Integrating Individual, Organizational and Market Level Reasoning for Agent Coordination

Mihai Barbuceanu[1] and **Wai-Kau Lo**[2]

Abstract. In this paper we articulate a multi-level view of agent coordination and provide solutions for an integrated agent architecture that addresses all the levels. At the *individual agent level*, we model the decision making problem faced by individual agents that need to discover their highest utility goals and the plans to achieve them. Individual level plans normally contain goals that lie outside the agent's control. To achieve them, the agent needs to team up with other agents in the organization. At the *organizational level*, we show how organizational structures can be used to form the minimum cost teams needed to achieve such goals. Individual and organizational reasoning rely on knowing the utilities of the options available to agents. Often, these utilities are not given in advance, they must be discovered dynamically by market driven interaction. At the *market level*, we give a constraint optimization formulation to Multi Attribute Utility Theory and introduce interaction processes that allow agents to discover how to cooperate to optimize their objectives. All levels translate their specific models into a common reasoning infrastructure integrating randomized and systematic search.

1 INTRODUCTION

As the connectivity provided by the Internet increases, the coordination problem faced by the interconnected agents and organizations becomes more and more difficult. Global systems for manufacturing and service provisioning for example, connecting suppliers, producers, distributors and customers all over the world need to coordinate the activities of agents inside their different component organizations and the organizations themselves on the global market place to provide value for all participants.

In this paper we articulate a multi-level view of coordination and we provide solutions for an integrated agent architecture that addresses all the levels. All levels use a *common representation and reasoning infrastructure* that represents an agent's goals, constraints and utilities (section 2). A MAXSAT-type solver produces a high (or the highest) utility behavior. This is used as the solution of the agent's decision problem at the *individual level* (section 3). Individual behaviors normally contain goals that the agent can't achieve without the collaboration of other agents in the organization. At the *organization level*, organization models are dynamically generated for these goals. These models, based on the Steiner tree problem [2], are compiled into a form solvable by the same MAXSAT-type solver, determining the structure of the teams that can achieve the goals with minimal cost (section 4). Individual and organizational reasoning both rely on knowing the utilities of the goals. In many cases, these are not given in advance, they have to be discovered dynamically by market interaction. At the *market level* (section 5), we use Multi-Attribute Utility Theory (MAUT) [6] to evaluate and negotiate the optimal exchanges between agents or organizations, given their multiple interdependent objectives and their preferences. This is done by giving a constraint optimization formulation to the MAUT problem and by using the same MAXSAT constraint optimization solver to find the best offerings or responses in negotiation. We end with related work, applications and future work hints.

2 BDL REASONING INFRASTRUCTURE

Agents act and thus we first need a language to describe and reason about agent behavior (the Behavior Description Language, or BDL). A behavior consists of partially ordered goal achieving activities that satisfy given constraints. In our representation there are two kinds of goals, *composed* and *atomic*. Composed goals consist of other (sub)goals, while atomic goals do not. We allow three kinds of compositions.

Sequential compositions, $a = seq(a_1, a_2, ...a_n)$ denote that all component goals a_i must be executed in the given order.

Parallel compositions, $a = par(a_1, a_2, ...a_m)$ denote that all component goals must be executed, but without imposing any order.

Choice compositions, $a = choice(a_1, a_2, ...a_p)$ denote execution of only a non-empty subset of sub-goals, also without imposing any order. *Exclusive choices* ($xchoice$) require the execution of exactly one component sub-goal.

From the execution viewpoint, choices have *or* (*xor* for xchoices) semantics in that a choice g is 'on' - meaning will be executed and written $On(g)$ - iff at least one component is on (exactly one for xchoices) and 'off' - meaning will not be executed and written $Off(g)$ - iff all components are off. Sub-goals can occur negated within choices and parallels, but not within sequences. Thus, $c = choice(a, -b)$ denotes a choice between achieving a and not achieving b. Sequences and parallels both have *and* semantics - 'on' iff all components are on, and 'off' otherwise. The difference is that sequences also require *ordered* execution of subgoals, while parallels don't. At the planning level we address this by enforcing specific constraints for sequences. For example, $On(seq(a_i, a_{i+1})) \supset Off(seq(a_{i+1}, a_i))$. From this we derive that if a sequence is 'on', all its sub-sequences are also 'on', and if a subsequence is 'off' then all its super-sequences are also 'off'. E.g., $Off(seq(a_1, a_3)) \supset Off(seq(a_1, a_2, a_3))$.

From an agent's viewpoint, a goal is *individually achievable* if the agent acting alone can achieve it. A goal is *collectively achievable* if a coalition of agents must be put together and coordinated in order

[1] Enterprise Integration Laboratory, University of Toronto, 4 Taddle Creek Road, Toronto, Ontario, Canada, M5S 3G9, email: mihai@eil.utoronto.ca
[2] Enterprise Integration Laboratory, University of Toronto, 4 Taddle Creek Road, Toronto, Ontario, Canada, M5S 3G9, email: wklo@eil.utoronto.ca

```
Biased-WSAT(goalNetwork){
 for i = 1 to maxTries{
  A = a randomly generated on-off assignment;
  for j = 1 to maxFlips{
   if A is a solution return it;
   else {G = randomly chosen inconsistent goal;
    With probability P,
     Flip G or one of its subgoals that results in
     the greatest utility increase;
    Otherwise (with probability 1-P)
     Flip G or one of its subgoals
     that results in the greatest
     decrease in the number of inconsistent goals;}}}
 return failure;}
```

Figure 1. Biased WSAT algorithm.

```
BranchAndBound(maxUtil){
 top:while(true){
      if Complete solution found with utility=util{
         if util > maxUtil{maxUtil = util;
                           Save current solution;}
         Backtrack;
         if Backtracking not possible exit;}
     else{goal=select most constrained goal;
          Push goal onto stack;
          while(goal inconsistent with past assignments
                or current solution worse than maxUtil){
           if Another assignment for goal and
              its subgoals exists
              {Continue with top;}
              else {Backtrack;
                    if Backtracking not possible
                       exit;}}}}}
```

Figure 2. Branch and Bound algorithm.

to achieve the goal. Often, one or more collectively achievable goals are part of an agent's plans. In such cases, the agent has to determine the best set of agents to collaborate with to achieve these goals. This is addressed later on.

Planning Behavior. Let $G = \{g_1, ...g_n\}$ be a set of goals, or a goal network. An on-off labeling of the network is a mapping $L : G \rightarrow \{on, off\}$ associating either 'on' or 'off' labels to each goal in G. A labeling is *consistent* iff the labels of each composed node and of its subgoals are consistent with the node's execution semantics. E.g. if a sequence goal is 'on', then all its subgoals are 'on' and no other contradictory sequences are 'on'. If a sequence is 'off', either some subgoal is 'off' or else the ordering of the sequence is not implied by the current 'on' sequences. Consistent labelings thus define *executable* behaviors.

Let $C = \{c_1, ...c_m\}$ be a set of constraints, where each c_i is either a simple constraint of the form $On(g_j)$ or $Off(g_k)$, or an implication on both sides of which there are conjunctions of simple constraints.

Let $U = \{(g_1, u_{on}(g_1), u_{off}(g_1))...(g_l, u_{on}(g_l), u_{off}(g_l))\}$ be a utility list, that is a set of goals with their associated utilities. $u_{on}(g)$ is the utility obtained if g is achieved. $u_{off}(g)$ is the utility obtained if g is not achieved. Given a utility list U and a consistent labeling L of G, the total utility of the labeling, $Util(L, U)$, is the sum of on-utilities for the 'on' labeled goals plus the sum of off-utilities for the 'off' labeled goals.

Finally, a behavior planning problem is a tuple $P =< G, C, U, criterion >$ where $criterion \in \{max, min\}$. A problem specifies a goal network, a set of constraints, a utility list and an optimization criterion, either max or min. A solution to a problem P is a labeling L such that $Util(L, U)$ is either maximal or minimal, according to the $criterion$.

The Solver. The problem of finding a consistent labeling (without enforcing the sequencing constraints) is equivalent to satisfiability (SAT). The problem of finding a labeling that maximizes (minimizes) utility is a form of MAXSAT. The Solver we provide integrates an incomplete random search procedure based on biasing the WSAT method of [11], and a systematic (complete) branch and bound procedure, both operating over the same representation of goal networks. In both cases enforcing the sequencing constraints is done polynomially each time the main loop of the algorithm finds a solution. As known, the random search method is not guaranteed to find a solution, but performs very well on large scale problems.

The random search method (figure 1) always keeps a complete on-off assignment to goals. Goals are 'flipped' (their value is changed) with probability P (usually around 0.2) to increase the utility of the assignment (unlike the standard WSAT which flips randomly), and with probability $1 - P$ to increase the number of consistently labeled goals. In each case ties are broken in favor of the other criterion. Based on the current experience, this method has consistently produced high utility solutions, in many cases finding the optimal solution.

The branch and bound method (figure 2) is a systematic backtracking search procedure that is guaranteed to find the optimal solution. The procedure maintains the utility of the current best solution. If the partial solution currently explored can not be extended to one with better utility than the current best, then it is dropped and a new one is explored. The procedure uses the variable selection heuristic of selecting the most constrained goal first (the one with most subgoals assigned) and forward checks any proposed assignment to ensure it is not inconsistent with past assignments (figure 2). The solver provides API-s for the integration of the two methods, for example by using random search first for a number of runs and then taking the utility of the best solution produced as the bound for branch and bound.

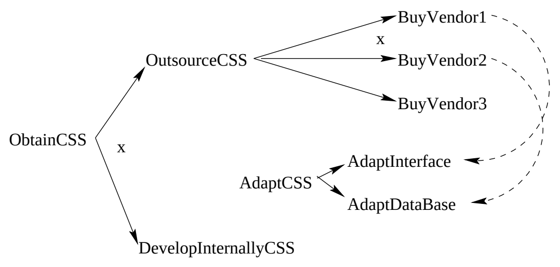
3 INDIVIDUAL LEVEL

Let us now introduce the example that will be used throughout the paper to illustrate the components of our architecture. The problem is that of an organization that needs to decide whether to make or buy a certain type of Customer Service Software (CSS). Figure 3 shows the goal network, the constraints and the best solution individual level solution, as viewed by the member of the organization who has the leading role in solving it, agent `VP-Sam`. In the figure, `ObtainCSS` and `BuyCSS` are both xchoices, while the other goals are choices.

4 ORGANIZATIONAL LEVEL

Several of the goals in figure 3 can only be achieved by *groups* of agents working together: `DevelopInternallyCSS`, `AdaptInterface` and `AdaptDataBase`. The organization will need to assemble *teams* for developing the software internally, or for adapting a vendor solution. Because of that, the utility of any behavior involving the goals in figure 3 can only be computed after considering the cost of the teams that can be formed for achieving the collective goals.

Determining these teams and selecting the min-cost one is the purpose of our organizational reasoning method. The key to the method is the use of *organization models*. Figure 4 shows such an organization model, a graph representation of the agents that can take part in achieving the collective goal `DevelopInternallyCSS`. The upper part of the figure contains all agents and groups that can be involved. Individual agents (like `ProjectMgr-Cindy`) and groups



Constraints: {On(ObtainCSS), On(BuyVendor1) => On(AdaptInterface),
On(BuyVendor2) => On(AdaptDataBase)}
Utilities: {(BuyVendor1, 3, 0)(BuyVendor2, 4, 0) (BuyVendor3, 3, 0)
(AdaptInterface, -1, 0) (AdaptDataBase, -2, 0) (DevelopInternally 4, 0)}
Best Solution: {On(ObtainCSS), On(DevelopInternallyCSS), Off(BuyCSS), Off(AdaptCSS)...}

Figure 3. Individual reasoning about the Make or Buy problem.

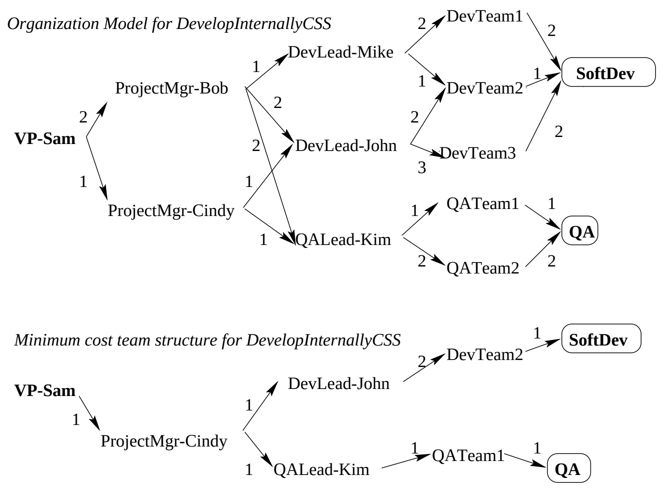


Figure 4. Organization model with team structure.

(like DevTeam2) are shown as nodes. Encircled nodes without outgoing arcs (like SoftDev and QA) represent the goals achievable by agents. Directed edges between an agent (or a group) and a goal node signify that the agent (group) will achieve the goal with the cost attached to the edge. Directed edges between two agents (groups) signify that the source node has *authority* to manage the target node for achieving some goal, according to the *social contract* that agents have agreed to abide by. These edges are labeled with the total cost of involving the target agent (group). Thus, ProjectMgr-Cindy has authority to manage both DevLead-John and QALead-Kim, at a cost of 1 each, to achieve the SoftDev and QA goals respectively. If a target agent can achieve several goals, we allow a different cost for each goal (this is not shown in the figure). Finally, we also require that any agent (group) must be on a path ending in a goal node - anybody contributes to some goal - and that an organization model is directionally acyclic.

An organization model represents the views of one agent. Each agent has a number of such models representing its beliefs about how authority is distributed, what itself and others can do and what the costs of doing things are. We use the notation $O =< a, A, E, G >$ to denote organization models representing agent a's viewpoint and containing agents (or groups) A ($a \in A$), labeled edges E and goals G. Given an organization model O, a team structure (or a coalition) T led by a and able to achieve the set of goals G is a tree rooted in a and spanning all goals G. In other words, it is a group of agents that a believes it can manage at the top, with specified management and goal execution roles that ensure that a set of goals will be achieved. The cost of a team structure T, $cost(T)$, is the sum of all costs of all edges in T.

A minimal cost team structure is a team structure that has the minimum cost in O. Figure 4 (lower part) shows a minimal cost team structure that achieves both the SoftDev (software development) and QA (quality assurance) goals for developing internally the CSS software).

Finding the Minimal Cost Team Structure. As defined above, the problem of finding a minimal cost team structure in an organization graph is equivalent to the Steiner tree problem studied in operations research [2]. Recently, [4] have shown that this NP-complete problem can be very competitively solved by a stochastic satisfiability method. They have reported very good time performance (sometimes orders of magnitude better than specialized OR methods) as well as scalability to complex problems with thousands of nodes. Based on these results, we have adopted their method for use in our architecture.

Given an organization model $O =< a, A, E, G >$, the method first pre-computes the best (least cost) k paths in the graph between the leader a and each of the nodes in G. The intuition is that the tree structured best team can be composed by assembling some of these paths together. By computing a maximum of k paths between the leader and every goal, as opposed to *all* such paths, we guarantee a propositional encoding with size linear in the number of edges in O (otherwise the encoding would be quadratic). This approximation works well in practice and there exists empirical evidence about the best values for k for various sizes of E and G (see [4] for details).

Second, we create a new MAXSAT problem P_O and a new goal network for it, according to the method of [4]. For each edge between two nodes in O we create an atomic goal. The on-utility of this goal is the cost of the edge in O. For each path between the leader and a goal in G (there are at most k such paths) we create another atomic goal. After creating all k such goals, we define a constraint imposing that exactly one of these goals can be true. That is only one path between the leader and each goal in G will be allowed. Finally, for each path goal created above, we add a constraint stating that if the goal is ‘on’ then all goals associated with the edges will be ‘on’.

The Steiner tree problem P_O thus created is solved for minimizing the sum of utilities. The result is the set of ‘on’ edges such that the total cost is minimal and there is a path between the leader and each organization graph goal. The elements of the tree (except for the leaves) are the agents (or groups) that need to be involved in the team. This team structure not only has minimal cost, but it also describes a number of aspects related to teamwork: what goal each agent is supposed to achieve, which agents are coordinators and whom they coordinate, the costs associated with every goal achievement or coordination relation. In general, the costs can be interpreted as containing the *pay-offs* to be paid to the team agents to obtain their commitment to the joint work.

In the end, to determine an agent's best behavior we have to combine the previous individual behavior planning with the organizational planning method presented here. Assuming S is a solution to the individual planning problem, there will be $S_c \subset S$ collective goals which are ‘on’. For each goal in S_c the agent has to use

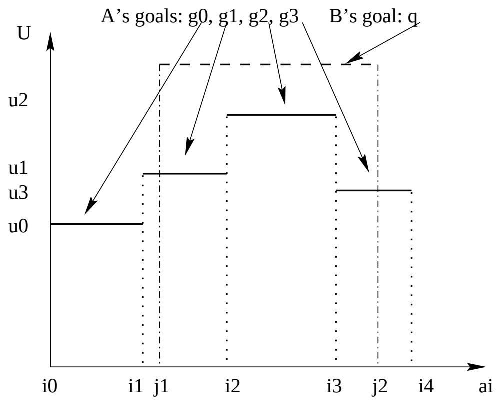


Figure 5. Representing and encoding attributes.

its corresponding organization model to generate and solve the min cost team formation problem. Then the final utility of a behavior S is $util_f(S) = util_i(S) - \sum_{c \in S_c} cost(t_c)$.

If branch and bound is used, we can generate the best solution wrt $util_f()$, by using $util_f(S)$ as the bound, because always $util_f(S) \leq util_i(S)$. This requires that the min cost team formation problem be solved at each iteration of branch and bound. Research is needed to find more efficient ways of integrating these methods in this case.

Once the min-cost team determined, the commitments of the agents composing it are obtained through negotiation. If members are not willing to commit, their pay-off can be increased (up to the cost of the next min-cost team), or new teams can be formed excluding the agents who do not wish to participate.

5 MARKET LEVEL

If our agent knew the utilities of buying the CSS software from the three vendors in figure 3, then the previous methods would have been enough to decide how to achieve its goals. In reality however, the utilities are not given in advance, they have to be discovered dynamically by interaction on the market. The utility of an offer for the CSS depends on how the offer satisfies a number of relevant and interdependent objectives, including e.g. the price, the overall quality, the level of support offered, the warranties, etc. We address this problem by using Multi-Attribute Utility Theory (MAUT) [6] in a constraint optimization formulation that allows the use of our MAXSAT solvers for determining the best offering and supports agent interaction by means of Pareto optimal negotiation protocols.

Encoding attributes. Let $A = a_1, a_2, ...a_n$ be a set of attributes shared by a number of agents. In our example, interesting attributes include the $price$, the overall $quality$, the level of $support$ etc. The domain of an attribute a_i, D_{a_i} is an interval $[l, h]$ where l and h are integers or reals. The domain describes the range of values that the attribute can take. Each value in the domain is assumed to be measured in a unit specific to the attribute. E.g., for $price$ a domain can be $D_{price} = [5, 100]$ measured in dollars. Agents interact by exchanging multi-attribute specifications formed by means of a shared set of attributes that have shared domain specifications.

An agent's utility function is approximated in the form shown in figure 5. The domain of the attribute is decomposed into a set of disjoint sub-intervals that cover the entire domain, such that on each sub-interval the utility is constant. Let $D_{a_i} = [i_0, i_1) \cup [i_1, i_2) \cup ...[i_{n-1}, i_n]$ be a decomposition of D_{a_i} into n subintervals such that $i_0 = l$, $i_n = h$ and for any $x \in [i_l, i_{l+1})$ we have $U_{a_i}(x) = u_l$ (figure 5). For each domain sub-interval $[i_l, i_{l+1}]$ we create an atomic goal $g^l_{a_i}$ which is 'on' iff the value of a_i is in the subinterval $[i_l, i_{l+1})$. As the subintervals cover the domain and are disjoint, in any situation only one of these goals can be 'on'. This is enforced by posting, for each attribute a_i, the constraint $On(xa_i)$ where $xa_i = xchoice(g^0_{a_i}, g^1_{a_i}, ...g^{n-1}_{a_i})$ (we call these attribute encoding constraints). The utility function of a_i is translated into a utility list where $u_{on}(g^l_{a_i}) = u_l$ and $u_{off}(g^l_{a_i}) = 0$.

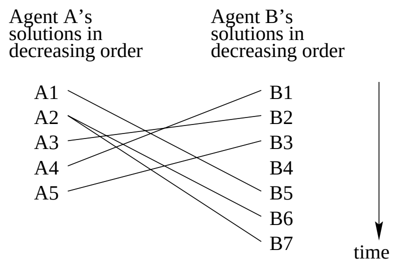
Agents may add their own constraints about what attribute values or combinations of values are acceptable. For example, an agent may only accept $support \in \{4, 5\}$, where support levels range between 0 and 5. Or an agent may accept to pay more than \$50 only if the quality is greater than some given limit. A proposal from another agent will not be accepted unless all these acceptability constraints are satisfied.

Using the common BDL infrastructure, we encode a MAUT problem as a behavior planning problem whose goals are all the goals generated for all attributes of interest, whose constraints are all the attribute encoding constraints plus all the acceptability constraints and whose utility list is obtained by merging the utility lists of each encoded attribute. A solution of a MAUT problem is an on-off assignment to the goals of the problem that satisfies all constraints. The optimal solution is the solution that has maximum utility.

Let S be a solution to agent A's MAUT problem and a_i an attribute of the problem. Because of the attribute encoding constraint, one and only one of the goals associated with the subintervals of the attribute will be 'on' in S. The subinterval associated with this goal defines the acceptable set of values for the attribute in the given solution in the sense that any value in the set is equally acceptable to the agent. Let now S_A and S_B be solutions to A's, respectively B's MAUT problem. The two solutions intersect iff for each attribute a_i the acceptable sets of values for the two agents have a non-empty intersection. In figure 5 $[i_3, j_2]$ is the intersection of A's goal g^3 and B's goal q for the represented attribute. If an intersection exists for each attribute, then the two solutions intersect. The existence of intersecting solutions represents a possible agreement between the agents, because each solution contains non-empty ranges of values for each attribute which are acceptable to each agent.

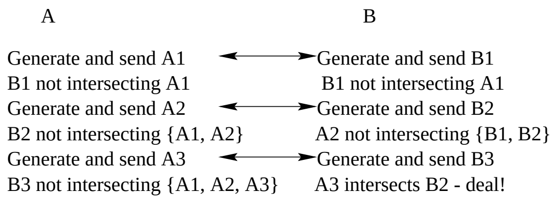
The Negotiation Process. Assume we have two negotiating agents A and B. For each agent, an acceptable solution specifies, for each attribute in part, an interval of acceptable values that the attribute can take. The branch and bound solver allows an agent to generate its solutions in decreasing order of utility. The first (best) solution is obtained by running the solver on the original constraints. The next best solution is obtained by logically negating the previous best solution, adding it as a constraint and solving the resulting more constrained problem. This allows each agent to generate all its acceptable solutions in decreasing order of utility. Figure 6 shows an interaction protocol allowing two agents to determine the best deal they can achieve given their valuations and acceptability constraints. Each agent generates its solutions in decreasing order of utility, and sends each solution to the other agent. If any agent discovers an intersection between the current solution received from the other agent and one of its own past solutions, then this is the Pareto optimal deal (no other pair of solutions can be better for both agents).

Going back to the situation in figure 3, we see that by representing the attributes of interest and the associated utility functions, our agent (the buyer) can negotiate with each of the three vendors the best deal for the CSS system. The utility of each deal can then be used as the



Agent A's solutions in decreasing order
Agent B's solutions in decreasing order
A1
A2
A3
A4
A5
B1
B2
B3
B4
B5
B6
B7
time

Connections define intersecting pairs of solutions of A and B.
(A3, B2) is the Pareto optimum.



A	B
Generate and send A1	Generate and send B1
B1 not intersecting A1	B1 not intersecting A1
Generate and send A2	Generate and send B2
B2 not intersecting {A1, A2}	A2 not intersecting {B1, B2}
Generate and send A3	Generate and send B3
B3 not intersecting {A1, A2, A3}	A3 intersects B2 - deal!

Figure 6. The negotiation process.

utility of the `BuyVendor1..3` goals. Knowing these, the agent can use the integrated individual and organizational reasoning method to discover its best course of action in the given situation.

6 CONCLUSIONS

Our integration of individual and organizational reasoning can be compared with a range of research on teamwork and coalition formation. Logical specifications for teamwork were first proposed in [7]. Practical architectures extending these specifications are reported in [13] and [3]. These models and systems address the issue of *teamwork execution*, for example stipulating when and how an agent should quit the team. We complement this work by addressing the *team design* problem as part of a single decision making problem whose solution specifies the goals to adopt, the plans to execute and the collaborations to pursue in order to maximize the agent's utility.

With respect to coalition formation, this generally includes three activities [10]. Coalition generation is the partitioning of the set of agents into sets (coalitions), usually disjoint and exhaustive. Solving the coalition problem is done by joining together the work and resources of member agents. Finally, the value generated by the activity of the coalition has to be divided among its members. [9] presents an anytime algorithm for coalition generation with worst case performance guarantees. This solution produces flat (unstructured) coalitions which specify neither what work will be done by each agent nor what pay-offs will they receive. Our Steiner tree based solution produces *structured coalitions* where the roles and goals of every agent are specified and the pay-offs to be received are determined. In [5] a coalition formation algorithm is presented which uses a two-agent auction method to determine the pay-off division and to introduce a limited structure inside a coalition (with one agent being the manager of the coalition and the others receiving an agreed pay-off).

The PERSUADER system [12] is an early system that used MAUT to resolve conflicts through negotiation in the domain of labour disputes. It has been suggested [1] that a combination of MAUT and constraint satisfaction could lead to agents that could automate multi-objective negotiations in e-commerce applications. However, we have not seen any concrete technical solution as to how this integration can be achieved, other the one presented here.

The levels of the architecture have been validated in several applications. Max-utility behavior planning has been used to determine how subscribers requests for multiple telecommunications "features" can be serviced consistently with the constraints and preferences of subscribers, thus providing a new solution to an important industry problem known as "feature interaction". The MAUT based negotiation system has been used to negotiate the acquisition of electronic components.

The next step is the integration of all levels into a significant application. Toward this goal, we are developing a supply chain system where market level interaction determines what to buy from which suppliers, team formation determines the teams of contractors to be involved in manufacturing an order and individual level planning decides on the best course of action for each team member. To integrate an application on this scale, we are extending the BDL infrastructure in a number of directions. First, we are integrating probabilistic reasoning to allow agents to make more informed decisions under uncertainty. At the market level, probabilities will be used to measure the likelihood of partners accepting offers, pruning unlikely offers and shortening the duration of negotiation. Second, BDL specifications will be made more expressive by allowing temporal and resource constraints. This will require the addition of a scheduler and a temporal execution system, turning BDL into a more complete behavior planning and execution infrastructure.

7 ACKNOWLEDGEMENTS

This research is supported, in part, by Materials and Manufacturing Ontario, Mitel Corp., Metex Systems and Communications and Information Technology Ontario.

REFERENCES

[1] Gutman, R., Moukas,A., and Maes, P. Agent Mediated Electronic Commerce: A Survey. *Knowledge Engineering Review*, June 1998.

[2] Hwang, F.K, Richards, D.S. and Winter, P. The Steiner Tree Problem. North-Holland (Elsevier Science Publications), Amsterdam 1992.

[3] Jennings, N. R. Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. Artificial Intelligence, 75 (2) pp 195-240, 1995.

[4] Jiang, Y., Kautz, H. and Selman, B. Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAXSAT. *First International Joint Workshop on Artificial Intelligence and Operations Research*, Timberline, Oregon, 1995.

[5] Ketchpel, S. Forming Coalitions in the Face of Uncertain Rewards. *Proc. of AAAI-94* vol1, 414-419, Seattle, WA, July 1994.

[6] Keeney,R. and Raiffa, H. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. *John Willey & Sons*, 1976.

[7] Levesque, H., Coehn, P., Nunes, J. On acting together. *Proc. of AAAI-90*, Boston. MA, 1990, 94-99.

[8] Pearl, J. Probabilistic Reasoning in Intelligent Systems, *Morgan Kaufmann*, 1988.

[9] Sandholm, T.V., Larson, K., Andersson, M., Shehory, O., and Tohme, F. Anytime Coalition Structure Generation with Worst Case Guarantees. *Proc. of AAAI-98*, Madison, WI, July 1998, 46-53.

[10] Sandholm, T. V. and Lesser, V.R. Coalitions among computationally bound agents. *Artificial Intelligence* 94, 1997, 99-137.

[11] Selman, B., H.J. Levesque and D. Mitchell. 1992. A new method for solving hard satisfiability problems. *Proceedings of AAAI-92* San Jose, CA, pp. 440446.

[12] Sycara, K. The PERSUADER. In *The Encyclopedia of Artificial Intelligence*, D. Shapiro (ed), JohnWilley & Sons, January 1992.

[13] Tambe, M. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7, 1997, 83-124.