WAIST: Wasp Inspired Scheduling for Real-Time Strategy Games

WAIST: Planeamento Inspirado em Insectos Sociais para Jogos de Estratégia em Tempo Real

Resumo

A jogabilidade dos jogos de estratégia em tempo real (RTS) parece estar confinada a um padrão onde a micro gestão económica é igualmente importante, senão mais, como a estratégia de combate. Uma possível maneira de criar jogos RTS focados na estratégia, sem o sacrifício das outras componentes chave, pode passar pela automatização das tarefas repetitivas e consumidoras de tempo. O jogador continuará a controlar a controlar tudo mas perderá menos tempo com as tarefas de micro gestão. Neste artigo apresentamos um sistema automático para o planeamento de produção de unidades, que acreditamos permitir a exploração de novos paradigmas de jogabilidade. De modo a ser aceite pelo jogador, o sistema deve, entre outras coisas, ser eficiente e robusto, o que não é uma tarefa trivial considerando a natureza dinâmica do ambiente deste género de jogos. De modo a suplantar tal desafio, propomos um sistema inspirado na inteligência colectiva demonstrada por insectos sociais, nomeadamente das vespas, e apresentamos um cenário no qual demonstramos possíveis mudanças na jogabilidade dos jogos de estratégia em tempo real.

Palavras: inteligência colectiva, planeamento dinâmico, alocação de tarefas, estratégia em tempo real

Abstract

Gameplay in real-time strategy games (RTS) seems to be somehow confined to a de facto standard where economical micro-management is equally important - if not more - as combat strategy. A possible way to create strategy-focused RTS games without sacrificing the other key aspects of the genre is to automate the repetitive and time-consuming tasks. The player will still be in control of everything but he'll spend less time executing the micro-management tasks. In this paper we present an automated system for unit production scheduling that we believe will allow the exploration of new paradigms of play. To be accepted by the player, such system must, among other things, be efficient and reliable, which is a non-trivial task when considering the highly dynamic nature of the environment in this genre of games. To overcome such challenge, we propose a system inspired in the swarm intelligence demonstrated by social insects, namely wasps, and present a scenario in which we demonstrate some possible changes to RTS gameplay.

Keywords: swarm intelligence, dynamic scheduling, task allocation, real-time strategy

1- Introduction

Analyzing the evolution of real-time strategy games over the past years it seems that its gameplay is somehow confined to a de facto standard; collect resources, construct a base camp and an army, conquer the enemy and repeat. Games as Company of Heroes (Relic Entertainment, 2006) or Ground Control (Massive Entertainment, 2000) differ from this "formula" and stand out due to its strategy-focused gameplay, which is achieved with the sacrifice of other key aspects of the genre, as economic management for example. We believe that by offering a mechanism that automate micro-management, for instance at the individual unit creation level, we will promote the exploration of novel paradigms of gameplay for such genre of games by allowing the player to focus on what's most important in a RTS game;

combat strategy. In particular, in this paper, we explore an approach that relieves the player from having to request specifically where each unit will be created, while maintaining the control on where each resource production factories will be placed in the game environment.
To earn the trust of the player and not hinder the game experience, such a system has to be efficient, robust and reliable within the unpredictable and instable environment that characterizes RTS games. To deal with the highly dynamic nature of the environment, we explore a solution based on the behaviors of social insects that despite their simplicity are capable of producing complex emergent behaviors as a colony (swarm intelligence), responding and adapting themselves to external perturbations in a de-centralized manner. In this document, we present an automated mechanism for scheduling production units, based on swarm intelligence algorithms that already proven themselves capable of handling intricate engineering problems (Colorni A., 1991). By providing such automated mechanism, we expect to change the focus of the player while playing the game and promote novel paradigms of gameplay for RTS games.
This document is organized as follows. First, in the next section, we will describe how insect behavior and particularly wasp behavior is adequate to tackle the problem of resource scheduling. Then, in section 3, we will describe the algorithm we are using to implement this behavior in Warcraft III (Blizzard Entertainment, 2002) and how we intend to evaluate it both in terms of performance and gameplay experience.

2 - Related Work

2.1 - Swarm Intelligence (SI)

SI is a term used to describe collective emergent behavior resulting from decentralized and self-organized systems. Its roots are the studies of self-organized social insects, like ants, wasps or termites (Ben-Jacob E. 2000; Hirsh A. 2001). In a colony, there is no central entity or mechanism controlling or even defining objectives, yet these simple creatures with strict sensory and cognitive limitations manage to perform complex tasks such as food foraging, brood clustering, nest maintenance and nest construction. The underlying mechanisms behind their complex behavior as whole became subject of great interest and study, resulting in innumerous natural models.

At first glance, SI may seem interesting only for biology purposes; however the problems dealt within a colony are analogous to scheduling/logistic engineering problems addressed by

Man. Quickly the SI theories created to explain insects' emergent behavior became adapted to solve such practical problems.

2.2 – Wasp Behavior

From its studies of the *Polistes dominulus* wasps, Theraulaz et al. created a model of dynamic task allocation (Theraulaz G et. al., 1990) that successfully emulates the self-organized behavior of wasps. The model consists in a wasp hive in which there are two possible tasks; foraging and brood care. Individuals decide which task to do according to their response threshold and stimulus emitted by the brood. The system has the following main features:

- Tasks have the capacity of emitting stimuli that affects the individuals' task selection decisions. (**stimulus**)
- Individuals possess response thresholds that represent their predisposition to perform certain tasks. (**response thresholds*)***
- Each individual has a force that is taken into account during dominance contests to determine the winner. Dominance contests form a hierarchy within the colony. (**force*)***
- Individuals have a set of response thresholds for each possible task. When an individual performs a task, the respective response threshold is decreased while the others are increased. This means that the more an individual performs a task the more likely he is to do it again, creating task specialists in the society. (**specialization**)

These four features provide the model with a powerful robustness. The system self-organizes itself towards optimal performance due the individuals' capacity of specialization. However this specialization isn't rigid, allowing them to dynamically adapt their work force according to the constantly changing colony needs or loss of individuals.

2.3 - Routing-Wasp (R-Wasp)

Based on the properties of the natural model created by Theraulaz et al., V. Cicirello et al. (2004) proposed an algorithm for dynamic task allocation that later was adapted to Morley's factory problem (Morley D., 1996) from General Motors, denominated as R-Wasp.

In this algorithm, each different possible job has a type and is capable of emitting stimuli according to the time they've remained unassigned. Each agent capable of performing jobs has a set of response thresholds, one for each type, that represent its propensity to bid for a job of a certain type.

For each unassigned job, agents stochastically decide to bid or not according to the respective response threshold and stimulus emitted. After the decision of all agents, if there is more than one candidate, a dominance contest occurs which consists in a tournament where duels are made until one last standing agent wins. The winner of a duel is stochastically settled taking in consideration the agent forces, which vary according to properties that make the agent more or less suited to perform the job it is competing for. The job is assigned to the dominance contest winner.

Response thresholds are updated according to the jobs an agent processes. The more an agent does a job of a type, the more the respective threshold is decreased and more likely will accept other jobs of the same type. The opposite is also true, i.e. the less it does of a type the less likely it is to accept jobs of that type. However, if the agent is idle all thresholds are gradually lowered to make sure it will accept other jobs.

The R-Wasp strength consists in two aspects. Response thresholds allow the creation of specialists for job types and gives them the ability to bid for the jobs most suited for them, without however discarding their ability to do other jobs if needed, Force permits a good distribution of the workload trough the agents by taking in consideration their characteristics for determining the job winner. The two aspects together allow the system to self-regulate and dynamically respond to unexpected events like lost of agents or variations in demand.

3 – R-Wasp in RTS games

3.1 – Scenario

The algorithm is currently implemented in the videogame Warcraft III (Blizzard Entertainment, 2002) as a system responsible for assigning units ordered by the player to the available factories. The player specifies the location and quantity of desired units and the system finds the near optimal schedule.

In order to evaluate the algorithm, a custom scenario has been set. This scenario differs from regular RTS games due to the introduction of heterogeneity in factories, i.e. factories have different production times, and also to the insertion of the concept of setup time which is an extra time required when the production changes from one type of unit to another. In this scenario:

- Two units can be produced: X and Y.
- The change of unit production from one type of unit to other results in extra setup time

- Five different factories are available, each one with different properties (Fig.1): setup time, production time and unit types produced.

The objective is to build a determined amount of units within a certain amount of time using the available buildings. Several variations will be introduced in the tests as the destruction of factories and the variation of orders distribution.

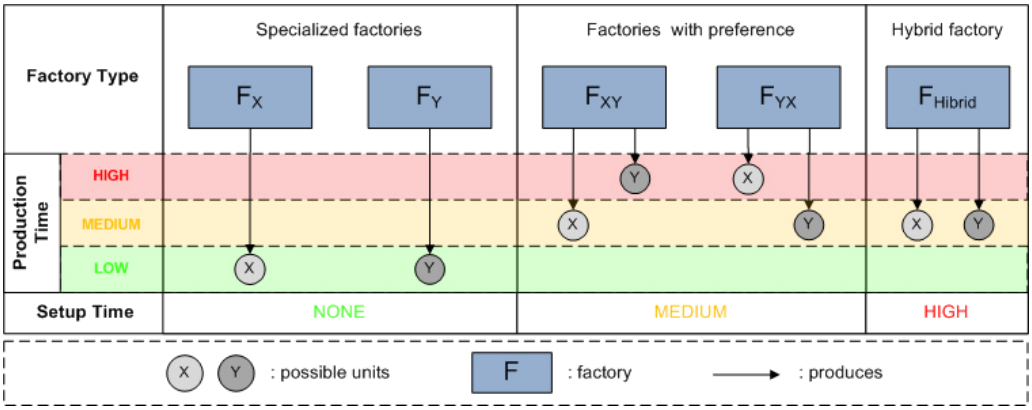

Figure 1 – Representation of the current scenario. For instance F_{XY} can produce X and Y but Y will take more time. If after producing a unit X the player orders a unit Y, it will be required a medium setup time before the production of Y may begin.

3.2 – Model

The algorithm currently implemented is the following:

```
1  list of unassigned orders
2  list of factories
3  list of candidates
4  foreach unassigned order
5          foreach factory
6                  decide to bid or not                    (1)
7                  if(bid)
8                          candidates ← factory
9          dominance contest between candidates            (2),(3)
10         assign order to the winner
11         update forces                                   (4)
12 update thresholds                                       (5)
13 update stimuli                                          (6)
```

Figure 2 – Pseudo code of the algorithm. The marked steps are detailed in the text.

Each factory capable of producing units corresponds to a wasp capable of executing jobs and each ordered unit corresponds to a job, being its type defined by the unit type. A factory *f* will bid for an unassigned job *j* with probability given by (1):

$$(1) \qquad P\left(bid \mid T_{f,j}, S_j\right) = \frac{{S_j}^2}{{S_j}^2 + {T_{f,j}}^2}$$

$T_{f,j}$ stands for the threshold of the factory for that type of job and S_j is the stimulus emitted by the job. If more than one factory applies to a job, the winner is selected after a tournament of dominance contest where its forces are taken in consideration. If an odd number of factories apply, N factories will automatically pass to the second phase where N is given by (2) and C stands for the number of competitors. The probability of factory *f1* winning factory *f2* in a duel is given by (3):

(2) $N = 2^{\lceil log_2(C) \rceil} - C$ (3) $P(f_1\ wins \mid F_1, F_2) = \frac{{F_2}^2}{{F_1}^2 + {F_2}^2}$

where F_1 and F_2 represent the forces of factory 1 and 2 respectively. The force of a factory is given by (4):

(4) $F_{factory} = T_{processing} + T_{setup} + T_{delivery}$

$T_{processing}$ is the sum of all the production times of the job currently being processed and all jobs in its queue, T_{setup} is the sum of all setups needed and $T_{delivery}$ is the time needed for the unit of the job being disputed to reach its destiny location.

Every cycle the response thresholds are updated according to the following criteria:

(5) $T_{f,j} = T_{f,j} - \delta_1$, if the last job in factory *f* queue is of type *j*

$T_{f,j} = T_{f,j} + \delta_2$, if the last job in factory *f* queue is not of type *j*

$T_{f,j} = T_{f,j} - \delta_3$, if factory *f* is idle

δ_1 actuates as a learning coefficient since it diminishes the threshold and encourages the factory to take jobs of the same type. δ_2, oppositely, actuates as an unlearning coefficient. δ_3 is used to encourage the factory to take any job if it is idle, by decreasing all thresholds.

Job stimuli are also updated every run time cycle (6) according to the time they have remained unassigned, granting that they never remain unassigned for a long time.

(6) $S_j = S_j + T_{unassigned} \times S_j$

This model possesses the following system parameters: T_{min} , T_{max}, δ_1, δ_2 and δ_3. They must be established with the aid of experimental simulations in order to determine the optimal values and how they affect the system performance.

3.3 – Evaluation

The evaluation will be made in a first phase using scripted actions in order to obtain results faster and tweak the system parameters. In the second phase tests will be made using human players, not only to evaluate their performance but also their satisfaction with the system, and consequently gameplay.

To evaluate the performance of the R-Wasp algorithm, the following metrics will be taken into consideration: time taken to execute all tasks; resources spent and number of setups required. The results will be compared against other strategies, such as, random order assignment for factories and order assignment based on factory's location.

4 - Final Remarks

In this paper, a new paradigm for scheduling resource in RTS games was presented, inspired by the social behavior of wasps. The properties of a natural model of wasp behavior related to task allocation was described and served as inspiration for the scheduling algorithm R-Wasp. Finally, a scenario in which the approach will be evaluated was presented. In the future, we intend to explore how such a paradigm change may enable new gameplay in such genre of games and evaluate the approach with human players.

References

Ben-Jacob E., Cohen I., Levine H. (2000) Cooperative self-organization of microorganisms, Advances in Physics, Volume 49, Issue 4, pp. 395 – 554

Cicirello V., Smith S. (2004) Wasp-like agents for distributed factory coordination, Journal of Autonomous Agents and Multi-Agent Systems, 8:pp.237–266

Colorni A., Dorigo M., Maniezzo V. (1991) Distributed optimization by ant colonies, Proceedings of ECAL'91, European Conference on Artificial Life, Elsevier Publishing

Hirsh A., Gordon D. (2001) Distributed problem solving in social insects, Annals of Mathematics and Artificial Intelligence, v.31 n.1-4, pp. 199-221

Morley D. (1996) Painting trucks at general motors: The effectiveness of a complexity-based approach, in Embracing Complexity: Exploring the Application of Complex Adaptive Systems to Business, The Ernst and Young Center for Business Innovation, pp. 53–58

Theraulaz G., Goss S., Gervet J., Deneubourg J.L. (1990) Task Differentiation in Polistes Wasp Colonies: A Model for Self-Organizing Groups of Robots, In: J.A. Meyer and S.W. Wilson, Editors, 1st Int'l Conf. on Simulation of Adaptive Behavior, pp. 346–355