Ground tactical mission support by multi-agent control of UAV operations

Jiří Vokřínek, Peter Novák and Antonín Komenda

Agent Technology Center
Dept. of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague
Czech Republic

Abstract Autonomous control of group of unmanned aerial vehicles based on task allocation mechanisms shows great potential for ground tactical mission support. We introduce experimental simulation system combining flexible mission control of ground assets in urban environment and autonomous aerial support utilizing multi-agent problem solving techniques. Two case-studies are presented for evaluation – cooperative area surveillance and dynamic target tracking with undervalued number of assets. We show the strength and benefits of multi-agent task allocation and delegation mechanisms in such dynamic scenarios mainly in case of limited number of assets.

1 Introduction

In the recent decade the state of technology in robotics achieved a degree allowing routine deployment of unmanned assets in real-world scenarios. Nowadays, many tactical missions, be it e.g., disaster relief search & rescue operations, military missions in mountainous, urban, air or underwater environments rely on deployment of tele-operated robots, such as various unmanned aerial vehicles including conventional fixed-wing aircrafts, vertical-take-off-and-landing helicopters, robotic cars or underwater gliders. With a growing need to deploy multiple assets in the field in order to efficiently fulfil a mission, the tele-operation control mode, however, results in a need for employment of multiple human operators coordinating and controlling the mission execution. This in turn results in high personnel, and subsequently also financial costs. As a consequence, there is a growing need for technology enabling a single human operator to control multiple robotic assets. This need in turn requires a high degree of autonomy on the side of the involved robotic teams which should cooperatively coordinate the mission execution and ideally resolve most of the dynamically arising issues autonomously with as little human operator intervention as possible.

TACTICAL-AGENTFLY 2 (TAF2) is an experimental multi-agent simulation system we have developed as a part of a larger on-going project initiative aiming at investigation of issues in and development of control and planning algorithms for unmanned aerial and ground assets that are engaged in ISTAR (*Intelligence, Surveillance, Target Acquisition, and Reconnaissance*) operations. The TAF2 system facilitates *execution of configurable missions* carried out by a set of aerial and ground assets in an operations theatre.

In the below described case-study, the system provides a platform for extensive testing and evaluation of various implemented coordination strategies for a team of unmanned aerial vehicles (UAVs) supporting a ground-mission by means of providing common operational picture, area surveillance and on-demand tracking of mobile targets. Architecturally, the application comprises three main subsystems: i) the MAS platform providing agent-development toolkit, services such as an agent life-cycle management, a communication middleware, and serves as a container for the whole simulation; ii) the simulation environment providing the implementation of the simulated environment in which the agents are embodied; and iii) a number of agents living in the MAS platform and embodied in the simulation. The set of agents running in the simulation is furthermore divided into three separate teams: a blue team (ground units representing the task force carrying out a mission in the environment), a red team (ground units representing the adversary force interacting with the blue force's mission) and a team of UAVs (supporting the blue force in their mission). While the interaction between the red and the blue teams provides the storyboard of the simulated ground mission, the main purpose of the TAF2 system is to test and evaluate various techniques for control and distributed decision making among the UAVs which provide a mission-centric support to the ground blue team.

Firstly, in Section 2, we provide an overview of the ground mission simulation technology for flexible control of the ground assets and simulated troops. At the core of the approach lies use of *Jazzyk* [3], a highly modular agent-oriented programming language based on the theoretical framework of *Behavioural State Machines* [4]. Secondly, in Section 3 we introduce a general framework for task allocation based on an abstract distributed multi-agent problem solver [8]. In the TAF2 system, the multi-agent problem solver is employed for distributed computation of task allocations among the members of the team of unmanned aerial vehicles supporting the ground operations. The main contribution of the presented paper is the proposal and evaluation of a distributed technique for performing cooperative surveillance and tracking by a team of UAVs with bounded resources discussed in sections 3 and 4. I.e., the main tackled problem can be formulated as the question: *"How can a multi-robotic team of M agents explore, surveil and track N targets (mobile units and/or areas), where $M < N$?"*

2 Ground mission simulation

As a test-bed scenario enabling evaluation of the algorithms for cooperative mission-centric information collection techniques we developed the following mission specification. A mission is configured with 1) a number of blue teams, each consisting of a given number of ground troops (blue force), deployed in an area of interest, 2) a team of UAVs providing airborne support by performing various information-collection tasks, and 3) a number of insurgents randomly distributed in the area of interest (red force). Furthermore, the starting configuration of blue force includes the GPS coordinates of their corresponding deployment points, the map of the urban area and the coordinate of a joint meeting point where the team will be lifted-off at the end of the mission. Finally, the area contains a point of interest called safe house.

Figure 1. Depiction of a situation during the exploration mission phase performed by a team of 3 UAVs. The green lines represent the actual plans of the UAVs. The blue arrows represent the blue force agents located in the deployment points. The red arrows represent the red force agents scattered around the map.

From the point of view of the blue force, the simulated mission unfolds in three stages. Firstly, the task of the blue force is to explore the area of interest and find the safe house. The mission stage is fulfilled cooperatively by the team of UAVs. Subsequently, the ground units approach the safe house and secure its perimeter. In this phase, the core of the mission, the team of UAVs provides 1) situational awareness to the ground units by performing a continuous surveillance of the area around the safe house, and 2) on-demand tracking of mobile targets disturbing the blue force's advance. Finally, the mission ends with a transport of the rescued VIP to the meeting lift-off point, while the team of UAVs still provides the on-demand mobile target tracking and lift-off point surveillance support to the ground units.

The role of the red team is to serve as a set of loosely-coordinating agents interfering with the mission of the blue team which might be tasked to track and capture them. The overall mission plot of the individual teams is supposed to be fully configurable in terms of compositions goals/tasks assigned to the teams and individual agents. The agents execute the mission by executing goal-oriented behaviours associated with and triggered by the overall mission goals and tasks, possibly also their decompositions into lower-level goals. An example mission visual is depicted in Figure 1.

Mission specification The above described mission is just a single instance of a possible mission story-board. One of the objectives of TAF2 project was to develop a technology allowing flexible mission specification and its subsequent execution in a large range of scenarios. We designed and implemented a rudimentary framework allowing us to encode mission specifications as sequences of higher-level behaviours/tasks the agents

and agent teams should perform. In particular, the mission in our implementation is specified as a Prolog-style list of, possibly parametrized, achievement goal specifications, the agents are supposed to be pursue during the mission execution. In the scenario implementation, the goals are associated with a composable reactive behaviours implemented as a reactive planning policies (cf. the next subsection). The parameters of the goal symbols are bound to information stored in the agent's belief base.

Example 1. One of the most complex entities involved in the above described simulated mission are the blue force troops. The following is an example of the actual mission specification of a single blue team agent:

```
[ wait_for_target, move_to(target_safe_house), cover_position, final([move_to(collection_point), cover_position]) ]
```

The behaviour of blue force team agent is thus specified to first wait until the target location of the safe house is broadcast by the team of supporting UAVs, subsequently move to the safe house, secure the target location and when the mission turns into the final phase move to the meeting point and secure the final target location.

Reactive planning as a simulated mission execution framework The simple mission specification language described above provides an input to the simulated agents which in run-time attempt to execute the specified mission taking into account their own, inherent, autonomous behaviour implementation. As an example of such, take a blue agent which should opportunistically, during any mission phase, start a pursuit of an encountered red insurgent.

Our design and implementation of a fully configurable multi-agent mission simulation framework is based on the *Belief-Desire-Intention* (BDI) agent architecture. In particular, the mission execution architecture is based on instantiation of the modular BDI architecture [5] and implemented in the agent-oriented programming framework *Jazzyk* [3]. The underlying design of the system is based on the design inspired by case-studies of autonomous robots for video-games introduced in [2].

Mission execution system The architectural decomposition of the simulated entities is heavily inspired by the BDI architectural scheme. In particular, the agents' belief base is further decomposed into two sub-modules: a *Prolog*-based component storing the agent's information about its environment and itself (belief base), and a *Java* module handling the agent's goals (goal base).

Agents' overall behaviours are implemented as a set of self-encapsulated composable behaviours encoded as reactive plans, policies, in the programming language *Jazzyk*. The elements of a mission specification correspond to high-level goals, which are associated with a single, or more concrete behaviours in the agent program. Additionally, agents often feature implicit behaviours, i.e., such which were not provided in the mission specification, but the agent adopts their associated goals because of its own decision in run-time. An example of such would be the inherent goal of blue troops to track down and capture red agents.

3 Task control of teams of UAVs

For task allocation among the simulated assets in the case-study, we used and adapted an abstract multi-agent problem solver architecture based on a task allocation and

local resource planning introduced in [8]. The architecture is usable for surveillance and tracking with undervalued numbers of aircrafts towards the numbers of targets. The abstract multi-agent problem solver is extended towards cooperative MxN surveillance and tracking by techniques of (i) tasking extension, (ii) task injection, and (iii) task groups implementation. The principle that enables limited number of assets to cover larger number of goals is based on task-oriented representation of goals and dynamic task assignment and exchange between agents. The applicability of this approach in vehicle routing domain has been validated in [9] and shows great potential in scenario introduced in this paper.

The *task sharing* approach is based on passing of tasks from a busy agent to a vacant agent(s). The process can be summarized in the following four steps:

task decomposition: the tasks of the agents are decomposed into subtasks and subtasks, which can be shared are selected.

task allocation: the selected tasks are assigned to the vacant agents or agents, which ask for them.

task accomplishment: each agent tries to accomplish its (sub)tasks. The tasks, which need further decomposition, are recursively processed and passed to other agents.

result synthesis: the results of the tasks are returned to the allocating agent since it is aware of the way to use it in the context of the higher tasks.

From the perspective of distributed problem solving, the crucial parts of the algorithm are the task allocation and result synthesis. From the planning perspective, however, the other two phases are more important. The allocation problem is usually solved by negotiation techniques resulting in use of inter-agent contracts (commitments). As a consequence, problems related to the resource allocation domain, such as e.g., cross-booking, over-booking, backtracking, and others have to be tackled in the overall task allocation architecture. In the allocation phase, a hierarchy of agents is established, which may not be effective in heterogeneous multi-agent systems. The decomposition and delegation principle is widely used in agent-based approaches for problem solving and planning and shows great applicability to realistic problems.

The abstract multi-agent solver architecture is defined as an interplay between three types of agents:

task agent solving the problem preprocessing. To accomplish this, it uses domain specific heuristics, generic ordering strategy, or randomised method. In our scenario the ground units agents undertake this role according to the scenario progress and generate tasks for the group of UAVs.

allocation agent is responsible for decomposition of the preprocessed task allocation problem received from the *task agent* and delegation of the resulting subproblems to the underlying *resource agents*. The allocation agent also performs the result synthesis and tracks the actual task allocation. In our scenario, the role of allocation agent is distributed across ground units – the allocation of particular task is controlled by the agent who introduces the task.

resource agents are responsible for the individual resource planning. In case of further decomposition, the subproblem is handed over to another Task Agent. The resource agents represents the particular UAVs in our scenario.

The multi-agent solver uses the principles of problem decomposition and delegation to autonomous agents that solve individually the parts of the problem. The overall solution is then obtained by merging the individual agents' results. The optimization based on interactions in cooperative environment is usually described as utilitarian social welfare maximization. The problem of minimization of social welfare of a team of agents can be reformulated as the problem of optimization of an abstract objective cost function $\sum_{t \in \mathcal{T}} cost(t)$, where $\mathcal{T}$ is a set of tasks delegated to a single resource agent agent undertaking task t. The cost is infinite in the case the agent is not able to fulfil the task t (assuming there is at least one agent with finite $cost(t)$). Further details on the problem analysis can be found in [8].

The multi-agent solver has been used for multiple goals allocation to a team of heterogeneous agents in the TAF2 system. The goals represent surveillance and tracking tasks and the agents represents the unmanned aerial vehicles capable to perform the tasks.

Formally, we consider a set of heterogeneous agent capabilities $\mathcal{C} = c_1 \dots c_n$ and population of agents $\mathcal{P}$, where each agent is able to provide one or more capabilities. The agents population $\mathcal{P}$ can be divided into groups of Resource Agents $\mathcal{R}_i = a_1 \dots a_n, a_i \in \mathcal{P}$, where all agents from $\mathcal{R}_i$ provide capability c_i and single agent can be member of one or more groups.

We divide the set of tasks $\mathcal{T}$ which ought to be allocated to n subsets $\mathcal{T}_1, \dots, \mathcal{T}_n$, such that the tasks in $\mathcal{T}_i$ can be undertaken by agents from $\mathcal{R}_i$ and at the same time $\bigcup_{i=1}^{n} \mathcal{T}_i = \mathcal{T}$ and $\bigcap_{i=1}^{n} \mathcal{T}_i = \emptyset$.

The allocation is executed for all tasks on different sets of resource agents according their capabilities. The allocation procedure finds the *winner* agent from the appropriate set of Resource Agents $\mathcal{R}_i$ corresponding to actual task $t \in \mathcal{T}_i$. It corresponds to infinite *cost* of insertion estimation of the agent A_j iff $t \in \mathcal{T}_i$ and $A_j \notin \mathcal{R}_i$. The insertion *cost* is computed by each agent locally according to its current plan and the *winner* is selected as the one with minimal insertion *cost*. For more details see [9]. In the case of multiple capabilities of single Resource Agent the weighting factor for the different task prioritization has to be captured by the local agent planner. This weighting factor has must not be the same for all agents. In our scenario the local agent planner is based on finding the shortest route across all assigned task using cheapest insertion heuristics [6] while taking into account dubins curves instead of euclidian distances [1].

The task allocation algorithm is based on local optimization of a single task insertion and improvement. Each iteration of the algorithm provides locally-optimized solution of resources utilization and order-dependent task allocation. The algorithm does not use any backtracking mechanism or exhaustive search of the state space. It has a significant impact on the algorithm's computational complexity, but it is susceptible to finding locally efficient solution only. The global solution quality is improved by execution of the following improvement strategies [9]:

delegate worst: each resource agent identifies its worst task (in terms of the highest removal estimation cost) and tries to delegate it to another agent if the removal cost (savings) is higher than the insertion cost of other agent;

delegate all: each resource agent tries to delegate all its tasks (only if the removal cost is higher than the insertion cost);

reallocate all: each resource agent successively removes all its tasks from the plan and allocates them again (undertakes Allocation Agent role at this moment). The result of the allocation can be the same as before, or a change of the position of the task in the current agent plan, or delegation to another agent.

The improvement strategies are executed as the mission progress to enhance the task allocation until the solution stops to improve. The improvement procedure is restarted each time a new task is allocated, plan execution deviation is detected or unpredictable environment changes apply. In this process each Resource Agent fixes some part of the plan to avoid unnecessary disturbances (e.g. first task). In following sections the techniques based on described multi-agent problem solver are noted as dynamic vehicle routing problem solver (DVRP).

3.1 Cooperative area exploration

The goal of the exploration task is to provide an overview of the whole area by the cooperating group of UAVs (the field of view of a UAV is 50 meters in the simulated scenario, the area to explore is rectangular 1500 meters to 1500 meters). In order to perform a thorough evaluation of the proposed exploration method we have implemented two approaches to the area surveillance:

Zig-zag The state-of-the-art cooperative exploration algorithm [7] flexible in the number of participating UAVs, which uniformly divide the explored area among themselves and fly in a coordinated formation through it. Each UAV has assigned a slice of the area and plans a route through it in the manner of the parallel lines with distance of the field of view of it's sensor between lines. This approach ensures the total coverage of the area minimizing the distance traveled.

DVRP Dynamic vehicle routing problem approach utilizes the multi-agent solver described before. The surveillance task is created for every point of interest in the target area (i.e. all the crossroads as well as straight segments of the streets of length over 10 meters and open spaces). While treating each point of interest as an independent task, the agents negotiate among themselves and using the principles described in the previous section divide the task among the team members. Each plane builds local route for allocated tasks using the cheapest insertion heuristics with dubins curves. The global routes cost minimization is secured by the task allocation algorithm and improvement strategies of the solver.

The main result of our evaluation of the two implemented exploration methods for exploration information collection task is the following claim:

Claim. In structured areas with apriori known structure and possibly irregularly distributed points of interests, the DVRP-based exploration method is significantly more efficient than naive implementations such as the zig-zag algorithm.

The above claim hinges on our experiments described below. Important in this context is the observation that naive methods, such as the zig-zag algorithm, uninformed about the topological structure of the target area spend significant time exploring areas without

much information, such as open spaces (e.g., desert in our example scenario). Observe also, that the two compared methods should perform equally in situations when the topology of the target area is *apriori* unknown. In such a situation, it makes only sense to feed the DVRP-based method with odes of interest uniformly distributed on a grid over the target area (and thus get the similar route patterns like in zig-zag case).

3.2 Cooperative MxN tracking

This multi-agent solver described earlier has been used in the implementation for MxN tracking scenario aiming to undervalued number of assets. The limited resource tracking scenario has been tested on two algorithms. There is a set of (more or less) cooperating planes and a set of randomly moving ground targets. The number of planes and targets vary from 1 to 10 in the experiments. The input for the algorithm is the set of last known positions of targets to track. This information is getting old rapidly and the position uncertainty is increasing over the time. The potential position of a lost target is represented as 10 meters grid around last known position with the size corresponding to last known position age and maximal target speed. Such grids of all the position uncertainty are transferred into set of tasks for multi-agent solver. When a target is re-identified the respective position uncertainty tasks are discarded. The algorithms tested provides different levels of planning and coordination and we evaluate their robustness to the changing number of planes and targets. The tested methods are following:

Greedy cooperative allocation In every moment of the mission simulation, each plane finds closest last known position or position uncertainty area and goes to explore it. The planes use simple voting method to avoid conflicts in selection of areas. It leads good utilization of resources and provides good implicit path conflict avoidance.

DVRP Dynamic vehicle routing problem approach based on multi-agent solver. The tasks to allocate are last known positions of the ground targets and position uncertainty areas. Again, each plane builds local route for allocated tasks using the same technique as in area surveillance scenario and global routes cost minimization is secured by the task allocation algorithm of the solver. This approach provides efficient dynamic task delegation and effective handovers. The complexity of this method grows with the number of target points (i.e. size of area of uncertainty), so in case of high number of lost targets the communication overhead increase.

4 Experimental results

The performance and scalability to the number of planes for algorithms in experimental scenarios has been evaluated.

In the **area exploration** scenario the complete area can be explored efficiently with about 6 planes with both zig-zag and DVRP algorithms. The performance of DVRP-based exploration proofs to be better than zig-zag in our scenario. Figure 2 shows the DVRP-based exploration method is about 100% faster than zig-zag exploration method. Experiment results show notable steps in the time, where only minimal number of nodes are explored. It corresponds to the situation, where the planes are flying over desert

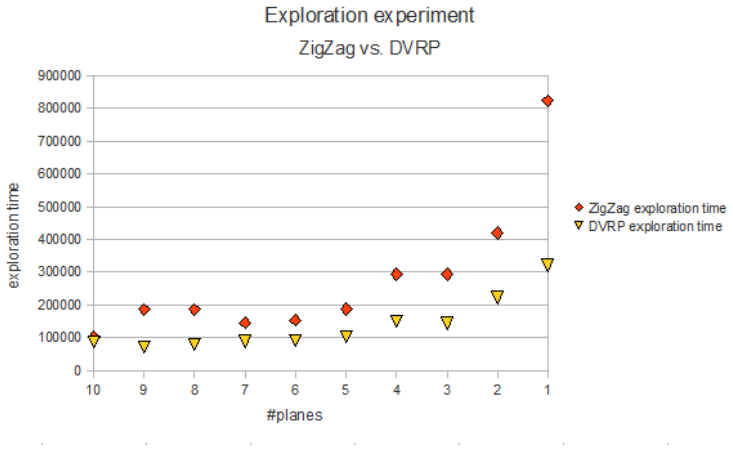

Figure 2. Area exploration – time needed for exploration of all points of interests by varying number of planes for zig-zag and DVRP-based exploration algorithm.

areas or making turns on the upper or lover part of the area. These blind steps are not significant in the results of DVRP-based method, because planes optimize their routes across the points of interests. The only exception is a situation when the route planning heuristics fails to provide non-repeating route across points of interest (i.e. planes is passing already explored nodes) or there is need for longer flight over non interesting area. The results prove the claim stated in Section 3.1 is true in our scenario. DVRP-based exploration method proves to be significantly more efficient than zig-zag algorithm in apriori known structured area.

The implementation of DVRP in the **MxN tracking** scenario was evaluated against greedy cooperative allocation. Figure 3 shows the maximum number of targets for varying number of planes for both algorithms in our scenario. Surprisingly, the greedy cooperative allocation approach proved to be relatively robust especially due to its low communication complexity which grows linearly with the number of the aircrafts in the team. This method provide robust n to n tracking with very low communication needs (communication is linear to the number of planes). With 8 planes the scenario is saturated and planes are able to track any number of mobile ground targets without loosing any of them in the scenario settings. DVRP-based approach provides more robust results. It proves to provide stable behavior of n to $n + k$ tracking (4 planes are able to track 5 targets or 6 planes successfully track 8 targets in our scenario). With 7 planes the scenario is saturated and planes are able to track any number of ground targets.

Acknowledgements The presented work was supported by the U.S. Army Communications-Electronics Research, Development and Engineering Center grant no. W911NF-08-1-0521, Czech Republic Ministry of Education, Youth and Sports, grant no. MSM6840770038, the Grant Agency of the Czech Technical University in Prague, grant no. SGS10/189/OHK3/2T/13 and research programme no. 1M0567 funded by the Ministry of Education of the Czech Republic

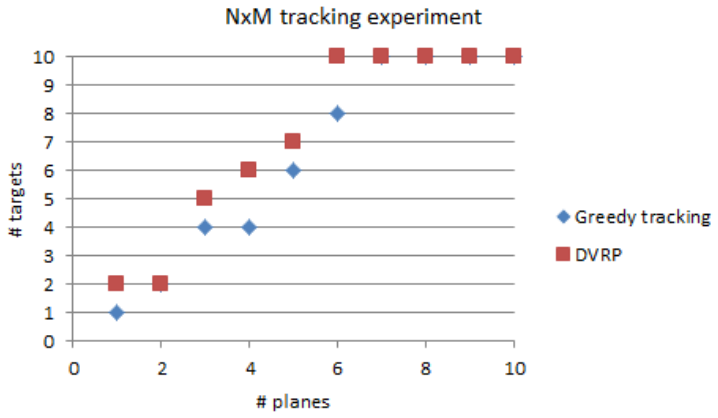

Figure 3. NxM tracking – number of tracked targets by varying number of planes for greedy and DVRP-based tracking algorithm.

References

1. L. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.
2. M. Köster, P. Novák, D. Mainzer, and B. Fuhrmann. Two case studies for Jazzyk BSM. In *Proceedings of Agents for Games and Simulations, AGS 2009, AAMAS 2009 co-located workshop*, volume 5920 of *LNAI*, pages 31–45. Springer Verlag, 2009.
3. P. Novák. Jazzyk: A programming language for hybrid agents with heterogeneous knowledge representations. In *Proceedings of the Sixth International Workshop on Programming Multi-Agent Systems, ProMAS'08*, volume 5442 of *LNAI*, pages 72–87, May 2008.
4. P. Novák. *Behavioural State Machines: Agent Programming and Engineering*. PhD thesis, Faculty of Mathematics/Computer Science and Mechanical Engineering, Clausthal University of Technology, Germany, September 2009.
5. P. Novák and J. Dix. Modular BDI architecture. In H. Nakashima, M. P. Wellman, G. Weiss, and P. Stone, editors, *AAMAS*, pages 1009–1015. ACM, 2006.
6. D. Rosenkrantz, R. Stearns, P. Lewis, et al. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6:563, 1977.
7. E. Semsch, M. Jakob, D. Pavlíček, M. Pěchouček, and D. Šišlák. Autonomous uav surveillance in complex urban environments. In C. M. D. E. S. Maxim Likhachev, Bhaskara Marthi, editor, *Proceedings of ICAPS 2009 Workshop on Bridging the Gap Between Task and Motion Planning*, pages 63–70, Greece, September 2009.
8. J. Vokřínek, A. Komenda, and M. Pěchouček. Abstract architecture for task-oriented multi-agent problem solving. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(1):31 –40, 2011.
9. J. Vokřínek, A. Komenda, and M. Pěchouček. Agents towards vehicle routing problems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 773–780, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.