A Genetic Algorithm Approach to Compute Mixed Strategy Solutions for General Stackelberg Games

Srivathsa Gottipati, Praveen Paruchuri
International Institute of Information Technology, Hyderabad, India
srivathsa.gottipati@research.iiit.ac.in,praveen.p@iiit.ac.in

ABSTRACT

Stackelberg games have found a role in a number of applications including modeling market competition, identifying traffic equilibrium, developing practical security applications and many others. While a number of solution approaches have been developed for these games in a variety of contexts that use mathematical optimization, analytical analysis or heuristic based solutions, literature has been quite sparse on the usage of Genetic Algorithm (GA) based techniques. In this paper, we develop a GA based solution to compute high quality mixed strategy solution for the leader to commit to, in a General Stackelberg Game (GSG). Our experiments showcase that the GA solution developed here indeed performs well in terms of scalability and provides reasonably good solution quality in terms of the average reward obtained.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**;

KEYWORDS

Stackelberg games,Mixed strategy, Genetic algorithms

ACM Reference Format:
Srivathsa Gottipati, Praveen Paruchuri. 2021. A Genetic Algorithm Approach to Compute Mixed Strategy Solutions for General Stackelberg Games. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3449726.3459419

1 INTRODUCTION

Stackelberg games have found a role in a number of applications including modeling market competition [9], identifying traffic equilibrium [4], practical security applications [7] and many others. General Stackelberg Games (GSGs) [1] contain N agents, and each agent n must be one of a given set of types θ_n. One player referred to as the leader, commits to a (mixed) strategy to optimize its utility function while the other players referred to as followers respond to the leader's strategy to optimize their own utility functions. The GSG we consider in this work has two agents namely a leader and a follower where θ_1 is the set of possible types for the leader (set to 1 type) and θ_2 for the follower. The leader therefore faces multiple follower types (both) with discrete payoff functions where the mixed strategy of the leader is known to the follower (but not the actual action taken in the round). Prior work that focuses on developing GA based solution(s) for GSGs is quite sparse with [6] developing a GA for computing a solution to a Stackelberg game that uses a continuous and differentiable payoff function for a two player game with a single type for each player.

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459419

2 GENETIC ALGORITHM APPROACH

Following are specifics of the GA approach we develop here:

Initial Population: The initial population is comprised of randomly generated mixed strategies and is seeded with the best deterministic strategy of the Stackelberg game [3]. The generation procedure for mixed strategies satisfies the following constraints:

(1) Probability assigned to each action must be between 0 and 1
(2) Sum of probabilities across all the actions must sum to 1

Fitness Function: We use the idea for fitness function presented in [6] (which presents a GA for a Stackelberg game with a continuous and differentiable payoff function) and make suitable modifications to account for the discrete payoff function here. In particular, we model a maximization function for the follower, given an arbitrary mixed strategy of the leader. We then use this 'functor' inside the objective function of the leader to develop a bilevel optimization model [8] inline with ideas behind game theoretic solution approaches for GSGs e.g., DOBSS algorithm [5] although there is no notion of a fitness function. The fitness computation presented in Fig. 1, can be mathematically described as follows: Given a GSG using normal form representation [3], we denote X as index set of pure strategies for the leader (row player), L denotes the set of follower types and Q^l denotes the index set of pure strategies for follower of type l (column player). R^l_{ij} and C^l_{ij} are the rewards of the leader and the follower of type l respectively. p^l denotes the a priori probability that a follower of type l will appear and Z_i denotes the probability of using pure strategy i, from the index set X of the leader. The *argmax* function computes the best response j of follower, for a given mixed strategy of the leader (as encoded in a chromosome of the population). The computed response j is then used in the objective function, to compute the maximum expected reward for the leader (i.e., fitness value).

$$\max \quad \sum_{i \in X} \sum_{l \in L} \sum_{j \in Q^l} p^l R^l_{ij} Z_i$$

$$\text{s.t.} \quad j \in \underset{j \in Q^l}{\text{argmax}} \{ \sum_{i \in X} C^l_{ij} Z_i \}$$

$$\sum_{i \in X} Z_i = 1$$

$$Z_i \in [0, 1]$$

Figure 1: Fitness Computation

Table 1: GA Parameter details

Parameter	Value
Population size	50
Crossover rate	0.9
Mutation rate	0.1
Selection	Tournament Selection with selection size=3
Crossover mode	Simulated Binary Bounded Crossover(SBX)
Upper($x^{(U)}$) / Lower($x^{(L)}$) bound	1/0
Distribution index η (for SBX)	0.1
Mutation mode	custom mutation
Generations	100

Selection and Crossover: We use Tournament Selection and Simulated Binary Crossover for these two operations.

Mutation: We develop a custom mutation operation which is a combination of exhaustive search along with a relaxed version of the same. However, only one of them is picked and used probabilistically (with a low probability for exhaustive search). In the exhaustive search, we vary the probability of an action in the mixed strategy by (tentatively) adding a small value δ in an attempt to find better mixed strategy (after normalization) and is used as replacement if found better than the current best identified so far. This operation is performed for every action over a set of δ's (i.e., all the possible combinations of actions and δ's) and the best identified is returned. The relaxed version is more like a random search wherein a random action is picked and is then tested with random δ's iteratively. If a better mixed strategy is found during this process, the procedure terminates, else will continue picking random δ's (without repetition), till all the δ's are tested for the action picked.

Normalization: Given that some of the operations can result in violation of probability constraints (as mentioned in "Initial Population"), we use normalization whenever necessary.

Replacement policy: At the end of each generation, if the new generation of offspring are less fitter than their respective parents, they would not be allowed in the next generation (borrowed from [2], to improve the speed of convergence.). If $c1$ and $c2$ are generated via crossover with $p1$ and $p2$ as parents, $c1$ is considered as child of $p1$ and $c2$ for $p2$ (no such notation is needed for mutation).

Termination Conditions: The GA terminates : (a) GA reaches 100 generations. (b) The time limit (of 1 hour) is crossed. (c) The standard deviation (of fitness) of the population is less than 1×10^{-4}. (d) The difference between the chromosome with best fitness value in the current and previous generation is less than 1×10^{-4} for 10 consecutive generations.

3 EXPERIMENTAL RESULTS

For experimentation purposes, we use the domain presented in [5], which is motivated by a patrolling and security application. We created (normal form) games with 10 and 20 houses involving 1 to 14 and 1 to 8 follower types respectively. Each game models a patrol route consisting of two houses and five instances of each game setting were generated for averaging purposes. DOBSS is a popular algorithm for benchmarking purposes [5] and used in this work to compute the optimal GSG solution. We used GUROBI 9.0 optimizer to implement the DOBSS algorithm and DEAP 1.3 to implement our GA. Table 1 showcases the parametric details of the GA we used. Both the algorithms have been implemented with

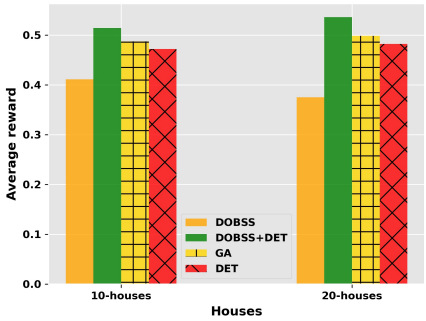

Figure 2: Average reward

Python interface on a machine with i5 processor and 16 GB RAM with a cutoff time of 1 hour (3600 seconds). In the case of DOBSS, the best deterministic strategy of the leader is used (referred to as DOBSS+DET), if the optimal mixed strategy could not be computed within the cutoff time. Given that our proposed GA is an anytime algorithm due to use of normalisation (to keep the chromosomes within the constraints), it performs well in terms of scalability.

Figure 2 showcases the average reward obtained using the different algorithms involving 10 and 20 houses for DOBSS, DOBSS+DET, GA and DET (best deterministic strategy). For the setting with 10 houses, the average reward obtained by DOBSS is 0.411, and DOBSS+DET is 0.514, 0.487 for GA, while the value obtained by DET is 0.472. Similarly, with 20 houses, the average reward for DOBSS is 0.375, 0.536 for DOBSS+DET, 0.498 for GA while the value is 0.482 for DET. The graph shows that our proposed GA performs reasonably well in terms of the average reward obtained. Our future work will explore ways to improve further the solution quality obtained using GA and explore possibilities to tailor the GA to better capture the domain characteristics/constraints of specific applications.

REFERENCES

[1] Carlos Casorrán, Bernard Fortz, Martine Labbé, and Fernando Ordóñez. 2019. A study of general and security Stackelberg game formulations. *European journal of operational research* 278, 3 (2019), 855–868.
[2] Yao-Chen Chuang, Chyi-Tsong Chen, and Chyi Hwang. 2015. A Real-Coded Genetic Algorithm with a Direction-Based Crossover Operator. *Inf. Sci.* 305, C (June 2015), 320–348. https://doi.org/10.1016/j.ins.2015.01.026
[3] Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on EC*. ACM, 82–90.
[4] W. Krichene, J. D. Reilly, S. Amin, and A. M. Bayen. 2014. Stackelberg Routing on Parallel Networks With Horizontal Queues. *IEEE Trans. Automat. Control* 59, 3 (March 2014), 714–727. https://doi.org/10.1109/TAC.2013.2289709
[5] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *AAMAS*. 895–902.
[6] João Pedro Pedroso et al. 1996. Numerical solution of Nash and Stackelberg equilibria: an evolutionary approach. In *Proceedings of SEAL*, Vol. 96. 151–160.
[7] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. 2008. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles International Airport.. In *AAMAS (Industry Track)*. 125–132.
[8] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. 2017. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation* 22, 2 (2017), 276–295.
[9] Heinrich Von Stackelberg. 2010. *Market structure and equilibrium*. Springer Science & Business Media.