ISTANBUL UNIVERSITY –
JOURNAL OF ELECTRICAL & ELECTRONICS ENGINEERING

YEAR : 2009 (951-958)
VOLUME : 9
NUMBER : 2

A MODIFIED GENETIC ALGORITHM FOR THE GENERALIZED ASSIGNMENT PROBLEMS

Mustafa B. AKBULUT[1], A. Egemen YILMAZ[2]

[1] : Ph. D. Student in Computer Engineering, University of Connecticut, mustafa@engr.uconn.edu
[2] : Ph. D. in Electrical and Electronics Engineering, HAVELSAN A.Ş., asimegemenyilmaz@yahoo.com

ABSTRACT

In this paper, a specialized Genetic Algorithm is proposed and applied for the solution of the Generalized Assignment Problem. Special crossover and the mutation operators called Common Element Crossover (CEX) and In-Pool Mutation (IPM) respectively has been defined by focusing on the special needs and nature of the Generalized Assignment Problem. The performance of the proposed method has been investigated in details via some test cases constructed for the multi-target tracking problems.

***Keywords**: Generalized Assignment Problem, Multi-Target Tracking, Genetic Algorithm, Data Fusion, Operational Research.*

1. INTRODUCTION

Assignment Problems appear in many practical applications that need to find a cost-effective way in the abundance of various solutions. Among such disciplines, operational research (for manpower/personnel scheduling) and data fusion (single/multi-sensor multi-target tracking) can be accounted.

One simple example to clarify this kind of problems would be as follows: A factory owner has n different tasks in his assembly line and a total of n workers to do them; however, the duration that each task takes for each worker varies. Therefore, the owner has to assign each worker to a different task and try to get the total job done in minimum time. An $n \times n$ assignment problem can be represented by means of an $n \times n$ matrix, where an example of a task-worker cost matrix for n=5 is given in Table 1. In this matrix, a number at an entry indicates the cost (e.g. the duration) of the assignment of that specific task-worker pair.

Table 1: An Example Task-Worker Cost Matrix for Assignment Problems.

	Workers				
tasks	3	8	1	5	2
	5	2	6	7	4
	1	3	6	8	3
	2	7	2	6	9
	7	5	8	2	7

Assignment Problems find application in various areas like personnel scheduling, single/multi-sensor multi-target tracking. On the other hand, in practical problems there are additional constraints such that:

i. The number of tasks is not equal to the number of workers; hence the problems is represented by an $n \times m$ matrix where $n \neq m$,

Received Date: 15.12.2008
Accepted Date: 05.11.2009

ii. There might be tasks which are not assigned to any worker,
iii. There might be workers which are not assigned to any task,
iv. There might be additional rules that for some worker-task pair, the assignment is unallowable.

With these additional constraints, the problem is referred to as the Generalized Assignment Problem.

For the multi-target tracking problems, "observations and tracks" are used instead of "tasks and workers" respectively. Observations can be simply considered as the set of measurements from a sensor in its last scan. Tracks can be considered as the trajectories of the targets in the environment. The problem is assignment of the observations to the tracks so that the total cost (distance) of assigned observation-track pairs is minimized. Certainly, the constraints (i-iv) are applicable to this problem:

- There is no guarantee that the number of observations in the last scan is equal to the number of tracks in the system (constraint i).

- There might be observations, which are not assigned to any current track (constraint ii); this phenomenon occurs in cases of false alarm or initiation of a new track.

- There might be tracks, which are not assigned to any observation in that scan (constraint iii); this phenomenon occurs in cases of missed detection.

- The assignment operation in multi-target tracking applications is performed after the "gating" procedure, in which the distances of all observations to all tracks are calculated; and compared to a predefined threshold. An observation-track pair is set to be as "not-assignable" if their distance is above the threshold. This leads to applicability of constraint iv in multi-target tracking applications. Certainly, the distance used in the gating is not necessarily limited to positional Euclidean distance (i.e. kinematic, parametric, statistical properties of the observations and the tracks might also be used in the distance definition; which is out of the scope of this work).

On the other hand, most algorithms proposed for the Assignment Problem have the termination condition of having all the tasks (observations) assigned to some workers (tracks). Naturally this fact causes a problem while applying these algorithms in the Generalized Assignment Problem due to constraint ii.

To overcome this problem, the task-worker cost (observation-track distance) matrix is augmented by means of the "tentative tracks" which in fact do not exist in the system. This yields an observation to be assigned to a tentative track if no suitable assignment can be made for it.

The matrix in Table 1 is reorganized and applied to a multi-target tracking problem (gating applied with a threshold of $d_t = 5$; and augmented) as seen in Table 2.

Table 2: Generalized Assignment Problem with Augmented Matrix and a Threshold of $d_t = 5$.

	tracks					tentative tracks				
observations	3	X	1	5	2	d_t	X	X	X	X
	5	2	X	X	4	X	d_t	X	X	X
	1	3	X	X	3	X	X	d_t	X	X
	2	X	2	X	X	X	X	X	d_t	X
	X	5	X	2	X	X	X	X	X	d_t

The (Generalized) Assignment Problem can be formulated as a linear programming problem (a 0-1 programming problem), which can be solved using different methods like the Enumeration Method, Simplex Method, Branch-Bound Method and Hungarian Method. However, since an Assignment Problem is highly degenerate, it might be difficult to solve it by the Simplex Method or the Branch-Bound Method. In fact, a very convenient and efficient iterative procedure for solving an Assignment Problem is the Hungarian Method. But for large Assignment Problems, this method will not be computationally effective.

2. THE TRAVELING SALESMAN PROBLEM AND SIMILARITIES TO THE GENERALIZED ASSIGNMENT PROBLEM

A solution for the Generalized Assignment Problem would be an array of length n which has distinct elements from 1 to n. The index of the array would represent the task while the number in that index would represent the worker that is assigned to that task. Thus any solution would be a permutation of $1 \ldots n$. This will be represented as an individual as explained in the following paragraphs.

The nature of the Generalized Assignment Problem reminds the renowned Traveling Salesman Problem (TSP). In the same way with TSP, the solutions to Assignment problems consist of a permutation of a certain set of numbers. Assignment Problems, however, enjoy some differences:

- For instance, solutions to Assignment Problems are not order-controlled, but rather position-controlled. On the other hand, in a path representation of a TSP, order of the cities visited is important but the positions of them in the solution do not matter (e.g. (1 3 4 2 5) = (2 5 1 3 4)).
- TSP also suffers from the danger of sub-tours, which can avoid a complete loop; however Assignment Problems do not have the same problem since they do not need to be connected in any way.

In solving TSP, several methods have been utilized so far. In this research, the initial idea is the modification of the existing solutions of TSP and application of them to the Assignment Problems.

3. GENETIC ALGORITHMS AND APPLICATIONS TO THE GENERALIZED ASSIGNMENT PROBLEM

Genetic Algorithm (GA) is a class of adaptive heuristic search and optimization algorithm based on the "survival of the best" principle of the natural selection. It is an iterative optimization procedure and it maintains a population of probable solutions within a search space (which is usually discrete) over many simulated generations. As will be detailed in the following paragraphs, the population members are string entities of artificial chromosomes. In natural terminology, each chromosome (represented by a binary string) is composed of genes. In each iteration (or generation), three basic genetic operations "selection, crossover and mutation" are performed.

The basic concepts of GA were primarily developed by [1]. After this work, numerous researchers have contributed to the development of this field. Detailed reviews on the development of the subject can be obtained in readings such as [2-7] and others.

In the last decade, several researchers developed different methodologies for solving the generalized assignment problem. Among them, the works [8-12] can be accounted. Also some special cases of the generalized assignment problem such as manpower and employee-scheduling etc. problems have been considered in [13-17].

Recently, Aickelin and Dowsland solved a nurse-scheduling problem by genetic algorithm using an indirect coding based on permutations of the nurses, and a heuristic decoder that builds schedules from these permutations [18]. Again, Harper *et. al.* developed a project assignment problem with the help of genetic algorithm [19]. Majumdar and Bhunia developed the so called "Elitist Genetic Algorithm" method which solves the Assignment Problem [20]. Among the above-mentioned papers, [9,12,14,15,18-20] used genetic algorithm for solving either generalized assignment problem or manpower scheduling problem or project assignment problem. In all these works, different initialization, crossover and mutation processes have been proposed and reported.

4. KEY POINTS IN THE PROPOSED ALGORITHM

Application of the Genetic Algorithms to the generalized assignment problem is not a straightforward task. The solutions shall be adequately represented and the relevant populations shall be appropriately generated. The crossover and mutation operators shall be defined carefully in such a manner that:

i. the application of these operators will not result in invalid solutions;
ii. the operators will provide the necessary diversity for efficient search of the solution space.

In this section, the definitions of the population selection, the crossover and mutation operators are given together with their reasonings.

4.1. POPULATION SELECTION

As stated before, Genetic Algorithms find a near-optimal solution by mimicking the process of evolution. For this purpose, a population of potential solutions is produced and a new population is generated based on the fitness of the old population, biased towards the fittest.

For GA that maximizes a function, individuals (potential solutions) in a population are usually represented as binary numbers, due to the ease of applying crossover and mutation operators. In

Assignment Problems, however, individuals in a population are merely different permutations of a limited set, making the conversion from and to binary representation an arduous task. Consequently, binary representation needs many different "repair algorithms." Due to this lack of convenience and possibly accuracy, we choose to represent the individuals as decimal number arrays of possible solutions.

In the first time that the population is created, random permutations of the solution set have been used as different individuals in the population. Since we have limited options due to the threshold and we have extended matrix due to the tentative track, we also limit choices of population to available candidates. Creating a population, in which only available (under-the-threshold) options are considered, is also important in our genetic operators like crossover and mutation.

4.2. COMMON ELEMENT CROSSOVER (CEX)

Crossover is the most important operator in Genetic Algorithms for it differentiates the population while carrying it towards an optimal solution. In solving TSP using Genetic Algorithms and path representation, certain methods for the crossover operator have been devised by different researchers. Three major ones of them are partially-mapped, order and cycle crossovers. Among these, partially-mapped and order crossovers try to preserve the order of the parent individuals of the population. This property, however, is trivial in Assignment Problems since the position of the elements in a solution set is crucial. Cycle Crossover is efficient for preserving positions of the elements for individuals that are permutations of the same set of numbers. In a matrix with a tentative track, however, the elements of different solution sets can very well be different, leading to a new way of crossover: Common Element Crossover (CEX).

In this technique, we acknowledge that parents can have certain elements in common as well as distinct ones. We also consider that if an element appears in the same position for many different solution sets, it is a good place for the element and should be preserved. In CEX technique, the elements that are common in the two parent individuals are noted as well as their indices. Then, preserving the common elements, the other elements are crossed over to generate new individuals.

As an example, let's say we have the parent individuals p_1= (7 3 2 6 1 9) and p_2= (5 6 8 3 1 4). The common elements are 3, 6 and 1 at indices 1, 3 and 4. Preserving the mentioned indices, and crossing over the rest, the first offspring o_1= (5 **3** 8 **6** **1** 4) and the second offspring o_2= (7 **6** 2 **3** **1** 9). By CEX, the positions of the elements are preserved and elements that are proved to be efficient in certain positions (e.g. 1 in the example above) carried over to the offspring.

4.3. IN-POOL MUTATION (IPM)

Mutation is complementary to crossover in genetic algorithms in order to diversify the population against crossover's tendency to converge into local minima (or maxima depending on the problem). In Assignment Problems, this is also the case.

Although it is easy to mutate a binary population, it is relatively harder to do it in decimal individuals with unique elements. There are also a number of ways to mutate this kind of populations. Some of them are inversion, insertion, displacement and reciprocal exchange. Each having its own virtue, the first three perform well with TSP rather than Assignment Problems, since they are tailored to preserve the order of the individuals at least partially. Although useful for some kinds of Assignment Problems, reciprocal exchange method is not suited for this case in which we have over-the-threshold values and a tentative track, because not every number is a good candidate for possible solutions due to the consideration of the threshold value. To address this problem, we devised another way of mutation called In-Pool Mutation (IPM).

In this technique, the randomly-selected element of any individual is mutated to any element from a certain pool of *available* (e.g. under-the-threshold and not repetitive) elements. By using IPM, any values that can result in an invalid solution, such as values over the threshold or used illegally, can be avoided without any extra repairing algorithm.

As a note, the probability for mutation to take place is usually kept low in order to keep the element of luck a little lower than a relatively *intelligent* crossover operator.

5. TEST CASES AND NUMERICAL RESULTS

The proposed scheme has been experimented by means of several controlled test cases. In the test stage, we tested the effect of number of iterations, population size, probability of crossover and mutation to the ability of the algorithm to find the minimum solution.

First, we tested them in a relatively small matrix (a 10×10 matrix), which has elements ranging from 1 to 50, uniformly and randomly distributed. The threshold value has been chosen to be 25, which would eliminate almost one half of the elements. The optimal solution for the matrix is 63.

In testing for iteration, the algorithm ran for different number of generations, ranging from 1 to 2048. The population size was constant at 50, probability of crossover was kept at 0.25 and probability of mutation was 0.01. After having 100 runs with each of the iteration choices, we calculated the minimum, maximum, mean and standard deviation of the values. The results are shown in Table 3.

Table 3: Testing for Iteration.

Matrix: 10×10; entries 1-50 (uniform random) **Population Size**: 50 **Threshold:** 25 **Probability of Crossover:** 0.25 **Probability of Mutation:** 0.01				
Solutions				
Number of Iterations	Min	Max	Mean	Standard deviation
1	84	135	115.3	10.4
2	77	134	114.8	10.5
4	90	137	113.63	10.05
8	83	131	113.65	10.55
16	76	130	113.42	10.55
32	73	128	104.21	11.36
64	63	124	96.75	12.07
128	63	117	91.08	12.21
256	63	111	86.1	11.06
512	63	108	79.7	10.7
1024	63	98	75.44	8.71
2048	63	94	76.09	7.78

The results show that, as the algorithm goes on for more generations, the results come closer to the optimum solution as can be noticed through the progression of mean values.

At the same time, population size proved to be an important factor in approaching an optimal solution. Iteration number kept at 64 and other variables specified as above, population is tested for values from 1 to 1024. The results are shown in Table 4.

Table 4: Testing for Population Size.

Matrix: 10×10; entries 1-50 (uniform random) **Iteration number**: 64 **Threshold:** 25 **Probability of Crossover:** 0.25 **Probability of Mutation:** 0.01				
Solutions				
Population Size	Min	Max	Mean	Standard deviation
1	114	201	156.1	20.34
2	74	175	144.0	17.49
4	95	158	131.6	14.79
8	72	167	121.1	15.17
16	81	158	111.7	13.82
32	68	125	102.6	13.53
64	69	118	96.4	11.83
128	63	112	89.8	11.00
256	63	102	81.2	8.97
512	63	94	76.18	7.65
1024	63	91	74.29	6.75

Similar to the results of the test for iteration number, increasing the population size helps the algorithm find more accurate solutions for the Assignment Problem.

Crossover operation's effect on the accuracy of results is tested by using a population size of 64, iteration number of 128, a mutation probability of 0.01 and a variety of crossover probabilities on the same matrix with the same threshold of 25. The results of this experiment are displayed in Table 5.

Under the light of the results, crossover operator, at first, seems almost trivial in affecting the outcome of the algorithm. It is, however, almost natural to attain the near-optimal results in a big population for a small matrix. Therefore, the conditions in the experimental results above are not a sufficient tool to judge the crossover operator.

The results above, on the other hand, are useful in order to compare the crossover operator's performance with the mutation operator's. Under the same conditions with the crossover operator and with crossover probability of 0.25, the mutation operator performed in Table 6.

Table 5: Testing for Crossover Probability.

Matrix: 10×10; entries 1-50 (uniform random) **Population Size:** 64 **Iteration number**: 128 **Threshold:** 25 **Probability of Mutation:** 0.01				
Solutions				
Crossover Probability	Min	Max	Mean	Standard deviation
0.00	63	115	89.6	11.13
0.01	63	112	85.9	10.82
0.02	64	120	90.13	10.97
0.04	63	115	89.33	10.67
0.08	63	117	90.44	11.23
0.16	63	115	88.13	11.37
0.32	63	111	89.02	10.12
0.64	63	121	89.78	11.83
0.96	63	117	89.34	12.81
1.00	63	119	91.46	11.98

Table 6. Testing for Mutation Probability.

Matrix: 10×10; entries 1-50 (uniform random) **Population Size:** 64 **Iteration number**: 128 **Threshold:** 25 **Probability of Crossover:** 0.25				
Solutions				
Mutation Probability	Min	Max	Mean	Standard deviation
0.00	90	132	111.70	9.70
0.01	63	114	87.46	11.83
0.02	63	101	82.65	9.39
0.04	63	99	76.07	9.15
0.08	63	91	71.7	7.55
0.16	63	94	76.07	7.1
0.32	64	94	82.01	5.83
0.64	64	99	85.6	6.27

The results for the mutation operator show more or less a random pattern, when compared to the crossover operator's results, accentuating the fact that mutation depends more on chance.

CONCLUSION

Genetic Algorithms prove to be efficient algorithms to find extremes of a function in a short time. Utilizing them in assignment problems can be crucial since assignment problems come on to the stage in many real-time applications as well as high-precision engineering challenges.

Although many researchers used Genetic Algorithms for a variety of problems and found many applications, specialized crossover and mutation techniques for Assignment Problems shall be devised.

There are three important points in our research that need to be accentuated as to give an idea in the efforts to solve Assignment Problems using Genetic Algorithms:

1- Use of Decimal Population: Being a limited set of discrete numbers, solutions for Assignment Problems do not need quantizing for which binary representation is used. This fact has also been noted by many different researchers, who studied Genetic Algorithms in order to solve problems including similar solution sets such as Traveling Salesman Problem.

2- Common Element Crossover: A distinguishing property of Assignment Problems from other similar problems like TSP is the importance of positions that elements occupy instead of their order. For example, a potential solution s_1= (2 4 3 5 7) represents totally different values from s_2= (3 5 7 2 4) in an Assignment Problem, whereas it defines the same solution for a TSP. Addressing this, we developed the Common Element Crossover (CEX) technique in which the common elements of the parents are unconditionally preserved in their places while other elements are swapped without changing their positions in the solution set. This technique safeguards the elements that take place in many different solutions, while not risking an over-the-threshold value, which would need extra repairing algorithms.

3- In-Pool Mutation: The commonplace techniques for mutation also risk the validity of a potential

solution since they disregard such a concern. In-Pool Mutation, on the other hand, changes a randomly selected element with another *legal* element that is in a pool of candidates for the specified index of the selected element. Another advantage of this technique over other techniques for decimal populations is that IPM affects only the selected element, thus being more controllable by the user, whereas other mutation ways affect at least two elements of an individual.

Aforementioned techniques enable the efficient use of Genetic Algorithms in Assignment Problems and yields to near-optimal solutions. Consistent with the nature of Genetic Algorithms, size and contents of the input matrix, threshold value, size of the population of potential solutions, number of generations and probabilities of crossover and mutation affect the accuracy of solutions.

It is hard to propose an evolutionary algorithm competent with the algorithms like Auction [21] and derivatives, which are specifically developed for the solution of the (Generalized) Assignment Problem. However, the following remarks shall be made:

- Evolutionary algorithms like Genetic Algorithms provide a very large set of high quality solutions. Using this fact, it can be claimed that this method will also computationally be effective in multi-hypothesis target tracking problems, where other possible solutions to the assignment problems are required.

- The performance of most algorithms depends on the sparsity of the cost-worker cost (observation-track distance) matrix; some algorithms which are developed for sparse matrices might be performing worse than the others for the dense matrices. The algorithm proposed in this work is considered to be generic; and hence although being not competent with some other algorithms for sparse matrices, its performance is believed to be comparable to most of them.

The current research will be extended in order to investigate the performance of the proposed method when applied to multi-hypothesis target tracking problems also considering the sparsity of the cost-worker cost (observation-track distance) matrix.

REFERENCES

[1] Holland, J. H., Adaptation of Natural and Artificial System, University of Michigan Press, Ann Arbor, MI, 1976.

[2] Goldberg, D.E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, New York, 1989.

[3] Michalewicz, Z., *Genetic Algorithms + Data structures = Evolution Programs*, Springer-Verlag, Berlin, 1999.

[4] Deb, K., *Optimization for Engineering Design-Algorithms and Examples*, Prentice Hall of India, New Delhi, 1995.

[5] Sakawa, M., *Genetic Algorithms and Fuzzy Multiobjective Optimization*, Kluwer Academic Publishers, 2002.

[6] Gen, M., Cheng, R., *Genetic Algorithms and Engineering Design*, Wiley, New York, 1997.

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

[8] Catrysse, D., van Wassenhove, L. N., "A survey of algorithms for the generalized assignment problem", *European Journal of Operational Research*, Vol: 60, pp. 260-272, 1992.

[9] Chu, P.C., Beasley, J.E., "A genetic algorithm for the generalized assignment problem", *Computers and Operations Research*, Vol: 48, pp. 17-23, 1997.

[10] Lorena L., Narciso, M.G., "Relaxation heuristics for a generalized assignment problem", *European Journal of Operational Research*, Vol: 91, pp. 600-610, 1996.

[11] Ross, G.T., Soland, R.M., "A branch and bound algorithm for the generalized assignment problem", *Mathematical Programming*, Vol: 8, pp. 91-103, 1975.

[12] Wilson, J.M., "A genetic algorithm for the generalized assignment problem", *Journal of the Operational Research Society*, Vol: 48, pp. 804-809, 1997.

[13] Bradley, D., Martin, J., "Continuous personnel scheduling algorithms: a literature review", *Journal of the Society for Health Systems*, Vol. 2, pp. 8-23, 1990.

[14] Easton, F., Mansour, N., "A distributed Genetic Algorithm for employee staffing and scheduling problems", *Proceedings of the Fifth International Reference on Genetic Algorithms*, San Mateo: Morgan Kaufmann Publishers, pp. 360-367, 1993.

[15] Tanomaru, J., "Staff scheduling by a Genetic Algorithm with heuristic operators", *Proceedings of the IEEE Conference on Evolutionary Computation*, New York, pp. 456-461, 1995.

[16] Dowsland, K.A., "Nurse scheduling with Tabu search and strategic oscillation", *European Journal of Operational Research*, Vol: 106, pp. 393-407, 1998.
[17] Dowsland, K.A., Thompson, J. M., "Nurse scheduling with knapsacks, networks and Tabu Search", *Journal of the Operational Research Society*, pp. 825-833, 2000.
[18] Aickelin, U., Dowsland, K.A., "An indirect Genetic Algorithm for a nurse-scheduling problem", *Computers and Operations Research*, Vol: 31, pp. 761-778, 2004.
[19] Harper, P.R., Senna, V., Vieira I.T., Shahani, A.K., "A genetic algorithm for the project assignment problem", *Computers and Operations Research*, Vol: 32, pp. 1255-1265, 2005.
[20] Majumdar, J., Bhunia, A. K., "Elitist genetic algorithm approach for Assignment Problem", *Advanced Modeling and Optimization*, Vol: 8, No: 2, 2006.
[21] Bertsekas, D. P., *Linear Network Optimization: Algorithms and Codes*, Cambridge, MA: The MIT Press, 1991.

Mustafa B. Akbulut was born in 1984. He received his B.Sc in Computer Engineering degree from Georgia Institute of Technology in Atlanta, Georgia, USA in 2006. During his studies, he worked at Havelsan Inc. as an intern where he conducted research on Genetic Algorithms for radar applications. He is currently pursuing a Ph. D. degree in Electrical Engineering at the University of Connecticut in Storrs, Connecticut, USA. His recent research interests include silicon-based nanoscale transistor technology and carrier transport mechanisms in small-scale devices.

A. Egemen YILMAZ received the B.Sc. degree in Mathematics in 1997, in addition to his B.Sc., M.Sc. and Ph.D. degrees in Electrical and Electronics Engineering from the Middle East Technical University (METU) Ankara, Turkey, in 1997, 2000, and 2007, respectively. Since 1996, he has been working in various defense industry companies and projects in Turkey and abroad. Since 2004, he has also been lecturing in Ankara University Electronics Engineering Department as a guest instructor. His research interests include especially the application of nature inspired optimization algorithms in numerical electromagnetics and multiple-target tracking.