ICARUS: Intelligent Coupon Allocation for Retailers Using Search

Stephen Swift
School of Information Systems, Computing and Mathematics,
Brunel University,
Uxbridge, Middlesex,
UB8 3PH, UK
Stephen.Swift@brunel.ac.uk

Amy Shi
Loyalty Logic Limited,
Logic House,
Waterfront Business Park,
Fleet Road,
Fleet, Hampshire,
GU51 3SB, UK
Amy.Shi@loyaltylogic.co.uk

Jason Crampton
Information Security Group,
Department of Mathematics,
Royal Holloway,
University of London,
Egham, Surrey,
TW20 0EX, UK
Jason.Crampton@rhul.ac.uk

Allan Tucker
School of Information Systems, Computing and Mathematics,
Brunel University,
Uxbridge, Middlesex,
UB8 3PH,UK
Allan.Tucker@brunel.ac.uk

Abstract- Many retailers run loyalty card schemes for their customers offering incentives in the form of money off coupons. The total value of the coupons depends on how much the customer has spent. This paper deals with the problem of finding the smallest set of coupons such that each possible total can be represented as the sum of a pre-defined number of coupons. A mathematical analysis of the problem leads to the development of a Genetic Algorithm solution. The algorithm is applied to real world data using several crossover operators and compared to well known straw-person methods. Results are promising showing that considerable time can be saved by using this method, reducing a few days worth of consultancy time to a few minutes of computation.

1 Introduction

One of the challenges facing more and more businesses today, particularly retailers, is how to meet individual customers' needs on a mass scale of millions. Loyalty schemes, directly connected to individual customers, play an increasingly important role in tackling this challenge. Loyalty schemes aim at building a long-term customer relationship, enabling the business to understand customers and their consuming habits, whilst offering customers rewards in line with their contribution towards the business. Customer segmentation, target marketing and personalised service are used a great deal by successful retailers to increase the customers' satisfaction (Humby 2003). However, one component that has been generally ignored by most in the past is that of personalised rewards. Research into consumer behaviour suggests that rewards such as monetary coupon, has a genuine positive impact on the customers (Schmitt 2003). However, the degree of influence varies between different customer segments and often leads to different response behaviour. Response analysis from previous marketing activities also reveals that the different layout of reward coupons, i.e., number of coupons and monetary value of the coupon, can result in a different response rate. For example, one segment responds better to four one-pound coupons while another segment preferred one four-pound coupon.

This paper discusses the business and practical issues associated with personalised rewards, and provides an intelligent, flexible and effective solution for generating a reward layout which suits different customer's needs and maximises the return on investment. A Genetic Algorithm (GA) (Holland 1975) based approach, ICARUS (Intelligent Coupon Allocation for Retailers Using Search), is presented within this paper. This method is compared with conventional search based techniques and with a number of different crossover operators. All methods employed are evaluated against a real world instance of the coupon allocation problem.

Section 2 describes the problem being addressed in this paper in more detail, including presenting a mathematical description which is fully exploited by ICARUS. Section 3 describes all of the components which make up the ICARUS method and Section 4 evaluates the technique against the conventional methods and appraises several crossover operators. Finally Section 5 draws some conclusions.

2 Background

2.1 Business Objectives

Reward coupons are usually mailed out once every four to six months by the retailer to its scheme members. The reward value to each member is a percentage of the contribution made by that member during a certain period, rounded down to the nearest integer pound. This means the reward value potentially varies from member to member, and the highest reward value varies from period to period. Each member receives a number of coupons which add up to the total monetary value of the reward. The monetary value

0-7803-9363-5/05/$20.00 ©2005 IEEE.

on each coupon is encoded as a barcode, and scanned in at the point of sale. Members can use one or more coupons to have money off their shopping but no change will be given if the coupon value exceeds the actual spend. The marketing objectives are to allow members to redeem their coupon(s) easily whatever their basket spend normally is, and to generate more store traffic and potentially higher spend.

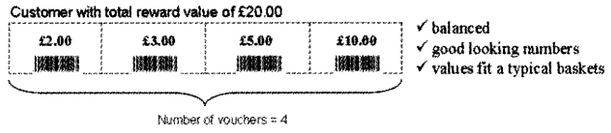
As mentioned in the Introduction, personalising the reward will enhance the positive influence, maximising the opportunities to achieve the marketing objectives. The challenge is to provide an intelligent coupon allocation solution which serves the marketing purpose and can take the business requirement into account, which is also practical, flexible and reasonably cheap to run in terms of time and resource required.

2.2 The Monetary Allocation Problem

There are several practical issues which need to be considered in association with the reward system:

1. The monetary value of each coupon should be well balanced/evenly distributed;
2. "Good looking" numbers are better received by customers, i.e. most people prefer to deal with a number divisible by five;
3. Coupon values should not exceed the typical basket value, which is between £25 and £35;
4. Each monetary value is bar-coded but not personalised (Mr. Sample1's £1 shares the same barcode as Mrs Sample2's £1 coupon);
5. The coupon value of zero should only be used where strictly necessary, since such coupons cannot be used/spent by a consumer.

An example of a simplified and well-allocated coupon is as follows:



An example of bad allocated coupon:

This paper will address the following two goals.

The first is the minimising of the number of different monetary coupons. This is so that the coupon bar-coding is easy to manage, i.e. the range of available values is not exhausted in too short a time. Each bar-code scheme has a fixed number of bar-codes available, which needs to encode the coupon denomination and expiry date (or validity date). The frequency of each run (every four to six months) is fixed along with the date information hence the only variable is the number of coupon denominations used within each run. For example, if there was room for a total of 64 coupon denominations, then eight denominations of coupons each run may mean that the scheme could last for four years (assuming a run every six months), whilst 32 denominations would mean the system would need redesigning after one year.

The second goal is that the range of coupon denominations must be able to represent the full range of customers' reward totals, with no exceptions.

2.3 Notation

The range of reward totals to be sent out to the customers will be denoted as the list V containing v_1 to v_n, where $v_1 \geq 0$ and $v_i < v_{i+1}$ for $1 \leq i < n$. Each value v_i will consist of exactly m coupon amounts where the set $A = \{a_1,\ldots,a_k\}$ will be used to denote the denominations the coupons can take. Each v_i is made up of a sum of m elements from A as defined in equation (1).

$$v_i = \sum_{j=1}^{m} x_{ij}, \text{ where } 1 \leq i \leq n \text{ and } x_{ij} \in A \tag{1}$$

The notation $[k]$ will be used to represent the set $\{0,1,\ldots,k\}$.

A trivial solution to this problem is to set A equal to $\{0,\ldots,v_n\}$ and then choose (search for) m numbers from A that equal each v_i as appropriate. However, the practical restrictions detailed in section 2.2 make this approach non-viable. The set A must be as small as possible since there are a finite number of barcodes available. The solution above would mean that the barcode format would need changing frequently. The elements of the set A must be aesthetically pleasing for the consumer, e.g. divisible by five and the elements $x_{i1},\ldots,x_{im}$ must be as close together as possible but dissimilar.

2.4 Mathematical Representation

Consideration will be first given to satisfying equation (1). Given $m > 1$ and a set of integers $A = \{a_1,\ldots,a_k\}$ and $a_i < a_{i+1}$, $1 \leq i < k$, does equation (1) hold? For convenience, let $\Phi(v_i, m, \{a_1,\ldots,a_k\})$ denote an instance of the problem.

For example $\Phi(7, 3, \{1, 3\})$ and $\Phi(9, 3, \{1, 3\})$ have solutions, but $\Phi(8, 3, \{1, 3\})$ does not.

An immediate observation is that $\Phi(v_i, m, \{a_1,\ldots,a_k\})$ has no solution if one of the following conditions holds:

$$v_i > ma_k \tag{2}$$

$$m = 1 \text{ and } v_i \notin \{a_1,\ldots,a_k\} \tag{3}$$

Note that an instance of the problem can be turned into a simpler instance of the problem. In particular, $\Phi(v_i, m, \{a_1,\ldots,a_k\})$ has a solution if and only if at least one of

$$\Phi(v_i - a_j, m-1, \{a_1,\ldots,a_k\}),\ 1 \leq j \leq k$$

has a solution. Hence, for example, it may be deduced that $\Phi(8, 3, \{1, 3\})$ has no solution because if it did either $\Phi(5, 2, \{1, 3\})$ or $\Phi(7, 2, \{1, 3\})$ would have a solution. By (2), $\Phi(7, 2, \{1, 3\})$ has no solution. Now $\Phi(5, 2, \{1, 3\})$ has no solution because neither $\Phi(2, 1, \{1, 3\})$ nor $\Phi(4, 1, \{1, 3\})$ has a solution by (3).

As another example consider $\Phi(14, 3, \{1, 3, 5\})$, which gives rise to three new problems:

$$\Phi(9, 2, \{1, 3, 5\}),$$
$$\Phi(11, 2, \{1, 3, 5\}),$$
$$\Phi(13, 2, \{1, 3, 5\}).$$

By (2), only the first of these can have a solution. However, this leads to the problem instances:

$$\Phi(4, 1, \{1, 3, 5\}),$$
$$\Phi(6, 1, \{1, 3, 5\}),$$
$$\Phi(8, 1, \{1, 3, 5\}).$$

By (3) none of these have a solution, hence $\Phi(14, 3, \{1, 3, 5\})$ does not have a solution.

Clearly, this provides a constructive method for not only determining whether a problem instance has a solution, but what that solution is. The procedure described above results in a "tree" rooted at the original problem; a "branch" between two problems can be labelled with the value deducted to arrive at the simpler problem. The solution can be determined by backtracking from a successful solution picking up the values from the branches. The procedure outline above is naturally recursive, with three different possible terminating conditions for $\Phi(v_i, m, \{a_1,\ldots,a_k\})$:

$$m = 1 \text{ and } v_i \in \{a_1,\ldots,a_k\}$$
$$v_i > ma_k$$
$$v_i < 1$$

The first of these means there is a solution; the remaining two mean that no solution is possible.

Before some mathematical properties of the problem are considered, further notation is needed.

Let $\pi(v_i, m, A) = \{x_1,\ldots,x_m\}$ be the first solution generated by $\Phi(v_i, m, A)$ or the empty set if there are no solutions.

Let $\Pi(v_i, m, A)$ be the set of all solutions generated by $\Phi(v_i, m, A)$ or the empty set if there are no solutions.

Proposition 1. If $\Phi(v_i, m, [k])$ has a solution then all $\Phi(w, m, [k])$ for $0 \le w \le v_i$ have a solution.

Proof. Clearly $\Phi(0, m, [k])$ has a solution. Proceeding by induction, it is assumed that that $\Phi(w, m, [k])$ has a solution for $0 < w < v_i$, and then $\Phi(w+1, m, [k])$ is considered. Now by assumption

$$w = \sum_{j=1}^{m} b_j, \text{ where } b_j \in [k]$$

and since $w < v_i \le m_k$ there exists an l such that $b_l < k$. Hence,

$$w+1 = \left(\sum_{\substack{j=1 \\ j \ne l}}^{m} b_j \right) + (b_l + 1).$$

That is, $\Phi(w+1, m, [k])$ has a solution and the result follows by induction. ■

Proposition 2. The smallest value that k can take for $\Phi(v_i, m, [k])$ to have a solution is:

$$k = \left\lceil \frac{v_i}{m} \right\rceil \quad (4)$$

where, $\lceil x \rceil$ is the smallest integer greater or equal to x.

Proof. If $\Phi(v_i, m, [k])$ has a solution, then $v_i \le m_k$. Since $m > 0$ then

$$k \ge \frac{v_i}{m},$$

Which implies that

$$\left\lceil \frac{v_i}{m} \right\rceil$$

is the smallest integer such that $\Phi(v_i, m, [k])$ has a solution. ■

2.5 Related Work

As far as the authors are aware, there has been little or no previous work in addressing this problem. However Genetic Algorithms and search techniques (Michalewicz 1998) in general have been used in many similar types of problems. There are many applications of using a GA to solve combinatorial, partitional or ordering problems, e.g. (Garey 1979, Sacerdoti 1977, Goldberg 1985). In the field of loyalty card data analysis, most work has concentrated on the processing of "basket data" using association rules (Agrawal 1994).

3 The ICARUS Method

Within this paper, the coupon allocation problem will be solved using a binary Genetic Algorithm. The following sub-sections detail the specifics of the technique.

3.1 Representation

Given the requirements set out in Section 2.2, the main goal will be to select the smallest set A. Proposition 1 and 2 can help in determining the size of the chromosomes. Given values for v_n and m then equation 4 gives us a minimum size that can be used. Since it is desirable to avoid using the zero valued coupon where possible, it is noted that the zero will only be needed for small values of v_i, and therefore will be added to A when $v_i < m$. Therefore A will consist of a subset

of the set $\{1,\ldots,k\}$ determined by the best individual from the GA. Each individual will be of length k, and the ith gene determines whether the corresponding set A contains the value i. For example, for subsets of $\{1,2,3,4,5\}$ then the chromosome 10101 corresponds to the set $A = \{1,3,5\}$.

3.2 Fitness

Given the requirements described in section 2.3, two functions can be designed, as shown in equations 5 and 6, that form the fitness function for the GA.

$$G(A) = \left(\sum_{i=1}^{|A|} g(a_i) \right) + 1$$
$$g(a_i) = \begin{cases} 1 & , a_i \bmod 5 = 0 \\ \alpha & , \text{otherwise} \end{cases} \quad (5)$$

The above function, $G(A)$, will score low values for small sets of numbers divisible by five.

The aim of the second function, $H(V, m, A)$, is to rate how well a set of coupons satisfy all of the reward totals (V). A penalising term is added for each v_i (each reward total) where there is no solution for $\Phi(v_i, m, A)$.

$$H(V,m,A) = \left(\sum_{i=1}^{n} h(v_i,m,A) \right) + 1$$
$$h(A,v_i) = \begin{cases} 1, & |\pi(v_i,m,A)| > 0 \\ \beta, & \text{otherwise} \end{cases} \quad (6)$$

The two parameters $\alpha, \beta > 1$ are penalising terms.

The fitness function for the GA is defined according to equation 7, where the smaller the function is, the better the solution.

$$F(V,m,A) = G(A)H(V,m,A) \quad (7)$$

3.3 Parameters and Operators

The application dependant and GA parameters, along with the GA operators used, are detailed in Table 1.

3.4 Post Processing

Once a suitable set A has been found given a particular instance of the problem, a post processing step can be performed to select the most suitable coupons for each reward total. This stage will make use of the function $\Pi(v_i, m, A)$, where it is used to generate all of the possible coupon combinations for a given total, and then some selection criteria can be used to choose the most appropriate set.

4 Evaluation

In this section a GA using three crossover operators, hill-climbing and simulated annealing are tested against a real instance of the coupon allocation problem and compared with a solution which was generated by hand.

Parameter	Value
Representation	Binary
Fitness	As equation 7
Generations	Determined by the number of fitness evaluations (~50)
Population	1000
Crossover	See section 4.1
Mutation	Binary, probability = $1/n$
Survival	Ranked (Baker 1985), (a minimisation problem) with Elitism = 25 (DeJong 1975)
Fitness Evaluations	50, 000; to ensure convergence, determined through experimentation
A	10.0, sufficiently larger than 1.0
B	100.0, a large number > α

Table 1: ICARUS Parameters

4.1 Crossover Methods

The GA was implemented using three standard binary crossover operators. The first was uniform crossover (Syswerda 1989), the second was one point crossover (Holland 1975) and the third was And/Or crossover. The latter is defined below and was added as this operator can create children with a small number of '1's. This was thought to be desirable since the solution will be as small a subset of A as is feasible, which therefore requires chromosomes with a small number of '1's.

And/Or crossover defines two children from two binary parent chromosomes. The process is similar to Uniform crossover, but rather than choosing to set a child's bit based on selecting the relevant bit from one of the parents (usually chosen at random), the first child's bits are determined by logically ANDing both the parent bits together, and the second child's are through using the logical OR operator.

4.2 Simulated Annealing

Simulated Annealing (SA) (Kirkpatrick 1983) is an attempt to improve upon the hill-climbing algorithm (HC) (Russell 1995) by building in the ability to escape local minimas, a problem associated with the HC algorithm. SA is exactly the same as HC but when the new solution is worse than the old one, it is not discarded, but accepted with a probability according to equation 8.

The representation is a single binary chromosome (denoted Z), similar to an individual within the GA. Algorithm 1 shows the pseudo code for the SA method, where *random*(a, b) is a function which returns a uniformly distributed random number between a and b inclusive. SA requires a number of parameters which are detailed in table 2.

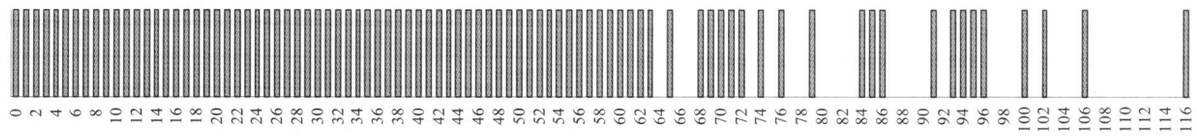


Figure 1: Range of Customer Reward Totals for the Real World Data, the Presence of a Column Indicates if an Amount Needs to be Sent Out or Not

$$P(\text{accept new}) = e^{-\left(\frac{\Delta Z}{\theta_t}\right)} \qquad (8)$$
$$\Delta Z = |F(Z') - F(Z)|$$

Parameter	Value
Representation	Binary
Fitness	As equation 7
Starting Temperature, θ_1	The search space is random walked for 1% of the total iterations, and the average of ΔZ is used (Swift 2004)
Cooling rate, c	Computed from θ_1 and the number of iterations
Iterations	As table 1, minus 1% (see above)

Table 2: SA Parameters

4.3 Hill-Climbing

The algorithm used for Hill-Climbing uses algorithm 1, but with the initial temperature set to zero. This ensures that a worse solution will always be rejected (line 6).

Algorithm 1. Simulated Annealing

```
1.  Generate a random binary string Z of
    length k
2.  Initialise starting temperature, θ1
3.  For t = 1 to Iterations
4.     Randomly mutate a single random bit of
       Z creating Z' and rescore
5.     Set probability p according to equation
       8 with current value for θt
6.     If the new score > old score And
       random(0,1) > p Then
7.        Undo the change
8.     End If
9.     θt+1 = cθt
10. End For
```

4.4 Real World Dataset

Figure 1 shows the values of V for a real world problem. Restating the problem, given the set V of 85 values between 0 and 116 and four coupons (m = 4), find a set of denominations A such that every value in V can be represented as the sum of four elements of A. As can be seen, V ranges from $v_1 = 0$ to $v_n = 116$, where $n = 85$; in this particular case, there were four coupons, hence $m = 4$. Given the size of v_n then by proposition 1 and 2, k (the size of the representations for ICARUS) must be at least 29. Given that a basket of shopping is between £25 and £35, it was decided that k will be 35, i.e. allowing coupons to range between £1 and £35. This will allow there to be potentially several solutions for larger values of v_i, i.e. setting k to 29 means that there is only one solution for v_n=116. Arguably one could make k very large and let ICARUS find the most appropriate set of coupons; however the search space is of order 2^k, hence the smaller the value for k, the quicker the execution of ICARUS.

It is apparent that as the coupon total increases (v_i), the series contains more and more gaps. A solution was found by hand but took a consultant 2-3 days of effort. The solution used is shown in figure 3, this solution contains all of the integers in the range 0-10 and the numbers divisible by 5 between 11 and 50 inclusive. The solution contains 18 numbers (not counting zero).

The coupon scheme was sent out to the customers for the retailer in question. Figure 2 shows the percentage rate of coupon redemption over the first month of a certain period during which the coupons were valid. Based on figure 2, a list of further requirements was drawn up:

1. Despite the differences between the segments, the £1 coupon has the lowest redemption rate. Solutions without this value would be desirable;
2. Coupons valued between £4 and £25 have the highest redemption rate. This should be taken into account.

Both these requirements can easily be added into the function $\Pi(v_i, m, A)$, for implementation during the post processing stage.

4.5 Experimental Results

This section describes the results of the experiments. Each of the methods was run 25 times since they are all stochastic techniques. Table 3 shows the minimum (Min.), maximum (Max.), mean and standard deviation (St.Dev.) fitness for all of the methods, figures 4 and 5 show convergence graphs, table 4 shows how consistent each of the results is, table 5 shows the mode (most common) result for each of the methods, and finally coverage is discussed. Finally, figure 6 shows the frequency of the number of valid solutions for each reward total as determined by the function $\Pi(v_i, m, A)$.

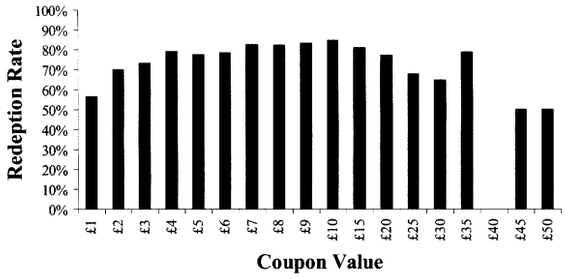


Figure 2: Coupon Redemption Rate for the Manually Prepared Scheme

$A = \{0,1,2,3,4,5,6,7,8,9,10,15,20,25,30,35,40,45,50\}$

Figure 3: The Manual Solution to a Real World Instance of the Coupon Allocation Problem

Method	Min.	Max.	Mean	St.Dev.
UNIFORM	3010	3096	3037.5	40.9
ONEPOINT	3010	3096	3034.1	39.4
ANDOR	3010	3096	3071.9	39.4
SA	3010	3096	3068.5	40.9
HC	4644	7052	6123.2	666.2
Manual	7826	7826	7826.0	0.0

Table 3: Summary Statistics for Method Results

Within table 3, *Manual* refers to the results from the manual (consultant) solution for the problem, UNIFORM refers to the results from the GA using uniform crossover, ONEPOINT for one point crossover and ANDOR from using And/Or crossover.

From table 3 it can be clearly seen that all of the search based methods get a much improved fitness than the manual method. Of course this assumes that the fitness in equation 7 is producing good results. All of the GA based methods and SA produce the same minimum and maximum, which are improvements on those for HC. Given that a low average fitness and a low standard deviation is desirable, the results from table 3 show that ONEPOINT performs the best, followed by UNIFORM, SA, ANDOR and then HC. However the results for ONEPOINT and UNIFORM are very close together, since ONEPOINT has a slightly better mean than UNIFORM and the standard deviations are almost identical. All of the search based methods (except HC) have very similar and low standard deviations, showing that the methods seem to be producing the same consistency of results.

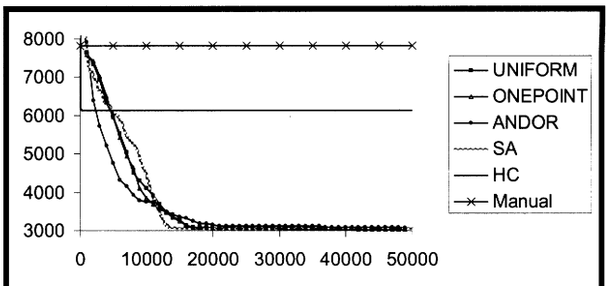


Figure 4: Convergence Graph for All Methods

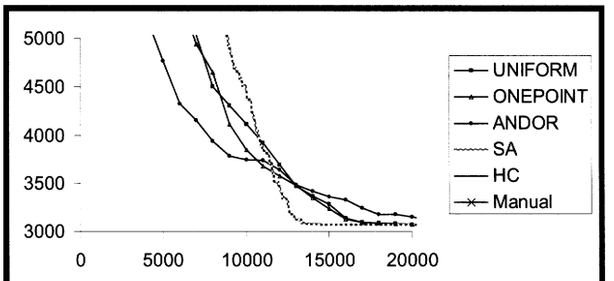


Figure 5: Magnified Convergence Graph for All Methods

Figures 4 and 5 show the method convergence graphs for all of the methods employed in this paper. Figure 5 is the same as figure 4, however the bottom left hand corner of the graph has been magnified. The plots are created from averaging the 25 runs for each method. It can be clearly seen that the HC method converges the quickest at first, before becoming stuck in a local optima, the next fastest is SA, followed by UNIFORM, then ONEPOINT and finally ANDOR. However it must be noted that ONEPOINT only performs marginally better than UNIFORM. Note that the manual results are constant within figures 4 and 5, and have been added for completeness.

Given two sets of results for the coupon allocation problem, say A and B, then the similarity between these two sets will be defined to be:

$$S(A,B) = \frac{|A \cap B|}{\max(|A|,|B|)} \tag{9}$$

Given a set of results $R = \{A_1,\ldots A_p\}$, then the overall similarity of the set will be defined to be:

$$S(R) = \frac{2}{p(p-1)} \sum_{i=1}^{p-1} \sum_{j=i+1}^{p} S(A_i, A_j) \tag{10}$$

Equation 9 is a ratio of the size of the intersection of two sets of results and the size of the largest results, which will range between 0 and 1. Equation 10 is the average of all the possible pairings of similarity.

Figure 6: Number of Valid Solutions for Each Reward Total

Method	Equation 10
UNIFORM	0.81
ONEPOINT	0.82
ANDOR	0.74
SA	0.76
HC	0.40
Manual	1.00

Table 4: Method Consistency

As can be seen from table 4, ONEPOINT produces the most consistent results, followed by UNIFORM, SA, ANDOR and then HC. It is interesting to note how poor the HC results are. Note the correlation between the standard deviation figures in Table 3 and the consistency results as would be expected.

Method	Coupon Denominations (Number)
UNIFORM	1 2 5 10 20 30 32 (7)
ONEPOINT	1 2 5 10 20 30 32 (7)
ANDOR	1 2 5 10 15 30 33 35 (8)
SA	1 2 5 10 20 30 32 (7)
HC	1 2 3 4 5 12 25 28 35 (9)
Manual	1-10 15 20 25 30 35 40 45 50 (18)

Table 5: Mode (Most Common) Method Result

Table 5 displays the modal (most common) result from all of the methods, note that 1-10 represents all of the integers between 1 and 10 inclusive. Also note that all of the results for the HC were different from each other, so the first set of results is displayed.

UNIFORM, ONEPOINT and SA produced the same set of coupon denominations, whilst ANDOR and HC produced a slightly larger result. These results are almost a subset of the manual set, where several values have been replaced for a coupon which is not divisible by 5.

This could this be explained by observing that the manual method appears to have placed more emphasis on the divisible-by-5 requirement.

It is worth noting that all of the results cover all of the 85 values specified by the real world dataset, i.e. the resultant set A for each method does not fail for any value v_i. Given the results presented in this section, it is clear that the methods along with the fitness function in equation 7 produce desirable and promising coupon allocations.

Finally, figure 6 shows the frequency histogram of the results of the function $\Pi(v_i, m, A)$, i.e. how many valid combinations of each reward totals can be generated from the coupon set $A = \{1\ 2\ 5\ 10\ 20\ 30\ 32\}$. This set was taken from the best results from table 5. Table 6 displays some summary statistics regarding this graph.

From figure 6 and table 6 it can be seen that there are only a small range of options available for each reward total. This is understandable, given that ICARUS is trying to create as small a set A as possible. It seems that the values in the middle of the reward total range have more combinations available.

Statistic	Value
Min.	1.0
Max.	5.0
Mean	2.0
St.Dev.	1.0

Table 6: Summary Statistics for Figure 6

The selection of the best combination has been omitted. Since there are relatively few for each reward total, the consultant could choose manually which one they thought was the best in a very short period of time.

5 Concluding Remarks

Within this paper a method for solving the reward scheme coupon allocation problem has been presented, named ICARUS. This method is achieved through the use of a binary Genetic Algorithm, where the fitness and representation have been tailored to suit particular commercial requirements. The results clearly show that

ICARUS using one point crossover is better than a number of other crossover operator alternatives, and two straw-person heuristic search methods, along with the manually prepared solution.

The ICARUS method runs in a few minutes, and can solve the coupon allocation problem for many different retail scenarios (for various sets V and sizes m).

5.1 Future Work

Future work falls into two distinct areas, algorithm improvement and feedback.

The ICARUS algorithm is really a prototype, demonstrating the *proof of concept*, that the problem can be solved using a GA. A more in depth and thorough analysis is now needed, looking at improved fitness functions, scalability, intelligent operators etc. In particular the use of a Multi-Objective GA (Deb 2001) could be explored so that the functions H and G are separated out from the fitness function F, and perhaps the post processing stage (selecting the most aesthetic solution from $\Pi(v_i, m, A)$) could also be integrated. In terms of the business application, the next step is to take customer segmentation into account and generate differing coupon allocations based on each particular segment.

It is intended that ICARUS be used to determine the next run for the retailer which provided the real world data used within this paper. Once this has been implemented, the coupon redemption rate can be compared with that of figure 2, to see if any improvement has been made, thus verifying if ICARUS truly outperforms the manual system.

Acknowledgments

The authors would like to thank the unnamed retailer who provided the dataset used within this paper, Janet McFall for her helpful comments and the reviewers for their constructive advice.

Bibliography

Agrawal R. (1994) "Fast algorithms for mining association rules", Proceedings of the 20th VLDB Conference, Santiago, Chile, pp. 487-499

Baker J.E. (1985) "Adaptive Selection Methods for Genetic Algorithms", Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, pp. 101-111

Deb K. (2001) "Multi-Objective Optimization Using Evolutionary Algorithms", John Wiley and Sons Ltd.

DeJong K.A. (1975) "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems", Doctoral Thesis – University of Michigan, Dissertational Abstracts International, 36, 10, 5140B

Garey M. and Johnson D. (1979) "Computers and Intractability – A Guide to the Theory of NP-Completeness", W. H. Freeman, San Francisco

Goldberg D. and Lingle R. (1985) "Alleles, loci and the travelling salesman problem", Proceedings of the first International Conference on Genetic Algorithms, Grefenstette J.J. (ed.), Lawrence Erlbaum Associates, Hillsdale

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

Humby C., Hunt T. and Phillips T. (2003) "Scoring Points", Kogan Page

Kirkpatrick S. and Gelatt Jr. C.D. and Vecchi M.P. (1983) "Optimization by Simulated Annealing", Science 220, No. 4598, 671-680

Michalewicz Z. and Fogel D.B. (1998) "How To Solve It: Modern Heuristics", Springer

Russell S. and Norvig P. (1995), "Artificial Intelligence, A Modern Approach", Prentice Hall, pp.111-112

Sacerdoti E. (1977) "A structure for plans and behaviour". American Elsevier, New York

Schmitt B. (2003), "The Customer Experience Management: A Revolutionary Approach to Connecting With Your Customers", John Wiley and Sons Ltd.

Swift S., Tucker A., Vinciotti V., Martin N., Orengo C., Liu X. and Kellam P. (2004) "Consensus clustering and functional interpretation of gene-expression data", Genome Biology, 5(11:R94)

Syswerda G. (1989) "Uniform Crossover in Genetic Algorithms", Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 10-19