation would therefore have cut out at generation 450.

The effectiveness of the FDAPCEA compared with the original results and the results of Todd (1997) as shown below in Table 2.

Problem	Original Solution	Todd (1997)	FDAPCEA
KUSIAK '91	7	6	6
STEWARD '81	93	24	24
AUSTIN '96	320	158*	157

*Todd later found a solution of 156 using a Multi-objective algorithm.

Table 2 – Comparative Results of Best Solutions to DSM problems.

6.3 Discussion of the Results.

The results shown in section 6.2 have shown that the FDAPCEA is able to produce results at least as good as those reported to date, using only simple genetic operators. The sample improvement curves show that the algorithm is consistently able to release itself from flat spots in the improvement curve.

7. Conclusions.

This paper has demonstrated the application of adaptive parameter control based on the differential improvement in fitness between successive generations. It has shown in general that this technique used in conjunction with basic genetic operators can provide optimization of the DSM to at least the currently best-known solutions.

7.1 Applicability of Evolutionary Algorithms to the solution of the DSM.

The work by Todd (1997) as well as the work presented in this paper has clearly demonstrated the suitability of evolutionary algorithms to the solution of the DSM. Evolutionary Algorithms have shown in both these works to produce results better than those produced by other non-Evolutionary methods.

7.2 Effectiveness of the FDAPCEA.

The FDAPCEA has demonstrated its effectiveness here by being able to equal best-known solutions to the benchmark problems, without becoming trapped in local optima. The authors believe that this warrants further investigation and experimentation into the use of adaptive parameter control in genetic algorithms. The algorithm has thus also shown its specific suitability to the DSM type scheduling problems.

8.0 Further Research and Potential Applications

In order to increase the practical applicability of this research it is intended to extend the application of this algorithm to the precedence network PSP. The algorithm has shown its suitability to the PSP problem (see Lancaster and Ozbayrak, 2007) and the authors believe there is practical application of this technique to real world project scheduling problems. Our on-going research will investigate application of the FDAPCEA to real world Resource Constrained Project Scheduling Problem (RCPSP) as well as special cases of the RCPSP.

In the current form the solution to the Design Structure Matrix can be utilised to improve the design process minimising iteration due to the interaction between various disciplines and information sources.

References

EIBEN A, HINTERDING R and MICHALEWICZ Z, Parameter Control in Evolutionary Algorithms, IEEE Transactions of Evolutionary Computation, 1999, 3, 2, 124 – 141.

GOLDBERG, D., 1989, Genetic Algorithms in Search, Optimisation and Machine Learning, Addison Wesley Longman Inc.

HERROELEN W, DEMEULEMEESTER E and DE REYCK B, Project Scheduling: A research Handbook, 1999, ISBN 1-402-07051-9, (Springer).

KOLISCH R and SPRECHER A, PSPLIB: A Project Scheduling Problem Library, Christian Albrechts Universitat zu Kiel, Germany. Available on line at: http://129.187.106.231/psplib/ , 1996.

LANCASTER J, Managing Iteration in Design, Engineering Designer – The Journal of the Institution of Engineering Designers, January/February 2003 26-28.

LANCASTER J and OZBAYRAK M, Evolutionary Algorithms applied to Project Scheduling Problems – A survey of the state-of-the-art, International Journal of Production Research, 2007, 45, 2, 425 – 450.

ROGERS J, Ordering Design Tasks based on Coupling Strength, in 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimisation, 1994.

ROGERS J, Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Process, NASA Langley Research Centre, Hampton Virginia, 1996.

SEWELL M, SAMARABANDU J, RODRIGO R and McIsaac K, The Rank-Scaled Mutation Rate for Genetic Algorithms, International Journal of Information Technology, 2006, 3, 1, 32 – 36.

THIERENS D, Adaptive Mutation Rate Control Schemes in Genetic Algorithms, Institute of Information and Computing Sciences, Utrecht University Report UU-CS-2002-056, 2002.

TODD D, Multiple Criteria Genetic Algorithms in Engineering Design and Operation, PhD Thesis University of Newcastle Department of Marine Technology, 1997.

URSEM R, Models of Evolutionary Algorithms and their applications in Systems identification and control optimisation, PhD Thesis Faculty of Science University of Aarhus, 2003.

WHITFIELD R, DUFFY A, COATES G and HILLS W, Efficient Process Optimisation, *Concurrent Engineering,* 2003, 11, 2, 83 – 92.

YOUNES M and RAHLI M, On the Choice of Genetic Parameters with Taguchi Method applied in Economic Power Dispatch, Leonardo Journal of Sciences, 2006, 9, 9 – 24.

ZHAUNG M and YASSINE A, Task Scheduling of Parallel Development Projects using Genetic Algorithms, in *DETC '04 ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference Salt Lake City, Utah USA*, 2004.