Using Game Theory to Analyze a Biologically-Inspired Agent Coordination Mechanism

Vincent A. Cicirello

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report AGENTS-08-002

October 2008

Using Game Theory to Analyze a Biologically-Inspired Agent Coordination Mechanism

Vincent A. Cicirello

Computer Science
Stockton University
Galloway, NJ 08205 USA
https://www.cicirello.org/

Technical Report AGENTS-08-002

Abstract

Computational frameworks motivated by the efficiency of biological systems abound—using mechanisms adapted from models of naturally occurring behavior such as foraging, immuno-response, dominance contests, behavioral thresholds for task performance, worker recruitment, etc. Often, the motivation is as simple as that it works well for the biological system's problem and perhaps a similar mechanism can work just as well for a problem-solving framework inspired by the biology. These nature-inspired systems are often robust, effective problem solvers. In this paper, we consider the application of game theory for the analysis of biologically-inspired agent coordination mechanisms. The objective of this paper is to provide examples of how game theory can be used to explain the emergent behavior of collective problem solving systems. To illustrate, we analyze an existing multi-agent task allocation protocol motivated by a computational model of wasp behavior.

Keywords: game theory, multi-agent coordination, multi-agent systems, swarm intelligence, wasp behavior

ACM Classes: I.2.8; I.2.11; G.m

MSC Classes: 68Q07; 68T42; 68W15; 91A05; 91A06; 91A10; 91A20; 91A68; 91A80

1 Introduction

Computational frameworks motivated by efficient biological systems abound: genetic algorithms [13, 11], ant colony optimization [10], artificial immune systems [9], among many other *swarm intelligence* systems [1]. Nature-inspired systems employ mechanisms adapted from models of naturally occurring behavior (e.g., foraging, immuno-response, dominance contests, behavioral thresholds, worker recruitment); and are often robust, effective problem solvers.

It is not always intuitively obvious as to why these systems are such effective problem solvers. But deeper theoretical analysis often leads to insightful explanations. For example, there are volumes of foundational theory behind why genetic algorithms (GA) are effective problem solvers. For example, the GA schema theorem states that above average fitness, short, low-order schema grow at least exponentially within

https://orcid.org/0000-0003-1072-8559

the population. Early on, Holland showed (through his analysis of the k-armed bandit [12, 13]) that the optimal exploration policy is to sample the observed best at a rate that grows exponentially—making an argument that the GA performs the optimal tradeoff of exploration and exploitation. Another example is where Parunak and Brueckner [20] use the second law of thermodynamics and Kugler and Turvey's [15] theory of self-organization to explain the self-organized behavior of an artificial ant colony.

In this paper, we consider the use of game theory for the analysis of biologically-inspired agent coordination mechanisms. To illustrate, we analyze an existing multi-agent task allocation protocol motivated by a computational model of wasp behavior [17, 19]. This task allocation protocol has been used by Cicirello and Smith for assigning jobs to machines (e.g., trucks to paint booths) [8, 3], as well as for minimizing weighted tardiness under the constraint of sequence-dependent setups [7]. Empirically, it has performed better (in simulation) than existing agent-based systems for the problem. The objective of this paper is to provide examples of how game theory can be used to explain the emergent behavior of collective problem solving systems.

Game theory is the study of decision-making of multiple entities in strategic interdependent situations (see [16, 14] for an introduction). In this paper, we assume that the nature-inspired agents are self-interested entities, incapable of entering into binding, enforceable agreements; and thus assume the application of *noncooperative* game theory. We assume that the reader is familiar with the language of game theory, and in our analysis employ *normal form* games, and *repeated* or *iterated* games, as well as the *Nash Equilibrium* (NE) concept.

We begin in Section 2 with an overview of the multi-agent system. Then, in Section 3, we present a game theoretic analysis of the system. A preliminary game theoretic analysis appeared in [2], which also applied game theory to analyze an ant-inspired multi-agent system [5]. But in the present paper, we have made significant improvements to the game theory model. We discuss the analysis and conclude in Section 4.

2 Wasp Behavior Inspired System

The multi-agent system analyzed in this paper is faced with the problem of assigning jobs to machines in a simulated factory. There is a set of multi-purpose machines each capable of processing more than one type of job, but with significant setup cost associated with processing a job of a type for which it is not currently set. At the completion of a job, it remains setup for that job type and is capable of processing another job of the same type with no additional setup cost. Otherwise, it is necessary to change the setup of the machine for the next job. Jobs arrive continuously, sequenced arbitrarily with respect to job type. The types of the jobs that arrive are not known until they arrive. Upon arrival, a job must be placed at the end of a queue. A machine must process jobs from its queue in a first-in-first-out order. The objective is to minimize the total setup costs.

The multi-agent system analyzed here is based on the task allocation behavior of wasps and is described briefly here with sufficient detail for analysis purposes (see [8, 3] for complete details). The game theoretic analysis considers the original version of the system [3] and the improved version [8]. To differentiate between the versions, the former is referred to as WaspsV1 and the latter as WaspsV2.

Each machine is represented by a wasp-like agent that uses a stimulus-response mechanism to choose jobs to add to its queue. Each agent maintains response thresholds, $\Theta_w = \{\theta_{w,0}, \ldots, \theta_{w,J}\}$, where $\theta_{w,j}$ is the threshold of wasp w to type j jobs. Lower thresholds correspond to a greater interest in jobs of a type; while higher thresholds correspond to a lesser desire for jobs of some type. When a job arrives, it broadcasts a stimulus, S_j, with a magnitude proportional to the time spent waiting for queue assignment, and continues to do so until it is assigned a queue. If an agent responds, the job is added to its queue. An agent w responds

with probability:

$$P(\theta_{w,j}, S_j) = \frac{S_j^2}{S_j^2 + \theta_{w,j}^2}. \quad (1)$$

An agent tends to respond to job types for which its response threshold is lower; but will respond to jobs of other types if a high enough stimulus is emitted.

If more than one agent responds to a job, the winner is chosen at random in WaspsV1; while in WaspsV2 the winner is chosen stochastically by a tournament of wasp dominance contests (see [17, 18] for model of wasp dominance hierarchies, or [6] for a stochastic sampling algorithm derived from the dominance hierarchy behavior). Define the force F_w of an agent w as: $F_w = 1.0 + T_p + T_s$, where T_p and T_s are the sum of the processing and setup times of all jobs in w's queue. A dominance contest between two competing agents determines which gets the job. Let F_1 and F_2 be the force variables of wasps 1 and 2. Wasp 1 gets the job with probability:

$$P(F_1, F_2) = \frac{F_2^2}{F_1^2 + F_2^2}. \quad (2)$$

Shorter queues have a higher probability of getting the job. If more than two wasps compete, a single elimination tournament of dominance contests decides the winner.

The thresholds $\theta_{w,j}$ vary in $[\theta_{min}, \theta_{max}]$. An agent updates the response thresholds according to the state of its machine. If it is processing type j or is setting up for type j, then $\theta_{w,j}$ is updated with: $\theta_{w,j} = \theta_{w,j} - \delta_1$. If it is processing or setting up for a type other than j, then $\theta_{w,j}$ is updated with: $\theta_{w,j} = \theta_{w,j} + \delta_2$. These rules imply that agents prefer to specialize in one job type. By decreasing the response threshold of the type it is processing, it increases the probability of taking on more jobs of that type. A third rule is invoked when a machine is idle. If idle, then for all types j the response thresholds $\theta_{w,j}$ are adjusted according to: $\theta_{w,j} = \theta_{w,j} - \delta_3^t$, where the exponent t is the time the machine has been idle. To summarize, agents prefer to specialize in one (or a few) job type(s), but also dislike being idle. If specializing results in a large amount of idleness, then the machines accept jobs of other types.

3 Game Theoretic Analysis of Wasp Inspired Coordination

The players of the game are the agents of WaspsV1 and WaspsV2. The set of strategies per player (possible actions) is infinite since the response thresholds are values in a continuous range. To simplify the analysis, we consider a reduced strategy set:

- A: Always respond to job type A and never to any other type. This corresponds to a response threshold for type A of 0 and thresholds for all other types of ∞. Recall that the response thresholds are strictly positive and bounded in some finite range, but for ease of analysis this has here been relaxed.
- B: Always responds to jobs of type B and never to jobs of any other type. This corresponds to a type B threshold of 0 and all other response thresholds of ∞.
- AB: Respond to both type A and type B jobs. Both response thresholds are 0.
- NONE: Never responds to any job and remain idle indefinitely. This corresponds to all response thresholds having values of ∞.

In computing the payoffs for this game-theoretic model, we consider the following preferences of the wasp-like agents:

- Idleness: The agents do not like being idle. This derives from the idle-machine update rule. If an agent finds itself idle, it reduces all response thresholds, thus increasing the probability of taking on any jobs. Therefore, if a strategy results in idleness, there is a penalty in the payoffs.

- Unprocessed jobs: The agents do not like when jobs are not processed. This derives from the continuously increasing stimulus emitted by a job until it is assigned to a queue. Eventually, given finite response thresholds and enough time, a job will emit a stimulus great enough to be selected by some machine. Once this occurs, its threshold for that type will decrease. If there are only two machines and they play (A, A), and if there exists at least one type B job, then eventually this type B job will force at least one of the machines to play either B or AB (and similarly for the playing of (B, B)). To account for this dislike of unproductive behavior, the payoffs assign large negative penalties to all players if at least one job is not processed.

- Speciality: Agents prefer to specialize to as few types as possible (without violating the dislike of idleness principle). Through the update rules, agents decrease response thresholds for types they are processing and increase them for others. This increases the probability of accepting more jobs of one type than another. If no machines have specialized (i.e., all play AB), they receive a payoff penalty.

The game-theoretic analysis that appears here considers two example problems. The first consists of two identical machines (M1 and M2) and two job types (A and B). The time required to process either type on either machine is 1 time unit, and to setup either machine for type A if the previous job is B is 2 time units (and likewise for setting up a machine for type B if the previous job is A). The queues are initially empty. Four jobs arrive. Jobs J1 and J2 are type A; and jobs J3 and J4 are type B. All 24 possible arrival sequences of the jobs are equally likely. In the analysis, we also consider an iterated version of the four job game, in which jobs arrive in sets of 4 at some constant rate with equal proportions of types A and B.

To describe the payoff computation, we present the normal form game for this example for the agents of WaspsV1. This game appears in Figure 1(a). In computing the payoffs, all possible arrival sequences of the jobs are equally likely. Initial machine setup is considered in a manner specific to the entry in the figure.

Note entries (A, A) and (B, B). In each of these strategy combinations, jobs remain unprocessed since there is one type of job in each case that is never accepted into the queue of either machine. The agents' dislike of unprocessed jobs results in a large penalty of $-\theta_{max} = -\infty$ for both players in each case.

Next, consider (NONE, A), (NONE, B), (A, NONE), (B, NONE). In these cases, both players receive a penalty of $-\theta_{max} = -\infty$ for the unprocessed jobs. If $-\theta_{max}$ was finite, then the player that played NONE should receive an additional penalty for being idle while jobs went unprocessed. The strategy combination (NONE, NONE) results in a payoff of $-\infty$ to each player for this same reason.

Consider the remaining entries. If a player's strategy is A, it is assumed that the initial setup is type A (and similarly for strategy B). The reason is that if a player is playing A (similarly B) then the queue must only be accepting jobs of type A and if we assume that the game is played iteratively (along with that strategy) then it must be setup for A at the start of each round. In computing payoffs for entries involving strategy AB, the assumption of the initial setup depends on the opposing player's strategy. If the players play (AB, AB), it is assumed that either initial setup is equally likely. For (AB, A), the initial setup for the player playing AB is assumed twice as likely B than A. If (AB, A) is played in WaspsV1, then player M1 gets both type B jobs and players M1 and M2 on average each get 1 type A. On average, player M1 processes twice as many type B jobs as A. With the equally likely arrival sequence assumption, the final setup of an iteration is twice as likely B and thus the initial setup of the next iteration if the players continue to play this strategy combination iteratively is also twice as likely to B.

	A	B	AB	NONE
A	$-\infty$,$-\infty$	**-2,-2**	-6.2,-6.2	$-\infty$,$-\infty$
B	**-2,-2**	$-\infty$,$-\infty$	-6.2,-6.2	$-\infty$,$-\infty$
AB	-6.2,-6.2	-6.2,-6.2	-8.7,-8.7	-18,-18
NONE	$-\infty$,$-\infty$	$-\infty$,$-\infty$	-18,-18	$-\infty$,$-\infty$

(a)

	A	B	AB	NONE
A	$-\infty$,$-\infty$	**-2,-2**	-2.4,-2.4	$-\infty$,$-\infty$
B	**-2,-2**	$-\infty$,$-\infty$	-2.4,-2.4	$-\infty$,$-\infty$
AB	-2.4,-2.4	-2.4,-2.4	-6,-6	-18,-18
NONE	$-\infty$,$-\infty$	$-\infty$,$-\infty$	-18,-18	$-\infty$,$-\infty$

(b)

Figure 1: Two-player games with 4 jobs (2 type A, 2 type B). All arrival sequences equally likely. Players: machines. Strategies: A, B, AB, NONE. The NE are in bold. **(a)** WaspsV1. **(b)** WaspsV2.

All remaining entries in the payoff matrix except (AB, AB), (AB, NONE), and (NONE, AB) give each player a loss equal to the expected length of time until all jobs are processed (i.e., $-1 *$ makespan). For example, the payoff of the combination (B, A) is (-2, -2) because each machine expects to complete all jobs assigned to it by these strategies in 2 time units. The combination (AB, A) results in a payoff of (-6.2, -6.2) because M1 (the row player) expects to be finished in 6.2 time units. Although, player M2 expects to be finished in 1 time unit, it also suffers a loss of -6.2 in the payoffs. All players are penalized according to the makespan of the entire set of jobs. This is due to the wasp-like agents' preference for keeping busy (or dislike of idleness). The other entries with payoffs of (-6.2, -6.2) are computed likewise.

The payoffs for (AB, AB), (AB, NONE), and (NONE, AB) are twice the expected completion time of all jobs. This penalty is related to the wasp-like agents' preference for specialization when possible. This penalty should probably be defined in terms of δ_1 and δ_2 since these system parameters control the adaptation of specialization; but it is not clear how to do so. In all three of these cases, neither wasp has specialized when it is clearly possible.

Figure 1(b) shows the game that models WaspsV2. It is similar to that of WaspsV1 except for payoffs involving strategy AB. In WaspsV1 when two or more wasps compete for a job, the winner is chosen randomly. This rule is assumed in computing the WaspsV1 payoffs. WaspsV2 uses a more complex tie breaking rule as discussed in Section 2. A simplification of this rule is used in computing payoffs. Recall that the shorter queue has a higher probability of getting the job in a multi-wasp competition. The simplification assumes that the shorter queue gets the job with probability 1.

In the games of Figure 1(a) and Figure 1(b), there are two pure strategy NE: (A, B) and (B, A). Each has payoffs of (-2, -2). Both WaspsV1 and WaspsV2, in the more complex problem with hundreds of jobs of equal proportions of the types, converge upon behavior equivalent to playing one of the NE in an iterated game (see [3] for an empirical evaluation). To illustrate, consider the following cases:

- If either NE is played in some round in the repeated game and if both players assume the other will continue to do so, there is no incentive for the other to deviate from it. To justify the assumption that the other will continue to play its part in this NE in the repeated game, consider the NE (A, B). In this

	A	B	AB	NONE
A	$-\infty,-\infty$	**-4,-4**	-6.38,-6.38	$-\infty,-\infty$
B	**-4,-4**	$-\infty,-\infty$	-6.48,-6.48	$-\infty,-\infty$
AB	-6.38,-6.38	-6.48,-6.48	-8.69,-8.69	-18.4,-18.4
NONE	$-\infty,-\infty$	$-\infty,-\infty$	-18.4,-18.4	$-\infty,-\infty$

Figure 2: WaspsV1. Two-player game with five jobs (4 type A, 1 type B). All arrival sequences are equally likely. Players: machine agents. Strategies: A, B, AB, NONE. The NE are in bold.

NE, the threshold updating rules keeps player M1 interested in type A and uninterested in type B. The opposite is true for M2. Thus, they continue to play (A, B).

- If either (A, A) or (B, B) are played, then the increasing stimulus of unprocessed jobs eventually results in one or both players accepting those jobs. Consider the case where (A, A) is played. After some number of iterations, a type B job will emit a stimulus great enough to be selected by machine M1. During the processing of this job, the response threshold updates will decrease its threshold for type B (and increase the threshold for type A). Depending upon system parameters this will either result in M1 playing B or AB. If B, then the players begin playing the NE (B, A) and all is well. The case of (AB, A) is considered next.

- If (AB, A) or (A, AB) are played by the WaspsV1 agents, then the player playing AB will get 2 type B jobs and 1 type A job (on average). Through this extra load of type B jobs, the response threshold updates will eventually increase the type A threshold and decrease the type B threshold to the point of specialization in type B. This leads to one of (B, A) or (A, B) depending on which of (AB, A) or (A, AB) were played to begin with. This is likewise true for (AB, B) and (B, AB). If (AB, A) or (A, AB) is played by the WaspsV2 wasps, then the player playing AB will on average get less than 1 type A job to every 2 type B jobs so this former argument is stronger towards convergence upon the NE.

- If (AB, AB) is played, a similar argument as in the preceding case can be made. The effects of the response threshold updates strongly lead to specialization.

- If any wasp plays NONE, the idle machine response threshold update rule will force the machine to reduce its response thresholds and begin taking jobs of one or both types. The strategy NONE, as can be seen in the game matrices, is strictly dominated and never played by the wasps of these systems.

Consider a second example. There are again two identical machines (M1 and M2) and two job types (A and B). The processing time is again 1 time unit. The setup time is left unchanged from the previous example (i.e., 2 time units). There are five jobs. Jobs J1, J2, J3, and J4 are of type A; and a single job of type B: J5. All 120 possible arrival sequences are equally likely in computing payoffs.

The normal form for WaspsV1 is in Figure 2. Payoffs are computed in the same way as they are for the four job example. There are again two NE: (B, A) and (A, B). Both NE are Pareto-optimal and equally desirable. The problem is one of coordinating upon an NE. Empirically, WaspsV1 does just that in the problem with hundreds of jobs in a ratio of four type A to every type B (see [3]). When viewed as a repeated game, the system converges upon the repeated play of one or the other of these NE. This convergence can be explained as in the four job case above.

Of greater interest is the normal form of the five job problem for WaspsV2 (shown in Figure 3(a)). For WaspsV2 the NE are: (AB, A) and (A, AB). These are superior in global system performance over those of the WaspsV1 NE. First, in WaspsV1 (see Figure 2), if one wasp specializes to type A jobs and the other accepts either type (strategy AB), then the player playing AB gets the one type B job and also has a 50/50 chance of getting each of the type A jobs. Further, this player is assumed twice as likely set for type A as for type B since on average this strategy gives half of the type A jobs to this player in an iterated game (twice as many A jobs as B). Playing AB while the other player plays A (in WaspsV1) can result in the queue: A,B,A. Assume that the slightly less likely initial machine setup of B is the case. This sequence requires 9 time units to complete; while the other machine processes two type A jobs in 2 time units. Not every arrival sequence is quite this bad, but cases such as this are why the payoffs in the game of Figure 2 are as poor as they are for (AB, A) (i.e., (-6.375, -6.375)). Contrast this with when agents of WaspsV2 play (AB, A). As in WaspsV1, the machine playing AB definitely gets the type B job. But, with respect to type A, the choice is no longer unbiased at random. Whichever queue appears shorter at the time of a type A arrival has a greater chance of getting that job (the payoffs assume the shorter queue gets the job with probability 1 to simplify analysis). The deficient result will not occur in WaspsV2. The player playing AB in NE (AB, A) or in NE (A, AB) will only take type A jobs when it appears to benefit both players (i.e., when otherwise the other machine will be working while it remains idle). The global system performance of the two NE (AB, A) and (A, AB) of the game of Figure 3(a) is an improvement over the previous game. That is, if WaspsV2 converges upon one of these, then it outperforms WaspsV1. Empirically, WaspsV2 converges upon one of the NE in the iterated version of this game (see [4] for empirical results). To explain, consider the following cases:

- The strategy NONE is never played by the agents for the reasons described previously. It is strictly dominated (very strongly so).
- If (A, A) is played, the type B jobs will eventually give off a stimulus great enough to result in some machine accepting them. This will result in either (A, B), (B, A), (A, AB), or (AB, A). This is similarly the case if (B, B) is played.
- If (A, B) or (B, A) is played, the player that plays B will suffer from an idleness penalty given that it gets one job each round while the other player gets four type A jobs. The idle machine response threshold updates will eventually cause that machine to play AB instead, resulting in either (A, AB) or (AB, A).
- If (AB, A) or (A, AB) are played, we are at a NE. Both machines are happy. The expected completion times of the individual machines are approximately equal. That is, the WaspsV2 rule that breaks ties when both machines request a type A job, on average, keeps approximately equal queue lengths in this instance. There is no idleness penalty and no idle machine response threshold updates occur. The machine playing A is clearly satisfied since it is specialized to a single task type. The machine playing AB processes enough jobs of both types to keep the response thresholds low enough to accept more jobs of both types; while at the same time is not over-allocated type A jobs due to the WaspsV2 tie breaking rule.
- If (AB, AB) is played, specialization pressures coupled with the very low demand for job type B will shift one machine to playing A, resulting in one of the NE.

Finally, consider a change to the problem (i.e., the payoffs change) after the agents have converged upon a NE in the repeated game. Consider the case where initially we have a ratio of four type A jobs to every

	A	B	AB	NONE
A	$-\infty,-\infty$	-4,-4	**-3.8,-3.8**	$-\infty,-\infty$
B	-4,-4	$-\infty,-\infty$	-4.1,-4.1	$-\infty,-\infty$
AB	**-3.8,-3.8**	-4.1,-4.1	-7.1,-7.1	-18.4,-18.4
NONE	$-\infty,-\infty$	$-\infty,-\infty$	-18.4,-18.4	$-\infty,-\infty$

(a)

	A	B	AB	NONE
A	$-\infty,-\infty$	-4,-4	-4.1,-4.1	$-\infty,-\infty$
B	-4,-4	$-\infty,-\infty$	**-3.8,-3.8**	$-\infty,-\infty$
AB	-4.1,-4.1	**-3.8,-3.8**	-7.1,-7.1	-18.4,-18.4
NONE	$-\infty,-\infty$	$-\infty,-\infty$	-18.4,-18.4	$-\infty,-\infty$

(b)

Figure 3: WaspsV2. 5 job problem. All arrival sequences equally likely. Players: machines. Strategies: A, B, AB, NONE. The NE are in bold. **(a)** 4 type A, 1 type B. **(b)** 1 type A, 4 type B.

one type B; and after some time has elapsed this ratio changes to four type B jobs to every one type A. The payoff matrix for the first set of rounds of this game is in Figure 3(a) for the WaspsV2 agents. This example would not be as interesting for the WaspsV1 agents in that the NE are the same before and after the payoff change occurs. After the proportions change, the remaining iterations have payoffs as in Figure 3(b). There are two pure strategy NE, but this time the NE are (B, AB) and (AB, B). Figure 4 illustrates how the agents respond to this change. Figure 4a shows plots of the response thresholds over time to type A and Figure 4b shows the response thresholds over time to type B. These plots are averages of 100 runs of 5000 time unit simulations. The problem change occurs at time 2500. In the first half of the simulations the wasps have converged upon the repeated play of the NE (A, AB). When the problem change occurs at time 2500, (A, AB) is no longer a NE. Almost instantly, player two adapts to this change and begins playing strategy B to improve its payoffs. This is seen in Figure 4a where M2's threshold for type A increases. At the same time, player M1 begins receiving an idle time penalty due to the smaller proportion of type A jobs and eventually begins playing AB as is seen in the sharp decrease in M1's threshold for type B (Figure 4b). The result is that the agents converge upon the repeated play of the NE (AB, B) of Figure 3(b).

4 Discussion and Conclusions

In this paper, we used game theory to model multi-agent task allocation protocols inspired by wasp behavior. Specifically, two instances of a factory control problem have been modeled as iterated games. Each game contains multiple NE and the problem is that of coordinating upon a NE in a repeated game.

The agents under analysis converge upon the repeated play of NE in iterated games corresponding to specific problem instances. The interactions of the WaspsV2 agents appear to lead to superior global system performance in these instances as compared to WaspsV1. This is the case both in the payoffs of the NE in the game that models the WaspsV2 agents as well as in empirical system results (presented elsewhere). A tracking of the response thresholds over time show a correspondence to NE convergence.

Although in this paper game theory was used only to model and analyze the behavior of an existing

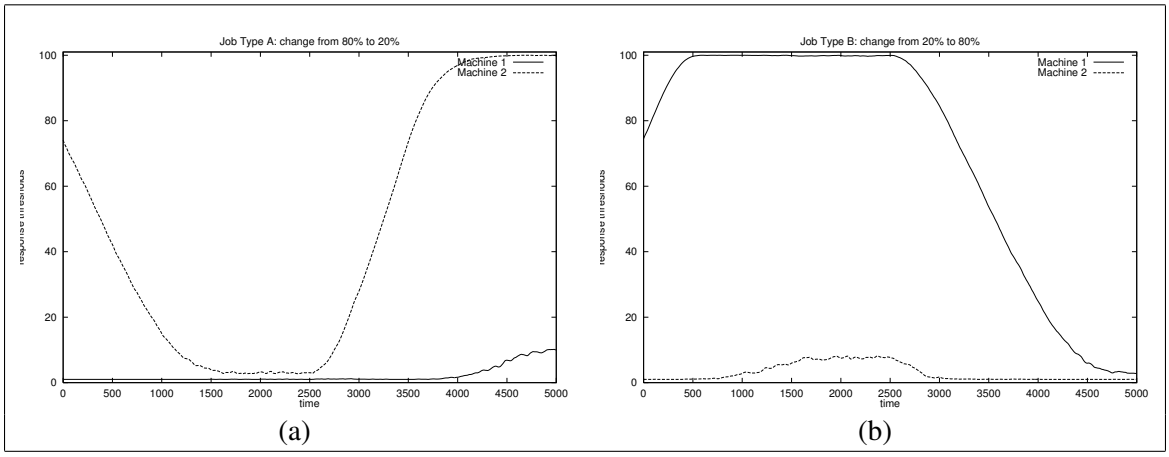

Figure 4: Average response thresholds over time of the agents of WaspsV2. Averages of 100 simulation runs for a problem consisting of jobs of two types arriving in a ratio of four type A to every one type B prior to time 2500 and a ratio of one type A to every four type B after time 2500.

multi-agent coordination protocol, potential future work can include using it during the design of such a protocol. For example, given a methodology of translating a parameterized protocol and a problem instance into a normal form game, one can then potentially choose values for the coordination protocol's parameters that lead to "better" NE in the game theoretic model.

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[2] V. A. Cicirello. A game-theoretic analysis of multi-agent systems for shop floor routing. Technical Report CMU-RI-TR-01-28, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2001.

[3] V. A. Cicirello and S. F. Smith. Wasp nests for self-configurable factories. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 473–480. ACM Press, May/June 2001. doi:10.1145/375735.376420.

[4] V. A. Cicirello and S. F. Smith. Improved routing wasps for distributed factory control. In *IJCAI-01 Workshop on Artificial Intelligence and Manufacturing, Working Notes*, pages 26–32, August 2001.

[5] V. A. Cicirello and S. F. Smith. Ant colony control for autonomous decentralized shop floor routing. In *ISADS-2001: Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems*, pages 383–390. IEEE Computer Society Press, March 2001. doi:10.1109/ISADS.2001.917443.

[6] V. A. Cicirello and S. F. Smith. Amplification of search performance through randomization of heuristics. In *Principles and Practice of Constraint Programming - CP 2002: 8th International Conference, Proceedings*, volume LNCS 2470 of *Lecture Notes in Computer Science*, pages 124–138. Springer-Verlag, September 2002. doi:10.1007/3-540-46135-3_9.

[7] V. A. Cicirello and S. F. Smith. Distributed coordination of resources via wasp-like agents. In *Innovative Concepts for Agent-Based Systems: First International Workshop on Radical Agent Concepts, WRAC-2002*, volume LNAI 2564 of *Lecture Notes in Artificial Intelligence*, pages 71–80. Springer-Verlag, January 2002. doi:10.1007/978-3-540-45173-0_5.

[8] V. A. Cicirello and S. F. Smith. Wasp-like agents for distributed factory coordination. *Autonomous Agents and Multi-Agent Systems*, 8(3):237–266, May 2004. doi:10.1023/B:AGNT.0000018807.12771.60.

[9] D. Dasgupta, editor. *Artificial Immune Systems and Their Applications*. Springer, 1999.

[10] M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*. McGraw-Hill, 1999.

[11] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.

[12] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Comput.*, 2(2):88–105, June 1973. doi:10.1137/0202009.

[13] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[14] D. M. Kreps. *Game Theory and Economic Modelling*. Clarendon Lectures in Economics. Clarendon Press / Oxford University Press, 1990.

[15] P. Kugler and M. Turvey. *Information, Natural Law, and the Self-Assembly of Rhythmic Movement*. Lawrence Erlbaum, 1987.

[16] R. D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey*. Dover Publications, Inc., New York, 1985.

[17] G. Theraulaz, S. Goss, J. Gervet, and J. L. Deneubourg. Task differentiation in polistes wasp colonies: A model for self-organizing groups of robots. In *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 346–355. MIT Press, 1991.

[18] G. Theraulaz, E. Bonabeau, and J. L. Deneubourg. Self-organization of hierarchies in animal societies: The case of the primitively eusocial wasp polistes dominulus christ. *Journal of Theoretical Biology*, 174:313–323, 1995.

[19] G. Theraulaz, E. Bonabeau, and J. L. Deneubourg. Response threshold reinforcement and division of labour in insect societies. *Proceedings of the Royal Society of London B*, 265(1393):327–335, February 1998.

[20] H. Van Dyke Parunak and S. Brueckner. Entropy and self-organization in multi-agent systems. In *Proceedings of the Fifth International Conference on Autonomous Agents*, AGENTS '01, pages 124–130, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/375735.376024.