Yugoslav Journal of Operations Research
9 (1999), Number 2, 223-233

A GENETIC ALGORITHM APPROACH TO THE BUYERS' WELFARE PROBLEM OF PRODUCT LINE DESIGN: A COMPARATIVE COMPUTATIONAL STUDY

Georgia ALEXOUDA, Konstantinos PAPARRIZOS

Department of Applied Informatics,
University of Macedonia,
156 Egnatia Str, POB 1591,
Thessaloniki 540 06, Greece

Abstract: In this paper we present a Genetic Algorithm based heuristic for solving the Product Line Design Problem using the Buyers' Welfare Criterion. The new approach is compared with a recently developed Beam Search method on randomly generated problems. Our method seems to be substantially better in terms of CPU time. Also, the solutions found by our method are better than those found by the Beam Search method in comparable times.

Keywords: Genetic algorithms, product line design, buyers' welfare problem, marketing.

1. INTRODUCTION

Optimal product design is well recognized as one of the most crucial decisions for a firm. Many practitioners and academicians deal with optimal product design, because the rates of failure of new products and their associated losses are very high [2]. It is well known that the product design problem is NP-Hard [10]. For this reason many researchers have proposed heuristic procedures to solve the problem (see [6] for a review).

A very important problem in marketing is the optimal product line design. Recently, many researchers have proposed preference-based procedures for this problem. There are two different approaches to attack the problem. The first approach considers a finite set of candidate items (reference set) from which a product line is selected [5, 7, 12]. Preference evaluations for each item are used to select a product line maximizing the buyers' welfare function (the "buyers' problem") or the sellers' return function (the "sellers' problem"). If the number of attributes and attribute levels is large

and most attribute level combinations define feasible products, it can be computationally infeasible to enumerate the utilities of candidate items. For this class of problems, it is preferable to use the second approach, which constructs product lines directly from parts-worth data. Kohli and Sukumar [11] and Nair, Thakur and Wen [14] have developed the most recent heuristic methods using the second approach. Our method constructs product lines directly from parts-worth data.

There are two basic approaches for modelling the single product or product line design problem: the multidimensional scaling approach (MDS) and the conjoint approach. Conjoint analysis is a very popular method for real applications [15]. Green and Krieger [7], McBride and Zufryden [12], Dobson and Kalish [5], Kohli and Sukumar [11] and Nair, Thakur and Wen [14] have used the conjoint method in product line design problems. Our method uses conjoint analysis for modelling the product line design problem.

2. PROBLEM DEFINITION

In the Buyers' Welfare Problem we consider that buyers choose the product that gives them maximum utility and the product line is designed so that the total utility of all the buyers of the product line is maximized. The Buyers' Welfare Criterion can be used by nonprofit organizations. However, buyers' welfare is very important for the survival of every firm.

The typical mathematical formulation of the product line design problem [10] considers the following sets: Let $\Omega = \{1,2,\ldots,K\}$ denote the set of K attributes. Let $\Phi_k = \{1,2,\ldots,J_k\}$ denote the set of J_k levels of attribute $k \in \Omega$. Let $\Psi = \{1,2,\ldots,PN\}$ denote the set of PN items to be selected, where the multi-attribute description of each item is to be determined by solving the buyers' welfare problem. Let $\Theta = \{1,2,\ldots,I\}$ denote the set of I buyers. Let w_{ijk} denote the part worth of level $j \in \Phi_k$ of attribute $k \in \Omega$ for consumer $i \in \Theta$. The parts-worth can be estimated using conjoint analysis.

Each buyer chooses the product that gives him maximum utility. The Buyers' Welfare problem is to select a product line that maximizes the total utility of all the buyers of the product line. Let x_{ijkm} be a variable which indicates whether level $j \in \Phi_k$ of attribute $k \in \Omega$ is assigned to product $m \in \Psi$ and consumer $i \in \Theta$. In particular it is set that:

$$x_{ijkm} = \begin{cases} 1, \text{if level } j \in \Phi_k \text{ of attribute } k \in \Omega \text{ is assigned to product } m \in \Psi \text{ and consumer } i \in \Theta \\ 0, \text{otherwise} \end{cases}$$

The Buyers' Welfare problem can be formulated as the following 0-1 integer program:

$$\max \sum_{i\in\Theta} \sum_{m\in\Psi} \sum_{k\in\Omega} \sum_{j\in\Phi_k} w_{ijk} x_{ijkm} \tag{1}$$

$$\text{s.t.} \sum_{j\in\Phi_k} \sum_{m\in\Psi} x_{ijkm} = 1, \quad i \in \Theta, k \in \Omega \tag{2}$$

$$\sum_{j\in\Phi_k} x_{ijkm} - \sum_{j\in\Phi_{k'}} x_{ijk'm} = 0, \quad k' > k, k, k' \in \Omega, i \in \Theta, m \in \Psi \tag{3}$$

$$x_{ijkm} + x_{i'j'km} \le 1, \quad i' > i, j' > j, i, i' \in \Theta, j, j' \in \Phi_k, k \in \Omega, m \in \Psi \tag{4}$$

$$x_{ijkm} \in \{0,1\}, \quad i \in \Theta, j \in \Phi_k, k \in \Omega, m \in \Psi \tag{5}$$

Constraints (2)-(4) force each buyer to be assigned to one of the items of the product line. The objective function (1) selects the items of the product line to maximize the total buyers' welfare and ensures that each buyer is assigned to an item from which he obtains the maximum utility.

3. EXISTING HEURISTIC METHODS FOR THE PRODUCT LINE DESIGN PROBLEM

Because the Buyers' Welfare Problem of a Product Line Design is NP-Hard, many researchers have proposed heuristic methods. Kohli and Sukumar [11] proposed a heuristic solution procedure that mimics a dynamic programming method using attributes as stages and attribute levels as states. Therefore, this solution procedure is called a dynamic-programming heuristic.

Nair, Thakur and Wen [14] proposed a Beam Search (BS) heuristic for the solution of the product line design problem. Their computational study showed that the BS method finds better solutions and takes less computation time than the dynamic programming heuristic. BS methods were developed in the 1970s for artificial intelligence search problems. BS is a breadth-first search process with no backtracking. At any level of BS, the b most promising nodes are explored further in the search tree, where b is called the beam width. Nair, Thakur and Wen [14] suggest a way to compute a good value of b. In our computational study we have followed this suggestion.

We must note that these heuristic methods work with 'partial' product profiles in the sense that they are described by the levels of only some attributes, which are considered in a concrete step. That's the reason why the first method can't use the solution of the second as a starting point and vice versa. This restriction doesn't exist for the Genetic Algorithms (GAs) and in our computational study we make use of this advantage. We initialize the GA in two different ways. In the first way we initialize the GA with a random first population. In the second way we include the solutions of the BS heuristic in the first population.

4. GENETIC ALGORITHMS

The GA is an approach that arose with computer science. They were first invented by John Holland in the 1960s and were developed by Holland and his students and colleagues at the University of Michigan in the 1960s and 1970s. GAs are intelligent probabilistic search techniques that mimic some of the processes of natural evolution and selection [13, 1, 9, 4]. All GAs consist of the following main components:

1. Chromosomal representation: GAs work with an encoding of the variables as strings of genes, which can take on some values from a specific finite range or alphabet. This string of genes, which represents a solution, is called a chromosome. It works with a population of solutions rather than a single solution.

2. Initial population: The initial population can be created randomly or using problem-specific information.

3. Fitness evaluation: Each solution in the population is evaluated according to some fitness measure.

4. Reproduction: This operator selects chromosomes from the current generation based on their fitness values.

5. Crossover: This operator creates new chromosomes by mating current chromosomes. There are different types of crossover, like one-point crossover, two-point crossover and uniform crossover.

6. Mutation: This operator is the occasional random alteration of the value at a string position.

A generic form of GAs can be stated as follows:

- Generate an initial population
- Repeat

  Fitness evaluation of current population
  Reproduction
  Crossover
  Mutation
- Until the stopping condition is satisfied.

Correctly applied GAs can provide good heuristic solution approaches for many integer programming problems [3, 8]. In recent years many researchers have dealt with solving marketing optimization problems using genetic algorithms (see [9] for a review). Balakrishnan and Jacob [1] have developed a GA for the single product design problem. Our method is an extension of this approach to the product line design problem.

5. A GENETIC ALGORITHM FOR PRODUCT LINE DESIGN

We present a GA for the product line design problem. We initialize the GA in two different ways. In the first way we initialize the GA with a random population. This algorithm is called GA1. In the second way we include the solutions of the BS heuristic in the first population of the GA. This algorithm is called GA2. We have used the Buyers' Welfare Criterion for the fitness evaluation of each population. In each iteration 40% of the new population is produced using the reproduction operator, another 40% using the uniform crossover operator and 20% using the mutation operator. If the best candidate solution does not improve in the last 10 iterations, the GA terminates.

The algorithm can formally be described as follows. In this description L is the set of candidate product lines and $M = |L|$ is the population size. The population is maintained in matrix POP_{M*PN*K}. The elements POP_{lmk}, where $l \in L$, $m \in \Psi$ and $k \in \Omega$, denote the selected level of each attribute. That is, if level $j \in \Phi_k$ of attribute $k \in \Omega$ is assigned to product $m \in \Psi$ of product line $l \in L$, then $POP_{lmk} = j$, otherwise 0. Let w_{ijk} denote the part worth of level $j \in \Phi_k$ of attribute $k \in \Omega$ for consumer $i \in \Theta$. The utilities of the PN different products of each product line which are obtained by each buyer are maintained in the matrix $PRODUTIL_{M*I*PN}$. The buyers' welfare for each product line is stored in the matrix $WELFARE_M$.

STEP 1: (***Initialization***) Generate an initial population of candidate product lines. Store the population in matrix POP_{M*PN*K}.

STEP 2: (***Fitness Evaluation***). Compute matrices $PRODUTIL_{M*I*PN}$ and $WELFARE_M$, as follows:

$$\text{set } PRODUTIL_{lim} = \sum_{k \in \Omega} w_{i(POP_{lmk})k}, \quad l \in L, i \in \Theta, m \in \Psi$$

and

$$\text{set } WELFARE_l = \sum_{i \in \Theta} \max_{m \in \Psi} PRODUTIL_{lim}, \quad l \in L.$$

STEP 3: (***Reproduction***) Choose the $(2/5)M$ best product lines.

STEP 4: (***Crossover***) Create randomly $M/5$ pairs of product lines chosen among the ones created in Step 3. Perform the uniform crossover operator on these pairs to generate $(2/5)M$ new candidate product lines.

STEP 5: (***Mutation***) Pick randomly $M/5$ product lines from the set of the $(4/5)M$ product lines which were created in Steps 3 and 4 and perform the mutation operator on them.

STEP 6: (***Stopping Rule***) If the best candidate solution does not improve in the last 10 iterations STOP. Otherwise, go to STEP 2.

6. COMPUTATIONAL RESULTS

In order to see whether the GA has any computational advantage, we compared it with the BS heuristic. We randomly generated 360 different problems. The problems are divided into two groups. In the first group the number of consumers is 100, and in the second 150. We considered 18 different problem sizes for each group. For each different problem size 10 problems were randomly generated. In order to generate 18 different problem sizes we gave different values to the number of attributes (K=5,6,7), the number of attribute levels (J=4,5,6) and the number of products (PN=2,3). The parts-worth were generated randomly from a uniform distribution and normalized within respondent. The normalized parts-worth are assumed mutually comparable for the buyers' problem.

In the implementation of the GAs the population size was set equal to 150. The algorithms terminate when in 10 consecutive iterations the best candidate solution doesn't improve. We must indicate that in some cases the results of the GAs can be improved, if we increase the population size or if we set a more strict stopping condition. However, according to our experience, the population size and the stopping condition we used in our implementation, are in general good for the problem sizes we studied.

For each class of 10 problems we computed the average CPU time of the algorithms GA1, BS and GA2. The CPU time needed to find the initial solution by the BS method is not included in the CPU time of GA2. For the algorithms GA1 and GA2 we computed the average number of iterations. Furthermore, for the cases where algorithm GA1 finds a better or an equivalent solution compared to that found by the BS method, we computed the average CPU time and the average number of iterations needed by algorithm GA1 to reach or to better the solution found by the BS method. These computational results are presented in Tables 1a and 1b. In all cases algorithm GA1 requires less CPU time than the BS method. This is more obvious for big problems. In almost all cases GA2 needs less iterations and CPU time than GA1. There is only one exception for PN=3, K=5, J=4 and I=100. For the cases where algorithm GA1 finds a better or equivalent solution compared to that found by the BS method, algorithm GA1 needs in the most cases about 50% of the number of iterations and CPU time to reach or to better the solution found by the BS method.

It is very interesting to compare the solutions of the heuristic methods. We compared the solutions of GA1, GA2 and BS. The comparative results are presented in Tables 2a and 2b. In all 36 problem sizes GA1 more often finds a better solution than BS. In 81.39% of the cases we examined, GA1 finds a better solution than BS, while in only 12.22% BS finds a better solution than GA1. In 62.22% of the problems GA2 improves the solution of BS. Also we compared the solutions of GA1 and GA2. In 43.89% of the problems GA1 finds a better solution than GA2, while in 31.94% of the problems GA2 finds a better solution than GA1. In each iteration of the GA2 the b product lines generated by the BS heuristic have a high possibility of being reproduced. A consequence of this fact is insufficient changes to the population. That is perhaps a

possible explanation for the superiority of GA1 over GA2. These results are presented separately for $I=100$ and $I=150$ at the end of Tables 2a and 2b.

The implementation was done using the programming language Borland C++ 4.5. The computational tests were conducted on a PC with a Pentium Processor (16 MB RAM, 150 MHz, using Windows 95).

Table 1a: Computational results of GA1, BS and GA2 for $I=100$

			GA1				BS	GA2	
PN	*K*	*J*	CPU time (secs)	CPU time* (secs)	Iterations	Iterations*	CPU time (secs)	CPU time (secs)	Iterations
2	5	4	12.47	5.44	21.20	9.00	21.86	8.12	14.20
2	5	5	14.51	6.29	23.60	10.22	22.63	8.94	15.20
2	5	6	15.96	6.28	26.70	10.33	24.04	10.38	17.30
2	6	4	15.19	5.99	24.60	9.63	43.89	8.87	14.60
2	6	5	17.59	7.79	28.30	12.57	43.71	11.10	17.60
2	6	6	21.91	11.89	34.00	18.13	44.00	9.53	15.10
2	7	4	18.59	8.79	29.50	13.75	45.58	11.86	18.90
2	7	5	19.08	8.64	28.40	12.78	41.83	13.25	20.80
2	7	6	21.19	8.59	32.30	12.78	46.07	12.17	18.60
3	5	4	16.79	8.08	25.30	12.00	41.78	17.19	25.40
3	5	5	25.65	14.01	36.50	19.80	45.55	15.46	21.60
3	5	6	23.04	11.05	33.50	16.00	45.44	21.49	31.30
3	6	4	24.66	12.28	34.50	17.00	84.72	13.17	18.40
3	6	5	29.32	13.64	39.40	18.29	85.37	20.01	26.50
3	6	6	31.54	16.78	41.40	21.86	85.54	20.46	27.40
3	7	4	28.40	13.16	37.00	16.90	84.32	13.66	17.50
3	7	5	36.97	19.63	46.00	24.25	85.95	20.46	25.90
3	7	6	35.93	15.75	46.20	20.10	88.57	15.49	19.70

*needed by GA1 to reach or to better the solution of the Beam Search heuristic only for the cases where GA1 finds a better or equivalent solution compared to those found by the BS method.

Table 1b: Computational results of GA1, BS and GA2 for $I=150$

			GA1				BS	GA2	
PN	*K*	*J*	CPU time (secs)	CPU time* (secs)	Iterations	Iterations*	CPU time (secs)	CPU time (secs)	Iterations
2	5	4	15.69	7.53	21.90	10.38	23.39	10.46	14.70
2	5	5	18.06	8.68	23.70	11.13	24.75	11.41	15.60
2	5	6	19.25	8.45	26.10	11.40	25.74	11.29	15.50
2	6	4	20.21	10.35	26.60	13.40	46.69	13.34	17.80
2	6	5	23.56	9.75	26.70	11.00	49.17	13.49	15.80
2	6	6	25.55	14.23	32.00	17.89	49.26	12.83	16.20
2	7	4	23.11	11.31	29.20	14.13	47.94	11.43	14.50
2	7	5	27.40	12.57	33.00	15.40	48.35	14.64	17.40
2	7	6	26.21	12.49	31.90	15.11	50.98	17.91	22.20
3	5	4	29.77	15.66	33.20	17.22	43.24	16.15	18.20
3	5	5	26.77	14.07	30.50	16.00	46.02	20.11	22.70
3	5	6	29.93	14.43	34.00	16.33	48.94	22.71	23.60
3	6	4	32.41	15.63	36.30	16.30	88.40	19.25	21.10
3	6	5	39.73	17.96	40.90	18.30	89.81	26.48	27.70
3	6	6	40.78	23.26	42.90	24.22	90.52	18.37	19.30
3	7	4	35.94	16.07	37.10	16.33	92.09	20.33	20.20
3	7	5	44.68	24.03	43.10	22.89	93.67	19.94	19.30
3	7	6	42.04	20.73	41.30	20.20	96.64	31.86	30.50

*needed by GA1 to reach or to better the solution of the Beam Search heuristic only for the cases where GA1 finds a better or equivalent solution compared to those found by the BS method.

Table 2a: Comparative results of the solutions of GA1, BS and GA2 for $I=100$

PN	*K*	*J*	GA1 better than BS	BS better than GA1	GA2 better than BS	GA1 better than GA2	GA2 better than GA1
2	5	4	5	0	5	1	0
2	5	5	7	1	5	2	3
2	5	6	8	1	6	4	2
2	6	4	8	2	5	5	2
2	6	5	7	3	6	5	4
2	6	6	6	1	4	3	2
2	7	4	8	2	6	3	5
2	7	5	9	1	9	4	1
2	7	6	9	1	6	4	3
3	5	4	10	0	9	5	4
3	5	5	10	0	8	6	3
3	5	6	9	1	8	4	3
3	6	4	10	0	5	6	4
3	6	5	7	2	9	4	4
3	6	6	7	3	6	4	6
3	7	4	10	0	5	6	2
3	7	5	8	2	7	2	5
3	7	6	10	0	6	7	3
%			82.22%	11.11%	63.89%	41.67%	31.11%

Table 2b: Comparative results of the solutions of GA1, BS and GA2 for $I=150$

PN	*K*	*J*	GA1 better than BS	BS better than GA1	GA2 better than BS	GA1 better than GA2	GA2 better than GA1
2	5	4	5	2	5	1	3
2	5	5	7	2	7	3	3
2	5	6	9	0	6	4	1
2	6	4	7	0	5	4	1
2	6	5	7	2	4	3	4
2	6	6	9	1	6	5	2
2	7	4	8	2	4	6	3
2	7	5	5	5	5	4	5
2	7	6	8	1	6	4	3
3	5	4	8	1	7	4	3
3	5	5	6	4	6	3	7
3	5	6	9	1	9	3	6
3	6	4	10	0	7	5	4
3	6	5	10	0	6	7	3
3	6	6	9	1	6	8	2
3	7	4	9	1	8	6	3
3	7	5	9	1	4	7	3
3	7	6	10	0	8	6	3
%			80.56%	13.33%	60.56%	46.11%	32.78%

7. CONCLUSIONS

A Genetic Algorithm (GA) for the Buyers' Welfare Problem of a Product Line Design was presented. In the implementation the GA was initialized in two different ways. In the first way the GA was initialized with a random first population. This algorithm is called GA1. In the second way the solutions of the Beam Search (BS) method were included in the first population of the GA. This algorithm is called GA2. The genetic algorithms GA1, GA2 and the BS method were compared on 36 different classes of problems each one containing 10 randomly generated problems. Algorithm GA1 seems to be substantially better than the BS method in terms of CPU time. Also the solutions found by GA1 are better than those found by the BS method. Furthermore, in many cases algorithm GA2 improves the solution of the BS method. However, in most cases GA1 finds a better solution than GA2.

Acknowledgement. The authors wish to thank the anonymous referee for his suggestions.

REFERENCES

[1] Balakrishnan, P., and Jacob, V., "Genetic algorithms for product design", *Management Science*, 42 (1996) 1105-1117.

[2] Business Week, "Flops: too many new products fail", August 16, 1993, 76-82.

[3] Chu, P., and Beasley, J., "A genetic algorithm for the generalised assignment problem", *Computers and Operations Research*, 24 (1) (1997) 17-23.

[4] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[5] Dobson, G., and Kalish, S., "Positioning and pricing a product line", *Marketing Science*, 7 (1988) 107-125.

[6] Green, P., and Krieger, A., "Recent contribution to optimal product positioning and buyer segmentation", *European Journal of Operational Research*, 41 (1989) 127-141.

[7] Green, P., and Krieger, A., "Models and heuristics for product line selection", *Marketing Science*, 4 (1985) 1-19.

[8] Hadj-Alouane, A., and Bean, J., " A genetic algorithm for the multiple-choice integer program", *Operations Research*,·45 (1) (1997) 92-101.

[9] Hurley, S., Moutinho, L., and Stephens, N., "Solving marketing optimization problems using genetic algorithms", *European Journal of Marketing*, 29 (4) (1995) 39-56.

[10] Kohli, R., and Krishnamurti, R., "Optimal product design using conjoint analysis: Computational complexity and algorithms", *European Journal of Operational Research*, 40 (1989) 186-195.

[11] Kohli, R., and Sukumar, R., "Heuristics for product-line design using conjoint analysis", *Management Science*, 36 (1990) 1464-1478.

[12] McBride, R., and Zufryden, F., "An integer programming approach to the optimal product line selection problem", *Marketing Science*, (1988) 126-140.

[13] Mitchell, M., *An Introduction to Genetic Algorithms*, MA: MIT Press, Cambridge, 1996.

[14] Nair, S., Thakur, L., and Wen, K., "Near optimal solutions for product line design and selection: Beam search heuristics", *Management Science*, 41 (1995) 767-785.

[15] Wittink, D., and Cattin, P., "Commercial use of conjoint analysis: An update", *Journal of Marketing*, 53 (1989) 91-96.