# Building Trading Agents: Challenges and Strategies

**Maria Fasli**[1] and **Ioannis Korres** and **Michael Michalakopoulos** and **George Rallidis**

**Abstract.** With the advent of the Internet, trading in electronic market places has become common practice for an increasing number of businesses and individuals. One of the most efficient ways of negotiating for goods and services is via auctions. Although implementing an agent to take part in an auction for a single good is relatively simple, developing an agent to participate in simultaneous auctions offering complementary and substitutable goods is a complex task. In this paper we discuss the challenges of the Trading Agent Competition which features a complex benchmark e-market problem and we present the strategies of two agents that participated successfully in the competition, one of them being placed among the eight finalists.

## 1 INTRODUCTION

Agent-based and multi-agent systems have become increasingly popular as a means of conceptualising and implementing a wide range of applications. More recently, with the advent of the Internet there has been a mounting interest regarding their utilisation in e-commerce applications. Agent technology being particularly suited for information rich environments can be applied in some of the stages involved in searching and buying products and servises [4]. For instance, semi-autonomous and personalised software agents can be used in the negotiation stage in order to automate the trading process. This has several potential benefits such as reduced costs and greater efficiency as well as savings in time and effort.

One of the most efficient ways of negotiating for goods and services is via auctions [5]. Although constructing an agent to take part in an auction for a single good is relatively simple, developing an agent to participate in simultaneous auctions offering complementary and substitutable goods is a complex task. This is the form of the problem tackled at the International Trading Agent Competition which features artificial trading agents competing against each other in a market-based scenario. The 2nd Trading Agent Competition and Workshop (TAC-01) was held in Tampa, Florida on October 14, in conjunction with the 3rd ACM Conference on Electronic Commerce.

In this paper we discuss the challenges presented by the benchmark problem of TAC-01 and the implementation of two trading agents that addressed those challenges and participated in the event. Both agents participated in the Preliminary Tournament of the competition and achieved good results. One of them qualified and participated in the Final Tournament obtaining a place among the eight finalist agents. The structure of the paper is as follows. In the next two sections we discuss the benchmark problem of TAC-01 and the challenges that it presents for building effective trading agents. The next two sections describe the strategies and implementation details of the two agents. We then present the competition's results and the paper ends with the conclusions and a pointer to future work.

[1] University of Essex, Department of Computer Science, Wivenhoe Park, Colchester CO4 3SQ, UK, email: mfasli@essex.ac.uk

## 2 THE TAC SCENARIO

The Trading Agent Competition is an open-invitation event featuring software agents from all over the world competing in a challenging market game. The aim of the competition is to stimulate research in trading agents with an emphasis on developing a successful strategy for maximizing profit in a constrained environment. The competition is based on the infrastructure of the AuctionBot server [10]. A TAC game (instance) lasts 12 minutes and involves eight agents competing against each another. Each agent acts as a travel agent whose objective is to create travel packages from TACtown to Tampa during a notional 5-day period for eight clients. In order to do so, the agent has to secure the necessary resources which consist of airline tickets, hotel reservations, and optionally some tickets for entertainment events, all being traded simultaneously in electronic auctions. Each one of those clients has its own individual preferences over the various aspects of the trip. The agents' objective is to construct travel packages as close as possible to their clients' preferences while at the same time minimizing expenditure. A travel package is feasible if it contains an in-flight ticket, an out-flight one and a hotel room for each day between the in-flight and out-flight days.

There are two types of airline tickets, flights to Tampa (in-flights) and flights from Tampa (out-flights). There are 4 single-seller auctions for the in-flight tickets (days 1-4) and 4 for out-flight tickets (days 2-5). There are no in-flights on day 5 and no out-flights on day 1. TACAir is the only provider of airline tickets whose availability is considered unlimited. In the beginning of the game the tickets are priced randomly between \$250 and \$400. Their price changes periodically (every 24-32 seconds) by a random value between -\$10 to $x(t)$. $x(t)$ is a linear function of the game time (0:00 to 12:00) and its value is always between [10,90]. This means that the ask price for airline tickets is expected to increase as the game progresses motivating agents to buy airline tickets early. The ask price is guaranteed to stay between \$150 and \$800. The flight auctions close at the end of the game and no resale of airline tickets is allowed.

There are two hotels available, the "nice" Tampa Towers (*TT*) and the the "not so nice" Shoreline Shanties (*SS*). Because of the difference in quality the price of the former is expected to be higher than that of the latter. There are 4 auctions for each hotel (days 1-4) and each hotel has 16 rooms available for each day. The rooms are traded in 16-th price English (ascending) auctions. This means that the 16 highest bidders win the rooms at the 16-th highest price. The agents have no knowledge of the other bids, their only information is the current ask price. Four minutes after the start of the game one hotel auction clears and closes. One (random) hotel auction closes every minute thereafter. Bid withdrawal or ticket resale is not allowed.

There are three types of entertainment events: alligator wrestling, amusement park and museum. Tickets are available from days 1 to 4, and they are traded in twelve auctions in total. These auctions are double auctions, that is the agents can act as both sellers and buyers [9]. Clearing takes place immediately when a match occurs and the remaining bids are left active in the auction. The auctions close when the game ends. Bid withdrawal and ticket resale are permitted.

When the game starts each agent is assigned a random set of preferences for the eight clients as well as a random endowment of entertainment tickets. The preferences consist of the preferred arrival day (*PA*) between days 1 and 4, the preferred departure day (*PD*) between days 2 and 5, the hotel premium value (*HP*) or bonus between \$50 and \$150, and entertainment values (*EV*) between \$0 and \$200 for each type of entertainment ticket. The hotel premium value represents the bonus that the agent will receive if the client is placed in the nice hotel. No hotel swapping is allowed during a client's stay.

The top scoring agent is the one that achieves the highest sum of the individual client utilities while minimizing the expenses of the goods bought. The client utility, measured in dollars, for a feasible travel package is computed by the formula:

*Utility=1000-TravelPenalty+HotelBonus+FunBonus*, where
*TravelPenalty=100 * (| AA-PA | + | AD-PD |)*$^2$
*HotelBonus*= *HP* if client is staying at *TT*, 0 otherwise.
*FunBonus=EV1+EV2+EV3*

If the travel package is not feasible the client is assigned zero utility. When the game ends the TAC server allocates the goods the agent holds to its clients in order to construct feasible packages and a score is computed for each participant agent. The server tries to make an optimum allocation in order to maximize the clients' utilities. A more detailed description of the game can be found in [7].

## 3 CHALLENGES

There are several challenging issues in designing and implementing a trading agent for the TAC market scenario. Finding optimal solutions via a brute force is computationally intractable. A more sophisticated approach, which may include an advanced search algorithm and an effective bidding strategy, is therefore required in order to succeed.

To achieve long-term success, the bidding strategy should take into account not only current prices, but an estimation of the future prices as well. Unfortunately, accurate price estimation is very difficult, if not impossible, because of the limited information the game provides. The only information that is available to the agent is the current asking prices. The agent knows nothing about the other agents' bids or ticket demand. Standard machine learning techniques are very hard to apply since the agent cannot observe the bidding patterns of the other participants.

Although the agents are not required to perform the final allocation of goods to their clients and report it back to the server, a major issue in TAC-00 [6], [1], the allocation problem is relevant here as well. What is referred to as the *completion problem* [3] can be stated as follows: Given the current ticket holdings, the current market prices (or maybe the estimated prices) and the client preferences find the most profitable combination of tickets that need to be traded in order to complete travel packages. The completion problem is a hard one since the number of the available tickets is constrained by the availability in the market and also the (estimated) prices need to be taken into account. This problem is classified as NP-complete. The optimum solution is not tractable via a brute force approach and thus sophisticated search algorithms have to be employed. The computation time of the algorithm is vital.

Having already decided on the optimum packages, that is what tickets and how many are needed, the problem is to decide when and how much to bid. The agent may also consider bidding for tickets not currently included in the optimum list to maintain flexibility later in the game. The bidding strategy can be divided into three parts, the strategy for obtaining each of the three available resources, flight, hotel, and entertainment tickets respectively.

The key resources, as it may have already become obvious from the game description, are the hotel reservations, since they are vital for the construction of feasible travel packages. Their limited availability makes them the most sought-after goods. Failing to acquire a particular hotel reservation may jeopardize the whole travel package. Things get even worse when the rest of the tickets have already been secured. Hotel auctions are ascending-price ones, and consequently once the price goes up there is no way back. An obvious strategy would be for the agent to place its bids at a high price, say \$1000. In this case, it is most probable that the agent will win the rooms at a price equal or lower than the bid price. An agent that bids according to this strategy does not have to monitor the hotel auctions in order to update its bids. If the bid is rejected this implies that the ask price is much higher and therefore the reservation will not be beneficial for the agent. Such a strategy has the drawback that the agent cannot change its initial decision since bid withdrawal is not allowed.

The limited availability of hotel rooms creates a further dilemma. Since it is almost certain that the prices of the flight tickets will increase as time passes, the expenses will be minimum for an agent that buys all the wanted tickets in the beginning of the game. This would be optimal in a game that the demand of hotel rooms is lower than the supply, for all the hotel auctions that an agent participates in. On the contrary, if the demand is higher than the supply (this is the case for most games), this strategy involves high risk. Buying the flight tickets in the beginning also means committing early on packages, and thus having less flexibility to move a client's arrival or departure days later in the game. The agent may fail to reserve some of the hotel rooms needed to assemble the travel packages, and thus additional flight tickets may have to be purchased later in the game in order to acommodate changes. This entails that some of the initially purchased flight tickets will remain unused while the expences will increase. On the other hand, an agent could purchase the flight tickets for each client when all the hotel rooms for this client are reserved, but this means that the flight ticket prices may be significantly higher, again leading to increased expenditure.

Although the entertainment tickets are not as important as the other resources, the additional bonus that can be obtained can make the difference and distinguish a good agent from a not so good one. These are the only goods that can be both bought and sold. Sometimes it is more profitable to sell a ticket than to assign it to a customer, a fact that adds to the complexity of the bidding strategy.

Last but not least, the agent has to take into consideration network disruptions and delays, a typical phenomenon for network applications, and make sure that its bids arrive on time.

## 4 AGENT *TYPHON*

Agent *Typhon* was implemented in *Delphi* and is an optimal agent, that is the agent always tries to achieve the best possible score given the situation. This agent was inspired by *RoxyBot* which was one of the most successful agents in TAC-00 [3], [1]. The high level strategy illustrated in Figure 1 which has been used by most TAC agents was followed for *Typhon*.

The key feature of this agent is the quality of the solution to the completion problem as described earlier. The agent is required to know at any given point the most profitable (or at least a very good)

[2] *AA* and *AD* are the actual arrival and departure days respectively.

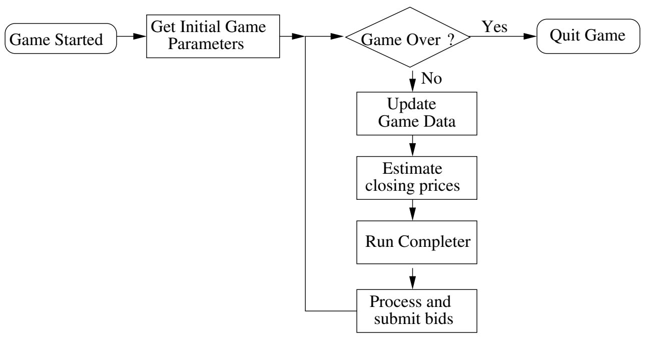


**Figure 1.** *Typhon*'s high level strategy.

combination of the goods to be traded. Exhaustive search, unfortunately, is not an option as the possible combinations are just too many. The agent can afford only a few seconds of searching, typically 2-3 seconds at most, and ideally below 1 second. A variation of the A* algorithm was implemented with a restriction on the maximum number of "open" nodes kept in order to achieve a good solution/computation time ratio. The algorithm has as input the clients' preferences, the market prices and the current ticket inventory. The output is a recommended travel package for each of the clients. The algorithm executes in two phases. In the first one it searches for flights and hotels and in the second one for entertainment events. The algorithm does not search for single tickets but for ticket packages. This is reasonable, since the goods are valuable only in combinations.

Each travel package consists of three numbers, the arrival day (between 1-4), the departure day (between 2-5) and the hotel type (0 or 1). There are 21 different packages including the null package. For instance, a travel package has the form (1,3,1), which means arrival on day 1, departure on day 3 and stay at the good hotel. An entertainment package consists of four numbers. The first number indicates the entertainment type (between 0-3) on day one, the second the entertainment type for day 2 and so one. Zero value means no entertainment event will be assigned for that day. A travel package cannot have two events of the same type. There are 73 different entertainment packages including the null package. For instance, the package (0,0,3,1) represents that the client will be assigned an entertainment of type three on day 3, an entertainment of type one on day 4 and no events for days 1 and 2.

The algorithm starts execution from one initial state (node) where no travel packages (TP) are assigned and proceeds by assigning TP to the clients and evaluates the utility it gets. Each time a TP is assigned, a different node is generated. Each node is evaluated by computing 3 numbers: $g$, $h$ and $f$. $g$ is the utility that has been obtained so far in the tree search (including the current node) minus the cost so far. $h$ is the utility that is expected to be obtained from the rest of the search. Finally, $f$ is the sum of $g$ and $h$ and constitutes the fitness value, which tells the algorithm how good the node is. $h$ is the heuristic function and is computed as follows: for each of the remaining clients the algorithm assumes that the maximum possible utility will be obtained from the travel part *(1000+HP)*. Then it counts the tickets left in the inventory to see if it has the necessary resources to satisfy the clients left. If there are not enough resources, the algorithm subtracts a cost for each ticket missing. If the ticket is an in-flight one, the cost is that of the cheapest in-flight ticket in the game at that time. The same applies to other missing resources (out-flight and hotels). The heuristic function is highly optimistic as it assumes that the tickets needed will be bought at the minimum possible cost, and also the maximum utility will be obtained. Notably, already purchased tickets are treated as a sunk cost (their cost is set to zero).

The algorithm keeps two lists of nodes. The open list, which contains the nodes that have not been searched yet and the closed list where all the nodes have been already searched. The high level structure of the algorithm is presented below:

```
Repeat
    1. get the best node from the open list (maximum $f$)
    2. add it to the closed list
    3. expand it
        i. generate all successors
        ii. compute $f$ for each one
        iii. add them to the open list
until the search reaches depth 8 (all clients have been considered)
```

As expected the algorithm requires too much time to perform the full search. The solution given to this problem is simple. Each node can only have a limited number of successors instead of keeping all of them (21 in total). Thus, although the algorithm generates all of them, it finally keeps the best and discards the rest. Of course by doing so the optimality of A* is gone, but this was necessary. The best number of successors to be kept was found empirically to be two. Keeping more does not improve the quality of the solution, but it does increase the computation time exponentially.

The bidding strategy is determined by the output of the completion algorithm. Once the completer assigns one travel package and one entertainment package to each customer, the list of tickets that need to be traded to satisfy the packages is generated. A maximum price and a suggested one is computed for each ticket.

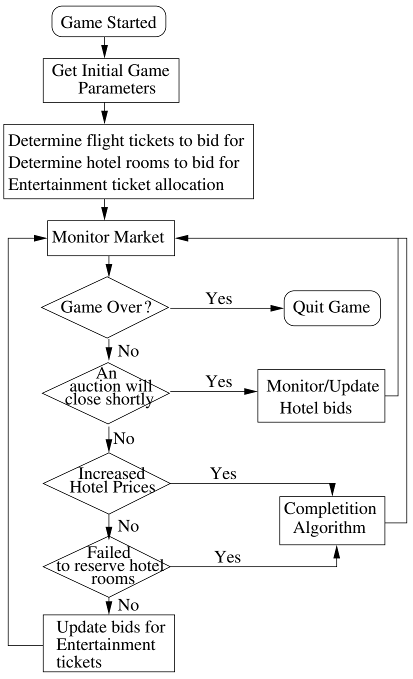
The agent computes a number, which expresses the probability to stick with the current package for the client. The probability depends on the total days of stay, the hotel (good or bad) and the days (it is easier to secure rooms for days 1,4 than for days 2-3). It may buy one flight for customers with average probability and no tickets for the rest of clients are purchased. The probability thresholds are parametrically defined. The agent waits until the game reaches 4+ minute, when the situation with the hotel rooms is supposed to be clearer, to buy the rest of the tickets. The agent buys any additional airline tickets needed immediately after the 4th minute.

The agent starts bidding with some delay in order to avoid affecting the hotel prices and help them stay at lower levels. It starts bidding at the 3rd minute, 1 minute before the first hotel auction closes, and usually has two bidding opportunities/rounds before the closing: it bids passively in the first round, but more aggressively in the second one in order to avoid loosing too many tickets when the first auction closes. The agent bids initially for both hotels but after the 4th minute it sticks with the hotel proposed by the completer. The agent bids also for hotel rooms not included in the clients' packages attempting to buy rooms cheaply and use them if it later fails to secure the desired rooms. The bids increase over time.

The agent both buys and sells entertainment tickets according to the completer's output. If there are tickets marked for selling, the agent starts with a very high asking price. The price decreases linearly as the game progresses. The agent sells unwanted tickets at their minimum price (set to $40) after the 11th minute. It does not accept to sell tickets for less than $40 and help its opponents improve their scores. For wanted tickets the agent starts with a low price and gradually increases it as the game progresses. After the 11th minute of the game the agent bids its maximum value, which is set at 90% of the bonus value for the ticket. The agent withdraws bids for tickets for which it is not interested any more.

## 5 AGENT *CAISERSOSE*

Agent *CaiserSose* was implemented in *Java* and its high level strategy is described in Figure 2.



Game Started
Get Initial Game Parameters
Determine flight tickets to bid for
Determine hotel rooms to bid for
Entertainment ticket allocation
Monitor Market
Game Over?
Yes
Quit Game
No
An auction will close shortly
Yes
Monitor/Update Hotel bids
No
Increased Hotel Prices
Yes
Completition Algorithm
No
Failed to reserve hotel rooms
Yes
No
Update bids for Entertainment tickets

**Figure 2.** *CaiserSose*'s high level strategy.

In the beginning of the game the agent only holds the entertainment tickets allocated by the server. In this *initial* state, given the fact that the preferences of all clients are known, it determines what flight and hotel tickets it needs to obtain.

The agent determines the flight tickets needed for each client, according to his preferred arrival and departure date. It also determines which flight tickets are likely to reach a high price (days 2-4) in the beginning of the game and buys those first. The rest are purchased later in the game. This provides for the flexibility to shorten the stay of some clients, in case the agent fails to reserve some hotel rooms. Although this leads to a penalty of 100 points for each day that the vacation is shortened, it may be preferable since hotel room reservations might reach a price as high as $1000. The flight bids are placed at a price a little higher than the ask price in each auction.

The agent decides which type of hotel will be reserved for each client according to the following rules:
• A client that arrives in Tampa on day1 or day4 and leaves on day2 or day5 respectively, will be accommodated in *TT*.
• A client that will stay in Tampa for only one day (but different than the above) will be accommodated in *TT* if he offers a bonus $>$ \$80. In the opposite case, he will be accommodated in *SS*.
• All clients who will stay in Tampa more than one day will be accommodated in *SS*, unless they offer a bonus $>$ \$120 for *TT*.

These decisions do not lead to the highest score since not all clients get the *TT* hotel. The agent attempts to balance the demand for hotel rooms for *TT* and *SS*. The hotel bids are placed at a price a little higher than the ask price in each auction.

The allocation of the initial endowment of entertainment tickets to clients and the decisions of which to buy/sell are taken as follows:
• Tickets for which no client offers bonus will be put up for sale.
• Tickets for which clients offer bonus $<$ 100 will be put up for sale for a price $>$ 100. If they are not sold, then they will be reallocated.
• Tickets for which clients offer bonus $>$ 100 will be held.
• If clients offer high bonus for tickets not in endowment, the agent will attempt to purchase those tickets at price $<$ bonus.

The buy bids for the entertainment tickets are initially placed at a low price whereas the sell bids at a high price. As time passes the price of the buy bids is increased and the price of the sell bids is decreased in order for the desired transactions to take place. The agent must consider that by selling or buying a ticket will help another agent make profit. If agent A sells a ticket to B for $10 and B gains $100 by assigning this ticket to one of its clients, it is obvious that the benefit is much greater for agent B. Thus, thresholds are set regarding the lowest selling and buying prices that the agent is willing to accept. Moreover, it seems reasonable to attempt to sell an entertainment ticket for which no client offers bonus. In some cases though it is preferable to sell a ticket even if some bonus can be gained from it. For example, if a client offers $70 for a certain ticket but it is possible to sell it at a higher price, the agent should try to do so.

In order to improve the agent's performance, an event-driven approach was adopted. In particular, the agent constantly monitors the hotel auctions until the game is over. Throughout the game, the agent is aware of whether or not a hotel auction is bound to close shortly. This is achieved by the use of an internal timer. The agent distinguishes between two cases (Figure 2):

1. A hotel auction will close shortly. In this case the agent monitors its hotel bids until the auction closes. The goal is not to be overbid at the last moment. If this happens, the agent places the bids again at a price higher than the current ask price, no matter what the ask price is. Since most agents follow the same strategy and place their bids just before an auction is about to close, there is a chance that the agent will not manage to update its bids, especially if there are network delays.
2. No hotel auction will close shortly. The agent gets information about the transactions that have taken place until this moment, as well as the current ask prices for the auctions. There are three cases that guide the agent's behavior:

    - There is an auction in which the agent participates and for which the ask price has reached the price of the agent's bid. In this case the completion function is called to deal with this situation (see below).
    - The agent has failed to reserve the wanted amount of rooms in an auction that just closed. The completion function is called.
    - If none of the above is the case, the agent updates the bids for the entertainment tickets. Firstly, it reruns the entertainment tickets allocation algorithm in order to determine what tickets need to be purchased/sold. This solves the problem of tickets remaining unused if the vacation of some clients is shortened. In this case the entertainment tickets allocation algorithm acts as a part of the completion algorithm. In addition, the agent lowers the prices for the sell bids, and increases the prices for the buy bids. The constraints here are that the agent will not sell a ticket for less than $50 and will not buy a ticket at a price higher than the bonus the client offers minus $30.

The objective of the agent is to reach a *final* state in which it has obtained all the goods that it initially bid for. However, in such a game it is highly unlikely that the agent will reach the final desirable state without having to reconsider. The completion algorithm has to determine a new *final* state that the agent will try to reach during the game. The agent needs to allocate the current holdings of goods, as well as the hypothetical quantities of goods that might be purchased to the eight clients, in a way that leads to an optimal score. At this point, solving this problem is much more difficult than it was in the beginning of the game, since some hotel auctions might have closed or reached a high ask price.

The algorithm implemented for *CaiserSose* is rule-based and uses greedy heuristics in order to find an acceptable solution that will decrease the expenses for the agent. This is achieved by switching clients from one hotel to the other, and by shortening the stay if possible. Each client is considered individually. The agent checks if it is possible for the client to get the *TT* or *SS* hotels. This is possible if either all the hotel auctions pertaining to this client's trip are still open or, although some of them are closed, enough hotel rooms have already been reserved so that it is possible to allocate one of them to this client. Four cases are considered:

1. It is possible to accommodate the client in both hotels. The agent calculates the expenses for each hotel and chooses whatever is more beneficial, considering of course the bonus for the *TT* hotel.
2. The client can only be accommodated in one type of hotel. In this case, this type of hotel is the only choice for the agent.
3. This client cannot be accommodated in any hotel without shortening its vacation. In this case the agent explores ways of shortening the client's stay and calculates the cost of the hotel rooms as well as the flight ticket(s) that must be now purchased. The most beneficial option is then chosen.
4. If the flight tickets for this client haven't been purchased yet, even if the agent is able to offer a certain type of hotel (cases 1 and 2), the completion algorithm will still consider shortening his vacation. This is done when it is more beneficial to receive a penalty than to reserve one or more expensive hotel rooms.

After the decisions are made for all clients the agent calculates the hotel rooms needed for each day and type of hotel and places the bids at a price a little higher that the current ask price of each auction. As for the flight tickets needed, the agent will bid for those tickets after the 9th minute of the game. The completion algorithm does not consider entertainment tickets at all. The reason is that the main concern is to determine which type of hotel should be reserved for each client. This does not interfere with entertainment tickets, since the client will still be allocated the same entertainment tickets independent of type of accommodation.

## 6 RESULTS AND CONCLUSIONS

In total twenty-eight teams, representing academic institutions, research centers and companies from ten different countries entered the competition [8]. The competition itself was organised into a preliminary tournament and a final tournament [2]. Participation in the preliminary round was open to all agents and its purpose was to determine the sixteen agents that would proceed to the final tournament as well as their division in two "heats". *Typhon* and *CaiserSose* performed well and achieved the 18th and 15th places respectively, and *CaiserSose* qualified for the finals[3]. The final tournament was held in Tampa where the participants and agent developers had the opportunity to present their work. The heat rounds determined the eight agents that would compete in the final rounds. The final ranking of the eight finalists along with their affiliations is given in Table 1. Livingagents developed by the German software company Living Systems AG achieved the highest average score. The agents developed by AT&T Research and Cornell University followed in the lead. *CaiserSose* performed very well and it was placed 7th among the eight finalist agents.

[3] Note that according to the TAC rules, even if two agents from the same institution qualify, only one is allowed in the finals.

**Table 1.** Final ranking in the Trading Agent Competition 2001.

| Pos. | Agent | Affiliation | Avg Score |
|---|---|---|---|
| 1. | Livingagents | living systems AG | 3670.0 |
| 2. | ATTAc | AT&T Labs - Research | 3621.6 |
| 3. | Whitebear | Cornell University | 3513.2 |
| 4. | Urlaub01 | Penn State University | 3421.2 |
| 5. | Retsina | Carnegie Mellon University | 3351.8 |
| 6. | SouthamptonTAC | University of Southampton | 3253.5 |
| 7. | CaiserSose | University of Essex | 3074.1 |
| 8. | TacsMan | Stanford University | 2859.3 |

We hope to be able to participate in future TACs and towards this direction there are a number of possible avenues for future development. One option is to integrate the two agents combining the A* approach of *Typhon* and the event driven behaviour of *CaiserSose*. This was not possible in TAC 2001 since the close proximity of the preliminary and final tournaments only allowed for fine tuning and not major changes in an agent's strategy. Another direction is to tackle the problem using a multi-agent system where a number of agents will be examining particular aspects of the trip and will negotiate among themselves a bidding strategy.

Although the TAC market scenario presents some interesting problems, other issues that are equally challenging such as negotiations for multi-attribute products and combinatorial auctions in which agents are allowed to bid on bundles are not being addressed.

Drawing on our experience from the Trading Agent Competition and motivated by our research interests in negotiation protocols, we are in the process of implementing a generic server capable of running market-based scenarios. Although in some respects, the server is similar in functionality to AuctionBot [10], it is more flexible regarding the implementation of e-markets by developers.

## REFERENCES

[1] M. Boman, 'Trading Agents'. AgentLink Newsletter, **6**, 15-16, (2001).

[2] M. Fasli, 'Trading Agent Competition 2001'. AgentLink Newsletter, **9**, 13-14, (2002).

[3] A. Greenwald and J. Boyan, 'Bidding Algorithms for Simultaneous Auctions: A Case Study', *Proceedings of the 3rd ACM Conference on Electronic Commerce*, ACM Press, 115-124, (2001).

[4] R.H. Guttman, A.G. Moukas and P. Maes, 'Agent-mediated electronic commerce: a survey', *Knowledge Engineering Review*, **13(2)**, 147-159, (1998).

[5] M. Kumar and S. Feldman, 'Internet Auctions', *3rd USENIX Workshop on Electronic Commerce*, Boston, Mass, 49-60, (1998).

[6] P. Stone, M.L. Littman, S. Singh and M. Kearns, 'ATTac-2000: An Adaptive Autonomous Bidding Agent', *Journal of Artificial Intelligence Research*, **15**, 189-206, (2001).

[7] TAC Game Description, http://auction2.eecs.umich.edu/game.html

[8] TAC List of Entrants, http://auction2.eecs.umich.edu/tac_entry.html

[9] P.R Wurman, W.E. Walsh and M.P. Wellman, 'Flexible double auctions for electronic commerce: theory and implementation', *Decision Support Systems*, **24**, 17-27, (1998).

[10] P. Wurman, M. Wellman and W. Walsh, 'The Michigan Internet AuctionBot: A Configurable Auction Server for Human and Software Agents', *Proceedings of the Autonomous Agents Conference*, 301-308, (1998).