PERGAMON

Transportation Research Part C 9 (2001) 175–190

TRANSPORTATION RESEARCH PART C

www.elsevier.com/locate/trc

Constraint handling methods in pavement maintenance programming

Weng Tat Chan *, T.F. Fwa, Kh. Zahidul Hoque

Department of Civil Engineering, National University of Singapore, Kent Ridge Crescent, Singapore 119260, Singapore

Received 13 March 2000; accepted 15 May 2000

Abstract

The problem of pavement maintenance management at the network level is one of maintaining as high a level of serviceability as possible for a pavement network system through reactive and proactive repair actions, whilst optimising the use of available resources. This problem has traditionally been solved using techniques like mathematical programming and heuristic methods. Lately, the use of genetic algorithms (GAs) to solve resource allocation problems like the network pavement maintenance problem has received increased attention from researchers. GAs have been demonstrated to be better than traditional techniques in terms of solution quality and diversity. However, the performance of the GAs is affected by the method used to handle the many constraints present in the formulation of such resource allocation methods. Penalty as well as generate and repair methods are the usual techniques used to handle constraints, but these have their drawbacks in terms of computational efficiency and tendency to get trapped in sub-optimal solution spaces. The paper proposes a third method that is computationally more efficient than the previous methods. The method is based on prioritised allocation of resources to maintenance activities and the maximum utilisation of resources. Constraints on maximum resource availability are no longer used passively to check on solution feasibility (as in the previous methods) but are used to help generate feasible solutions during the resource allocation phase of the algorithm itself. It is demonstrated that the GA with the prioritised resource allocation method (PRAM) outperforms the traditional GA with repair or penalty methods. PRAM was able to consistently outperform the other two GA based methods, both in terms of solution quality as well as computational time. It is concluded that PRAM can be used as the basis of more efficient resource allocation procedures in the area of pavement maintenance management. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Pavement maintenance programming; Constraints; Genetic algorithms; Optimisation

* Corresponding author. Tel.: +65-8742576; fax: +65-7791635.
E-mail address: cvecwt@nus.edu.sg (W.T. Chan).

0968-090X/01/$ - see front matter © 2001 Elsevier Science Ltd. All rights reserved.
PII: S0968-090X(00)00023-1

1. Introduction

The problem of pavement maintenance management at the network level is one of maintaining as high a level of serviceability as possible for a pavement network system through reactive and proactive repair actions, whilst optimising the use of available resources. Optimisation techniques like priority ranking, linear programming, dynamic programming and heuristic search have an important role to play in determining the quality and cost-effectiveness of the maintenance program. Many of these optimisation methods have been described in the pavement maintenance literature. The priority ranking method (Fwa and Chan, 1993; Hajek and Phang, 1989), in particular, has been the traditional favourite among practitioners because of its intuitive appeal and ease of application. Lately, the genetic algorithm (GA) has emerged as a successful general purpose and robust alternative, even in the face of the 'combinatorial explosion' due to the large number of maintenance programming combinations (Chan et al., 1994; Fwa et al., 1994a,b).

The following elements need to be identified in any optimisation problem:

1. a set of decisions, with alternatives for each decision;
2. a set of constraints which collectively (and implicitly) describe the set of feasible combinations of choices for the decisions; these usually take the form of limits on the feasible values for the decisions; and
3. an objective function which measures the aggregated consequence of the decision choices made; this is expressed in terms of maximising or minimising a criterion like cost or service level.

An optimisation procedure is a systematic means of generating candidate solutions, distinguishing the promising from the mediocre, as well as improving the current best solution in order to attain the optimal solution. There are two key difficulties in the process of generating promising candidate solutions, namely, (1) directing the search for new solutions to the more promising regions of the search space, and (2) making sure that the solutions generated are within the feasible search space defined by the constraints.

In the GA-based approach to pavement maintenance programming, each solution is expressed as a set of decision choices called a *chromosome*. The choices are encoded (by position) in the '*genes*' that make up the chromosome. Typically, there would be many such solutions in a population pool; GAs work with a succession of solution populations. The search is conducted by constructing new solutions through the traditional GA operators of *crossover* and *mutation*. In crossover, two candidate solutions (represented as chromosome strings of '1s' and '0s') mutually exchange corresponding parts of themselves to form two new strings. These new strings represent new solutions that potentially have characteristics of both parents. In mutation, a new chromosome string is obtained by changing some of the bits in the string at random. This causes the search to be directed to new regions of the search space. Selection pressure is applied to these new strings by evaluating them with the objective function; this evaluation determines their chances of participating in the next round of crossover and mutation operations (with the better solutions being awarded more chances). The premise is that good traits lead to better evaluations of the individual in which they appear, thus favouring those individuals in the selection process and increasing the chances that the same traits survive to the next round.

GAs have traditionally worked well for unconstrained optimisation problems for if there are no constraints, then all new solutions generated by the GA are feasible solutions to the optimisation problem. However, real world problems like the pavement maintenance programming problem

have a multitude of constraints. These typically arise either from technological considerations (the particular work practices or technology adopted) or from limits on the availability of resources. Constraints pose a considerable challenge to GAs because of the way in which a GA generates new candidate solutions. The crossover and mutation process introduces the possibility that the new chromosomes produced will violate some of the constraints. For linear constraints and a convex feasible region, it is well known that linear combinations of points within the feasible space will always remain within the feasible space; it is therefore possible to modify the crossover process so that only linear combinations of the solution vectors are formed. However, this expedient solution does not carry over to the case of non-linear constraints and non-convex feasible regions. If constraints are not handled properly, much effort will be wasted in generating and evaluating infeasible solutions. There is also the strong likelihood that the search will not converge on the global optimum but get trapped instead in a sub-optimal part of the search space. It is for these reasons that constraints are said to pose considerable challenges to GAs in practical application. In this paper, we will review the performance of two popular constraint handling methods and introduce a new method developed by the authors which shows much promise.

All three methods are tested on a practical pavement maintenance problem described by Fwa et al. (1988) in order to evaluate their relative performance and merits. The rest of the paper is organised as follows: the pavement maintenance problem used for comparison is briefly described, followed by a review of existing constraint handling methods and a description of the proposed method. We then present the results of an experiment that compares the performance of the various methods on the chosen problem, as well as our observations on these results. The paper concludes with a summary of the key points made in the paper.

2. Pavement maintenance at network level

The problem of pavement maintenance at the network level was first described by Fwa et al. (1988) and solved using an integer programming approach. It was later solved again using genetic algorithms (Fwa et al., 1994b) with slightly better results being obtained. This problem is interesting for the present study because of the many constraints involved. The problem formulation is presented below in Eqs. (1)–(7). Table 1 defines the variables used in Eqs. (1)–(7). The objective is to maximise the total weighted work production resulting from undertaking a programme of maintenance activities (Eq. (1)). There are six constraints, namely production requirements, budget, manpower, equipment, material and rehabilitation constraints (2)–(7). However, it is assumed that there are enough materials for all types of maintenance work and the material constraint (6) is therefore relaxed. Except for production requirements and rehabilitation constraints, all the other three (budget, manpower and equipment) are considered as constraints on resource availability. The constraint on production requirements ensures that the required maintenance work is within their maximum limits. The rehabilitation constraint accounts for possible interference between routine and rehabilitative maintenance. The problem parameters are the highway types, urgency levels and the type of maintenance activities. Four highway types are considered: urban interstate, urban arterial, rural interstate and rural primary pavement segments. Four types of maintenance action are selected – crack sealing, premix levelling, shallow patching and deep patching. Maintenance urgency levels are a function of the extent and severity

Table 1
Definitions of variables in Eqs. (1)–(7)

Variable	Definition
W_{ijk}	Equivalent workload units in number of workdays of pavement repair activity j of need urgency level k performed on highway i
F_{ijk}	Priority weighting factor for pavement repair activity j of need urgency level k on highway i
N_1	Total number of highways considered
N_2	Total number of pavement repair activities considered
N_3	Total number of need urgency level considered
T_{ijk}	Total workload of pavement repair needs expressed in work measurement units for pavement repair activity j of need urgency level k on highway i
ϕ_{ijk}	Rehabilitation constraint factor for pavement repair activity j of need urgency level k, $0 \leqslant \phi_{ijk} \leqslant 1$
U_{ijk}	Work productivity for pavement repair activity j of need urgency level k on highway i
C_{ijk}	Cost per production unit of pavement repair activity j of need urgency level k on highway i
B	Total budget amount allocated for the analysis period considered
b_j	Budgeted fund for pavement repair activity j
h_{jg}	Number of mandays of work crew type g required for each unit of pavement repair activity j
H_g	Total available number of mandays of work crew type g
G	Total number of crew type
q_{jr}	Number of equipment days of equipment type r required for each production day of pavement repair activity j
Q_r	Total available number of equipment days of equipment type r
R	Total number of equipment type
m_{js}	Quantity of material type s required for each production day of pavement repair activity j
M_s	Total available quantity of material type s
S	Total number of material types
d_{ijk}	Number of working days before a scheduled rehabilitation during which no pavement repair activity j of need urgency level k would be performed on highway i
D	Total number of working days in the analysis period

of the pavement distress. High, medium and low urgency levels are considered for each of the maintenance activities. The maintenance work requirement for any urgency level of a highway pavement segment is expressed in terms of equivalent workdays. The problem is to decide the number of equivalent workdays for each decision variable (representing a maintenance action for a particular class of pavement segment at a certain urgency level). There are 48 ($4 \times 4 \times 3$) decision variables in the problem and each decision variable can have any value from 0 to 45. A value of 0 represents no action being assigned, whilst the upper limit of 45 comes from an analysis of the maximum value that can be assigned given the resource utilisation rates of the various pavement maintenance activities and the resource limits. The actual input data are not reproduced here and can be obtained from Fwa et al. (1988).

$$\text{Maximise} \quad \sum_{i}^{N_1}\sum_{j}^{N_2}\sum_{k}^{N_3} W_{ijk}F_{ijk}, \tag{1}$$

where $i = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, N_2$, $k = 1, 2, \ldots, N_3$.

(a) Production requirements:

$$0 \leqslant W_{ijk} \leqslant \frac{T_{ijk}\phi_{ijk}}{U_{ijk}}. \tag{2}$$

(b) Budget constraint:

$$\sum_{i}^{N_1}\sum_{j}^{N_2}\sum_{k}^{N_3} W_{ijk}U_{ijk}C_{ijk} \leqslant B. \tag{3a}$$

$$\sum_{i}^{N_1}\sum_{k}^{N_3} W_{ijk}U_{ijk}C_{ijk} \leqslant b_j \quad j = 1, 2, \ldots, N_2. \tag{3b}$$

(c) Manpower constraint:

$$\sum_{i}^{N_1}\sum_{j}^{N_2}\sum_{k}^{N_3} W_{ijk}h_{jg} \leqslant H_g \quad g = 1, 2, \ldots, G. \tag{4}$$

(d) Equipment constraint:

$$\sum_{i}^{N_1}\sum_{j}^{N_2}\sum_{k}^{N_3} W_{ijk}q_{jr} \leqslant Q_r \quad r = 1, 2, \ldots, R. \tag{5}$$

(e) Material constraint:

$$\sum_{i}^{N_1}\sum_{j}^{N_2}\sum_{k}^{N_3} W_{ijk}m_{js} \leqslant M_s \quad s = 1, 2, \ldots, S. \tag{6}$$

(f) Rehabilitation constraint:

$$\phi_{ijk} = \frac{D - d_{ijk}}{D}. \tag{7}$$

3. Constraint handling methods in GAs

The original GA formulation (Holland, 1975) had no explicit means of handling constraints. Indeed, constraints were conspicuously absent from the early problems handled by GAs. When practitioners interested in the application of GAs to real world problems added consideration of constraints to the formulation, many different ad hoc methods were introduced into the basic GA mechanism. It soon became apparent that the efficiency of GAs, when applied to constrained optimisation problems, is largely determined by how well constraints are handled in the algorithm.

The common constraint handling methods are: (1) the penalty method (PM), (2) the decode and repair method (DRAM), and (3) the genetic algorithms for numerical for constrained problems (GENOCOP) system. The first two methods have been in use for quite some time while the GENOCOP system (Michalewicz and Janikow, 1991) is the most recent.

3.1. Penalty method

In this method, a constrained problem is converted into an unconstrained problem by associating a penalty function with the evaluation function for each constraint violation. If a candidate solution violates any constraint, its evaluation value is penalised (reduced or increased) by a selected amount (depending on whether it is a maximisation or minimisation problem). Although this method is easy to use, careful consideration has to be given to the selection of an appropriate penalty value. There are no hard and fast rules on how to do this; the penalty value has to be large enough to prevent feasible solutions from being dominated by infeasible solutions, yet not too large as to trap the search in a local optimum.

3.2. Decode and repair method

This name is given to a class of methods that avoid the creation of invalid individuals or attempt to repair invalid individuals when they are created. Decode methods use special solution mapping representations that ensure that only feasible solutions are generated. Repair methods attempt to correct any invalid (infeasible) solutions that may be produced by the GA operations. This class of methods has been used for a long time but has not found wide usage because of several difficulties associated with their use. Firstly, these methods are computationally intensive because of the special representation used and the effort needed to repair infeasible solutions. Secondly, it is often not easy to code the repair algorithms, especially when several constraints are involved. Lastly, the special representations and repair algorithms are very problem specific and do not translate readily to another problem.

3.3. GENOCOP system

Michalewicz and Janikow (1991) introduced GENOCOP which was a generic system for handling both linear and non-linear constraints in a problem independent way. Equalities are first used to simplify the constraints, thus reducing the total number of problem variables and the resulting search space. Initial feasible solutions are then generated from the modified inequalities. This technique works well for convex search spaces formed by linear constraints but encounters difficulty with solution generation in non-convex search spaces. We dropped the method from further consideration after initial trials on our example problem revealed this.

4. Prioritised resource allocation method

From the insights gained in experimenting with the various constraint handling methods on our example, we developed a new method called the prioritised resource allocation method (PRAM). We believe PRAM is more efficient than any of the earlier methods reviewed and is based on the following insights:

1. As far as possible, the basic GA mechanism should not be tampered with – in particular, the selection pressure which drives the search into more promising parts of the solution space. The main problem with penalty methods is that the penalty term introduced into the objective function distorts the feedback received from the evaluation of the solutions.

2. As far as possible, the GA should avoid generating illegal solutions; otherwise, the GA could expend a great deal of computational effort generating illegal solutions, only to have to repair or discard them. It is also difficult to come up with special problem representations and operators that avoid constraint violations. Repair methods also distort the feedback given by the selection pressure.
3. When resource allocation involves several resources, local optimisations involving a single resource often lead to sub-optimal decisions. It is better to consider all active constraints simultaneously when making decisions.
4. Good solutions occur when several resources are stretched to the limit; indeed, a valuable insight from linear programming says that the optimal solution lies at the intersection of several constraints.
5. Whilst GAs perform admirably in searching globally for promising sub-regions of the search space, they are notoriously slow in converging to the local optimum in any sub-region. Fortunately, insight (4) also allowed us to overcome this limitation of GAs.

PRAM differs from the traditional GAs in several respects. Firstly, the chromosome string encodes more than the value of the decision variables – there are additional genes representing each of the resources appearing in the problem. Secondly, in PRAM the GA does not work on the value of the decision variable directly. Instead, the gene values represent priorities with which decision variables are allocated resources. The actual amount of resources to be allocated is determined dynamically from the constraints by another procedure. Lastly, the extra genes representing resources is used to optimise the performance of a separate resource allocation routine.

4.1. Description of the PRAM algorithm

The chromosome string representation used in PRAM is shown in Fig. 1 for a problem involving 48 decision variables. Besides the genes representing these decision variables (numbered from 1 to 48), there are three additional genes for the percentage of the requested budget, manpower and equipment that can be allocated each time (denoted with roman numerals (i)–(iii)), thus making a total of 51 genes in all. In Fig. 1, the integers inside the first 48 positions of the string represent the priority of the corresponding maintenance activity to receive its requested allocation. The purpose of having resource genes is to restrict the resource consumption by any single decision variable (even those with high priority).

All the strings in the pool are initialised randomly with each maintenance activity gene getting an integer in the range of 0–100. When a string is decoded, a sorted list of all the maintenance activities is constructed based on the values encoded by the genes. The maintenance activity with the highest priority receives a rank of 1 and the lowest is ranked 48.

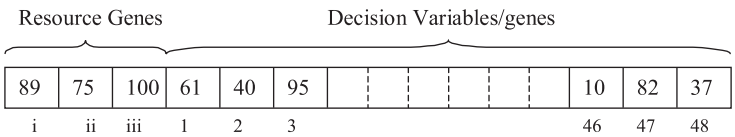


Fig. 1. String representation in PRAM.

Resource distribution is then performed in the order specified by the ranked list. An attempt is first made to allocate as many resources as are requested by the activity, subject to resource availability and the percentage limit imposed by the resource genes. If there are not enough available resources, the demand is adjusted proportionally to fit within the allowable resources. Allowable resource amounts are obtained simply by multiplying the resource limiting factors (obtained from the resource genes) by the amount of remaining available resources. If the minimum demand for the current activity still cannot be met, it is not allocated any and the process continues with the next decision variable on the list until all the activities have been considered.

PRAM does interfere with any of the normal GA operations but instead uses these operations on the chromosome strings to evolve the best combination of priority and resource limit values for the resource allocation algorithm. It is also obvious that PRAM always satisfies the resource constraints within the resource allocation algorithm itself – it does not need to make a trial guess, check on its validity and then either discard or repair the invalid trials.

5. Setting GA parameters

In the study, three constraint handling methods were implemented and their performance compared through an experiment. As far as possible, we tried to standardise the values of the GA parameters across the different constraint handling methods.

Population size. The penalty and DRAM methods required relatively large populations ($\geqslant 400$ numbers in order) whereas PRAM was able to produce equivalent results with only a fraction of the pool size (about 100). In order not to disadvantage PM and DRAM, the population size was set at 400 for these two methods but set to 100 for PRAM. Our experiments indicated that larger population sizes did not lead to much improvement in PRAM's results. Hence, the population size was limited to 100.

Population initialisation. PM did not require that all solutions in the initial pool be valid solutions, so the initialisation was straightforward. Each gene could be assigned a random number within the domain range. In contrast, DRAM required that all population members had to be valid solutions. This was achieved by applying the repair algorithm on a randomly initialised population until all strings are valid solutions. It was relatively easy to form the initial population for PRAM method since the genes could be randomly initialised.

Genetic operators. Although there are a number of variants of the crossover and mutation operators to choose from, we chose to use the basic two-point crossover and a simple bit mutation operator for all the constraint handling methods.

Stopping criterion. The quality of the solutions achieved in a GA run can depend on the stopping criterion used since this determines the number of solution trials allowed. We chose to use a simple criterion based on maximum number of iterations allowed. This maximum was set at 250 for all methods after some experimentation. This value strikes a balance between allowing enough trials for the slower methods to reach asymptotic performance and a reasonable amount of computational time for each run.

Penalty value. This parameter is only applicable to PM. We found that a penalty of between 500 and 1000 for each constraint violation could be used without adversely affecting the results obtained. If the penalty value was too small, few valid solutions would be produced because

invalid solutions achieved very high evaluations and the population tended to be dominated by infeasible solutions. On the other hand, too big a penalty value traps the GA in sub-optimal parts of the search space as a single valid (though poor) solution will quickly dominate the population.

6. Results and discussion

The performance of the various constraint handling methods (PRAM, DRAM and PM) can be compared in various ways. The most obvious is to look at the schedules produced by each of these methods at different points in time. From Table 2, it is obvious that PM fares the worst, both in terms of the best and average performances. Since there is no restriction on the validity of the solutions generated by PM (infeasible solutions were merely penalised), all the solutions in the initial pool for PM turned out to be invalid. This resulted in a large negative objective function value and it took more than 50 iterations for the first valid schedule to be produced. The best schedule produced by PM is only marginally better than the average of the schedules and by far, inferior to the best that PRAM and DRAM could produce. From the large negative average performance of PM throughout the run, it is evident that a significant portion of the population pool consists of infeasible schedules.

DRAM and PRAM start out about equal (since their initial average performances are comparable) but PRAM rapidly draws ahead of DRAM both in terms of best performance as well as average performance. Although the performances of the initial pool for PRAM and DRAM are comparable, DRAM requires more computational effort to decode and repair infeasible schedules. The contrast between the rate of improvement for all three methods is more evident from Fig. 2. PRAM is always ahead of the other two methods in achieving objective function values and the solutions in the PRAM pool are always better compared to those in the DRAM pool. Although the figures in Table 2 and Fig. 2 are obtained from a single run, all the runs we made showed qualitatively similar results. PM was never able to produce a schedule with an evaluation better than 4000 no matter how the pool size and the number of iterations were adjusted. Similarly, DRAM

Table 2
Best and average objective function values by the three methods[a]

After iteration	Objective function (OF) values					
	PRAM		DRAM		PM	
	Best	Average	Best	Average	Best	Average
1	3632.0	2672.3	3309.5	2610.5	−232421.5	−305950.0
50	4245.0	3754.6	4049.5	3372.9	−2967.5	−48221.0
100	4304.5	3876.8	4141.0	3464.6	2667.0	−34965.0
150	4315.0	3854.7	4153.5	3483.2	3235.5	−49091.0
200	4315.0	3849.0	4153.5	3457.4	3510.5	−44892.0
250	4323.5	3901.7	4176.0	3497.4	3622.5	−48482.0

[a] All these values are from a particular run that produced best OF values. Various combination of pool sizes and number of iterations were tried. But PM never produced an OF value >4000. Similarly, DRAM never reached the best OF values obtained by PRAM in any combination applied to both.

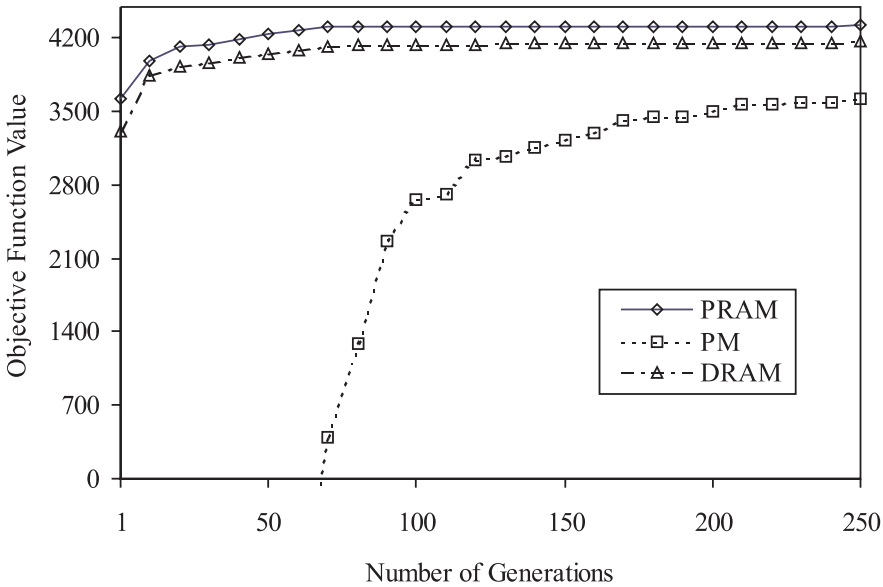


Fig. 2. Convergence pattern of the three methods.

was never able to reach the best value obtained by PRAM indicating that the former gets trapped easily in some sub-optimal local solution. For comparison purposes, the best evaluation values for the same problem was 3865.5 using an integer programming solution method (Fwa et al., 1988) and 4056.0 using a conventional GA solution method (Fwa et al., 1994b).

The three methods can also be compared by the diversity of the solutions produced. We analysed the maintenance actions suggested by the best 10% of the solutions in the final pool returned by each method. Table 3, which shows the results of our analysis, suggests that in all three methods, the top 10% of the schedules are largely homogeneous. In DRAM, there were two distinct solutions differing only by the value of two variables. One of these solutions (the second best) made up 90% of the top 10% of solutions in the pool. PRAM also produced two unique solutions differing by the values of four variables. In PRAM, one solution was present in 85% of the top 10% of solutions. PM produced three unique (and valid) solutions in the top 10% range; these solutions differed by the values of five variables.

We can also compare the three methods by the way they utilise resources. Table 4(a) compares the utilisation rates of the budget, manpower and equipment resources. All three methods use budget and manpower more heavily than equipment although PRAM achieves the highest utilisation factor. Despite the similarities in resource utilisation rates, the results produced by the three methods are significantly different, as we have previously discussed. PRAM achieves the highest equivalent work production of the three (the non-weighted total number of workdays for all three methods is roughly the same). It comes as no surprise that PRAM was able to maximise the use of the resources since it is designed to do just that. Surprisingly, DRAM also did well but further analysis points to a plausible reason: the repair algorithm in DRAM also used the concept of maximising resource utilisation. Table 4(b) shows the range of values obtained by the best 10% of solutions in the final population pool produced by the three methods. The figures show that

Table 3
Suggested decision values in equivalent workdays and their occurrence in the final pool of solutions

Highway type	Need urgency level	Repair strategy	Equivalent workdays		
			PRAM	DRAM	PM
($i=1$)	$k=1$	$j=1$	3 (100%)	3 (100%)	3 (100%)
		$j=2$	4 (100%)	4 (100%)	3 (95%), 4 (5%)
		$j=3$	8 (100%)	7 (100%)	8 (100%)
		$j=4$	1 (100%)	0 (100%)	0 (100%)
	$k=2$	$j=1$	4 (100%)	4 (100%)	2 (10%), 4 (90%)
		$j=2$	3 (15%), 2 (85%)	1 (100%)	1 (100%)
		$j=3$	1 (100%)	1 (100%)	0 (100%)
		$j=4$	3 (100%)	0 (100%)	1 (100%)
	$k=3$	$j=1$	3 (100%)	2 (100%)	2 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (100%)
		$j=3$	0 (100%)	0 (100%)	0 (100%)
		$j=4$	0 (100%)	0 (100%)	0 (100%)
($i=2$)	$k=1$	$j=1$	1 (100%)	1 (100%)	1 (100%)
		$j=2$	0 (100%)	0 (100%)	0 (100%)
		$j=3$	0 (15%), 1 (85%)	7 (100%)	6 (10%), 4 (90%)
		$j=4$	1 (100%)	0 (100%)	0 (100%)
	$k=2$	$j=1$	1 (100%)	0 (100%)	1 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (100%)
		$j=3$	0 (100%)	0 (100%)	0 (100%)
		$j=4$	0 (100%)	0 (100%)	0 (100%)
	$k=3$	$j=1$	3 (100%)	3 (100%)	1 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (100%)
		$j=3$	0 (100%)	0 (100%)	0 (100%)
		$j=4$	0 (100%)	0 (100%)	1 (100%)
($i=3$)	$k=1$	$j=1$	4 (100%)	4 (100%)	4 (100%)
		$j=2$	6 (15%), 7 (85%)	8 (100%)	0 (100%)
		$j=3$	6 (15 %), 5 (85%)	0 (100%)	1 (100%)
		$j=4$	4 (100%)	0 (100%)	2 (100%)
	$k=2$	$j=1$	3 (100%)	3 (10%), 2 (90%)	1 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (5%), 2 (95%)
		$j=3$	1 (100%)	0 (100%)	0 (100%)
		$j=4$	1 (100%)	9 (100%)	0 (100%)
	$k=3$	$j=1$	1 (100%)	3 (100%)	3 (100%)
		$j=2$	0 (100%)	0 (100%)	0 (100%)
		$j=3$	0 (100%)	0 (100%)	0 (100%)
		$j=4$	0 (100%)	0 (100%)	0 (100%)
($i=4$)	$k=1$	$j=1$	3 (100%)	3 (100%)	3 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (100%)
		$j=3$	0 (100%)	0 (100%)	1 (5%), 3 (95%)
		$j=4$	0 (100%)	0 (100%)	0 (100%)

Table 3 (continued)

Highway type	Need urgency level	Repair strategy	Equivalent workdays		
			PRAM	DRAM	PM
	$k=2$	$j=1$	0 (100%)	2 (10%), 3 (90%)	1 (100%)
		$j=2$	0 (100%)	0 (100%)	4 (100%)
		$j=3$	0 (100%)	0 (100%)	1 (100%)
		$j=4$	0 (100%)	0 (100%)	1 (100%)
	$k=3$	$j=1$	0 (100%)	0 (100%)	0 (100%)
		$j=2$	0 (100%)	0 (100%)	1 (100%)
		$j=3$	0 (100%)	0 (100%)	1 (100%)
		$j=4$	0 (100%)	0 (100%)	3 (100%)

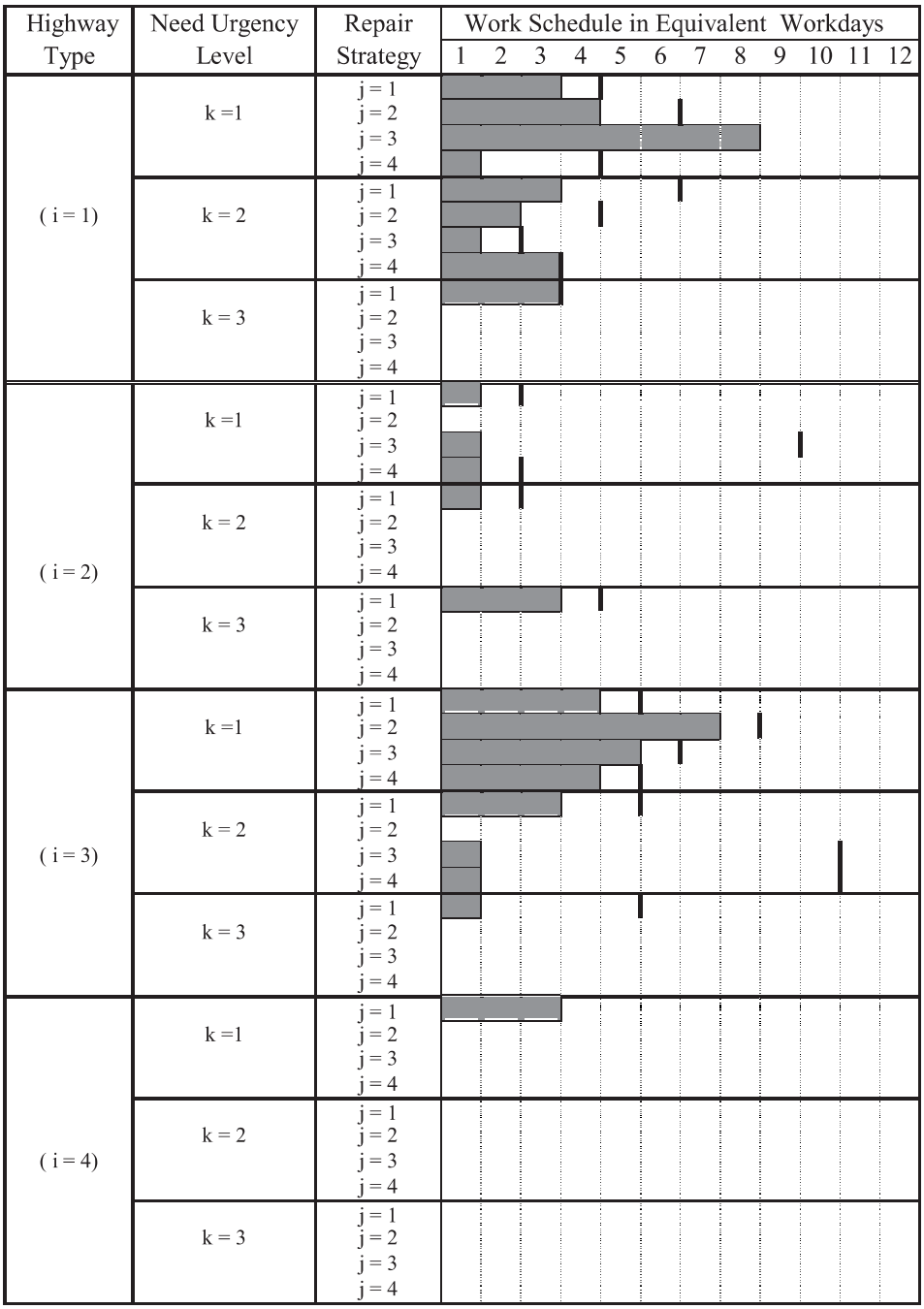
Table 4
Comparison on the results of the three methods

Item	Constraint handling methods		
	PRAM	DRAM	PM
(a) Comparison on the basis of work production and resource utilization of the best solutions			
Work production	4323.5	4176	3622.5
Budget utilized	54581.7 (90.9%)	52500.8 (87.5%)	52224.8 (87%)
Manpower utilized	526 (89.91%)	518 (88.54%)	522 (89.23%)
Equipment utilized	243 (67.5%)	234 (65.0 %)	247 (68.61%)
Total workdays	65	65	63
(b) Comparison on the basis of top 10% solutions in the pool			
Range of OF[a] values	4315.0–4323.5	4153.5–4176.0	3587.5–3622.5
Unique solutions	2	2	3

[a] OF – Objective function.

these ranges do not overlap, perhaps indicating different rates of improvement in the objective function.

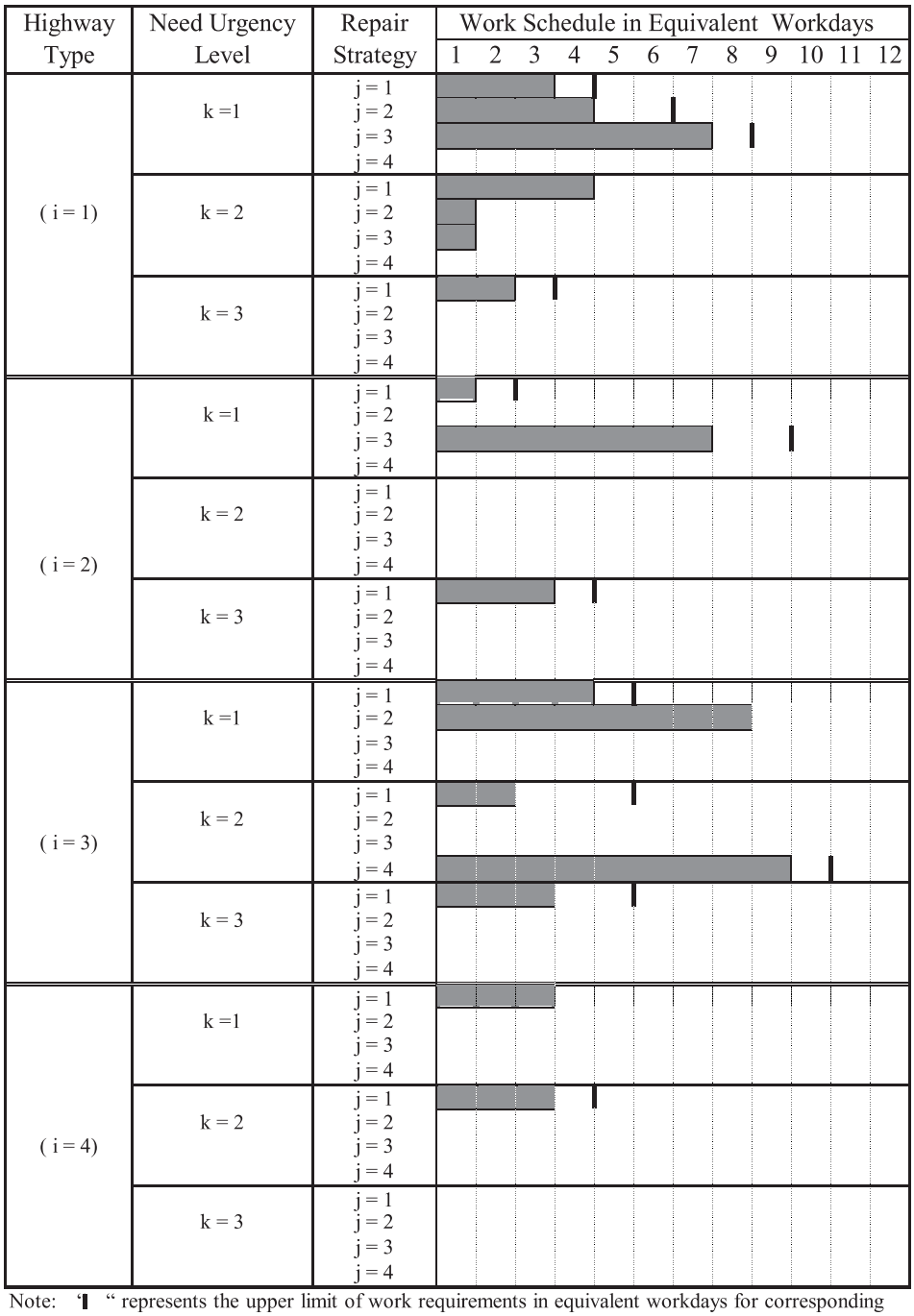
Finally, the proposed work schedules by the three methods are presented in Figs. 3(a)–(c). Both PRAM and DRAM schedules show a great deal of similarity with each other and seem to indicate a preference to assign work to road types that are high in priority, and to road defects that are severe (and rightly so!). Urban interstate and rural interstate with urgency levels 1 and 2 got the maximum repair work allocation, whilst other roads got very little or none at all. However, DRAM also assigned jobs to less important roads, which might account for less work production in the DRAM schedule. The PM schedule showed that it assigned values for even the least important roads with the least severity; this could be the main reason that PM failed to produce results as good as that of PRAM and DRAM. Close examination of the schedules produced by each of the three methods also indicates that the PRAM schedule operates closer to the upper limit of the work requirements as compared to the other two schedules, especially for the more heavily weighted decision variables.



Note: "▮" represents the upper limit of work requirements in equivalent workdays for corresponding decision variables.

(a)

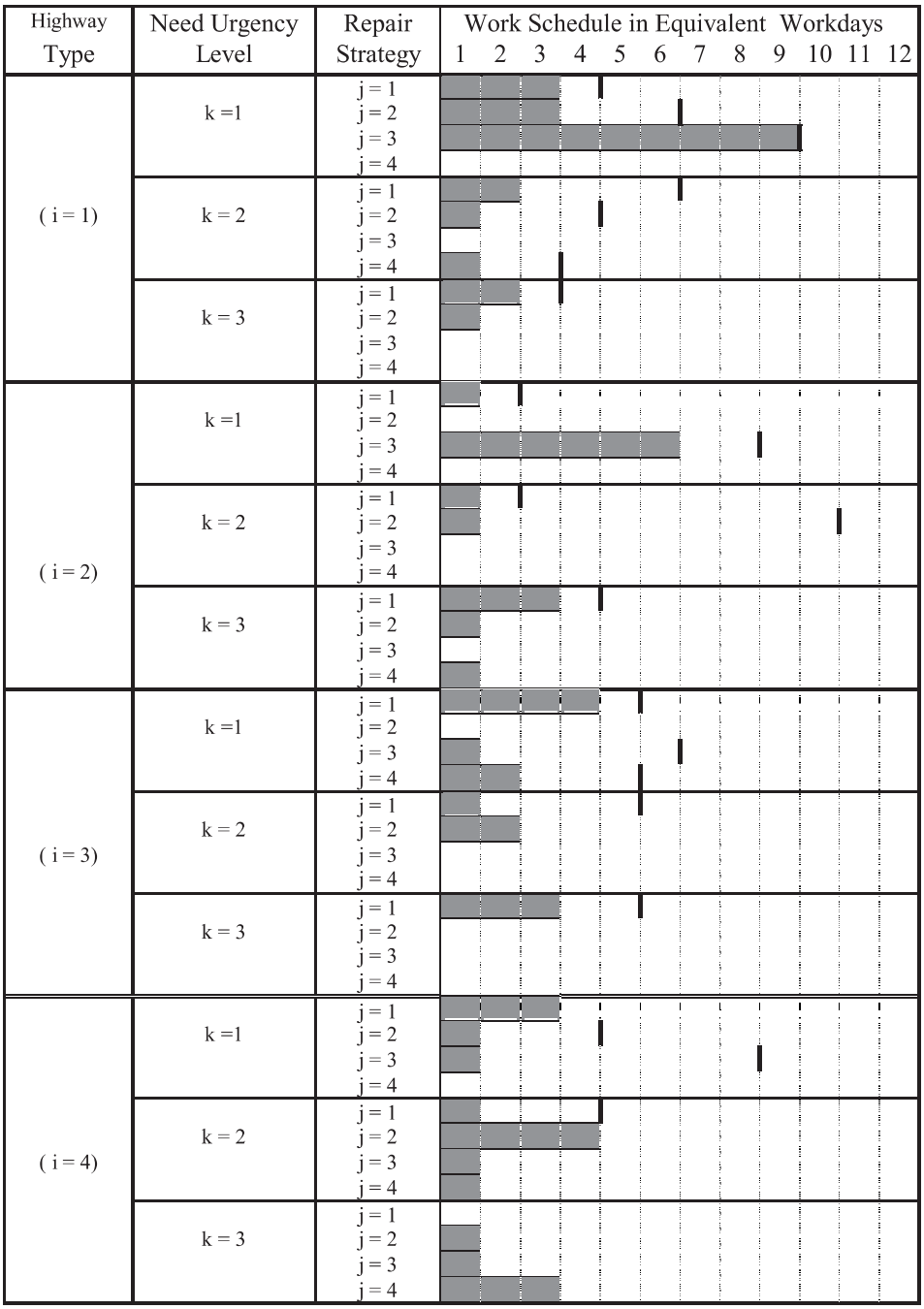
Fig. 3. (a) Proposed work schedule by PRAM, (b) proposed work schedule by DRAM, (c) proposed work schedule by PM.



Note: "▌" represents the upper limit of work requirements in equivalent workdays for corresponding decision variables.

(b)

Fig. 3. (*continued*).



Note: '▌ " represents the upper limit of work requirements in equivalent workdays for corresponding decision variables

(c)

Fig. 3. (*continued*).

7. Conclusions

In this paper, a new constraint handling method named PRAM was introduced for handling resource constraints when optimising pavement network maintenance programmes. Its performance was compared against two common constraint handling methods namely, the PM and the DRAM. PRAM showed great promise – it returned the best performance measures of the three methods as well as being the fastest computationally. The key concept behind PRAM is that of maximum resource utilisation. When we employed this same concept in the repair part of DRAM, it also worked very well. The traditional penalty method fared the worst.

References

Chan, W.T., Fwa, T.F., Tan, C.Y., 1994. Road-maintenance planning using genetic algorithms I: Formulation. Journal of Transportation Engineering ASCE 120 (5), 693–709.

Fwa, T.F., Sinha, K.C., Riverson, J.D.N., 1988. Highway routine maintenance programming at network level. Journal of Transportation Engineering ASCE 114 (5), 539–554.

Fwa, T.F., Chan, W.T., 1993. Priority rating of highway maintenance needs neural networks. Journal of Transportation Engineering ASCE 119 (3), 419–433.

Fwa, T.F., Chan, W.T., Tan, C.Y., 1994a. Road-Maintenance planning using genetic algorithms II: Analysis. Journal of Transportation Engineering ASCE 120 (5), 710–722.

Fwa, T.F., Chan, W.T., Tan, C.Y., 1994b. Optimal Programming by Genetic Algorithms for Pavement Management. Transportation Research Record No. 1455, Transportation Research Board, Washington, DC, pp. 31–41.

Hajek, J.J., Phang, W.A., 1989. Prioritization and Optimization Pavement of Preservation Treatments. Transportation Research Record No. 1216, TRB, National Research Council, Washington DC, pp. 58–68.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Michalewicz, Z., Jainkow, C., 1991. Handing constraints in genetic algorithms. In: Belew, R., Booker, L. (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms. Morgan Kaufmann, San Diego, pp. 151–157.