Implementing an Agent-based Decision Support System for Task Allocation: a Multi-criteria Approach

Nikolaos F. Matsatsinis, Pavlos Delias

Decision Support Systems Laboratory, Technical University of Crete
73100, Chania, Greece
Email: {nikos, mirouvor}@ergasya.tuc.gr

Abstract. The Task Allocation problem runs oftentimes in project management. Artificial Intelligence is trying to finger out a solution to model this ill-structured problem. In this paper, we propose an innovating multi – criteria approach, implemented in a decision support system, AgentAllocator. This multi – criteria method ends up with an optimal allocation plan. AgentAllocator is an easy to use application which allows the decision maker to model the problem (according to his policy) through its inputs dialogs and employ the final solution proposed by the system. In this paper, although some features of the DSS are discussed, we focus more on the proposed methodology and the theoretical background of the task allocation decision.

1 Introduction

The task allocation problem occurs very often in multiplex environments. This problem is considered as one of the most classical problems in the operational research, where the allocation is decided by validating a single optimization criterion (mono-criterion problem) or multiple optimization criteria (multi-criteria problem) [4, 5]. This paper attends to face the problem of task allocation in a multi-agent system. The allocation is based on the agent's attributes and on its ability to execute a specific task. The final decision comes upon the evaluation criteria used by the decision maker. A really significant feature of this model is the reevaluation of agents' attributes each time a task is assigned.

The method used by AgentAllocator [6] is delineated in section 3, while in section 2 we refer to four different approaches for task allocation in multi-agent systems and we try to underline their basic differences from the proposed model. In Section 4, illustrative examples as well as a few screenshots of AgentAllocator are attached. Finally, in the last section we try to criticise both the method and the DSS, revealing our feature goals.

2 Background

The task allocation in multi-agent systems has occupied to a large extend the scientists of Decision Science and Artificial Intelligence.

Smith [13] proposed a "Contract Net Protocol" that agents can adopt. According to this approach, the agents negotiate during run time and come to a decision for the task allocation. Since agents are autonomous, the negotiation breadth can be really huge and the time needed until the agents come to decision may not be available. In addition, negotiations themselves demand continuous communication among the agents, and that means great cost.

In order to reduce the negotiation's breadth that is demanded from the "Contract Net Protocol", Tidhar, Rao and Sonneberg [14] proposed a more guided process of selecting a team of agents with complementary skills to achieve a goal. The instructor is the developer of the multi-agent system, because he/she knows a priori all the alternative agents and their attributes. Each time that a goal is to be achieved the instructor split it to sub-goals and for each sub-goal he/she specifies the attributes that an agent or a group of agents must have in order to bring it successfully to an end. In other words the instructor determines agents' types that could be candidate for a task allocation. If an agent does not belong to the definite for a task "type", he cannot participate in the negotiation that will take place for the assignment of this specific task. Consequently, the number of the candidates for a task allocation is significantly reduced and the allocation process is less time-consuming. Of course, since an agent or a group of agents is considered as a candidate for a specific task allocation and takes part in the negotiation procedure, he acts autonomously.

The total autonomy that Smith [13] as well as Tidhar,Rao and Sonneberg [14] assigned to agents causes in several occasions serious problems. Such an occasion is presented and faced by Walsh and Wellman [15]. In that case, the various agents can perform as suppliers of primitive goods or as producers of output goods and the goal is a consumer-agent to acquire the goods he desires. Each agent regularly sends bid messages for some of the goods that he wishes to buy or sell, specifying also the price below/above which the agent is willing to buy/sell. Each time that a new bid arises, the rest of the agents may choose to revise their own bids. Agents can respond only to the most recent bid sent to the system. Bidding continuous until a state where all messages have been received and no agent chooses to revise his bids. The problem that arises and that obviously the "Contract Net Protocol" approach can not face is that the bid messages, that agents send, have to be valid. For instance, a producer should not be able to claim that he will sell an output product if he has not ensured the purchase of the necessary primitive goods. Additionally, the bids have to follow a specific economy policy. To ensure these, it is necessary for the system to include several bidding policies, so that every agent adopts his favorite. This means that agents partially loose their autonomy and follow strictly several main rules.

In the above approaches, task allocation is brought into effect after a process of negotiation among the candidate agents themselves. A different approach is the one that indicates the existence of a "master-agent" who will be responsible for the allocation of the tasks to several other agents (slave agents). Such an approach is presented by Fujita and Lesser [2]. In that case there is a goal to achieve until a deadline and with a standard quality level. Firstly, the master-agent splits this goal to two or more sub-goals that must be achieved. There is the possibility that several sub-goals might need to be executed sequentially, meaning that the one's results are essential for the execution of the rest. Using heuristics algorithms the authors determine the order according to which the sub-goals will be allocated and executed. The sub-goals assignment to

the slave-agents is based to a single criterion, the minimization of the central goal's execution duration. Specifically a slave-agent will undertake the execution of a sub-goal if and only if he is the most immediate available among the other agents. In case that the execution of a sub-goal prerequisite for more than one other sub-goals, both of them will be assigned to the same slave-agent. Of course if a sub-goal is prerequisite for more than one of the other sub-goals, then the agent who has undertake it, will also undertake and one of the other sub-goals, while the rest will be allocated to agents who will be available by the time that their execution will be feasible (that is by the time that the prerequisite sub-goal will be completed). Finally, Fujita and Lesser [2] provide to the master agent the possibility to evaluate periodically the initial plan of tasks allocation, to ensure that the time and quality limits of the central goal are satisfied.

Some important works study the principles of an agent decision making function in a range of disciplines including economics, sociology and time restrictions. Traditionally this function has been based in the optimization of a single criterion. Thus the Agents Decision Makers (ADM) choice was based on the minimization of the performance time or the results accuracy. Another approach (Parsons and Jennings [9]) makes a more social confrontation. The candidate agents are ranked not only in the base of their individual skills, but also according to their social behavior (past cooperations). Moraitis and Tsoukias [8] propose a multicriteria approach, which considers a set of criteria of each candidate agent, and through direct pairwise comparisons establish a graph representation of the candidate agents' behavior. The above multicriteria approaches are based on value focused procedures and aim to construct a unique function, aggregating the partial preferences on multiple criteria. In that case the ADM is directed by this value system to its final decision. Shehory et al. [11] mention the problem of task allocation to a group of agents in order to attain tasks execution and to improve tasks efficiency. We will confront the problem from a different point of view.

In this paper a new method for task allocation is presented. There are no more master and slave agents or negotiation among candidate agents. The multi-agent system itself is responsible for the task allocation. The candidate agents are predefined. Each of them has attributes – capabilities while each task is described by specific attributes – demands. The system counts all the possible assignments and evaluates them all through a multi-criteria method described below. As a result the final allocation plan is as much as possible consistent with the decision maker's preferences.

3 The Multi-criteria Method

AgentAllocator uses the method proposed by Matsatsinis et al. [7]. This is about a value-focused approach established in three main functions:

1. Modeling a consistent family of measurable and/or ordinal criteria to evaluate the performances of every possible task assignment to agents.
2. Assessing an additive value function aggregating the evaluation criteria in order to provide a ranking of alternative assignments.
3. Implementing an allocation mechanism which finally assigns the task to agents according to their value collected and which reviews agents' attributes.

3.1 Defining the Method and Making the Concessions

The focus of the method is to assign k tasks to m agents while the loads of tasks may be smaller, equal or even greater of the crowd of agents. Obviously, every agent is eligible to undertake more than one task. The method declares that each task can be executed just by one agent and that since the allocation is decided the agent has no right to negotiate this decision. Moreover, agents do not advance any of the tasks. At this point it is necessary to employ the following notations and definitions. Let:

T	be the set of tasks
A	set of agents
k	number of the tasks
m	number of the agents
t_i	the i-th task in T
a_j	the j-th agent in A
(t_i,a_j)	assignment action of t_i to a_j
$v(T)$	set of demands describing T
$v(A)$	set of attributes describing A
F	a consistent family of criteria
N	number of these criteria
g_j	the j-th criterion in F
$g_j(t_i,a_l)$	the performance of (t_i,a_l) assignment on the j-th criterion
$G(t_i,a_l)$	vector of performances of (t_i,a_l) assignment on the N criteria
g_{j*}	worst level of criterion g_j
g_j^*	best level of criterion g_j
$u_j(g_j)$	a non decreasing marginal value function on $[g_{j*}, g_j^*]$ varying from 0 to 1
p_j	weighting factor of criterion g_j (the sum of N weights equals 1)
$U(G)$	the global value of the performance vector G, varying from 0 to 1.

3.2 The Task Allocation Process

The various tasks are not identical but all can be described through a common set of demands. The same stands for the agents. They are also distinct and can be described by a common set of attributes. The agents' attributes may change dynamically during the allocation process. They may be improving, reducing or even remain constant. Both these demands and attributes varying between several rating levels. These levels are preferably descriptive. All these sets are formulated by the decision maker and entered in the system as input files. The next critical step is to shape the evaluation criteria. The evaluation criteria have to be modeled in such a way that they compose a consistent family of criteria and (as knowing from the multi-criteria analysis theory) have to fulfill three fundamental conditions: monotony, exhaustively and non-pleonasm [11]. Each criterion is modeled by a set of sub-criteria, defining hereby a sub-criterion as a combination of task's demands and agent's attributes. Those sub-criteria have also rating levels varying from 0 to 1(a numerical correspondence is attached to every descriptive level of the sub-criterion). Rating the sub-criteria, we have to ponder that:

- High performance agents should not be wasted in tasks with low demands.
- No sub-criterion should be advanced, unless it is a completely satisfactory one.
- All not workable assignments should be rated with a 0.

This is going to be clearer in the example that follows. Modeling the criteria may be the most critical step in this method as it incorporates the decision's maker logic and experience.

With a view to evaluate each assignment's performance the following heuristic process is executed:

1. Divide each criterion g_i to several level-characterizations and correspond each level with a numerical interval
2. For each assignment (t_i,a_j):
   - According to the content of each sub-criterion find the relevance between task's demands and agent's attributes and value (t_i,a_j) to the specific sub-criterion
   - For each criterion g_i sum up the values that (t_i,a_j) has achieved in the corresponding set of sub-criteria and calculate the average value
3. For each criterion g_i, give to each assignment the appropriate characterization based on the interval that the calculated average value belongs to.

Keeping on with the method, the decision maker is asked to rank a set of reference assignments, in such a way that the ranking reveals preference or indifference among the alternatives assignments. This ranking is used by the UTA [3, 12] method when assessing the criteria weights. UTA method is explained in section 3.3.

Applying this global utility function all assignments are being evaluated and given a score. The assignment that achieved the highest score is selected and the allocation is considered to be done. The task component of this assignment is removed from the set of task and the agents' attributes are reviewed according to the following single rule: Let q be the total of tasks that this specific agent has undertaken and let s be the total of the rating levels of each attribute. Then, for each attribute:

- If $q \geq P/s$, the attribute switch upwards or downwards its rating level
- If $q < P/s$, the attribute's rating level stands still, where P is a percentage defined by the decision maker.

The method acts exactly the same, in an iterating way, until the set of tasks is empty. Of course the assignments' scores are reevaluated in each step, since agents modify their attributes. The process of the proposed method is depicted in figure 1.

3.3 Multiple Criteria Decision Making. The UTA Method

In decision-making involving multiple criteria, the basic problem stated by analysts and decision-makers concerns the way that the final decision should be made. In many cases, however, this problem is posed in the opposite way: given the decision, how is it possible to find the rational basis through which the decision was made? Or equivalently, how is it possible to assess the decision–maker's preference model leading to the exact same decision as the actual one or at least the most "similar" decision? The philosophy of preference disaggregation in multicriteria analysis is to assess/infer preference models from given preferential structures and to address decision-aiding activities through operational models within the aforementioned framework.

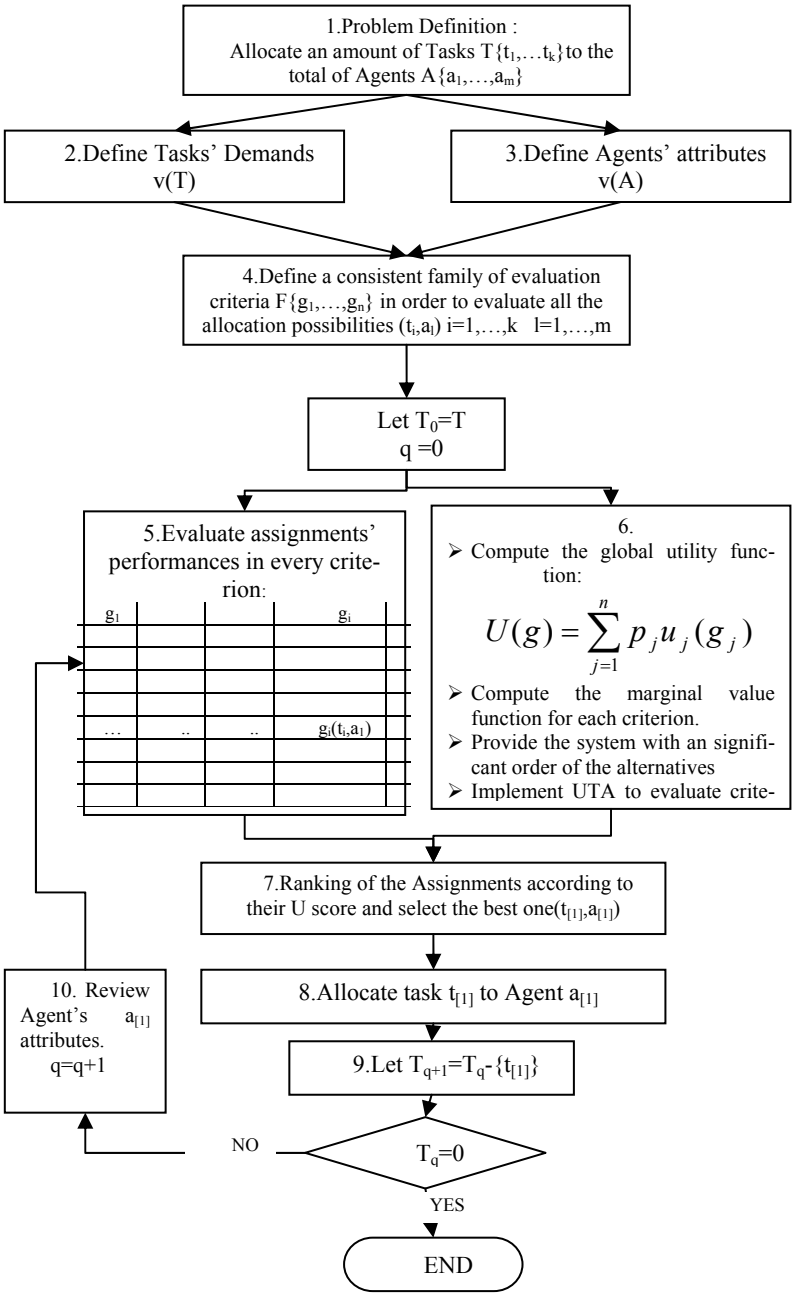

Fig. 1: Task Allocation's Logic Diagram

Under the term "multicriteria analysis" two basic approaches have been developed involving:

1. a set of methods or models enabling the aggregation of multiple evaluation criteria to choose one or more actions from a set A and
2. an activity of decision-aid to a well-defined decision-maker (individual, organization, etc.)

In both cases, the set *A* of potential actions or decisions is analyzed in terms of multiple criteria in order to model all the possible impacts, consequences or attributes related to the set *A*.

Roy [10] outlines a general modeling methodology of decision-making problems, which includes four modeling steps starting with the definition of the set *A* and finishing with the activity of decision-aid, as follows:

- Level 1: Object of the decision, including the definition of the set of potential actions A and the determination of a problematic on A.
- Level 2: Modeling of a consistent family of criteria assuming that these criteria are non-decreasing value functions, exhaustive and non-redundant.
- Level 3: Development of a global preference model, to aggregate the marginal preferences on the criteria.
- Level 4: Decision-aid or decision support, based on the results of level 3 and the problematic of level 1.

In level 1, Roy [10] distinguishes four referential problematics, each of which does not necessarily preclude the others. These problematics can be employed separately or in a complementary way in all phases of the decision-making process. The four problematics are the following:

- Problematic α: Choosing one action from A (choice).
- Problematic β: Sorting the actions in well defined categories which are given in a preference order (sorting).
- Problematic γ: Ranking the actions from the best one to the worst one (ranking).
- Problematic δ: Describing the actions in terms of their performances on the criteria (description).

In level 2, the modeling process must conclude on a consistent family of criteria $\{g_1, g_2, \ldots, g_n\}$. Each criterion is a non-decreasing real valued function defined on *A*, as follows:

$$g_i : A \to [g_{i*}, g_i^*] \subset \square \ / \ a \to g(a) \in \square$$

where $[g_{i*}, g_i^*]$ is the criterion evaluation scale, g_{i*} and g_i^* are the worst and the best level of the *i*-th criterion respectively, $g_i(a)$ is the evaluation or performance of action a on the *i*-th criterion and $\mathbf{g}(a)$ is the vector of performances of action a on the *n* criteria.

From the above definitions the following preferential situations can be determined:

$$\begin{cases} g_i(a) > g_i(b) \Leftrightarrow a \succ b \ \left(a \text{ is preferred to } b\right) \\ g_i(a) = g_i(b) \Leftrightarrow a \ \square \ b \ \left(a \text{ is indifferent to } b\right) \end{cases}$$

So, having a weak-order preference structure on a set of actions or objects, the problem is to adjust additive value or utility functions based on multiple criteria, in such a way that the resulting structure would be as consistent as possible with the initial structure. This principle underlies the disaggregation-aggregation approach presented in the next section.

UTA (**UT**ilité **A**dditive) method proposed by Siskos [3, 12] tries to analyze the decision maker's preferences in a multicriteria environment while the alternatives decisions are predefined and finite. UTA provides a global additive utility function that represents the decision maker's preferences following the next steps:

Step 1: Give a demonstrative ranking of some reference alternatives

Step 2: Express the global values of the reference assignments in terms of weights:

$$U[g(A_i)] = \sum_{j=1}^{n} p_j u_j \left[g_j(A_i) \right] \qquad \forall\ i = 1, 2, \ldots, k \tag{1}$$

g_j stands for the criteria, p_j for criteria weights and A_i for the alternatives.

Step3: For each criterion g_j , j = 1,…,N (where N is the total of criteria), evaluate the ordinal marginal utility functions $u_j(g_j)$ in such a way that:
$u_i : \{g_{j*}, g_j^*\} \rightarrow [0,1]$, where g_{j*} is the worst and g_j^* the best value of the j-th criterion.
$u_i(g_{j*}) = 0$
$u_i(g_j^*) = 1$

Step 4: Introduce two errors σ_i^+ and σ_i^- per assignment A_i and formulate for each pair of consecutive assignments in the ranking, the analytical expressions:

$$\Delta\left(A_i, A_{i+1}\right) = U[g(A_i)] - \sigma_i^+ + \sigma_i^- - \left[U[g(A_{i+1})] - \sigma_{i+1}^+ + \sigma_{i+1}^- \right] \tag{2}$$

The purpose and the significance of these errors is to represent the distance of each ranking preference given by the decision maker from the global utility function U.

Step5: Solve the linear program:

$$\text{Min } z = \sum_{i=1}^{k} (\sigma_i^+ + \sigma_i^-) \tag{3}$$

Subject to:
For i= 1,2,…k-1

$$\Delta(A_i, A_{i+1}) \geq \delta \text{ if } A_i > A_{i+1} \tag{4}$$
$$\Delta(A_i, A_{i+1}) = 0 \quad \text{if } A_i \approx A_{i+1}$$

$$\sum_{j=1}^{n} p_j = 1 \tag{5}$$

$$p_j \geq 0,\ j = 1,2,\ldots,n \tag{6}$$

$$\sigma_i^+ \geq 0,\ \sigma_i^- \geq 0,\ I = 1,2,\ldots,k \tag{7}$$

δ being a small positive number(AgentAllocator uses a default value of 0.05).

Step 6: (Stability Analysis) Test the existence of multiple optimal or near optimal solutions of the linear program (1)-(7); in case of non uniqueness, find the mean weights of those (near) optimal solutions which maximize the objective functions $z_j = p_j$ for all j = 1,2,…,n on the polyhedron (4)-(7) bounded by the new constraint:

$$\sum_{i=1}^{k} \left(\sigma_i^+ + \sigma_i^- \right) \leq z^* + \varepsilon \quad (8)$$

where z^* being the optimal value of the linear program in step 3 and ε is a very small percentage of z^*. AgentAllocator uses a default value for ε 0.05 but the decision maker is eligible to change this percentage through the options menu of the DSS.

4 Applying AgentAllocator in an Illustrative Example

Let us monitor AgentAllocator during the following example. Consider an employer trying to manage his enterprise's executive officers-agents. For the purpose of this article, let three be the total number of these officers (we call them agents from now on). Agents have a certain profile, composed by the following attributes:

Agent's name	*Technical Knowledge*	*Problem Solving Ability*	*Reliability*	*Management Skills*	*Availability*
a1	Basic	Undistinguished	Trustworthy	Clement	High
a2	Expert	Satisfactory	Unproved	Poor	Medium
a3	None	Satisfactory	Unproved	Expert	Medium

Table 1 Agents' Profiles

At this point, we should remind that agents have the ability to adjust dynamically their performances in certain attributes. So, in this case agents' *Availability* decreases and their *Management Skills* are worked up during run time. Let us also suppose that the tasks that occur in the enterprise environment can be described by the demands of: Immediacy, Importance, Technical Demands and their Social Minded aspect. All these demands have their own rating levels as shown in table 2.

Task name	*Technical Demands*	*Immediacy*	*Importance*	*Social Minded*
t1	Basic	Normal	Normal	Normal
t2	Expert	Urgent	Normal	Normal
t3	Average	Urgent	High	Normal
t4	None	Urgent	High	Critical
t5	Basic	Low	High	Critical

Table 2 Tasks' requirements

The goal is to figure out the best allocation plan. For this reason, three evaluation criteria are employed: the *speediness* of the task's execution, the *risk* undertaken by each allocation decision and the *functionality* of the assignments. These criteria are modeled as declared in table 3:

Speediness	*Risk*	*Functionality*
Availability – Immediacy	Problem Solving Ability - Importance	Technical Knowledge-Technical Demands
Problem Solving Ability	Management Skills-Social Minded	Management Skills-Social Minded
	Reliability	

Table 3 Modeling the Criteria

You may have noticed that the components of the sub-criteria are either agents' attributes or tasks' demands or both. For each sub-criterion, the decision maker has to correspond a numerical value at every possible combination of its components rating levels. Criteria, as mentioned in section 3, have their own rating levels. Table 4 gives an example of this sub-criteria rating procedure.

Speediness Criterion	*Availability*				
	Immediacy		Low	Medium	High
		Low	1	0.6	0.4
		Medium	0.4	1	0.6
		Urgent	0	0.6	1
	Problem Solving Ability				
	Non Satisfactory		Undistinguished		Satisfactory
	0		0.4		1

Table 4 Rating Sub-Criteria of Speediness Criterion

All this data can be inserted in the system using its input dialogs. In figure 1, the input dialog for the agents' data is presented.

AgentAllocator includes four types of input dialogs. In Task Dialog, the user inserts the data needed to model the tasks, such us task's demands and its rating levels. In Agent Dialog, the decision maker should insert agent's attributes, attributes' cases (improving, reducing or constant) and their rating levels. Criteria Dialog is used to insert the data for the evaluation criteria, which means: criteria's intervals, sub-criteria and sub-criteria's rating levels. Finally, in Assignments Dialog, the user can watch the performance of all assignments in the evaluation criteria and insert the ranking for the set of a few reference assignments.

In order to define the global utility function, UTA method is implemented. UTA assesses the criteria weights analyzing a reference set of alternatives. In a reference set, ranking has the following meaning:

$$ranking(\text{Assignment}_i) > ranking(\text{Assignment}_j) \text{ indicates preference,}$$
$$ranking(\text{Assignment}_i) = ranking(\text{Assignment}_j) \text{ indicates indifference.}$$

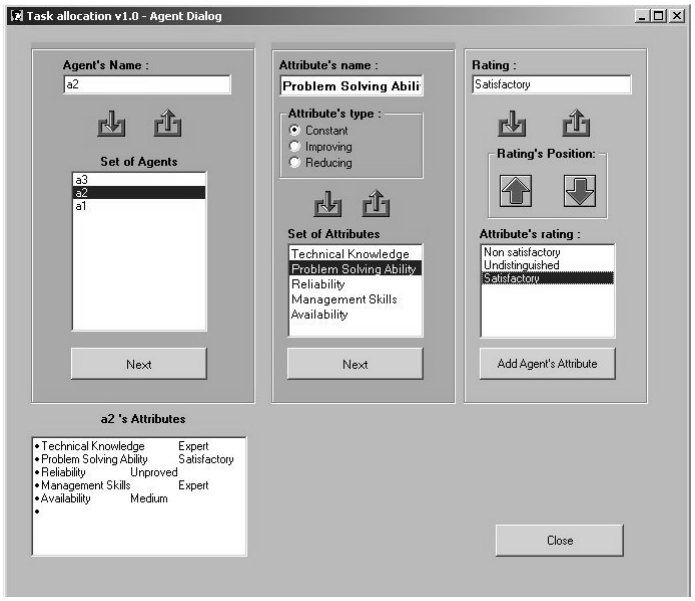

Fig. 2 Agent's Data Input Dialog

The reference set used in this example is figured out in table 5. Given the above reference set, UTA assess *Functionality* a weight of 61.8% while *Speediness* achieved a 26.7% and *Risk* a 11.5%. AgentAllocator calculates the scores of all possible assignments and ends up with an optimal allocation plan. The scores of the remaining possible assignments are recalculated each time a task is allocated. These scores may differ from step to step because agents' attributes change dynamically during the process. The user is able to watch the entire allocation process in the "task allocation board" which is attached to the system as represented below:

Preference Assignment	*Rank*	*Risk*	*Speediness*	*Functionality*
This is a Real Good Case!	1	Certainty	Big	Normal
Just Good	2	Admissible	Acceptable	Efficient
Not Satisfactory	3	Admissible	Big	Bad
Not Satisfactory too	3	Risky	Small	Normal
A Bad One	4	Risky	Acceptable	Bad

Table 5 Ranking of Reference Alternatives.

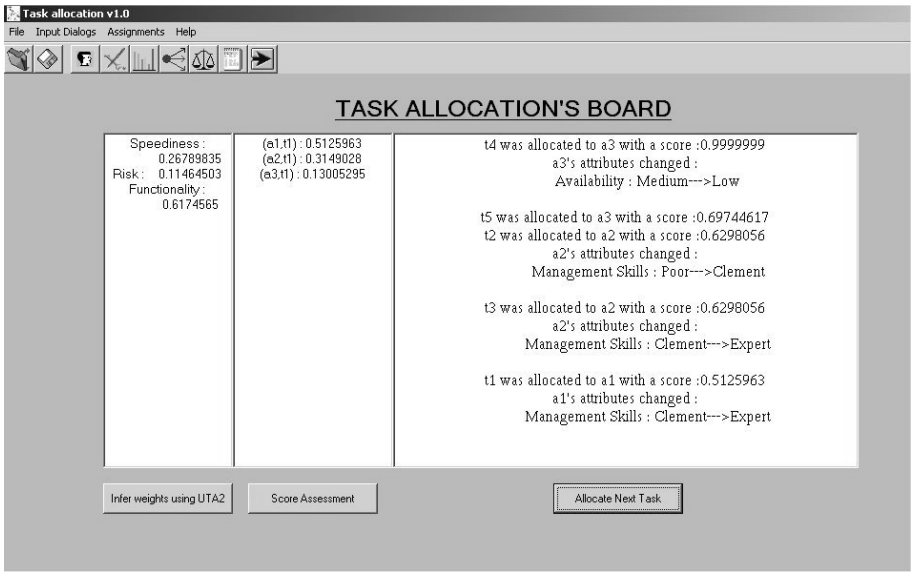

Fig. 3 Task Allocation's Board.

More specifically, the evolution of the task allocation process took place as follows:

Step1: The task called t4 is assigned to agent a3. The a3's attributes are reviewed and his *Availability* is reduced from *Medium* to *Low*. t4 is being removed from the set of tasks. The scores of the rest of the assignments are recalculated.

*Step 2:*t5 is allocated to a3.

Step3: t2 is allocated to a2. His *Management Skills* are shifted up to *Clement*.

*Step 4:*a2 undertakes t3. His *Management Skills* augments once more. a2's current *Management Skills* are rating with an *Expert* value. This double augmentation of a2's *Management Skills* can be avoided by increasing the Percentage *P* mentioned in section 3.2. But as long as this example is being used just for demonstration's reasons we shall allow to a2 this impressive evaluation in his *Management Skills!*

*Step 5:*One task has remained. That is t3 and it is assigned to a1. Now it is a1's turn to improve his attributes, so his *Management Skills* shift up to *Expert*.

5 Conclusions

The approach for task allocation proposed in this paper includes a multi-criteria methodology. That guides the final solution to be as possible consistent with the decision maker's preferences. Therefore, the final solution depends in a very direct way on the decision maker's policy. This infuses the system with an impressive flexibility but also with a disagreeable subjectivity. To prevent this subjectivity from leading to

dissatisfactory decisions, this approach should employ a way to measure the final allocation plan's effectiveness. There should be a feedback reporting the results of the selected allocation plan, so that the method could evaluate its own techniques. This feedback information would help even if it was implemented through every step of the allocation process, so the DSS could decide whether to improve more agents' attributes or to detract their improvement. A researching focus of this method is to model group allocation and some constraints that may task impose, such as precedence constraints. In addition, a future ambition is this DSS to include more allocation methods to support more efficiently the decision maker.

Concluding, AgentAllocator may subsist a compact and useful tool to model and analyze the task allocation decision in complex environments, which is hard to be modeled and analyzed otherwise.

References

1. Delias P.: Design and Analysis of an Decision Support System Based on Agent Technology, Diploma Thesis, Technical University of Crete, (2002)
2. Fujita S., Lesser V.: Centralized Task Distribution in the Presence of Uncertainty and Time Deadlines, *Proceedings ICMAS Conference*, (1996)
3. Jacquet-Lagreze E., Siskos Y.: Assessing a Set of Additive Utility Functions for Multi-criteria Decision Making, *European Journal of Operational Research*, 16(1): 48-56 (1982)
4. Kenny R.L., Raifa H.: *Decision with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley, NY (1976)
5. Klein G., Moskowitz H., Maheesh S., Ravindran A.: Assessment of Multi-attribute Measurable Value and Utility Functions via Mathematical Programming, *Decision Sciences*, 16(3):309-324 (1985)
6. Matsatsinis N.F., Delias P. : AgentAllocator: an Agent-Based Multi-Criteria Decision Support System for Task Allocation, *Proceedings 1st International Conference on Applications of Holonic and Multi-Agent Systems*, (2003)
7. Matsatsinis N.F., Fostieri M., Koutsouraki E., Moraitis P.: A Multi-Criteria Approach for Task Allocation in a Multi-Agent System, *Proceedings ESIT Conference*, Chania, (1999)
8. Moraitis P., Tsoukias A.: Graph Representation of Collective Attitudes for Teamwork in Dynamic Environment, (1998)
9. Parsons S., Jennings N.R.: Argumentation and Multi-agent Decision Making, *Proceedings AAAI Spring Symposium on Interactive and Mixed-Initiative Decision Making,* Stanford, CA, (1998), 89-91
10. Roy B.: *Methodologie Multicritère d'Aide à la Decision*, Economica, Paris, (1985)
11. Shehory O., Kraus S.: Task Allocation via Coalition Formation among Autonomous Agents, *Proceedings 14th International Joint Conference on Artificial Intelligence (IJCAI),* (1995).
12. Siskos Y.: Comment Modéliser les Preferences au Moyen de Functions d' Utilité Additives, *RAIRO Recherche Opérationnelle*, 14:53-82 (1980)

13. Smith R.: The Contract Net Protocol: High Level Communication and Control in a Distributed Problem-solver, *IEEE Transaction on Computers*, 29(12):1104-1113, (1980)
14. Tidhar G., Rao S., Sonneberg E.: Guided Team Selection, *Proceedings ICMAS Conference*, (1996)
15. Walsh W., Wellman M.: A Market Protocol for Decentralized Task Allocation, IEEE, (1998)