



Interactive Evolutionary Computation for Strategy Discovery in Multi-Phase Operations

Martin Masek
School of Science
Edith Cowan University
Joondalup, WA, Australia
m.masek@ecu.edu.au

Luke Kelly
School of Science
Edith Cowan University
Joondalup, WA, Australia
l.kelly@ecu.edu.au

Jacob Snell
Edith Cowan University
Joondalup, WA, Australia
j.snell@ecu.edu.au

Daniel Wheat
School of Science
Edith Cowan University
Joondalup, WA, Australia
dwheat@westnet.com.au

Chiou Peng Lam
School of Science
Edith Cowan University
Joondalup, WA, Australia
c.lam@ecu.edu.au

ABSTRACT

Complex adversarial operations typically involve the allocation of finite resources to meet a set of objectives over a number of phases. This poses a challenge for AI-based strategy discovery. A strategy for one phase cannot be developed in isolation as the resources available in any one phase are dependent on the outcome of previous phases. Our proposed solution is to combine an evolutionary algorithm search with human-guided evaluation. The approach uses simulation-based fitness evaluation, where a human operator can view the fittest solution after every set number of generations. The operator can ‘lock in’ strategies for particular phases, and ‘suggest’ alternative strategies to guide further evolution. Key to our approach is a representation encoding that allows relative proportions of resources to be represented where actual levels may not be known a priori. We evaluate our solution on a three-phase scenario of a real-time strategy game and compare the effectiveness of strategies that were purely human-devised, purely evolved, and those resulting from the human-evolution collaboration. The collaborative approach shows promising results in being able to find an optimum solution earlier.

CCS CONCEPTS

• Computing methodologies > Genetic algorithms • Applied computing > Operations research

KEYWORDS

Interactive Evolution, Operations Research, Real Time Strategy Game

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal
© 2023 Copyright is held by the owner/author(s).
ACM ISBN 979-8-4007-0120-7/23/07.
<https://doi.org/10.1145/3583133.3590644>

ACM Reference format:

Martin Masek, Luke Kelly, Jacob Snell, Daniel Wheat and Chiou Peng Lam. 2023. Interactive Evolutionary Computation for Strategy Discovery in Multi-Phase Operations. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion 2023 (GECCO '23)*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3583133.3590644>

1 Introduction

A typical adversarial scenario involves decision makers on opposing sides, each with competing objectives. In devising a strategy, each decision maker typically uses their understanding of the situation and past expertise to propose, test and revise their strategy with the aim of meeting their objectives. One approach to planning complex strategic operations is to break them up into distinct phases where each phase has different priorities.

Artificial Intelligence (AI) techniques have shown promise in assisting human decision makers in formulating a strategy for conducting various single-phase operations. AI techniques that feature human collaboration, such as interactive evolutionary computation (IEC) [5], attempt to harness the ability of AI to search large combinations of potential strategies efficiently while using human input to counter the limitations of fidelity, subjectivity and context in the evaluation of solutions. However, applications of these techniques to multi-phase operations have been limited, with IEC being explored for problems that do not typically decompose into phases.

A common platform used to understand and explore strategy discovery in adversarial operations is the real-time strategy (RTS) genre of games, in which multiple sides use resources in a quest for dominance, typically framed in the narrative of armed combat. Such games can also be broken up into phases. For example, consider the scenario depicted in Figure 1, where a blue team of fighting units (initially placed in the bottom left corner of the map) has the end goal of destroying red team units, capturing red team bases (Red 1, Red 2 and Red 3) whilst preserving its own blue team

units. A human blue team planner might partition this operation into three successive phases where the goal of a particular phase is to either commit the entire blue team force to capture a single red team base, or to partition the blue force to attack multiple red team bases in a single phase. This simplified scenario, where strategy is tightly scoped to warfare strategy, will be used in this paper to demonstrate our approach.



Figure 1: A scenario where the blue team (blue base in the bottom left corner) must capture three bases of the red team (Red 1, Red 2 and Red 3).

In this paper, we present an approach where a human decision maker can interact with an AI system, based on a genetic algorithm [2], collaboratively searching for an optimal strategy. Our gene encoding allows for the representation of relative resources, as the actual resource levels for a phase are not known a priori, while maintaining a solution's validity during mutation and crossover. An agent-based simulator is used to evaluate solutions, both by the evolutionary algorithm and the human operator. The human operator has two means of affecting the subsequent evolutionary run: they can either 'lock in' parts of a strategy corresponding to particular phases or suggest an alternate strategy. Evaluation was performed by comparing baseline strategies, produced and fine-tuned by human operators using the simulator, with strategies generated through evolution alone and through human-AI collaborative evolution.

The rest of this paper is organised as follows: details of our approach to multi-phase interactive evolution are given in Section 2, experiments and results in Section 3, followed by analysis and discussion in Section 4 and conclusion in Section 5.

2 Multi-Phase Interactive Evolution

Our approach extends the standard genetic algorithm (GA) [2] by providing a means for user interaction to guide evolution in the context of a problem that has a number of phases. As with a standard GA, the evolutionary cycle consists of individual solutions in an initial population being evaluated and then evolved through several generations, with individuals in each subsequent generation being constructed from the previous using operators for selection (based on evaluated fitness), mutation and recombination. In this section, we focus on the novel aspects of the proposed approach,

namely the means of user interaction for the human operator and the representation employed in mapping a solution into a chromosome.

2.1 Representation

When using an evolutionary algorithm, an important step is to determine how a solution to the problem can be encoded as an evolvable chromosome. A common approach is to use real-valued schemes, where the solution is quantified into a set of numbers, each forming a gene in the chromosome. With evolution, individual genes in a solution change through mutation and parts of chromosomes in individuals change using crossover. In doing this, care must be taken that individuals remain valid after the mutation and crossover process.

In our representation, each gene in the chromosome is a real number that represents the relative proportion of one resource to be assigned to one objective in one phase. The actual amount of the resource that will be allocated depends on: the amount available, the relative proportion assigned to that objective, and the relative proportions of that resource assigned to all the other objectives in that phase. For example, consider that a resource type, R , can be assigned to four different objectives $[a, b, c, d]$ in a particular phase, and the relative proportions assigned to each are p_a, p_b, p_c, p_d . In this case, the chromosome consists of four genes having the values p_a, p_b, p_c , and p_d each. If the phase prior to this one resulted in an amount M of resource R being available, the amount of R allocated to objective a , R_a , is given by Equation (1). Similarly, the actual allocation of resource R to objectives b, c and d will also be normalised by the sum of all the proportions p_a to p_d . In our RTS case study, resources can only be assigned in discrete combat units, so the final allocations are converted to integers.

$$R_a = (M p_a) / (p_a + p_b + p_c + p_d) \quad (1)$$

This encoding allows each gene to be treated independently, where its value can be modified without needing to re-evaluate the other genes in the chromosome to keep it valid. A side effect of this is that the mapping from genotype to phenotype loses its uniqueness in that multiple genetically different chromosomes could produce the same solution. For the example above, two chromosomes, one with gene values $[0.1, 0.1, 0.1, 0.1]$ and another with values $[0.9, 0.9, 0.9, 0.9]$, whilst genetically very different, produce the same resource allocation, equally apportioning the resource to each of the four objectives. This is not that 'unusual', as similar behaviour of genetically very different organisms displaying the same characteristics is found in nature; it is worth noting so that others that implement this encoding can consider its impact in the context of their application.

2.2 Human-AI Interaction

Existing systems that employ some form of interactive evolution commonly focus on directly evaluating solutions using humans to determine a fitness value. This is typically achieved by presenting several solutions to the human and asking them to rank or order them based on fitness. Our work focuses on the human operator evaluating a single solution each time they interact with

the system, that solution being the fittest one in the current population. To help the human operator better understand the selected solution, they can experiment with and fine-tune the solution within the simulation, potentially improving it before passing it back to the algorithm.

Initially, prior to evolution occurring, the human operator is not presented with any strategy and can manually devise their own by iteratively assigning their resources (all located at the Blue base at the start) to the other bases, and visualising the simulated outcome in terms of attrition and the Loss Exchange Ratio (LER). This interaction is facilitated through a touchscreen interface designed for an electronic planning table. For example, if the human operator taps first on the Blue base and then on Red base 3, the view shifts to focus on the supply of combat unit resources from the Blue base to Red base 3, shown in Figure 2.



Figure 2: Assignment of resources from Blue base to Red base 3 during Phase 1.

Each time the human operator makes a change, the scenario is re-simulated, and the movement of units and attrition are visualised. This allows the human operator to evaluate and fine tune their strategy before handing over to the evolutionary step. After each set of generations has elapsed, the interface displayed is similar to the initial interface. However, instead of starting with a blank slate, the interface shows the fittest strategy from the previous generation. This strategy is visualised to the human, and its chromosome is mapped to actual unit numbers displayed in the right-hand panel as allocations. The human operator can edit the allocation strategy currently shown to them via two mechanisms: locking genes and editing individual gene values.

The effect of locking part of the chromosome is that the particular allocation becomes mandated for every individual in every subsequent generation (unless the human operator releases the lock at some later time). This is implemented by copying those particular gene values to all individuals in the current population (overwriting their prior values) and disabling the mutation operator for those particular genes. This locking mechanism offers the human operator the option of focusing the GA on a subset of the problem space. For example, if they would like to investigate an optimal strategy given a particular set of prior resource allocations has taken place.

Editing the values of individual genes changes how the resources are allocated to objectives in each phase. This allows the human operator to either 'tweak' the existing solution or to re-engineer the strategy completely.

Once an operator has devised a satisfactory solution, the chromosome associated with the edited solution is injected into the current population before resuming the evolutionary cycle. Case injection has been applied in various domains, including RTS games [3], typically by using an archive of promising solutions and periodically injecting them into the population. For our implementation, the human operator edited chromosome is injected into the population at the initialisation of the first generation and at set generational intervals thereafter. This mechanism influences the trajectory of the search, whilst still allowing the evolution to explore alternatives, with the injected chromosome competing in the population based on the fitness evaluation and being a candidate for crossover and mutation (whilst its initial human edited form remains in the archive and is periodically re-injected into the population).

2.3 Implementation

The overall system consists of three main components, the user interface, the evolution engine and the evaluator. These will now be introduced.

2.3.1 The Evaluator. The evaluator is the RTS simulation used to evaluate solutions and determine the behaviour visualised in the user interface, where MicroRTS [4] was used in this work. The behaviour of the red and blue teams' combat units is controlled using AI controllers that MicroRTS provides. The red team is controlled with a 'defend' type controller. The red agents are designed to stay within a specific range of their base, only engaging enemy units within that range; if this base is destroyed, the agent will immediately move to the next closest red base or stay in place if no other bases are left to defend. The blue team's agent AI is controlled with a 'rush' type controller, where each agent moves towards the base it was allocated to in each phase.

2.3.2 The Evolution Engine. The evolution engine was built in Python using the Distributed Evolutionary Algorithms in Python (DEAP) framework. The individuals are maintained with a generational population model where the initial population has a case-injected individual is inserted, as discussed in 2.2 and the remaining individuals generated randomly. Elitism was chosen to guarantee the preservation of the single best individual to the next generation, and a hall of fame was used to preserve a set of the best individuals between generations.

An individual's fitness was measured using a fitness function calculated at the conclusion of a simulation as (2).

$$\text{Fitness} = \text{Blue units remaining} - \text{red units remaining} + \text{Blue bases remaining} - \text{red bases remaining} \quad (2)$$

The tournament selection algorithm [1] was used as the selection scheme, with a tournament size of two. The mutation scheme employed was Gaussian mutation, with two-point crossover [2] used as the crossover scheme.

2.3.3 The User Interface. The user interface was implemented using the Unity Game Engine (Unity®). This was utilised to provide a graphically rich, interactive experience to the user, instead of utilising and extending the user interface components of MicroRTS or the Evolution Engine.

3 Experiments and Results

To evaluate the human-AI collaboration aspects of the approach, we divided the scenario into three phases and evaluated four approaches for devising a strategy in this context. Each approach was run 20 times, with the distribution shown in Figure 3. Each evolutionary experiment used a population size of 30, a mutation probability of 0.3 and a crossover probability of 0.9, where the parameters were chosen after initial tuning.

Two of the approaches can be considered as baselines, one for strategies designed purely by a human operator working with the simulator to fine-tune their strategy (but no evolutionary cycles) and a second baseline for strategies determined purely by evolution from a random population to 60 generations with no human operator input or feedback. For the human-only baseline, we obtained strategies from human operators that had some familiarity with the RTS genre and a prior introduction to the scenario. For each strategy, we imposed a time limit of five minutes for the human operator to devise and fine-tune their solution.

We compared the baselines to two approaches that combined human input with evolution. One approach where case injection was used with a human-provided solution (each run using one of the 20 strategies from the human baseline) with no further human involvement. The other approach also used case injection but sought human input at generations 20 and 40. Evolution during these experiments was limited to 60 generations to provide a strategy in approximately 3 minutes for a human operator.

4 Analysis and Discussion

Comparing the two baselines with the interactive approaches in Figure 3, the interaction of human and AI appears, on average, to result in more successful strategies than either human-only or evolution-only approaches. The plot shows that the fitness of human-only strategies, tuned using the simulator, varies widely and the performance of the human-only approach is significantly lower than evolution only, this difference being statistically significant ($p < 0.0001$) using an independent T-Test. On the other hand, the collaborative approaches perform significantly better than the baselines (for both at $p < 0.0001$). Comparing the two collaborative approaches, the approach with human input during the evolution performed better on average than the scheme where only the initial case was injected, this difference also being significant ($p = 0.0268$).

Examining the twenty individual strategies in the human-only baseline saw seven result in the complete elimination of the enemy (red) agents and bases. In the other approaches, involving evolution, all strategies resulted in the complete elimination of the enemy agents and bases.

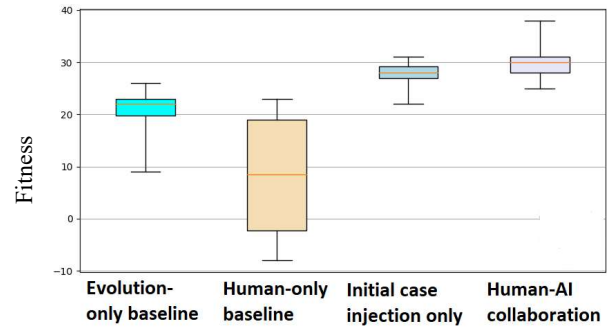


Figure 3: Box and whiskers plots of the best fitness of 20 individual solutions for each approach.

5 Conclusion

In this paper, we have shown how multi-phase adversarial operations, using the running example of combat in an RTS game, can benefit from human-AI collaboration in determining how resources should be allocated to objectives. Specifically, we focused on using a GA-based approach in which resource allocation for each phase is encoded in a chromosome, using a representation invariant to the actual level of resources that may become available in a particular phase and can be modified by mutation and crossover whilst remaining valid.

The human operator can examine and augment the current fittest solution and guide the subsequent evolution by locking parts of the strategy from further evolution while using case injection principles to inject a proposed solution into the evolving population regularly. Through a set of experiments, we compared the fitness of resource allocation strategies developed through the human-AI collaboration for a three-phase RTS scenario against the baseline performance of strategies developed and tuned by a human without the use of evolution and another set of strategies produced purely through evolution with no human input. Results indicate that the human operator could use the system to effectively collaborate with the AI, as the fitness of solutions developed through this collaboration was higher than the baseline.

ACKNOWLEDGMENTS

This research was supported by the Defence Science Centre, an initiative of the State Government of Western Australia.

REFERENCES

- [1] Goldberg, D. E., & Deb, K. 1991. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms* (Vol. 1, pp. 69-93). Elsevier.
- [2] Holland, J. H. 1992. Genetic algorithms. *Scientific american*, 267(1), 66-73.
- [3] Louis, S. J., & McDonnell, J. 2004. Learning with case-injected genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(4), 316-328.
- [4] Ontañón, S. 2013. The Combinatorial Multi-Armed Bandit Problem and its Application to Real-Time Strategy Games, In *AIIDE 2013*, pp. 58 - 64.
- [5] Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9), 1275-1296.