# A RANDOM KEYS BASED GENETIC ALGORITHM FOR THE TARGET VISITATION PROBLEM

ASHWIN ARULSELVAN, CLAYTON W. COMMANDER, AND PANOS M. PARDALOS

ABSTRACT. The objective of the TARGET VISITATION PROBLEM is to determine a path for an unmanned aerial vehicle that begins at a point of origin and needs to visit several targets before returning to its starting point. An optimal visitation sequence is one which minimizes the total distance traveled and maximizes the utility of the visitation order. This utility measure is defined for each pair of targets and represents the relative value of visiting a particular target before another. In this paper, we present the results of a preliminary study investigating the effectiveness of a genetic algorithm for the TARGET VISITATION PROBLEM. The encoding scheme is based on random keys. Numerical results are presented for a set of randomly generated test problems and compared with the optimal solutions as computed by a commercial integer programming package.

## 1. INTRODUCTION

Path planning problems represent an enormous amount of the literature on cooperative control and optimization problems. This is particularly true for those involving military applications [2, 3, 18]. In this paper, we consider the so-called TARGET VISITATION PROBLEM (TVP), whose objective is to determine a path for an unmanned aerial vehicle (UAV) which starting at an origin must visit a set of targets before returning to its starting point. The objective is to determine an optimal path which minimizes the total distance traveled and maximizes the utility of the visitation sequence. The TVP has many military applications including combat search and rescue and disaster relief [13].

To date, the only work on the TVP is the original contribution by Grundel and Jeffcoat [13] in which the problem was first proposed. Here the authors present the TVP and provide a basic analysis examining the similarities between the TVP and other well-known combinatorial problems. In addition, they describe the implementation of a metaheuristic for the TVP based on the greedy randomized adaptive search procedure (GRASP) [20].

In this paper, we describe the implementation of a genetic algorithm (GA) for the TARGET VISITATION PROBLEM. Genetic algorithms represent a very active area of research in computational optimization and have been applied with great success to a myriad of combinatorial problems [10]. The chapter is organized as follows. In the next section, we present the problem statement and analyze some basic properties of the TVP. In Section 3, we describe in detail the genetic algorithm implementation. Section 4 presents the preliminary results of the GA when tested on a set of randomly generated instances. As a basis of comparison, we examine the performance of the GA to the optimal solutions as computed by a commercial integer programming package. We provide concluding remarks in Section 5 and discuss directions of future research.

*Date*: April 2007.

Air Force Research Laboratory Technical Report #:

Submitted to *Advances in Cooperative Control and Optimization*, M.J. Hirsch, P.M. Pardalos, R. Murphey, and D. Grundel (editors), Springer, 2007.

Revision submitted to *Naval Research Logistics*.

## 2. Problem Description

Before formally defining the problem statement, we introduce the symbols and notations we will employ throughout this paper. We use the symbol "$b := a$" to mean "the expression $a$ defines the (new) symbol $b$" in the sense of King [16]. Of course, this could be conveniently extended so that a statement like "$(1-\epsilon)/2 := 7$" means "define the symbol $\epsilon$ so that $(1-\epsilon)/2 = 7$" holds [6]. Also, let $|N|$ denote the cardinality of the set $N$. Finally, we will use *italics* for emphasis and SMALL CAPS for problem names. Any other locally used terms and symbols will be defined in the sections in which they appear.

An instance of the TARGET VISITATION PROBLEM consists of a set $N = \{1, 2, \ldots, n\}$ of targets located at distinct points in the plane. There is a matrix $\mathbf{D} = \{d_{i,j}\}_{m\times m}$, where $m := n + 1$. The extra node, say node 0 represents the UAV's origin. The values of $d_{i,j}$ represent the distances between each pair of targets. There is also a value $d_{0,j}$ which represents the distance from the UAV's point of origin to target $j$, for all $j \in N$. We note that the distances need not be symmetric. That is, $d_{i,j} \neq d_{j,i}$ in general. Lastly, the instance consists of a matrix $\mathbf{R} = \{\rho_{i,j}\}_{n\times n}$ where $\rho_{i,j}$ represents the preference or utility of visiting target $i$ before target $j$. The intuition is that targets for which $\rho_{i,j}$ is relatively large should be visited earlier in the sequence as they are assumed to have a higher "threat level" or "cause of interest".

As Grundel and Jeffcoat mention in [13], the values of $d_{i,j}$ are usually easy to obtain since literal distance measures or other metrics such as travel time are available. However, the values of $\rho_{i,j}$, the value added by visiting target $i$ before target $j$, are not always so simple to obtain. This is because utility is largely based on personal preference and opinion, both measures which are usually more qualitative than quantitative. To overcome this, there are several methods used by military strategists to arrive at the values of $\rho_{i,j}$. The most common method, and the one we adopt in this paper, is known as "target value reconciliation" [13]. In this method a group of experts offer a set of pair-wise rankings for the targets from which the preference matrix is derived. More specifically, for all targets $i$ and $j$, each expert is to specify a preference of visiting target $i$ before $j$ [4]. The value of $\rho_{i,j}$ is simply the cumulative number of experts who prefer to visit $i$ before $j$. For a discussion on other techniques, the reader is referred to [13].

Let $\pi$ be a permutation of the set of integers $[1, \ldots, n+1) \cap \mathbb{Z}$, such that $j =: \pi(i)$ implies that target $j$ is the $i^{\text{th}}$ position of the visitation sequence. Then a feasible solution to the TVP is one in which the UAV leaves its starting point, visits each target exactly once, and returns to the origin. An optimal visitation sequence is one which minimizes the total distance traveled and maximizes the utility of the sequence. With this, we can formulate the TVP as the following combinatorial optimization problem [13].

$$\text{Maximize } Z(\pi) = \Big[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \rho_{\pi(i),\pi(j)}\Big] - \Big[d_{0,\pi(1)} + \sum_{k=1}^{n-1} d_{\pi(k),\pi(k+1)} + d_{\pi(n),0}\Big]. \quad (1)$$

In [13], the authors provide a nice discussion of the similarities between the TVP and TRAVELING SALESMAN PROBLEM (TSP) [17] and the LINEAR ORDERING PROBLEM (LOP) [9]. It is easy to see that if there were no added benefits of visiting one target before another, then the components of the utility matrix would all be equal. Hence the contribution of this terms in the objective function would be constant. In this case, the problem would reduce to a TSP since the objective would only be a function of the distance traveled [9]. On the other hand, if the distances were irrelevant, then the TVP reduces directly to a LOP. Grundel and Jeffcoat provide example graphs further illuminating these similarities and the reader is referred to their paper for further discussion [13].

Before proceeding to the description of the heuristic for the TVP, we note a crucial observation first made in [13]. Notice that for a given instance of the TVP it might be the case that the entries in one of the matrices in (1) dominates the other. However, for our consideration both distance and utility should play an equal role in determining an optimal

visitation sequence. To circumvent this issue, we adopt a simple heuristic which balances the matrices. Let $\pi_r$ be a random permutation of the targets to be visited. Define $\gamma \in \mathbb{R}$ such that $\tilde{\mathbf{R}} := \gamma \mathbf{R}$. In order to normalize the $\mathbf{D}$ and $\mathbf{R}$ matrices, we adjust the particular value of $\gamma$ so that

$$\frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \tilde{\rho}_{\pi_r(i),\pi_r(j)}}{d_{0,\pi_r(1)} + \sum_{i=1}^{n-1} d_{\pi_r(i),\pi_r(i+1)} + d_{\pi_r(n),0}} \approx 1. \tag{2}$$

The value of $\gamma$ associated with a given permutation (visitation sequence) is calculated as the total distance traveled divided by the utility of the corresponding sequence. We see that the particular value of $\gamma$ can be adjusted to serve as a weight if it is deemed that the relative importance of the distances and utilities are not equal. For larger values of $\gamma$, the solution will tend to favor the utility of the sequence over the total distance traveled [13].

Using the permutation based formulation in (1), we now describe the implementation details for a genetic algorithm for finding near optimal solutions for the TARGET VISITATION PROBLEM.

## 3. GENETIC ALGORITHM

Genetic algorithms (GAs) receive their name from an explanation of the way they behave. It comes as no surprise, they are based on Darwin's Theory of Natural Selection [8]. GAs store a set of solutions, or a *population*, and the population *evolves* by replacing these solutions with better ones based on certain fitness criteria represented by the objective function value.

In successive iterations, or *generations*, the population evolves by *reproduction*, *crossover*, and *mutation*. Reproduction is the probabilistic selection of the next generations elements determined by their fitness level. Crossover is the combination of two current solutions, called *parents*, which produces one or more other solutions, referred to as *offspring*. Finally, mutation is the random modification of the offspring. Mutation is performed as an escape mechanism to avoid getting trapped at a local optimal solution [10]. In successive generations, only those solutions having the best *fitness* are carried to the next generation in a process which mimics the fundamental principle of natural selection, *survival of the fittest* [8]. Figure 1 provides pseudo-code for a standard genetic algorithm. Genetic algorithms were introduced in 1977 by Holland [11], and were greatly invigorated by the work of Goldberg in [10].

We note that though the GA does converge in probability to the optimal solution, it is common to stop the procedure after some "terminating condition" (see line 3) is satisfied. This condition could be one of several things including, a maximum running time, a target objective value, or a limit on the number of generations. For our implementation, we use the latter option and the best solution after `MaxGen` generations is returned.

When designing a genetic algorithm for an optimization problem, one must provide a means to encode the population, define the crossover operator, and define the *mutation* operator which allows for random changes in offspring to help prevent the algorithm from converging prematurely. The encoding scheme we propose for our GA is based on random keys and follows exactly as described by Bean [1]. As mentioned in [1], GAs often have a difficult time maintaining feasibility of solutions in successive generations. This problem is overcome by the use of random keys as an encoding mechanism for the population. Random keys work by encoding the solution vector using random numbers. The feasibility issue is then moved into the objective function, and subsequently all offspring produced are guaranteed to be feasible solutions.

For the GA implementation for the TVP, we have the following definitions. As mentioned above, solutions are represented by a random vector. To determine the visitation sequence, a random deviate from a distribution which is uniform onto $(0, 1) \in \mathbb{R}$ is generated for each target. The tour is determined by sorting the random numbers and sequencing

```
procedure GeneticAlgorithm
    Generate population P_k
    Evaluate population P_k
    while terminating condition not met do
        Select individuals from P_k and copy to P_{k+1}
        Crossover individuals from P_k and put in P_{k+1}
        Mutate individuals from P_k and put in P_{k+1}
        Evaluate population P_{k+1}
        P_k ← P_{k+1}
        P_{k+1} ← ∅
    end while
    return best individual in P_k
end procedure GeneticAlgorithm
```

FIGURE 1. Pseudo-code for generic genetic algorithm.

the targets in descending order of the sort. For example, suppose there are $n = 3$ targets to visit. Then a chromosome such as

$$(.34, .71, .28)$$

would correspond to the visitation sequence

$$2 \to 1 \to 3.$$

The objective value of the sequence can be evaluated, thus determining the fitness of the chromosome.

In order to evolve the population over successive generations, we use a reproduction method which copies the best individuals in the current generation to the next. We aptly refer to this set the BEST set. This technique ensures that the best solution is monotonically improving in every generation [1]. To breed new solutions, we implement a strategy known as *parameterized uniform crossover* [21]. This method works by selecting two solutions to serve as parents. In our implementation, one parent is chosen at random from the BEST set, and the other is chosen from the entire population (including BEST). Then, for each target to be visited, a biased coin is tossed. If the result is heads, then the allele of the BEST parent is chosen, otherwise the allele is taken from the other parent. The probability that the coin lands on heads is known as CrossProb, and is determined empirically. Figure 2 provides an example of a potential crossover when the number of targets is $5$ and $\texttt{CrossProb} = 0.65$.

| Coin Toss | H | T | H | T | H |
|---|---|---|---|---|---|
| Parent 1 | 0.56 | 0.81 | 0.22 | 0.7 | 0.86 |
| Parent 2 | 0.29 | 0.49 | 0.98 | 0.12 | 0.32 |
| Offspring | 0.59 | 0.49 | 0.22 | 0.12 | 0.86 |

FIGURE 2. An example of the crossover operation. In this case, $\texttt{CrossProb} = 0.65$.

Notice that after sorting the keys, the visitation sequence for Parent 1 is given by $\pi_1 = (5, 2, 1, 3, 4)$. Likewise, the sequence for the second parent is given as $\pi_2 = (3, 2, 5, 1, 4)$. From the figure, we can sort the keys of the offspring and we see that the visitation sequence is given as $\pi_{os} = (5, 1, 2, 3, 4)$. We see that in fact, the offspring tends to have more characteristics of Parent 1 which is consistent with idea of weighting the value of CrossProb to favor alleles of the parent from the BEST set.

Finally, the mutation operator is defined as follows. Instead of introducing random perturbations to selected offspring, we instead replace a set of individuals having the worst fitness with new solutions generated at random from the same distribution as the original population. This replacement set is referred to as the `WORST` set. Using this method, we are able to ensure that the GA does not converge prematurely. This is a common method, sometimes referred to as *immigration* and appears throughout the literature [1, 12]. An overall pictorial view of the generational evolution of the proposed GA is provided in Figure 3.

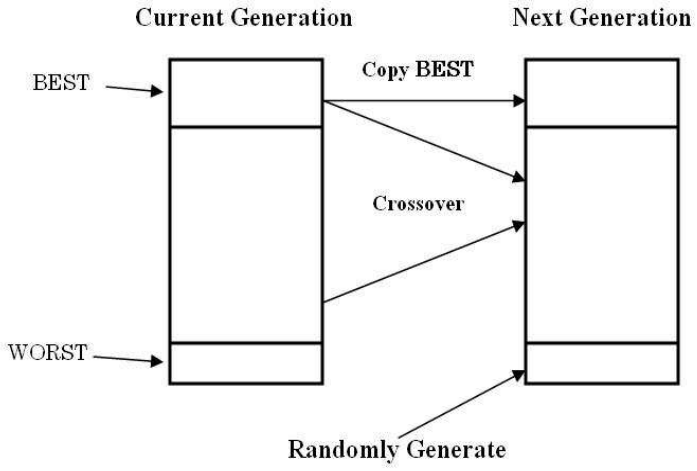


FIGURE 3. Graphical representation of generational evolution.

## 4. COMPUTATIONAL RESULTS

The proposed heuristic was implemented in the `C++` programming language and complied using GNU `g++` version $3.4.4$, using optimization flags `-O2`. It was tested on a `PC` equipped with a 1700MHz Intel® Pentium® M processor and $1.0$ gigabytes of RAM operating under the Microsoft® Windows® XP Professional environment.

In order to have a means to compare the results of the GA, we have implemented the integer programming model for the TARGET VISITATION PROBLEM using the CPLEX™ optimization suite from ILOG [7]. CPLEX contains an implementation of the simplex method [14], and uses a branch and bound algorithm [22] together with advanced cutting-plane techniques [15, 19]. The algorithms were tested on a set of randomly generated instances varying in size from $8$-$16$ targets. For each instance, the number of "experts" used to derive the utility matrix was $10$. Each of the "expertly defined" pair-wise preferences were generated uniformly. That is, for each target pair $(i, j)$ a random integer in the set $\{0, \ldots, 10\}$ was generated. This value represented the number of experts preferring to visit target $i$ before $j$. Also, the matrices were balanced using the heuristic described in Equation (2) above. For each instance, the maximum distance between the targets varied from $20$ to $150$ units.

For all of the instances tested, the parameters used for the genetic algorithm (GA) are provided in Table 1.

The comparative results of $250$ independent runs of the proposed heuristic on the $25$ randomly generated instances are presented in Table 4. The table is organized as follows. The first two columns provide the instance name and number of targets to be visited. The following two columns provide the optimal solutions as computed by CPLEX as well as

| | |
|---|---|
| $\texttt{CrossProb} = 0.7$ | Population Size ($\texttt{PopSize}$) $= 2 * \|N\|$ |
| $\texttt{MaxGen} = 10000$ | $\|\texttt{BEST}\| = .1 * \texttt{PopSize}$ |
| | $\|\texttt{WORST}\| = .2 * \texttt{PopSize}$ |

TABLE 1. Parameters used for the GA and HGA heuristics.

| Instance | | IP Model | | Genetic Algorithm | | | | |
|---|---|---|---|---|---|---|---|---|
| Name | Targets | Optimal Solution | Execution Time (s) | Max Soln | Min Soln | Avg. Soln | Avg. Time (s) | Avg. Dev (%) |
| rand8-1 | 8 | 60.2766 | 0.01 | 60.2766 | 56.3404 | 59.8895 | 0.054 | 0.642 |
| rand8-2 | 8 | 115.944 | 0.02 | 115.944 | 112.653 | 115.681 | 0.044 | 0.27 |
| rand8-3 | 8 | 195.333 | 0.01 | 195.333 | 194.96 | 188.555 | 0.032 | 5.006 |
| rand8-4 | 8 | 29.0074 | 0.02 | 29.0074 | 25.8592 | 28.888 | 0.052 | 0.412 |
| rand8-5 | 8 | 314 | 0.03 | 314 | 314 | 314 | 0.008 | - |
| rand10-1 | 10 | 157.404 | 3.01 | 157.404 | 140.133 | 154.084 | 0.123 | 2.109 |
| rand10-2 | 10 | 208 | 2.87 | 208 | 200 | 207.36 | 0.044 | 0.308 |
| rand10-3 | 10 | 520.679 | 0.01 | 520.679 | 437.12 | 518.698 | 0.049 | 0.380 |
| rand10-4 | 10 | 532.5 | 2.45 | 532.5 | 489.667 | 529.891 | 0.107 | 0.49 |
| rand10-5 | 10 | 365.125 | 4.87 | 365.125 | 303.615 | 349.457 | 0.068 | 4.291 |
| rand12-1 | 12 | 124.179 | 45.37 | 124.179 | 106.645 | 121.022 | 0.148 | 2.54 |
| rand12-2 | 12 | 318.38 | 61.17 | 318.38 | 266.641 | 308.668 | 0.128 | 3.050 |
| rand12-3 | 12 | 420.959 | 51.89 | 420.959 | 341.866 | 403.31 | 0.095 | 4.193 |
| rand12-4 | 12 | 594.546 | 16.44 | 594.546 | 487.099 | 580.956 | 0.137 | 2.286 |
| rand12-5 | 12 | 472.354 | 14.68 | 472.354 | 409.102 | 456.735 | 0.131 | 3.307 |
| rand14-1 | 14 | 137.609 | 303.55 | 137.609 | 110.948 | 128.208 | 0.225 | 6.832 |
| rand14-2 | 14 | 405.774 | 370.01 | 405.774 | 334.807 | 383.21 | 0.29 | 5.561 |
| rand14-3 | 14 | 631.711 | 184.82 | 631.711 | 508.412 | 594.765 | 0.267 | 5.849 |
| rand14-4 | 14 | 176.631 | 301.71 | 176.631 | 146.979 | 164.377 | 0.250 | 6.938 |
| rand14-5 | 14 | 679.625 | 2700.78 | 679.625 | 530.617 | 638.161 | 0.225 | 6.101 |
| rand16-1 | 16 | 381.934 | 1353.77 | 381.934 | 298.207 | 351.275 | 0.351 | 8.027 |
| rand16-2 | 16 | 431.531 | 2556.77 | 431.531 | 333 | 387.659 | 0.322 | 10.167 |
| rand16-3 | 16 | 415.338 | 462.5 | 415.338 | 332.868 | 380.319 | 0.329 | 8.431 |
| rand16-4 | 16 | 421.658 | 2810.45 | 417.109 | 314.109 | 386.355 | 0.397 | 8.372 |
| rand16-5 | 16 | 249.939 | 3788.49 | 249.939 | 187.592 | 234.192 | 0.326 | 6.3 |

TABLE 2. The numerical results for a set of 25 randomly generated instances are provided. The optimal solutions are also shown.

the required computation time. The heuristic data is listed next. Namely, for each instance we provide the best, worst, and average solutions computed during the $250$ runs. The average time to compute the best solution is provided as well as the average deviation from the optimal solution for each instance.

We begin our analysis of the data by noting that for all $6250$ runs, the GA computed optimal solutions $95.95\%$ of the time requiring $0.168$ seconds on average. This compares favorably with respect to CPLEX which required on average $601.428$ seconds to compute the optimal solutions. The CPLEX time was greatly improved for the instances containing $16$ targets which we supplied with a feasible solution as a starting point. Without the starting solution, the computation time for these instances was on the order of $75000$ seconds. The algorithm also scaled well, averaging less than $0.5$ seconds of computing time for each instance. It is reasonable to assume that better performance could be achieved provided the algorithm was able to run for more generations.

## 5. CONCLUSION

In this paper, we described the implementation of a genetic algorithm for the TARGET VISITATION PROBLEM. We began by formally introducing the problem statement. Then we described the details of the proposed heuristic. We presented numerical results comparing the GA solutions with the optimal solutions as computed by the commercial integer programming package CPLEX.

With the current literature on the TVP being slight, there exist many avenues to pursue future research. A natural extension of the work presented here is to augment the GA to include a local search intensification. The resulting, so-called hybrid genetic algorithm is gaining popularity and often adds significant improvement for a minimal amount of computation time. Alternatively, other metaheuristics could be applied as well as advanced cutting plane techniques to try to obtain optimal solutions for larger instances. Lastly, we suggest an investigation of approximation algorithms to produce solutions which have a guaranteed worst-case lower bound.

## REFERENCES


[1] J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[2] S.I. Butenko, R.A. Murphey, and P.M. Pardalos, editors. *Cooperative Control: Models, Applications, and Algorithms*. Springer, 2003.

[3] S.I. Butenko, R.A. Murphey, and P.M. Pardalos, editors. *Recent Developments in Cooperative Control and Optimization*. Springer, 2004.

[4] B.H. Chiarini, W. Chaovalitwongse, and P.M. Pardalos. A new algorithm for the triangulation of input-output tables in eEconomics. In P. Pardalos, A. Migdalas, and G. Baourakis, editors, *Supply Chain and Finance*. World Scientific, 2004.

[5] C.W. Commander, P.M. Pardalos, V. Ryabchenko, O. Shylo, and S. Uryasev. Jamming communication networks under complete uncertainty. *Optimization Letters*, published online, DOI 10.1007/s11590-006-0043-0, 2007.

[6] C.W. Commander, P.M. Pardalos, V. Ryabchenko, and S. Uryasev. The wireless network jamming problem. *Journal of Combinatorial Optimization*, published online, DOI 10.1007/s10878-007-9071-7, 2007.

[7] ILOG CPLEX. http://www.ilog.com/products/cplex, Accessed October 2006.

[8] C. Darwin. *The Origin of Species*. Murray, sixth edition, 1872.

[9] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[10] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[11] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers, 1989.

[12] J.F. Gonçalves, J.J.M. Mendes, and M.G.C. Resende. A hybrid genetic algorithm for the job scho scheduling problem. *European Journal of Operations Research*, 167:77–95, 2005.

[13] D.A. Grundel and D.E. Jeffcoat. Formulation and solution of the target visitation problem. In *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, 2004.

[14] F.S. Hillier and G.J. Lieberman. *Introduction to Operations Research*. McGraw Hill, 2001.

[15] R. Horst, P.M. Pardalos, and N.V. Thoai. *Introduction to Global Optimization*, volume 3 of *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, 1995.

[16] J. King. Three problems in search of a measure. *American Mathematical Monthly*, 101:609–628, 1994.

[17] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley, 1985.

[18] R.A. Murphey and P.M. Pardalos, editors. *Cooperative Control and Optimization*. Springer, 2002.

[19] C.A.S. Oliveira, P.M. Pardalos, and T.M. Querido. A combinatorial algorithm for message scheduling on controller area networks. *Int. Journal of Operations Res.*, 1(1/2):160–171, 2005.

[20] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.

[21] W.M. Spears and K.A. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

[22] L. Wolsey. *Integer Programming*. Wiley, 1998.



(A. ARULSELVAN) CENTER FOR APPLIED OPTIMIZATION, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA.

*E-mail address*: ashwin@ufl.edu

(C.W. COMMANDER) AIR FORCE RESEARCH LABORATORY, MUNITIONS DIRECTORATE, AND, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL USA.
*E-mail address*: clayton.commander@eglin.af.mil

(P.M. PARDALOS) CENTER FOR APPLIED OPTIMIZATION, DEPT. OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, GAINESVILLE, FL, USA.
*E-mail address*: pardalos@ufl.edu