Using Domain Knowledge in Coevolution and Reinforcement Learning to Simulate a Logistics Enterprise

Ying Zhao
Naval Postgraduate School
yzhao@nps.edu

Erik Hemberg
MIT, CSAIL
hembergerik@csail.mit.edu

Nate Derbinsky
Northeastern University
n.derbinsky@northeastern.edu

Gabino Mata
United States Marine Corps
gabino.mata@usmc.mil

Una-May O'Reilly
MIT, CSAIL
unamay@csail.mit.edu

ABSTRACT

We demonstrate a framework (CoEv-Soar-RL) for a logistics enterprise to improve readiness, sustainment, and reduce operational risk. The CoEv-Soar-RL uses reinforcement learning and coevolutionary algorithms to improve the functions of a logistics enterprise value chain. We address: (1) holistic prediction, optimization, and simulation for the logistics enterprise readiness; (2) the uncertainty and lack of data which require large-scale systematic what-if scenarios to simulate potential new and unknown situations. In this paper, we perform four experiments to investigate how to integrate prediction and simulation to modify a logistics enterprise's demand models and generate synthetic data based. We use general domain knowledge to design simple operators for the coevolutionary search algorithm that provide realistic solutions for the simulation of the logistic enterprise. In addition, to evaluate generated solutions we learn a surrogate model of a logistic enterprise environment from historical data with Soar reinforcement learning. From our experiments we discover, and verify with subject matter experts, novel realistic solutions for the logistic enterprise. These novel solutions perform better than the historical data and where only found when we include knowledge derived from the historical data in the coevolutionary search.

CCS CONCEPTS

• **Theory of computation** → *Evolutionary algorithms*;

KEYWORDS

coevolutionary algorithms, reinforcement learning, risk, logistics

ACM Reference Format:
Ying Zhao, Erik Hemberg, Nate Derbinsky, Gabino Mata, and Una-May O'Reilly. 2022. Using Domain Knowledge in Coevolution and Reinforcement Learning to Simulate a Logistics Enterprise. In *GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528990

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528990

1 INTRODUCTION

A logistics enterprise (LE) can be large and complex since it can contain multiple business processes from the areas of maintenance, transportation, and finance as a whole system. E.g. the U.S. Marine Corps (USMC) maintenance and supply chain is a complex enterprise. Logisticians in e.g. the Department of Defense, are trained consistently and they simultaneously work from clear operating procedures that outline every alternative and that help them make decisions. Moreover, the LE is in constant flux due to changes in components, component drift and attrition. A change, addition, or loss to a component has a ripple effect, impacting time to completion, servicing, purchases, transportation, etc. Finally, the LE have constantly changing demands or conditions from users and situational context.

These properties highlight the importance of analyzing LEs, plus forecast risks. However, the LE is frequently not accessible (e.g. proprietary), difficult to probe, or, cannot be experimented with (information is sensitive). This implies that a model of the LE cannot be provided and experimented with. Our goals are to:

(a) Provide the LE management with demands or conditions, within an envelope of realism, that pose risk on the LE's performance.

(b) Provide the LE management with sets of novel logistician decisions on each alternative that can provide better performance.

The only information available are historical records stored in multiple databases. Information is extracted and fused from these databases and fused it, to offer snapshots. Each snapshot indicates: *(a)* from each possible alternative every logistician could face, which ones were decided upon *(b)* from all possible situational demands or conditions, which ones were made *(c)* a "performance measure of the LE" as a score is possible to derive to use as a proxy for something like average time through system, costs, etc.

Our effort is focused on working with a snapshot of historical data to provide novel demands or conditions, within an envelope of realism, that are risky. These will help the LE management assess and mitigate unknown risk. Furthermore, it provides the LE management with novel sets of logistician decisions on each alternative that result in better performance.

To solve this challenge we develop and demonstrate a two component framework containing: *(a)* a surrogate model of the LE *(b)* a competitive coevolutionary algorithm We abstract the logistics training discipline and situational demands or conditions without loss of generality. Thus, we can assume there exists knowledge of

a set of possible binary alternatives that actors in the LE, i.e. "logisticians" can decide upon, such as resource allocations. Moreover, there is also knowledge of possible binary alternatives that describe the presence or absence of demands or conditions from users or exogenous conditions.

First we train our surrogate model with Soar-RL [5] to translate from the historical performance scores predicted from a set of (binary, known) demands or conditions and corresponding decisions. In a second step we abstract the logisticians' decisions as solutions and demands or conditions as tests[1], then, using the model from Soar-RL, we run a competitive coevolutionary algorithm [2, 4] (i.e.. min-max objectives) that allows a population of candidate solutions (trying to maximize the performance) against a population of candidate tests (trying to minimize performance as a means of finding potential risk), where a solution engages with a test and the outcome is derived from the surrogate model. We seed the initial pops with best from history. This is difficult unless we *(a)* Add constraints of realism *(b)* Consider pairwise value by comparison *(c)* Use domain-specific operators, i.e. mutation and crossover We show that this finds novel LE solutions and tests that translate to answers verified by subject matter experts of the USMC LE.

2 BACKGROUND

Soar Reinforcement Learning (Soar-RL). Soar-RL [5] is a cognitive architecture that scalably integrates a rule-based AI system with many other capabilities, including reinforcement learning. By using Soar-RL, a self-player or opponent take state/action attribute combinations to learn its preference $f_d(D)$ and $f_a(A)$, respectively. D (for self-player) and A (for opponent) could be different and asymmetric. A state attribute is an input variable that a player can not change and an action attribute is an one the player can decide its value. To translate this into a coevolutionary algorithm simulation, a preference is the contribution of a rule f_k to be selected for a self-player to win. We define preferences $p(f_k, v_1, c_1)$, $p(f_k, v_0, c_1)$, $p(f_k, v_1, c_0)$, and $p(f_k, v_0, c_0)$, where $p(f_k, v_1, c_1)$ means "if an attribute f_k is present ($v = 1$) there is a preference (probability) of $p(f_k, v_1, c_1)$ for the player to win the game in the end ($c = 1$)," i.e. win the game means fitness is one.

The total preference is the summation of the preferences from each of the state/action attribute combinations. There are K attributes. The Soar-RL reward is calculated as:

$$r = \sum_{k=1}^{K_1} p(f_k, v_1, c_1) - p(f_k, v_1, c_0) + \sum_{k=1}^{K_0} p(f_k, v_0, c_1) - p(f_k, v_0, c_0) \tag{1}$$

where K_1, K_0 denote the number of 1 (present) and 0 (not present) for an input data point with K attributes. The self-player gains a positive reward 1 if a correct action is taken at time t or a negative reward -1 if an incorrect action is taken.

Coevolutionary Algorithms. Coevolutionary algorithms [4] are related to evolutionary algorithms [1] and explore domains in which the quality of a candidate solution is determined by its ability to successfully pass some set of tests (attacks). For example, solutions (defenses) in a logistics chain need to pass the known operating conditions that are difficult or adversarial tests (attacks).

A basic coevolutionary algorithm evolves two populations with e.g. tournament selection and for variation uses crossover and mutation. One population comprises tests (attacks) and the other solutions (defenses). In each generation, engagements are formed by pairing attack and defense. The populations are evolved in alternating steps: first the test population is selected, varied, updated and evaluated against the solutions, and then the solutions population is selected, varied, updated and evaluated against the tests. Each test–solution pair is dispatched to the engagement component and the result is used as a part of the fitness for each of them. Fitness is calculated as the mean expected utility of the engagements.

3 DOMAIN SPECIFIC OPERATORS

Domain specific knowledge is provided by information from historical data, $\mathbf{X} \in \mathbb{R}^{N \times n}$, where each row is a set of decisions $\mathbf{d}$. We use general and simple domain knowledge to inform the coevolutionary search operators. The aim is to concentrate the search around realistic regions provided by the historical data, i.e. provide "knowledge" of the domain to the operators with information from the historical data. From this we count the positive decisions from the data, d_p, , the average number of positive decision $\overline{d_p}$, and standard deviation, s_d , as well as the max positive decisions d_m.

Mutation We provide knowledge to mutation by taking the number of positive decisions into account for mutation. We use decision flip mutation, i.e. flipping a positive (1) to a negative (0) decision, negative to a positive decision, or both.

Flip 1 to 0 and 0 to 1 Flip a positive decision to a negative and flip a negative decision to a positive decision. This leaves the number of positive decisions unchanged, $d_p(\mathbf{d}) = d_p(\mathbf{d'})$. We uniformly randomly chose a positive decision and make it negative. Then we uniformly randomly chose a negative decision and make it positive. The probability of this mutation, p_m is $p_m = \min(0.9, p'_m)$, $p'_m = \frac{1}{(|d_p(\mathbf{d}) - \overline{d_p}|/s_d)^2}$.

Flip 1 to 0 Flip a positive decision to a negative. This reduces the number of positive decisions by one, $d_p(\mathbf{d}) - 1 = d_p(\mathbf{d'})$. We uniformly randomly chose a positive decision and make it negative. The probability for this is $p_d = -p_m^2 + p_m$

Flip 0 to 1 Flip a negative decision to a negative decision. This increases the number of positive decisions by one, $d_p(\mathbf{d}) + 1 = d_p(\mathbf{d'})$. We uniformly randomly chose a negative decision and make it positive. The probability is $p_a = p_m^2 - 2p_m$

Crossover Domain specific positive decision aware variation by crossover is a modification of one-point crossover. The crossover point is selected to prevent one of the new solutions to have too many positive decisions (and the other too few). The maximum number of positive decisions for a new solution is based on the Gaussian distribution of positive decision in the data, $x_m \approx \mathcal{N}(\overline{d_p}, s_d)$. The crossover point based on the positive decisions in the first individual is uniformly randomly chosen as $1 < x_p^1 < x_m$ and for the second individual as $1 < x_p^2 < d_p - x_m$

Initialization The initial population are the top k (population size) decision from the historical data.

[1]We use solutions, self-player, defender and blue interchangeably to refer to the logisticians decisions. Tests, opponent, attacker and red interchangeably refer to the demands or conditions on the logistic enterprise

Fitness Function The number of positive decisions are also reduced by penalizing the fitness. Previous studies showed that taking many positive decision increased the fitness, but is not realistic [6]. Fitness, $y \in [-1, 1]$ is based on the Soar-RL reward, r and "realism" derived from the data. *Basic fitness:* $y = r$ *Length fitness:* $y = r - r_d$ Positive decision penalty if $d_p > d_m$ is $r_d = C_l(d_p(\mathbf{d}) - d_m)^2, C_l \in \mathbb{R}$ is a length penalty constant. *Lift fitness:* $y = r + 1/2 \sum_{d \in \mathbf{d}} L(d)$ If association information is use the fitness is adjusted. Association information, lift (L) is normalized in $a \in [0, 1]$. *Lift and length fitness:* $y = r - r_d + 1/2 \sum_{d \in \mathbf{d}} L(d)$ The fitness of an individual is the solution concept mean expected utility, i.e. the mean of all the fitness function values for an individual.

Domain knowledge with Lexical Link Analysis (LLA) We use an unsupervised text analysis method called lexical link analysis (LLA) to construct domain-dependent models for CoEv-Soar-RL [7]. In our case a word describes decision. The links between two nodes represent the likelihood of the two word features (i.e., attributes and their corresponding value or value ranges) occurring in the same context in the data. LLA computes the strength of a link between two word features w_l and w_k using the *lift (L)* measure $L_{lk} = \frac{P(w_l|w_k)}{P(w_l)}$ where $P(w_l|w_k)$ is the probability of word features w_l occurs in the same context where word feature w_k occurs and $P(w_l)$ is the prior probability of occurrence of the word feature w_l.

4 COEV-SOAR-RL FOR A USMC LOGISTICS ENTERPRISE

We use a proprietary USMC maintenance and supply chain data set. Some operational cost and risk comes from the uncertainty and unknown operation conditions that constitute the key challenges for the USMC maintenance and supply chain.

CoEv-Soar-RL. We use the following setup: **Opponent:** The test conditions of the USMCS LE. **Self-player:** Logistics chain solutions with data fields are decisions and actions that can be taken to pass the tests and improve performance. **Performance :** The probability to have the maintenance days between deadlined and closed less than an average computed from the historical data set.

We use Soar-RL to learn a surrogate model to use as a fitness function between test, solution, and fitness, i.e. $f_d(D)$ and $f_a(A)$, separately for both a self-player and opponent. For coevolution we use Soar-RL as a surrogate model [3]. The test (opponent) minimizes the fitness and the solution (self-player) maximizes the fitness.

Setup. We fuse data for a type of USMC equipment from a few databases including maintenance, supply, and equipment usage. We then aggregate the data for each service ticket. A total of 489 aggregated variables represent states and actions for both the self-player and opponent. We divide the variables into two groups: the opponent has 369 variables labeled `(O)` and the self-player has 120 variables labeled `(S)`. The number of rules learned from the data set using Soar-RL for the fitness functions $f_d(D)$ and $f_a(A)$ are $489 * 4 = 1956$, respectively.

We used a version of `Donkey GE`[2] with each decision encoded as positive (1) or negative (0). Duplicates are disallowed in the population.

[2]https://github.com/flexgp/donkey_ge

Experiments. We want to show that the CoEv-Soar-RL can use the association patterns discovered from the data, consequently, to bias the search towards novel high performing realistic solutions. The experiments are: **Baseline (E1)**: The fitness is calculated by *Basic fitness*. **Length (E2)**: Use a length penalty. Fitness is calculated by *Length fitness*. **Lift (E3)**: Use the context-dependent models without length-penalty. The fitness is calculated by *Lift fitness*. **Lift and length (E4)**: Use both context-dependent models and length penalty. Fitness is calculated by *Lift and length fitness*.

The fitness function is different for the four experiments. So the results are analyzed in terms of Soar-RL reward, r in Eq. (1), which represents the probability to have the maintenance days between deadlined and closed is less than the average computed from the historical data set.

5 RESULTS AND DISCUSSIONS

This sections answers the research questions regarding the impact on performance from using context information in the form of length penalty and association lift.

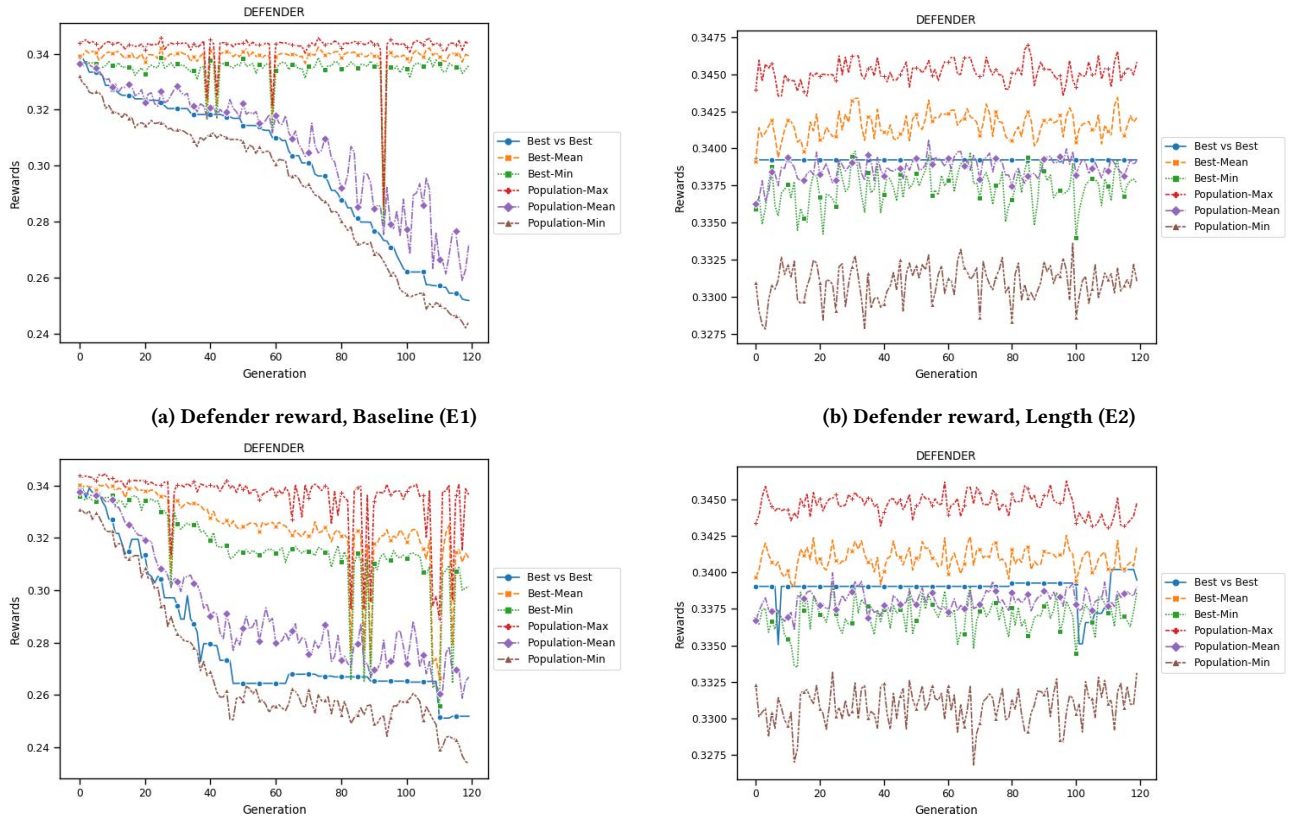
Baseline (E1). Figure 1a show the Defender's rewards in the baseline model over generations. The fitness used in the search is the Soar-RL reward(r in Equation (1)). The trends show that *(a)* the best Defender, representing logistics solutions, gets worse against the best Attacker (Best vs Best) while the Attacker, representing logistics tests, gets better against the best Defender (Best vs Best). *(b)* the best Defender does not get worse against average Attackers (Best-mean) while the Attacker gets better (Best-mean).

Effect of length (E2). Context is provided by penalizing the fitness when the number of positive decisions are greater than what was observed in the historical data. Figure 1b show that the rewards for Attacker and Defender's Best vs Best do not change.

Effect of lift (E3). The fitness is impacted based on historically good solutions by using association lift to provide context to the search. Figure 1c show that the rewards for the best Defender versus the best Attacker is decreasing initially, but is stabilized around generation 40 to 110. In a sense, the enterprise might be vulnerable and at risk for handling more serious tests.

Effect of lift and length (E4). Figure 1d show the trends of context-dependent models using associations from LLA. Associations from LLA are used to weight on realistic solutions. The plot shows: (1) The best Defender does not get worse and even get better around Generation 80-100 against the best Opponent or Attacker (Best vs Best) while the Attacker can get worse against the best Defender (Best vs Best). (2) The best Defender does not get worse against average Attackers (Best-mean) while the Attacker does not get better (Best-mean). The two seem to oscillate. (3) In Generation 120, the Best vs Best Defender generates higher Soar-RL reward than the ones in the original database, corresponding to new realistic solutions that are novel.

Future work is other problem formulations and sensitivity testing, as well as investigating the constraints that can be introduced by the grammar and rules when there is no historical data.



(a) Defender reward, Baseline (E1)

(b) Defender reward, Length (E2)

(c) Defender reward, Lift (E3)

(d) Defender reward, Lift and length (E4)

Figure 1: Line plot of reward over generations for the defender. x-axis is generation, y-axis is the Soar-RL reward for defender; Population-Mean is the mean reward in the population; Population-Max is the max reward in the population; Population-Min is the minimum reward in the population; The best solution is the leftmost solution shown in heatmaps (which has the best mean fitness against all the attackers in the population during the coevolutionary search); Best-Min is the min reward for the best solution; Best-Mean is the mean reward for the best solution; Best vs Best is the reward for best solutions against each other (top left corners in heatmaps).

ACKNOWLEDGMENTS

We would like to thank the Naval Postgraduate School (NPS)'s Naval Research Program (NRP) and the Office of Naval Research (ONR)'s Naval Enterprise Partnership Teaming with Universities for National Excellence (NEPTUNE 2.0) program for supporting the research. The views and conclusions contained in this document are those of the we and should not be interpreted as representing the official policies, either expressed or implied of the U.S. Government.

Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] Thomas Back. 1996. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford university press.

[2] Ying-Hua Chang. 2010. Adopting co-evolution and constraint-satisfaction concept on genetic algorithms to solve supply chain network design problems. *Expert Systems with Applications* 37, 10 (2010), 6919–6930.

[3] Torsten Hildebrandt and Jürgen Branke. 2015. On using surrogates with genetic programming. *Evolutionary computation* 23, 3 (2015), 343–367.

[4] Krzysztof Krawiec and Malcolm Heywood. 2019. Solving complex problems with coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 975–1001.

[5] John E Laird. 2019. *The Soar cognitive architecture.* MIT press.

[6] Ying Zhao, Erik Hemberg, Nate Derbinsky, Gabino Mata, and Una-May O'Reilly. 2021. Simulating a logistics enterprise using an asymmetrical wargame simulation with soar reinforcement learning and coevolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1907–1915.

[7] Ying Zhao, Douglas J MacKinnon, Shelley P Gallup, and Joseph L Billingsley. 2016. Leveraging Lexical Link Analysis (LLA) to discover new knowledge. *Military Cyber Affairs* 2, 1 (2016), 3.