2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Model Representation and Cooperative Coevolution for Finite-State Machine Evolution

Grant Dick* and Xin Yao†
*Department of Information Science, University of Otago, New Zealand
grant.dick@otago.ac.nz
†Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA)
School of Computer Science, University of Birmingham, UK
x.yao@cs.bham.ac.uk

***Abstract*—The use and search of finite-state machine (FSM) representations has a long history in evolutionary computation. The flexibility of Mealy-style and Moore-style FSMs is traded against the large number of parameters required to encode machines with many states and/or large output alphabets. Recent work using Mealy FSMs on the Tartarus problem has shown good performance of the resulting machines, but the evolutionary search is slower than for other representations. The aim of this paper is two-fold: first, a comparison between Mealy and Moore representations is considered on two problems, and then the impact of cooperative coevolution on FSM evolutionary search is examined. The results suggest that the search space of Moore-style FSMs may be easier to explore through evolutionary search than the search space of an equivalent-sized Mealy FSM representation. The results presented also suggest that the tested cooperative coevolutionary algorithms struggle to appropriately manage the non-separability present in FSMs, indicating that new approaches to cooperative coevolution may be needed to explore FSMs and similar graphical structures.**

***Keywords*—Finite-state machines, evolutionary search, representation, cooperative coevolution**

I. Introduction

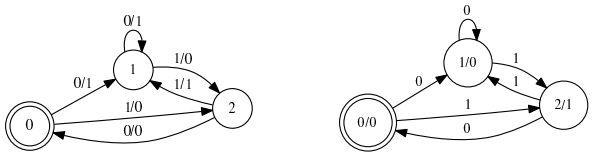
The evolutionary search of finite-state machine (FSM) representations can be traced back to some of the earliest work in evolutionary computation [1]. FSMs provide simple, yet flexible, solutions to many problems in control, prediction and pattern recognition. However, their expressiveness is often at the expense of requiring numerous variables to manage state transition and environmental output. Consequentially, the search space of finite-state machines is often large and complex, and becomes increasingly difficult to explore as problem size increases.

Recent work has explored the evolutionary search of Mealy-style FSMs for use in the Tartarus problem [2]. This work found that the resulting FSMs were very effective controllers within the Tartarus domain, and outperformed control methods from previous work. However, the Mealy representation used in the work required almost 80000 variables to implement the resulting machines, and this required considerable effort from the evolutionary search to find good solutions. The aim of this paper is to extend this previous work by undertaking a comparison of Mealy and Moore FSM representations. The resulting Moore representations require half the parameters of a Mealy FSM with the same number of internal states, which reduces the size of the search space considerably. Despite this reduced complexity, the resulting Moore FSMs are able to offer equal or better performance on the examined problems. Additionally, the paper explores the use of cooperative coevolutionary methods in the search of finite-state machines. The results suggest that existing methods of cooperative coevolution do not adequately manage the degree of non-separability present in FSM representations, and subsequently cannot improve the evolutionary search for good FSMs.

The remainder of this paper is structured as follows: §II examines previous work related to evolutionary computation and finite-state machines, and provides a brief overview of cooperative coevolutionary models; §III outlines the evolutionary model and test problems used in the experimental component of this paper; §IV performs a comparison of evolutionary search of Moore and Mealy-style finite-state machines; §V examines the impact of introducing cooperative coevolutionary models into finite-state machine evolution; finally, §VI concludes the paper, and suggests possible future work.

II. Related Work

Finite-state machines are a computational model in which a limited number of internal states act as memory, and transition between these states depends upon a response to the machine's *current state* and environmental input. Typically, a finite-state machine is defined as a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_i)$, where Q is the set of internal states of the machine, Σ is the set of permissible environmental inputs (e.g., sensory inputs of an agent), and Δ is the set of possible environmental outputs (e.g., agent actions). The function δ, maps the current state and environmental input into a *transition* into the machine's next internal state, while λ is the *output function* that maps the machine's current state and (optionally) environmental input into a suitable environmental output from Δ. Finally, q_i denotes the *initial state* in which the machine is started. Many different types of finite-state machine exist in the literature — in this paper, we limit exploration to Mealy and Moore implementations [3], [4]. The two models, examples of which as depicted in Fig 1, differ in the way that the output function, δ is defined: in the Mealy-style of FSM, δ takes both the current machine state and the environmental input as parameters in order to determine the output action. Conversely, the Moore



(a) *Mealy FSM* (b) *Moore FSM*

Fig. 1. The Mealy and Moore finite-state machine models. In the Mealy representation, environment actions (outputs) are assigned to each state transition, while in the Moore model, outputs are assigned to destination states.

FSM approach defines δ purely as a function of the current machine state. The two approaches present a trade-off between simplicity, flexibility, and the number of required internal states, with the Mealy representation argued as requiring fewer internal states than an equivalent Moore FSM, at the expense of needing to define an output label for each state transition. Despite their differences in implementation, it is possible to transform a Moore FSM into an equivalent-behaving Mealy FSM (and vice versa) by using additional internal states [5].

A. Evolution of Finite-State Machines

As mentioned previously, finite-state machines feature prominently in the history of evolutionary computation. The seminal work of Fogel et al. evolved FSMs for predicting sequences of symbols, ultimately creating the field of evolutionary programming [1]. This work was later extended to examine the evolution of FSMs in the iterated prisoner's dilemma problem [6]. Subsequent work explored the use of genetic algorithms to search for controllers for agent behaviour, with the Tartarus problem a frequently explored benchmark [7], [2]. Additionally, FSMs have been evolved to produce defensive behaviours in resource protection simulations [8]. Various other examples of FSM evolution also appear in the literature [9], [10], [11], [12], [13].

Most work in evolutionary computation focuses on Mealy FSM evolution, with the state transition and output functions modelled as strings of value-pairs denoting the matching input and corresponding output for the given state transition. Some work has explored the evolutionary search of Moore-style FSM representations [14], which typically requires fewer parameters for implementation: each state transition requires only a single parameter, while the output symbol attached to each state is either evolved as part of the representation, or externally assigned outside of the evolutionary search. Given that the number of state transitions is typically much greater than the number of internal states, Moore FSMs should require around half the parameters of an equivalent-sized Mealy FSM. However, there appears to be a lack of work directly comparing the evolutionary search behaviours of Moore and Mealy-style FSMs.

B. Cooperative Coevolution

The concept of coevolution features prominently in evolutionary computation. Early work focused on *competitive coevolution*, in which two or more populations evolve in a predator-prey like fashion, with the fitness of each population negatively correlated [15]. In the context of finite-state machines, competitive coevolution has been explored in the context of the Tartarus problem [16], but will not be explored in this paper. In contrast to competitive coevolution, *cooperative coevolution* presents a divide-and-conquer approach to the problem of evolving solutions. In cooperative coevolution, several populations coexist, each evolving subsets of the parameters to the overall problem [17]. When an individual from a given population needs to be evaluated, *representatives* from the other populations are selected and combined with the individual to form a complete solution to the problem. The individual can then be evaluated against the target problem.

Two challenges are present in establishing a cooperative coevolutionary approach to a problem. The first is the previously mentioned action of selecting population representatives. A common solution to this, as used in this paper, is to select the *fittest* individual from each population. Other solutions include selecting an ensemble of individuals (such as the best, median and worst fitness individuals) and performing multiple fitness evaluations to obtain a clearer measure of fitness, albeit at the expense of additional fitness evaluations.

In addition to selecting representatives, the second issue that must be addressed in cooperative coevolution is determining the number of populations and which parameters each population maintains. Some problems, such as the evolution of finite-state machines or recurrent neural networks, offer a natural decomposition of parameters by state or neuron, so this becomes straightforward exercise [18]. An alternative approach is to use *random regrouping*, in which each population is responsible for a random subset of the problem parameters [19]. To encourage coupled evolution of parameters during a run, these random groups are resampled at regular intervals. This approach has been shown to be very effective on high-dimension problems. Yet another approach is to attempt to "learn" parameter interactions during the course of the run, and use this to identify suitable parameter groupings [20].

III. Evolutionary Model and Test Problems

The previous work from which this paper is based explored Mealy FSM representations that required almost 80000 variables. While the resulting FSM performed well, the evolutionary search was computationally expensive. As such, there is a need to either reduce the number of required parameters to encode the solution, or establish another model of evolutionary search that more effectively searches the underlying parameter space.

This paper uses the evolutionary model from previous work [7], [2]. A population size of 200 individuals is used. In each generation, the population is divided into 50 groups of four individuals. Within each group, the fittest two individuals undergo recombination and mutation, and the resulting

offspring replace the two weakest individuals in the group. In this way, half the population is replaced in each generation, while an "elitist" strategy preserves the stronger population members. For recombination, uniform crossover is used, while point mutation is applied with probability 0.001.

The above breeding strategy was used for both Mealy and Moore representations. Therefore, the only difference between the configurations was in the underlying representations: for Mealy FSMs, the representation was a vector of value-pairs for each state transition, with one value denoting the destination state, and another to determine the environmental output for the transition. Crossover swapped complete value-pairs between parents, while mutation operated on either the destination value or output with equal probability. For the Moore FSM representation, a simple array of integers was used, with each value indicating the target state for the given state transition. Rather than evolving the output function, we hard code the output action to each state: for a given FSM with m states, each state is assigned a unique identifier from 0 to $m-1$. The output attributed to the state is then determined by the remainder of the division of the state's id by the size of the output alphabet. This approach to modelling Moore-style FSM is similar to previous work, in which only the state transition function was evolved [13]. However, in this work, we hard-code the state outputs prior to evolutionary search. Given that the labelling of internal states is arbitrary, and that initialisation of the populations is done randomly, this should not limit the flexibility of the resulting FSMs. Additionally, it decouples the output function from the transition function, and may make the search space smaller and easier to navigate.

A. Test Problems

Two test problems from previous work are explored in this paper. The first problem, *Tartarus* is a control problem, in which an agent is tasked with pushing boxes from the interior locations of a bounded world to the exterior walls using minimal sensory input in a given amount of time [21]. It presents a difficult challenge in which the agent must build a complex mental model of the world in order to effectively perform its tasks. At each time step, the agent is provided with the status of the eight cells in its immediate neighbourhood. Each cell can take on the one of the values EMPTY, WALL, or BLOCK. As a consequence, the agent must be able to respond to $3^8 = 6561$ environmental inputs. At each time step, the agent performs one of three actions: turn LEFT, turn RIGHT, or move FORWARD. The move forward action serves as the mechanism for moving boxes; if the agent is facing a box, and the cell immediately beyond the box is empty, then the box is moved to the empty cell as part of the agent's movement. If the cell beyond is not empty, then the agent performs no action and a unit of time is consumed. Similarly, if the agent performs a move forward action while facing a wall, then no action is taken in the world and a unit of time is consumed. Complete details for this problem can be found in previous work [21], [2].

The second problem, *competition for resources*, is a simulation that models the attacker-defender actions required to respond to an attack on resources in a networked environment [8]. In this problem, two agents (a player and opponent) are placed in a square toroidal grid, and each can sense the state of the four grid locations to their immediate north, south, east and west (the von Neumann neighbourhood). Each cell can take on three states - visited by the AGENT, visited by the OPPONENT, or EMPTY (not previously visited by either). Consequently, the agent must be able to respond to $3^4 = 81$ environmental inputs.[1] At each time step, the agent must move to one of the four adjacent cells that has not been visited by their opponent in a previous step. The goal of the simulation is to visit, or "capture", as many cells in the grid as possible, either by trapping the opposing agent, or rushing to grab empty cells as quickly as possible. For this problem, we use the "greedy" opponent agent from previous work, which attempts to move to empty cells if present, but otherwise moves to random locations that it controls. We do not use the greedy agent initialisation from previous work, but rather initialise each state transition output to any available action.[2] The remaining parameters for this problem were the same as for the original work [8].

To enable consistent comparisons between the finite-state machine representations, we used 12 internal states for the Tartarus problem, and eight internal states for the competition for resources problem. Varying the number of states, and examining the change in performance and evolutionary search, is left for future work. The number of states adopted is in-line with previous work exploring Mealy FSMs. Given that Moore FSMs are argued as requiring more internal states than Mealy FSMS, if any bias is introduced by the number of internal states chosen, it is probably in the favour of the Mealy FSM representation.

Results presented in this paper are averages obtained from 30 independent runs from each configuration. Where possible, results are presented with 95% confidence intervals (indicated as shaded regions around the plotted mean).

IV. Comparison of FSM Representations

The evolution of best fitness for each finite-state machine representation on the two test problems is shown in Figs 2 and 3. Two plots are shown in each figure: the first graph shows the direct comparisons of fitness over time, while the second graph attempts to normalise out the underlying trend to allow more direct comparison to be made. The relative fitness in the second graph is therefore:

$$f_{rel}(t) = \frac{f_{Moore}(t) - f_{Mealy}(t)}{f_{Mealy}(t)} - 1 \qquad (1)$$

where $f_X(t)$ is the mean best fitness of method X at generation t.

[1] In actuality, there are only 80 inputs, as the case where an agent is completely surrounded by cells visited by its opponent can never happen.

[2] The initial work initialised individuals by prioritising empty cells over visited cells, but this was not necessary for this work.

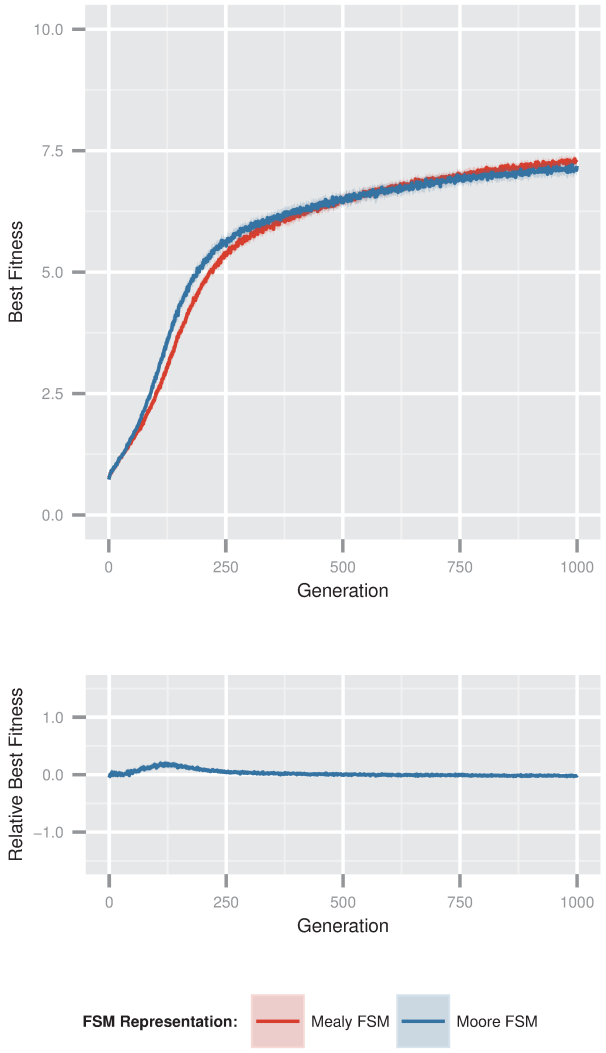


Fig. 2. Impact of FSM model representation on the evolution of best fitness on the Tartarus problem. The top graph shows the mean observed fitness over time, while the lower graph shows the fitness of the Moore-style FSM relative to the Mealy representation: values lower than zero indicate that the observed fitness when using the Moore representation lags behind the equivalent Mealy representation, while values greater than zero suggest improved performance. The shaded area around each series indicates the 95% confidence interval of the measured statistic.

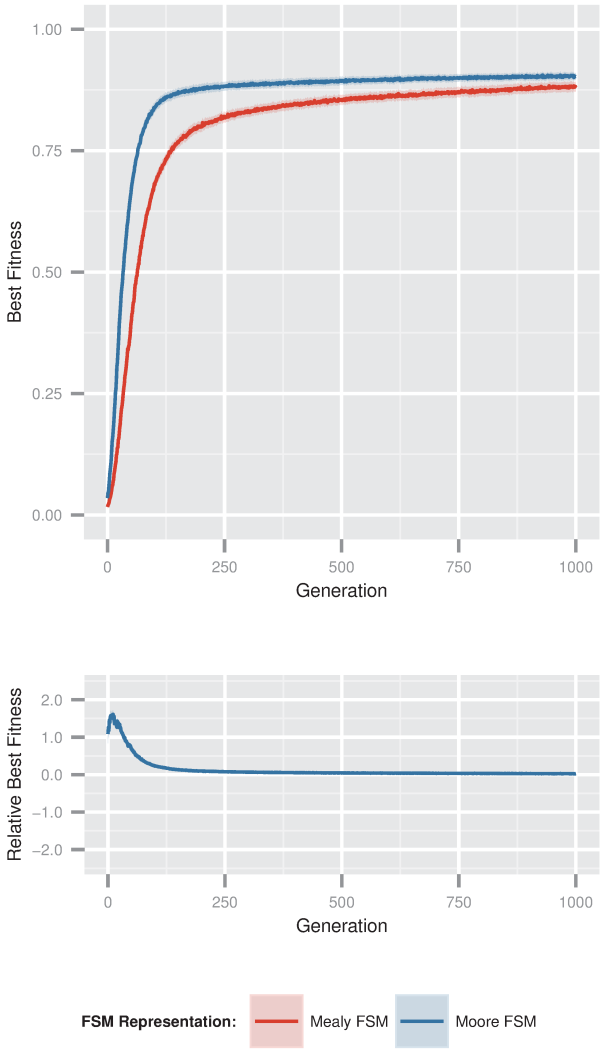


Fig. 3. Impact of FSM model representation on the evolution of best fitness on the competition for resources problem. The top graph shows the mean observed fitness over time, while the lower graph shows the fitness of the Moore-style FSM relative to the Mealy representation: values lower than zero indicate that the observed fitness when using the Moore representation lags behind the equivalent Mealy representation, while values greater than zero suggest improved performance. The shaded area around each series indicates the 95% confidence interval of the measured statistic.

When tested on the Tartarus problem, the Moore-style FSM representation evolves at slightly higher rate than the Mealy FSM representation, although the fitness is slightly lower (albeit not by a significant margin). The fitness curve presented by the Moore FSM configuration appears to be trailing off by 1000 generations, so it is possible that, given more generations to evolve, the Mealy FSM approach would have outperformed by a greater margin. It should be pointed out, however, that the Moore FSM result is competitive to 1000 generations, despite requiring only half the parameters of the Mealy FSM approach. This gives the Moore FSM more headway to include more internal states, which would have likely improved performance.

The results on the competition for resources problem are also interesting: despite using a fewer number of states, the evolution of Moore-style FSMs was both faster and converged to a higher fitness solution. The early stages of evolution are particularly impressive, with the fitness of the Moore FSM approach being up to 1.5 times greater than the equivalent Mealy FSM configuration.

In both test cases, the Moore FSM approach offered comparable or better performance to that of the equivalent Mealy FSM approach. A possible explanation for this is that, by using fewer variables, the Moore FSM approach produces a

much smaller search space. This smaller search space may be easier to navigate using evolutionary search, making it faster to identify good solutions. The ability to locate good solutions is then carried on into subsequent generations, maintaining good search performance.

V. Cooperative Coevolution and FSM Evolution

Previous work has demonstrated the ability of cooperative coevolution to increase the scalability of evolutionary search, and to exploit inherent separability in problem domains. The motivation for this work is inspired, in part, by previous work demonstrating cooperative coevolution in neural network evolution [18], [22]. In that work, each population was responsible for maintaining the weights for a single neuron of a fixed network topology. To evaluate an individual, representatives from each population were taken to assemble the complete weight set for the neural network, the resulting network was applied to the test cases, and the measured fitness was assigned to the individual. The results suggested that the coevolutionary process was able to identify and exploit the "limited separability" present in the representation [18]. It is therefore interesting to see if this carries over to finite-state machine evolution.

A standard coevolutionary strategy was adopted for this paper. The parameters were grouped into several distinct populations — each population had 200 members and used the same breeding strategy outlined in §III. A "generation" acted upon a single population in a round-robin fashion. In this way, the same number of evaluations was performed in both the coevolutionary and single population models, allowing simple comparisons to be made. Grouping of parameters was done in two ways: the first method was to group all of the state transition variables pertaining to a given state, meaning that there was a population for each internal state in the FSM. The second method used the random grouping approach, with the parameters of the representation shuffled and then allocated to subpopulations in equal proportions. After 100 generations, a new permutation of the parameters was established, and the new parameter groupings were redistributed amongst the populations. For this configuration, the same number of populations as the number of internal states was used, allowing for easy comparison. Other population configurations were explored, but the results are not presented here.

For evaluation purposes, the cooperative coevolutionary framework needs to select representative individuals from the populations not currently being worked upon. A simple "best individual" strategy was adopted — the best individual from each population not currently undergoing evolution was selected and combined with each individual from the current population as it underwent evaluation. Other strategies were investigated, but this strategy appeared to provide the best performance.

For a point of comparison, a recurrent neural network (RNN) architecture was also evolved through cooperative coevolution. The network architecture had five hidden nodes, with an output node for each possible environmental action. Hidden nodes used tanh activation functions, while the output nodes used linear activation nodes. The output node with the highest activation provided the environmental action for the system.[3] Given the fixed network topology, the representation for this approach was a simple weight vector, with BLX-0.5 crossover [23] used for recombination, and Gaussian mutation applied to each element with probability 0.05.

A. Results and Discussion

The evolution of best fitness over time for the cooperative coevolutionary approaches is shown in Figs 4 and 5. As done in §IV, the "raw" results are presented, along with an attempt to normalise the performance relative to the single population evolutionary model.

The RNN results for the Tartarus problem using the simple model of cooperative coevolution appear to concur with previous work; the framework is able to find good RNN solutions faster than an equivalent single population, non-cooperative, approach. However, neither of the FSM representations were able to improve upon the single population model by introducing cooperation. While the initial generations demonstrate a small improvement in fitness over a single population, the improvement is not maintained over the course of the run. The performance of all representations using the random regrouping approach is also poor, and the regrouping events are clearly delineated by a substantial and immediate drop in fitness, which takes time to overcome.

A similar story to the Tartarus problem is present in the results for the competition for resources problem. In this case, the cooperative approaches for both FSM representations are able to achieve parity with the single population model, albeit at a much slower rate. The RNN approach sees an initial benefit from using cooperative coevolution, although the fitness in the final generations lags behind that of the single population model. In all three representations, the random regrouping approach offers the worst overall performance, although the impact of regrouping appears to be less significant than on the Tartarus problem.

The results presented here raise some interesting points for discussion. In all cases using cooperative coevolution, the fitness of individuals in the very early generations is greater than those in the single population model. This is likely due to the "best individual" method used to select population representatives; by selecting the "best" representatives from each population, evaluation takes place under more controlled circumstances. In a non-cooperative model, the random initialisation that takes place means that good parameter values are likely to be paired with bad parameter values within the same individual. However, in the cooperative model, each individual samples only a subset of all the parameters, with the remaining parameter values being fixed through the representative solutions. Fixing these representative parameters reduces the

[3] In the competition for resources problem, the output nodes attached to actions that were not currently possible were ignored, so only *permissible* actions were considered for output.

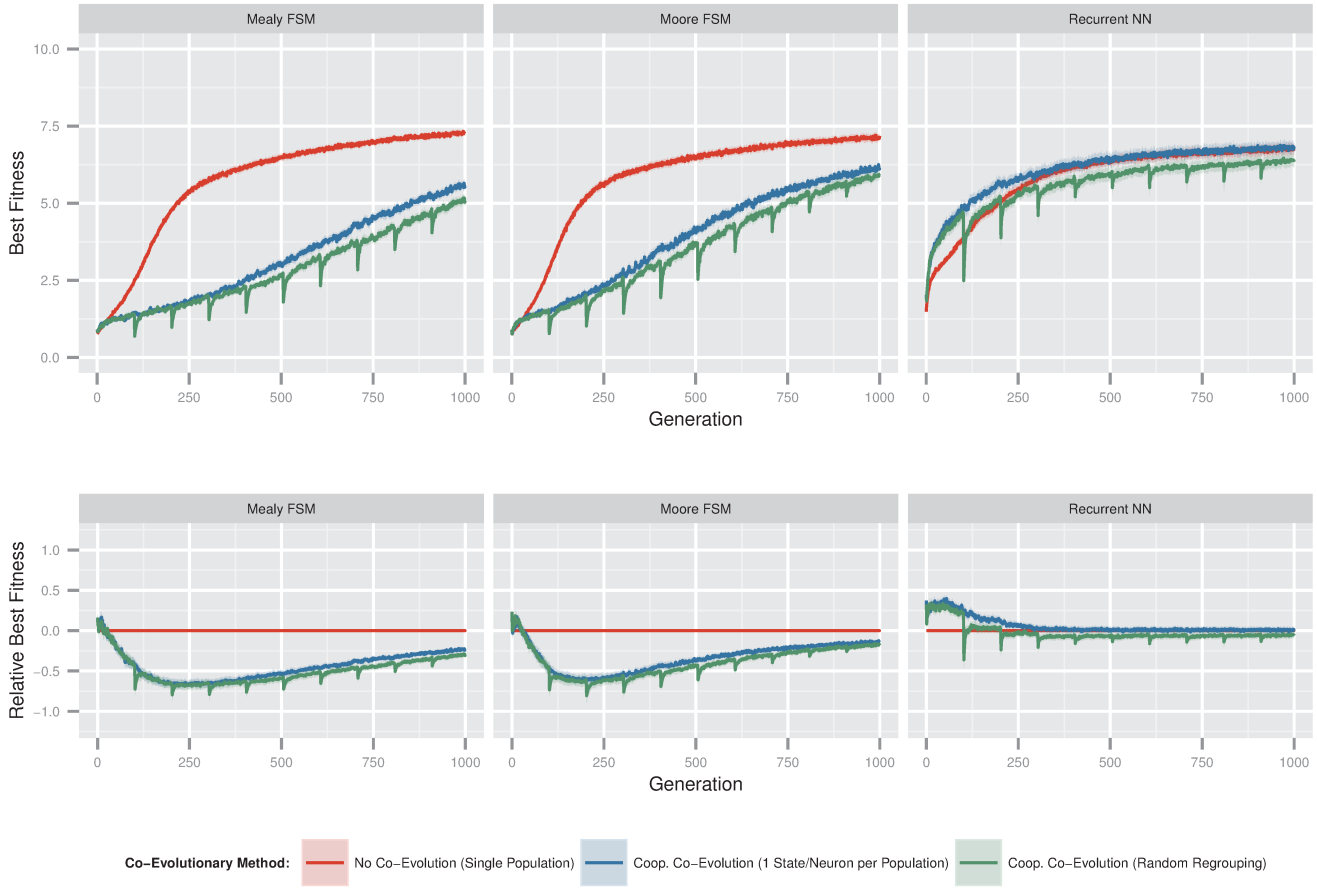


Fig. 4. Impact of cooperative coevolution on the evolution of best fitness on the Tartarus problem. The top row of graphs show the mean fitness scores for each method of cooperative coevolution, while the bottom row shows the fitnesses relative to the single population model: values lower than zero indicate that the fitness of the corresponding coevolutionary method lags behind the non-cooperative, single population model, while values greater than zero suggest improved performance. The shaded area around each series indicates the 95% confidence interval of the measured statistic.

noise in the evaluation process, meaning that a clearer picture of "good" parameter values is easier to quantify.

Unfortunately, the good performance observed in the early generations of cooperative coevolution does not carry forward into the later stages of the run. There are a couple of possible reasons for this. First is that, within each separate population, evolution is essentially slowed, as fewer generations of breeding take place due to the round-robin strategy. Despite being combined with the "best of breed" from the other populations, the actual parameters to not get as many opportunities to explore the search space as the single population models. The second possible explanation for the observed poor performance is that the ideal representative from a given population may change from individual to individual within a population and also between populations. Both the FSM and RNN representations are quite flexible and can encode many different, but equally viable, solutions. This gives rise to the possibility of multiple species existing within each population. This could lead to a "competing conventions" problem like that already known for recombination operators in neural network evolution [24]. Essentially, the representative from a given population may be adopting a different convention from that of the individual being evaluated, resulting in a sub-optimal RNN or FSM for evaluation.

VI. Conclusion

Finite-state machines offer simple yet flexible solutions to a range of problems. However, they suffer from an explosion of parameters in the face of increasingly complex input spaces and the required number of internal states to manage memory. This paper has explored two approaches aimed at controlling this parameter explosion. The first models FSMs through the Moore representation and fixes the output function to a set of hard-coded outputs determined by a unique identifier assigned to each state. The second approach adopted cooperative coevolutionary approaches that has exhibited good scalability properties in previous work. The results of this paper suggest that the first approach of hard-coding outputs in a Moore FSM is promising, while the cooperative coevolutionary methods

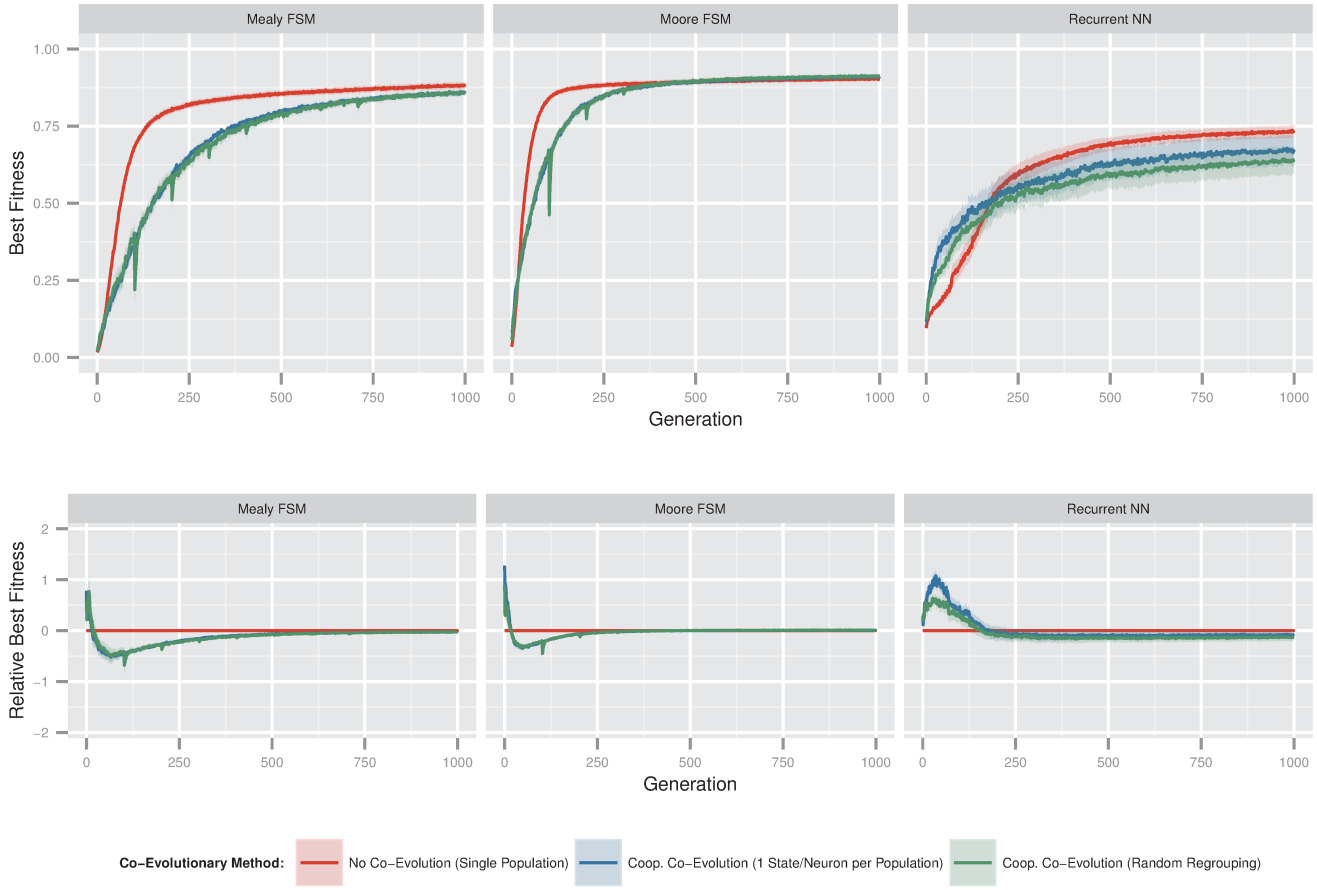


Fig. 5. Impact of cooperative coevolution on the evolution of best fitness on the competition for resources problem. The top row of graphs show the mean fitness scores for each method of cooperative coevolution, while the bottom row shows the fitnesses relative to the single population model: values lower than zero indicate that the fitness of the corresponding coevolutionary method lags behind the non-cooperative, single population model, while values greater than zero suggest improved performance. The shaded area around each series indicates the 95% confidence interval of the measured statistic.

raise several issues that warrant further investigation in future work. In particular, the issue of representative selection within cooperative coevolutionary frameworks needs to be explored further in light of the potential "competing conventions" problem.

A. *Future Work*

The results presented in this paper suggest several possible avenues of future work. Naturally, there is scope to apply the Moore FSM representation to problems from the literature not examined in this paper. this would certainly help to further characterise the behaviour of this representation.

The cooperative coevolutionary methods used in this paper make no attempt to learn the underlying parameter interactions in the problem. Rather, they rely on naïve grouping, or random permutations to encourage any interacting parameter co-adaptation. Work in identifying linkages and parameter interactions is an ongoing concern in cooperative coevolution [20]. Incorporating the results of this research into the evolutionary search of finite-state machine representations could yield useful results.

The concept of species in finite-state machine representations was alluded to in §V-A. This would certainly benefit from further characterisation, which would help towards designing solutions for identifying appropriate representatives within cooperative coevolution.

A key area that needs further investigation is how to select representatives within cooperative coevolution. A number of relatively straight-forward solutions are possible here: one solution may be to introduce tagging, similar to that of "simple subpopulation schemes" [25], and using this tagging information within selecting representatives. In theory, these tags would define a set of overlapping subpopulations within the cooperative populations, and these could adapt as small niches within the global population. Alternatively, an explicit distance measure could be introduced, with some measure of affinity used to select representatives. In such a configuration, selecting representatives could then be done in a multi-objective fashion.

REFERENCES

[1] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[2] G. Dick, "A true finite-state baseline for tartarus," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, ser. GECCO '13. New York, NY, USA: ACM, 2013, pp. 183–190. [Online]. Available: http://doi.acm.org/10.1145/2463372.2463400

[3] G. H. Mealy, "A method for synthesizing sequential circuits," *Bell System Technical Journal*, vol. 34, no. 5, pp. 1045–1079, 1955.

[4] E. F. Moore, "Gedanken-experiments on sequential machines," in *Automata Studies*, C. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton University Press, 1956, pp. 129–153.

[5] A. Klimovich and V. Solovev, "Transformation of a mealy finite-state machine into a moore finite-state machine by splitting internal states," *Journal of Computer and Systems Sciences International*, vol. 49, no. 6, pp. 900–908, 2010. [Online]. Available: http://dx.doi.org/10.1134/S1064230710060080

[6] D. B. Fogel, "Evolving behaviors in the iterated prisoner's dilemma." *Evolutionary Computation*, vol. 1, no. 1, pp. 77–97, 1993.

[7] D. Ashlock and J. Freeman, "A pure finite state baseline for tartarus," in *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000, pp. 1223–1230.

[8] W. M. Spears and D. F. Gordon, "Evolving finite-state machine strategies for protecting resources," in *Foundations of Intelligent Systems*, ser. Lecture Notes in Computer Science, Z. Ra and S. Ohsuga, Eds. Springer Berlin Heidelberg, 2000, vol. 1932, pp. 166–175.

[9] A. Pinter-Bartha, A. Sobe, and W. Elmenreich, "Towards the light — comparing evolved neural network controllers and finite state machine controllers," in *Intelligent Solutions in Embedded Systems (WISES), 2012 Proceedings of the Tenth Workshop on*, 2012, pp. 83–87.

[10] J. W. Horihan and Y.-H. Lu, "Improving fsm evolution with progressive fitness functions," in *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, ser. GLSVLSI '04. New York, NY, USA: ACM, 2004, pp. 123–126.

[11] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang, "Evolution as a theme in artificial life: The Genesys/Tracker system," in *Proceedings of the Workshop on Artificial Life (ALIFE '90)*, C. G. Langton, C. Taylor, D. J. Farmer, and S. Rasmussen, Eds. Reading, MA: Addison-Wesley, 1991. [Online]. Available: http://www.mpi-sb.mpg.de/services/library/proceedings/contents/alife90.html

[12] K. Chellapilla and D. Czarnecki, "A preliminary investigation into evolving modular finite state machines," in *CEC 99. Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 2, 1999, pp. 1349–1356.

[13] N. Niparnan and P. Chongstitvatana, "An improved genetic algorithm for the inference of finite state machine," in *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, vol. 7, 2002, pp. 5 pp. vol.7–.

[14] H. Shayani and P. J. Bentley, "A more bio-plausible approach to the evolutionary inference of finite state machines," in *Proceedings of the 2007 GECCO Conference Companion on Genetic and Evolutionary Computation*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 2937–2944. [Online]. Available: http://doi.acm.org/10.1145/1274000.1274039

[15] W. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Physica D: Nonlinear Phenomena*, vol. 42, no. 13, pp. 228 – 234, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0167278990900762

[16] D. Ashlock, S. Willson, and N. Leahy, "Coevolution and tartarus," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 2004, pp. 1618–1624.

[17] M. Potter and K. Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary computation*, vol. 8, no. 1, pp. 1–29, 2000.

[18] R. Chandra, M. Frean, and M. Zhang, "On the issue of separability for problem decomposition in cooperative neuro-evolution," *Neurocomputing*, vol. 87, no. 0, pp. 33 – 40, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231212000720

[19] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985 – 2999, 2008.

[20] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*, ser. PPSN'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 300–309. [Online]. Available: http://dl.acm.org/citation.cfm?id=1887255.1887289

[21] A. Teller, "The evolution of mental models," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. MIT Press, 1994, ch. 9, pp. 199–219.

[22] R. Chandra and M. Zhang, "Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, no. 0, pp. 116 – 123, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231212001014

[23] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata." in *Foundation of Genetic Algorithms 2*, D. L. Whitley, Ed. San Mateo, CA: Morgan Kaufmann., 1993, pp. 187–202.

[24] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI'89. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 762–767. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623755.1623876

[25] W. M. Spears, "Simple subpopulation schemes," in *Proceedings of the Fourth Annual Conference on Evolutionary Programming (EP94)*, A. V. Sebald and L. J. Fogel, Eds., 1994.