AN EFFICIENT GENETIC ALGORITHM PARADIGM FOR DISCRETE OPTIMISATION OF PIPELINE NETWORKS

Z. Y. Wu, Ph.D candidate and A. R. Simpson, Sr. Lect, Dept. of Civ. & Envir. Engrg, Univ. of Adelaide, South Australia 5005. Email: {zwu, asimpson}@civeng.adelaide.edu.au.

ABSTRACT Engineering and science disciplines make use of genetic algorithms. A number of genetic-based search paradigms have been developed and applied to different problems. A growing demand for algorithms to solve new problems and a never-ending process of designing algorithms strongly suggests the need for more efficient and more robust genetic-based optimisation techniques. Ideally these algorithms should make few assumptions regarding the objective functions and use as little domain knowledge as possible. In this paper, different genetic-based search paradigms including the standard GA, the messy GA and the fast messy GA are compared for the discrete optimisation of pipeline networks.

1. INTRODUCTION

Design of pipeline networks involves selecting pipe sizes from a set of commercially available pipe sizes. It is a discrete optimisation problem. The discrete optimisation problem appears to be easier to solve than the continuous one because fewer possible solutions exist. In general, however, it is more difficult to solve. This is due to the fact that the discrete design space is non-differentiable and nonconvex. The standard gradient-based programming techniques and optimality criteria cannot be applied directly. A global optimal solution of the discrete optimisation problem can be obtained only by an *exhaustive search* (Arora et al. 1994). The genetic algorithm is proven to be robust for the optimisation of pipeline networks but usually requires a large number of function evaluations (Simpson et al. 1994; Dandy et al. 1996).

Genetic-based search techniques have been developed to exploit the information (fitness and/or objective function value) gathered from samples taken from the search space. The space is quantified by different regions. The algorithms use the information to decide which region to explore next. In other words, the search space can be seen as being classified into different classes which represent a certain of relations between samples. Thus, the search space is decomposed into relation space, class space and sample space (Kargupta 1995). For example, for a 4-bit problem, # # # *f* is a relation between the samples, where *f* means fixed bit, # # # 1 and # # # 0 are two classes of the relation, where # is called don't-care symbol which can be either 0 or 1, 0110 and 1110 are two samples of class # # # 0. Searching for an optimum is to search for right relation and class.

A standard genetic algorithm (sGA) (Holland 1975) searches for relations and classes implicitly. The sGA population combines the relation space, class space and sample space all together. This results in sGA being a poor search for relations. The messy GA (Goldberg et al. 1989) is designed to search for the relations and classes by using a variable-length genotype, an explicitly enumerative initialisation, *cut* and *splice* genetic operator. However, the original messy GA usually requires a large number of members in the initial population. It is difficult to apply the original mGA to a highly dimensional problem. A fast messy GA (Goldberg et al. 1993), using a probabilistically complete initialisation and building block filtering process, was introduced to eliminate this bottleneck. In this paper, the messy GA and the fast messy GA are applied to the optimisation of design of pipeline networks. The efficiency of the messy GAs are compared with standard GA paradigms.

2. STANDARD GENETIC ALGORITHMS

The standard genetic algorithm (Holland 1975; Goldberg 1989) is characterised by using a fixed length genotype representation, crossover operators and mutation operator. It considers the relations defined by the fixed-length representation (binary bits or many others). It is possible to define a richer source of relations through representation. However, a sGA crossover favours those relations in which positions in sequence space are closer to each other and neglect those relations that contain positions far apart. This is usually called the *linkage problem*. A random mutation of the sample string also results in the disrupting proper evaluation of the relations. The sGA is not able to accomplish proper search in the relation space and has the major problems as (Kargupta 1995):

- Relation, class, and sample space are combined.
- Only a poor search for relation can occur.

Since the relation, class, and sample space are combined, the decision process becomes noisy and susceptible to error. The sGA does not have any way to explicitly decide about relation or classes.

3. MESSY GENETIC ALGORITHMS

A messy genetic algorithm (mGA) was developed to solve bounded difficult search problems (Goldberg, Deb & Korb 1989, 1990), which most likely lead a simple

GA search to a local optimum. The mGA allows the variable-length string to be used and each bit has attached a tag usually the order of the bit in a string. This flexible coding scheme usually generates an underspecified string in which some bits are missed or an overspecified string in which more than one value is specified for one bit. The underspecified string can be filled in by using *a competitive template*, while the overspecified string is reduced by using *first-come-first served* rule scanning from left to right. This variable locus representation solves the linkage problem.

The messy GA uses explicitly enumerative initialisation and two main operators namely thresholding selection, cut and splice operator.

Explicitly enumerative initialisation

At least one copy of all possible building blocks of a specified length k is provided. The initialisation of the messy GA is not exactly random. Rather, it provides all possible order-k equivalence classes (building blocks) in the initial population.

Thresholding selection

Comparing strings makes sense only when they are from the same relation. Thresholding selection tries to ensure that only classes belonging to a particular relation compete with each other. A similarity measure θ is used to denote the number of common genes among two strings. Two string are allowed to compete with each other if the θ is greater than a threshold value.

Cut and splice

Genetic operators, *cut* and *splice*, have been designed and used for the mGA instead of crossover and mutation as in the simple GA. *Cut* divides chromosome into two, while *splice* links two chromosomes as one individual. A cut operation is activated by a cut probability $P_c = P_k(l-1)$, where P_k is the specified bitwise cut probability, and l is the length of the string. Splicing is initiated by a prescribed probability P_s.

4. LESSONS FROM THE MESSY GENETIC ALGORITHM

Explicit enumeration of the classes and selection in the presence of thresholding aid precise evaluation and ordering of classes. This makes the messy GA more applicable than the sGA to a large class of problems. The advantages of a mGA are as follows.

(1) It decomposes the search space into the sample space —template space, and the class and relation space—the population strings. During the primordial phase every string in the population has a length less than the problem length. Each of them defines a class. Therefore, the strings in the primordial stage represent the class space. In the meantime, the template is always a full string and defines the sample space in the mGA.

(2) Less noise occurs in decision making in the mGA compared to the sGA. Since the main decision making in the mGA is made during the primordial stage, decomposition of the search space into the class space and relation space makes the decision less noisy when compared to sGA.

(3) A better search relation can be achieved. The search for a proper relation in the mGA is more accurate and less susceptible to error because of the explicit enumeration and the use of a local optimum as a template for class evaluation.

Despite these advantages, the main problem for the mGA is explicit enumeration of the building blocks. It prevents the mGA from being applied to highly dimensional problems. An improvement is the fast messy genetic algorithm (fmGA) that preserves the mGA search for good relations but provides the benefit of implicit parallelism.

5. FAST MESSY GENETIC ALGORITHM

The fast messy GA (Goldberg et al. 1993) replaces the explicitly enumerative initialisation by employing probabilistcially complete initialisation and a building-block filtering process.

Probabilistcially complete initialisation

The idea of the probabilistically complete initialisation is that all the classes can be defined using a much smaller number of strings, when the string length is much higher than k (the order of building blocks). In other words, multiple combinations of classes can be defined by the same string. This reduces the initial population size from $O(l^k)$ to $O(l)$, and consequently improves the search efficiency. The initialisation scheme generates a population of $O(l)$ of strings of length $l' = (l - k)$ for gene filtering process.

Building-block filtering process

The building-block filtering process offers a way for gradually detecting a certain order-k classes from string length l'. During this stage, a string is selected in the presence of thresholding and genes are randomly deleted by reducing the string length from l' to k.

6. MESSY GA APPLICATIONS TO PIPELINE NETWORKS

6.1 The Two Reservoir Network

A network with two water supply sources and fourteen pipes, studied by Simpson et al. (1994) as shown in Figure 1, has been chosen for this study. A complete analysis has been carried out previously by applying complete enumeration, linear programming, non-linear programming and the standard GA (Simpson et al. 1994; Simpson & Goldberg 1994). The global

optimum solution and a set of ranked solutions for this problem was found by using discrete pipe sizes using complete enumeration of every alternative. The results from the previous studies provide an excellent example for comparing the performance of the messy GA approach with the standard GA approach.

The network contains two reservoirs, five new pipes to be sized, and nine existing pipes-three of which may be rehabilitated by a pipe in parallel (referred to duplication although a different diameter may actually be selected), cleaning or alternatively left as they are. In Figure 1, solid lines represent the existing system, and dashed lines represent the parts of the system where pipe links [1], [4] and [5] may be rehabilitated, and pipe links [6], [8], [11], [13] and [14] that are to be sized with at least a minimum diameter pipe. Table 1 gives the pipe costs and available diameters. Three demand cases including two fire loading cases and one peak day loading case and the associated minimum allowable pressure heads are shown in Table 2.

Table 1 Available Pipe Sizes and Associated Costs for the Two Reservoir Network

Diam. (mm)	Cost for a new pipe ($/m)	Cost for a pipe in parallel to an existing pipe ($/m)	Cost for cleaning a pipe ($/m)
152	49.54	49.54	47.57
203	63.32	63.32	51.51
254	94.82	94.82	55.12
305	132.87	132.87	58.07
356	170.93	170.93	60.70
407	194.88	194.88	63.00
458	232.94	232.94	-
509	264.10	264.10	-

Table 2 Demand Patterns and Associated Minimum Allowable Pressures for the Two Reservoir Network

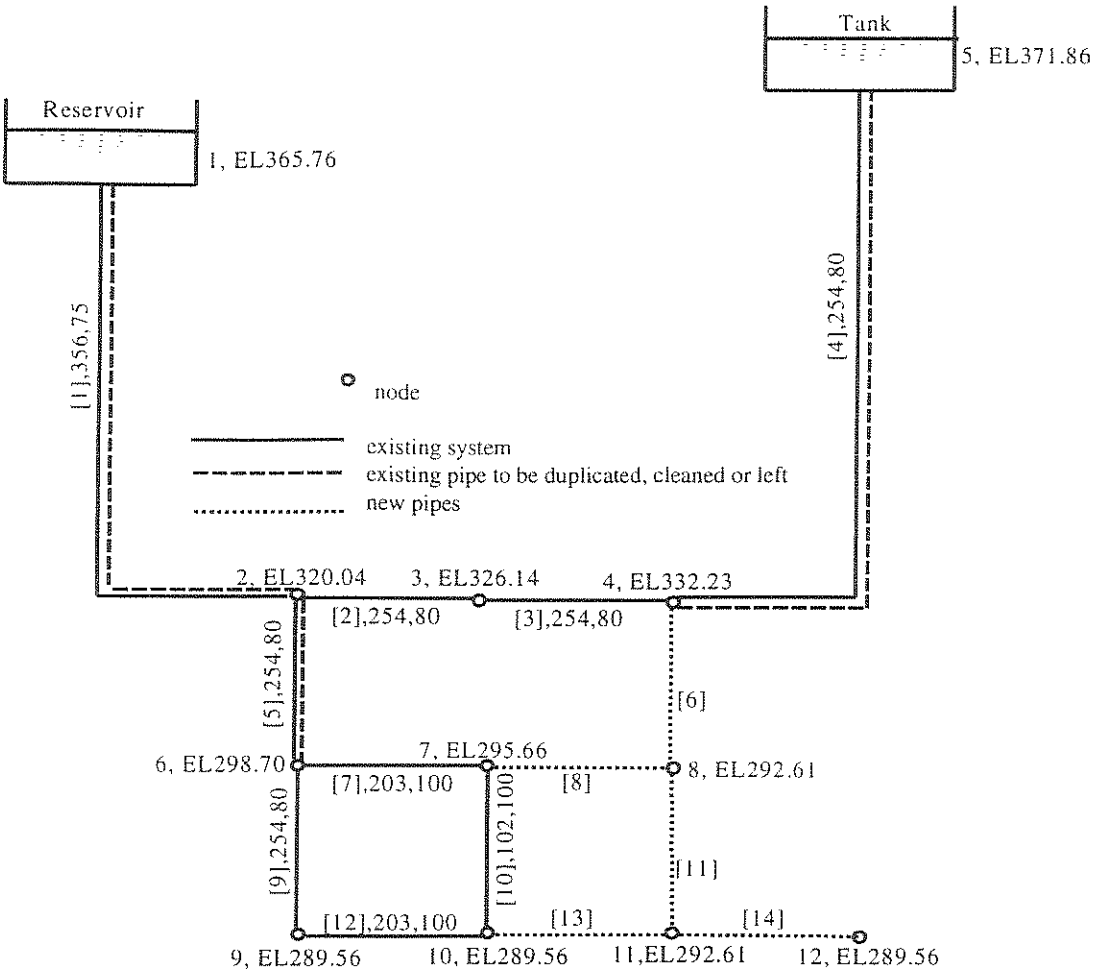
Node	Demand Pattern 1		Demand Pattern 2		Demand Pattern 3	
	Demand (L/s)	Minimum allowable pressure head (m)	Demand (L/s)	Minimum allowable pressure head (m)	Demand (L/s)	Minimum allowable pressure head (m)
2	12.62	28.18	12.62	14.09	12.62	14.09
3	12.62	17.61	12.62	14.09	12.62	14.09
4	0	17.61	0	14.09	0	14.09
6	18.93	35.22	18.93	14.09	18.93	14.09
7	18.93	35.22	82.03	10.57	18.93	14.09
8	18.93	35.22	18.93	14.09	18.93	14.09
9	12.62	35.22	12.62	14.09	12.62	14.09
10	18.93	35.22	18.93	14.09	18.93	14.09
11	18.93	35.22	18.93	14.09	18.93	14.09
12	12.62	35.22	12.62	14.09	50.48	10.57

Table 3 Population Sizes of Messy Genetic Algorithms for Optimisation of the Two Reservoir Network

Messy GAs	Original messy GA			Fast messy GA		
Eras	era 1	era 2	era 3	era 1	era 2	era 3
Initialisation	300	435	4060	60	60	60
Juxtapositional	150	150	150	150	150	150

6.2 Messy GA Coding, Decoding and Parameters

A genotype representation and the fitness formulation by Wu and Simpson (1996) have been used in this study. Three binary bits have been used to represent each pipe size variable for the five new pipes and three existing pipes to represent 8 possible choices of pipe sizes. Two binary bits have been used for each existing pipe to represent 3 possible choices of rehabilitation actions that include cleaning, leaving or duplicating an existing pipe. Thus 30 bits are needed for solving the problem by using the discrete diameter formulation. A decoding and mapping scheme from genotype to phenotype for optimisation of design and rehabilitation of water distribution systems is given by Wu and Simpson (1996). The penalty factor for pressures which do not meet the minimum allowable pressure constraints for this problem was chosen to be $5000/m of deficit to match the value taken by Simpson et al. (1994). Table 3 shows the population sizes used by the original and fast messy GA. The other parameters are splice probability P_s = 1.0, cut probability P_k = 0.017, Mutation Probability P_m = 0.01 and maximum generations N =10.



Pipes: [1], 356,75 [pipe number], diameter(mm), Hazen-Williams roughness C

Note. 1. All pipe lengths are 1609m, except pipe[1]=4828m and pipe[4]=6437m.

2. C=120 for new pipes and cleaned pipes.

Nodes: 2, EL320.04 node number, node elevation(m)

Figure 1 The Two Reservoir Network ***(from Simpson et al. 1994)***

6.3 Results and Comparison

The optimum discrete solution for this problem was found by the messy GAs with different random seeds and compared with the standard GA results (Simpson et al. 1994; Simpson & Goldberg 1994) as in Table 4. Typical convergence rates of the messy GA with the seed 0.7 are respectively given for using explicitly enumerative initialisation and probabilistically complete initialisation in Figure 2.

As shown in the Table 4, the fast messy GA is the most efficient at searching for the global optimal solution of discrete optimisation of pipeline networks. The messy GAs have found the lowest cost solution (global optimum) in each of the 10 runs with different random seeds. The numbers of original mGA evaluations needed for achieving the global optimal solution are less than for the standard GA. The messy GA using enumerative initialisation required only one third to half of the evaluation numbers of the standard GA (Simpson et al. 1994), and also less than the GA with tournament selection (selection pressure s = 2). Simpson and Goldberg (1994) observed that increasing tournament pressure (s = 5) for the standard GA could reduce the number of evaluations, and thus improve the search efficiency, but too much pressure (s =20) might lead the search to a local optimal. The fast messy GA, using probabilistically complete initialisation and building block filtering process, has further reduced the number of the evaluations, being approximately one third of the evaluation numbers of the original messy GA

Most of the evaluations in the original messy GA have been taken by the explicitly enumerative initialisation. It requires k-bit strings of $2^k\binom{l}{k}$ which provide all possible combinations of building blocks of order k, where l (l = 30 for the two reservoir network) is the

length of a string representing the problem. The original messy GA started the first era evolution with order 1, that is, one-bit combinations. 300 one-bit strings were generated by explicitly enumerative initialisation and evaluated by using a randomly generated gene string as a competitive template. The messy GA was followed by a *primordial phase*. During this stage, the genotypes were selected by performing the thresholding selection only, which enriches the highly fit strings in the population. Meanwhile the population size was reduced to 150 at the end of primordial phase for era 1. The next step was the *juxtapositional phase*, in which messy genetic operations such as *cut* and *splice* are applied to reproduce next generation. In the second era, 435 two-bit strings were initialised with order 2 building blocks and evaluated by using the best solution as the competitive template from the first era. For the third era, 4060 three-bit strings were initialised by partially complete enumeration with order 3. The number of evaluations required for the initial population for all 3 eras is more than 50% of the total number of evaluations. In general, the number of initial strings required by the original mGA is of $O(l^k)$. This disadvantage has been overcome in the fast messy GA.

The fast messy GA employed the probabilistically complete initialisation and the building block filtering process. The initialisation just required a population of $O(l)$ of string length $(l - k)$. The fast mGA started the first era (order 1 with $k = 1$) with 60 strings of 29-bit strings. It was followed the building block filtering process, in which the strings are cut in half every generation and evaluated by using a random template. Members for the next generation were selected by thresholding selection. The building block filtering process continues until the string length is equal to 1. The population size was increased to 150 at the end of building block filtering process. As for the original messy GA, it was then followed by the juxtapositional phase. For the second and the third eras the same population size as for the first era was used except the building block filtering was carried out for order 2 and order 3 respectively. Thus, the fast messy GA overcomes the bottleneck of the original messy GA and provides a more efficient search algorithm for the discrete optimisation of the pipeline networks.

7. CONCLUSIONS

The standard GA defines the relations and classes implicitly by using a fixed-length representation. It combines the relation space, class space and sample space all together, thus a poor and noisy decision process occurs. Increasing the tournament selection pressure can improve the search efficiency, but too much pressure may leads the search to a local optimal.

Messy GAs emphasise searching for appropriate relations. The original messy GA uses a competitive template and explicit enumeration of good classes—building-blocks—to ensure correct decision making. However, using an initialisation procedure where building-blocks are explicitly enumerated essentially limits the messy GA to be applied to a highly dimensional problem. The probabilistically complete initialisation and the building-block filtering process are introduced into the fast messy GA to detect better classes from better relations. The application of the messy GA to optimisation of pipeline networks in the case study in this paper shows that the fast messy GA is the most efficient algorithm among the genetic-based search paradigms. It eliminates the major bottleneck of the original messy GA —the explicitly enumerative initialisation and thus provides a promising optimisation algorithm for solving highly dimensional discrete optimisation problems.

8. REFERENCES

Arora J. S. & Huang M. W. (1994). "Methods for Optimisation of nonlinear problems with discrete variables: a review." *Structural Optimisation* 8, 69-85.

Dandy G. C., Simpson A. R., Murphy L. J. (1996). "An improved genetic algorithm for pipe network optimisation," *Water Resources Research*, Vol. 32, No. 2, 449-458.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimisation and machine learning*, Addison-Wesley Publishing Company, Inc., 412pp.

Goldberg, D.E., Korb, B., & Deb, K. (1989). "Messy genetic algorithms: Motivation, analysis, and first results." *Complex Systems, 3*, 493-530.

Goldberg D. E., Deb K., Kargupta H. & Harik G. (1993). "Rapid, Accurate Optimisation of Difficult Problems Using Fast Messy Genetic Algorithms." *IlliGAL Report*, No. 93004.

Holland, J.H. (1975). *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, Michigan.

Simpson A. R., Dandy G. C. & Murphy L.J. (1994) "Genetic algorithms compared to other techniques for pipe optimisation." *Journal of Water Resource Planning and Management*, ASCE, Vol. 120 No. 4, 423-443.

Simpson A. R. and Goldberg D. E. (1994). "Pipeline optimisation via genetic algorithms: from theory to practice." *2nd Int'l. Conf. on Pipeline Systems*, Edinburgh, Scotland.

Kargupta H. (1995). "SEARCH, Polynomial Complexity, and the Fast Messy Genetic Algorithm." *Ph.D Thesis*, University of Illinois at Urbana-Champaign.

Wu Z. Y. and Simpson A. R. (1996) "Messy genetic algorithms for optimisation of water distribution systems." *Research Report*, R140, Dept. of Civil and Envir. Eng., The University of Adelaide, Australia.

Table 4 Results of Comparison of GA Paradigms for the Two Reservoir Network

Run No.	Standard Genetic Algorithms					Messy Genetic Algorithms			
	Roulette wheel selection (Simpson et al. 1994) (N = 100; Pc = 0.9; Pm = 0.02)		Tournament selection (Simpson & Goldberg 1994) (N = 1000; Pc = 0.5; Px = 0.5; Pm = 0.0)			Explicitly enumerative initialisation (Wu & Simpson 1996)		Probabilistically complete initialisation	
	Cost (m$) (% difference from optimum)	Achieved at evaluation number	Cost ($m)	Achieved at evaluation number (s = 2)	Achieved at evaluation number (s = 5)	Cost ($m)	Achieved at evaluation number	Cost ($m)	Achieved at evaluation number
1	1.7910 (2.3%)	23,400	1.7503	9,000	4,000	1.7503	6,148	1.7503	1,112
2	1.7503*	10,350	1.7503	9,500	4,000	1.7503	6,148	1.7503	2,243
3	1.8417 (5.2%)	22,410	1.7503	8,500	4,500	1.7503	6,148	1.7503	1,855
4	1.8390 (5.1%)	15,660	1.7503	9,500	5,500	1.7503	8,958	1.7503	3,004
5	1.7503	17,190	1.7503	8,000	4,500	1.7503	2,957	1.7503	4,053
6	1.7503	11,070	1.7503	8,000	5,000	1.7503	2,522	1.7503	2,722
7	1.7503	10,080	1.7503	8,000	4,000	1.7503	8,758	1.7503	3,053
8	1.7999 (2.8%)	4,410	1.7503	7,500	4,500	1.7503	10,042	1.7503	2,622
9	1.7503	12,510	1.7503	10,000	4,000	1.7503	3,977	1.7503	1,622
10	1.7503	19,890	1.7503	10,000	3,000	1.7503	6,148	1.7503	1,722
Average		14,697		8,800	4,300		6,181		2,400

*The global optimum solution

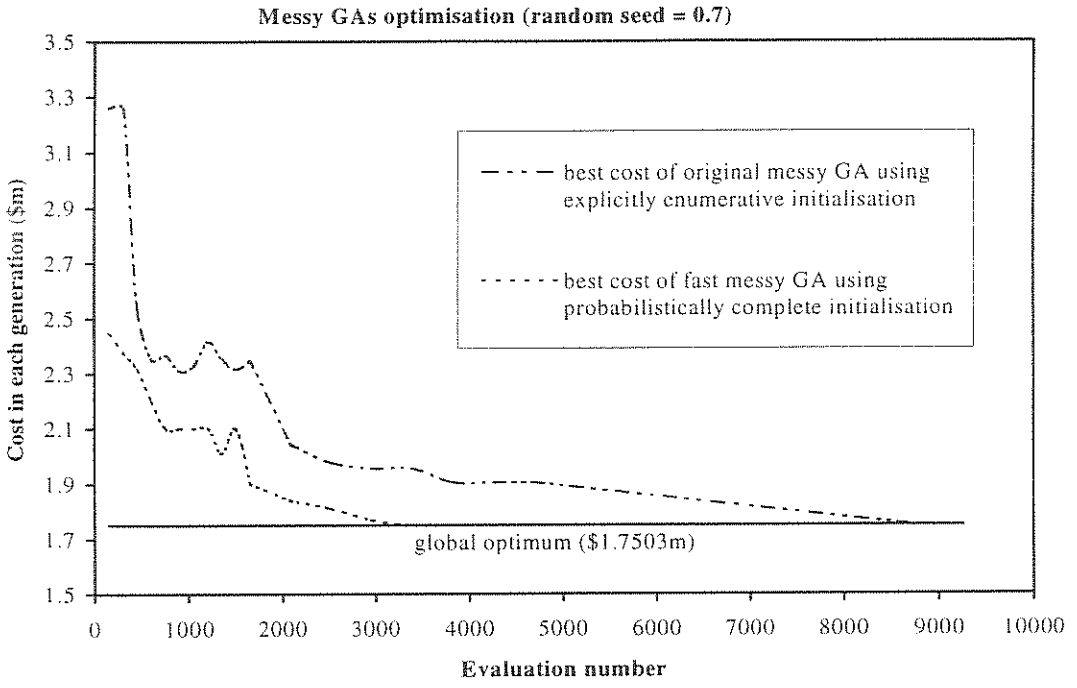


Figure 2 Comparison of Generation Best Cost for Messy Genetic Algorithm Optimisation of the Two Reservoir Problem