Self-Crossover Based Genetic Algorithm for Performance Augmentation of the Traveling Salesman Problem

Conference Paper · March 2011

DOI: 10.2514/6.2011-1633

2 authors:

Nicholas Ernest
Psibernetix Inc.
14 PUBLICATIONS **45** CITATIONS
SEE PROFILE

Kelly Cohen
University of Cincinnati
207 PUBLICATIONS **994** CITATIONS
SEE PROFILE

Infotech@Aerospace 2011
29 - 31 March 2011, St. Louis, Missouri

AIAA 2011-1633

Self-Crossover Based Genetic Algorithm for Performance Augmentation of the Traveling Salesman Problem

Nicholas Ernest[1] and Kelly Cohen[2]
School of Aerospace Systems, University of Cincinnati, Cincinnati, OH,45221

A genetic algorithm (GA) based approach to the path representation of the Traveling Salesman Problem (TSP) will be presented towards potential application to co-operative control of a group of UAVs. The scenario involves a random distribution of cities on a two dimensional grid or a pre-defined set of targets, and determines the optimal path solution of the TSP by means of an iterative algorithm. Similar codes based on genetic algorithms are in existence; however, a unique set of modifications are utilized in this study. Generally speaking, when applying a GA to the TSP, traditional crossover is not permitted, as a city could be visited twice using this method. To circumvent this issue and optimize algorithm efficiency, we introduce a concept referred to as SCROOGE (Self CROssover Optimal GEnetic algorithm). This algorithm will be utilizing a combination of self-crossover and mutation of the strings making up the population. Through self-crossover, a single string is chosen for breeding, and has a chance to produce an offspring based off of its own genetic material, much as a starfish or many other animals would in nature. Additionally, SCROOGE's parameters morph with time, becoming more focused on avoiding local minima as iterations progress.

Nomenclature

TSP	=	Travelling Salesman Problem
GA	=	Genetic Algorithm
Cr	=	(self) crossover rate
Mr	=	mutation rate
K	=	quantity of cities
N	=	string population
P	=	number of strings in tournament poll

I. Introduction

THE Travelling Salesman Problem is a NP-hard (non-deterministic polynomial-time hard) mathematical and computational scenario that has a wide array of applications as described by Korte and Vygen[1]. In the most basic form, the TSP involves finding the shortest path, referred to as tour, of visiting a given set of cities and returning to the starting point. A city can be any necessary target, with distance of round trip being a measure of fitness, whether the actual variable be cost, time, etc. The TSP has seen applications in several diverse areas such as aerospace, logistics, genetics, manufacturing, telecommunications, and neuroscience[2]. The TSP lends itself to formulate the assignment of a UAV to multiple tasks as depicted by Rasmussen and Shima[3]. In recent times, research has focused on finding new and unique strategies which enables UAV teams to optimize the use of their combined resources to accomplish their mission given the need for real-time task allocation using leanring methods based on artificial neural networks[4,5].

UAV swarm intelligence is closely related to the TSP, for use in scenarios such as both air to air and air to ground combat, as well as even extinguishing wildfires or hurricane data collection among many other possibilities. With an end goal of a target allocation and flight planning program for a UAV swarm in sight, an algorithm's performance in solving the TSP will be a good measure of its capability to handle the more complex UAV problems. Many TSP solvers exist, however, Genetic Algorithms[6] show great inherent potential in finding an optimal tour. In GA's, a population of random possible solutions, or strings, is seeded. During each iteration, or generation, the fitness is determined for each string. In the case of the path representation of the TSP, each string is a series of city numbers, representing the order of the cities that the salesman will follow. Just as in evolutionary theory, the fittest of the specimens is more likely to mate and produce offspring to populate the next generation. With each generation, the fittest member of the population continually grows in strength until a final result is obtained. How mating and

[1] Graduate Student, School of Aerospace Systems, University of Cincinnati, AIAA Student Member
[2] Associate Professor, School of Aerospace Systems, University of Cincinnati, AIAA Associate Fellow

fitness are determined, how offspring differ from parents, and population size all are determined by the GA. During this study, different mechanisms were analyzed, and a variety of parameters optimized in order to develop an efficient and robust algorithm. Future work will include the incorporation of intelligent control into the algorithm, however this study will solely focus on the strength of the GA.

A. Background

GA's traditionally utilize crossover (two parents creating an offspring which has traits of both) and/or mutation (some form of random small transformation in an offspring, such as switching a single value from 0 to 1). Due to the special constraints of the TSP, traditional crossover cannot be utilized, as breeding of two parent strings could cause an offspring route to visit a city twice. However, there are methods for a GA to work around this problem. The simplest of these solutions is to avoid crossover by utilizing mutation and/or self-crossover.

Parents	Crossover	Self-Crossover	Mutation	Self-Crossover + Mutation
1 2 3 4 5 6	1 2 3 **:** 3 2 1	5 6 **:** 3 4 **:** 1 2	1 2 **4 3** 5 6	5 6 **:** **4 3** **:** 1 2
6 5 4 3 2 1		2 1 **:** 4 3 **:** 5 6	6 **3** 5 **4** 2 1	2 **3** **:** 4 **1** **:** 5 6

Table 1. Example GA reproduction methods for a six city TSP tour.

During self-crossover, one string selected for breeding produces its own offspring, with a random distribution of the sections of the parent. As can be seen in Table 1, self-crossover behaves more like an extreme mutation, rather than actual crossover.

An algorithm equipped with self-crossover and mutation, titled SC_GA, was developed in order to analyze the effectiveness of these reproduction methods. The user defines the number of randomly placed cities, Cr, Mr, N, and P. Cr and Mr are both percentages which dictate the chance that a chosen string's offspring will go through any of the alteration methods. If neither self-crossover nor mutation occurs, the offspring is identical to the parent. Optimal values for each of these parameters were determined through extensive trade studies. Upon completion of SC_GA's optimization, a publically available TSP solving GA, TSP_GA[7], was acquired from the Matlab fileshare. SC_GA was implemented into TSP_GA's matlab format, in such a way that the only difference between the two codes was the algorithms themselves. A set 100 city map with a known optimal solution was acquired from the TSP Library, removing the random distribution of cities from both algorithms. These allowed for comparison between the two algorithms, and gave data on the effectiveness of self-crossover in this application. Lastly, SC_GA went through another round of improvements to become SCROOGE, with morphing parameters that boost performance.

II. Algorithms

While determining possible reproduction methods for a TSP solving GA, an algorithm titled SC_GA was created. A variety of polling styles, or methods of selction, were studied, and a the tournament polling style was selected. P strings from the population were selected at random, and the fittest of these was chosen for breeding.

All parameters were optimized for a K value of 100 through a set of trade studies. Tested values of Cr and Mr range from 0 to 1, with 0.05 intervals. N varied from 0.5K to 3.5K, at 0.25K intervals, and the ranges of P were 2 to 32, at a rate of 2. However, simply altering these parameters one at a time does not sufficiently optimize the algorithm, as coupling is present to some extent. Trade studies of 100 runs of the algorithm for each parameter were completed three times in total. Initial values were educated estimates, with values used for all parameters in the subsequent trade studies set as optimals from the last. Since optimal parameters found in the third trade study were the same as those from the second, convergence was assumed and these values were accepted as optimal.

SC_GA was then compared to TSP_GA by inputting SC_GA into a copy of TSP_GA's Matlab code formatting and making the two codes as identical as possible except for the actual algorithm. Both algorithms were then set to the "KroA100" map from TSP library, which has a known minimum tour of 21282 miles. Using the same computer for all runs (HP Pavillion dv6 notebook, 4GB RAM, 2.20 GHZ dual core processor) 100 additional runs of 30 second lengths were completed for each algorithm, collecting best tour distance and time at each iteration.

Additional improvements were later given to SC_GA, transforming it to the optimal algorithm deemed SCROOGE. Throughout the trade studies conducted during SC_GA optimization, trends were noticed for each parameter. Some parameter values had higher performance at the beginning of the run, but lost steam quickly, or vice versa. SCROOGE's parameters start at optimal initial values, and gradually change gear with each set of iterations until reaching the preferred parameter values later in the run. This allows SCROOGE to quickly hone in on a general solution and then gradually switch to a set of parameters that are better at mitigating local minima. It is important to note that this morphing of parameters does not include any knowledge of the optimal tour; the parameters simply change with iteration number. This keeps SCROOGE from having an unfair advantage. SCROOGE was then run through a similar comparison scenario to the previous two algorithms.

III. Results

The average data for the comparisons is presented in Figure 1. As can be seen SC_GA and SCROOGE show improvements over the mutation-only algorithm. 30 seconds was selected as the cutoff time due to practicality; each of these codes have the ability to produce any of the possible strings. While they may hit a local minima that effectively stops the code, most runs were even improving after 10 minutes. Another important note is the fact that these codes were created in Matlab, which is rather inefficient in terms of computational time and efficiency. In order to reduce these times considerably, translating to a different programming language would be proper. However, the comparison between algorithms is what is of interest here.

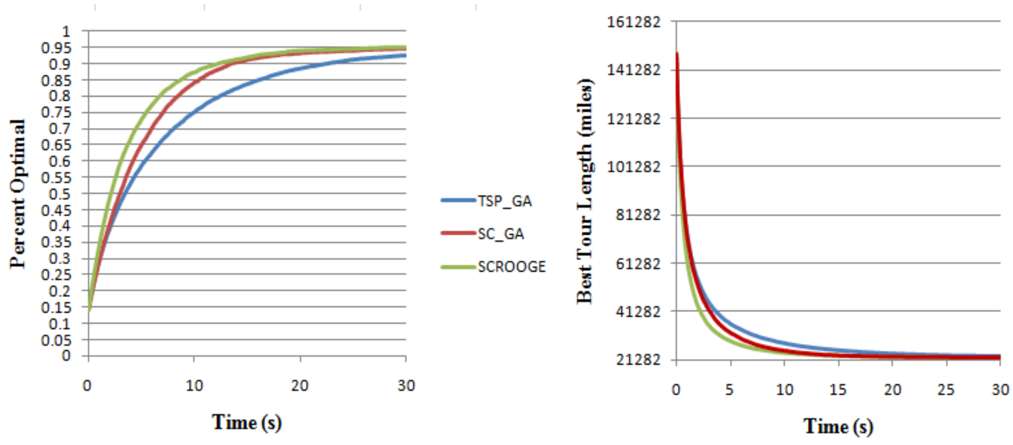


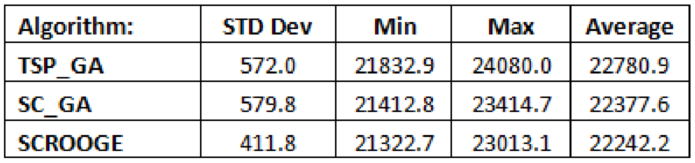
Figure 1. (a) Average percent optimal vs. time and (b) average best tour length vs. time.

Table 2 shows some important comparative data from Figure 1. An appropriate percent optimal is dependent on the type of problem, and 95% was desired in this case study. TSP_GA and SC_GA were unable to reach this benchmark, on average, within the 30 second window, unlike SCROOGE. SC_GA and SCROOGE both have a very similar trendline, with the difference being that SCROOGE's parameters during the beginning allow it to hone in on the optimal solution much more rapidly. This could be damaging to the population, as a lack of selectism in breeding greatly increases the risk of nearing an inescapable local minima. SCROOGE's parameters are set up to change before this risk would occur, as witnessed by the additional data presented in Table 3.

	Algorithm				
Time Til:	TSP_GA	SC_GA	SC_GA Improvement Over TSP_GA	SCROOGE	SCROOGE Improvement Over TSP_GA
85% Optimal	16.05 s	10.48 s	153.15%	8.68 s	184.91%
90% Optimal	22.4 s	13.6 s	164.71%	12.22 s	183.31%
95% Optimal	53.51 s	35.95 s	148.85%	27.626 s	193.69%

Table 2. Comparitive data from Figure 1.

Having the lowest standard deviation, minimum, maximum, and average end value after 30 seconds, SCROOGE is clearly the strongest algorithm. Its lower deviation compared to other algorithms' means that SCROOGE is able to mitigate local minima to a greater extent and that it's quicker to reach the final value before damaging local minima are present. SCROOGE's best run was able to obtain a tour length of 21,322.7 at 23.37 seconds into the run, which is 99.8% true optimal. While all of the algorithms perform well and can solve the TSP to find a near-optimal solution, SCROOGE is shown to do this faster, better, and more reliably.



Algorithm:	STD Dev	Min	Max	Average
TSP_GA	572.0	21832.9	24080.0	22780.9
SC_GA	579.8	21412.8	23414.7	22377.6
SCROOGE	411.8	21322.7	23013.1	22242.2

Table 3. Statistcal data for algorithms at 30 seconds (miles).

IV. Conclusions and Recommendations

During this study, SCROOGE was created in an effort to obtain a GA that can most efficiently solve the TSP. GA's are inherently simple, meaning additional constraints such as obstacles and turning radii can be implemented relatlively easily compared to other codes. While this algorithm can be a building block for a large variety of applications, its use in UAV swarm path planning and target allocation is of primary interest. SCROOGE's efficiency and robustness, coming from a combination of self-crossover and morphing parameters, will allow it to benefit from intelligent control to a greater extent. Results for the cases studied in this effort clearly demonstrate the effectiveness of this approach.

Future work will include the application of this approach to more complex cases and a more thorough statistical analysis of the effectiveness of this approach. Furthermore, it will include an examination concerning the scalability of this approach and the implications concerning memory and processing for both the off-line and on-line phases of development and implementation respectively.

V. References

[1]Korte, B., and Vygen, J., 2008, *Combinatorial Optimization – Theory and Application*, 4th Edition, Algorithms and Combinatorics, Vol. 21, Springer-Verlag Berlin Heidelberg.

[2]Applegate, D.L., Bixby, R.E., Chvatal, V., and Cook, W.J., 2006, *The Traveling Salesman Problem – A Computational Study*, Princeton Series in Applied Mathematics, Princeton University Press, New Jersey.

[3]Rasmussen, S.J., and Shima, T., 2008, "Tree Search Algorithm for Assigning Cooperating UAVs to Multiple Tasks", International Journal of Robust and Nonlinear Control, Vol. 18, pp135-153.

[4]Cohen, K., and Rasmussen, S., "Application of Proper Orthogonal Decomposition and Neural Networks to UAV Task Assignment", AIAA Infotech@Aerospace Conference, Atlanta, GA, April 20-22, 2010, AIAA Paper 2010-3541.

[5]Humphrey, L., Cohen, K., and Rasmussen, S., "Application of Proper Orthogonal Decomposition and Artificial Neural Networks to Multiple UAV Task Assignment", Invited Paper, AIAA Guidance Navigation and Control Conference, August 2-5, 2010, Toronto, Canada, AIAA Paper 2010-8439.

[6]Goldberg, D., E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, 1st edition, 1989.

[7]Kirk, J., "Traveling Salesman Problem – Genetic Algorithm," *Matlab File Exchange*, http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm [retrieved 30 June 2010].