Planning in Multiagent Expedition with Collaborative Design Networks

Y. Xiang and F. Hanshar

University of Guelph, Canada

Abstract. DEC-POMDPs provide formal models of many cooperative multiagent problems, but their complexity is NEXP-complete in general. We investigate a sub-class of DEC-POMDPs termed *multiagent expedition*. A typical instance consists of an area populated by mobile agents. Agents have no prior knowledge of the area, have limited sensing and communication, and effects of their actions are uncertain. Success relies on planing actions that result in high accumulated rewards. We solve an instance of multiagent expedition based on collaborative design network, a decision theoretic multiagent graphical model. We present a number of techniques employed in knowledge representation and demonstrate the superior performance of our system in comparison to greedy agents experimentally.

1 Introduction

Decentralized partially observable Markov decision processes (DEC-POMDPs) (e.g., [1]) extend POMDPs to multiagent systems. DEC-POMDPs provide formal models of many cooperative multiagent problems. However, in general, their complexity is nondeterministic exponential time complete (NEXP-complete) [2].

We consider a sub-class of DEC-POMDPs which we term as *multiagent expedition*. A typical instance consists of a large area populated by objects as well as mobile agents. Activities of agents include moving around the area, avoiding dangerous objects, locating objects of interests, and manipulating objects in various ways depending on the nature of application. The effect of an action is generally uncertain. Agents have no prior knowledge on the area. That is, they do not know, *a priori*, where dangerous or interesting objects are located. Instead, they try to identify nearby objects based on limited sensing of the local environment. Successful manipulation of an interesting object sometimes requires proper actions of a single agent and sometimes requires cooperation of multiple agents through limited communication. The success of an agent team depends on the number of objects successfully manipulated as well as the quality of each manipulation. Practical examples of multiagent expedition include undersea adventure, planet expedition, disaster rescue, anti-air defense, etc. In this paper, we specify precisely the instance of multiagent expedition used in this investigation.

Our knowledge representation is based on collaborative design networks (CDNs) originally proposed [10, 11] as a decision-theoretic framework for multiagent, optimal industrial design. The expressive power of the framework, however, goes beyond design. As long as domain dependencies can be encoded into sparse and distributed graphical structures, the framework supports autonomous, optimal,

and efficient multiagent decision making. To further explore its generality, this work investigates the application of CDNs to multiagent expedition.

Section 2 introduces background on CDNs. The instance of multiagent expedition that we investigate is specified in Section 3. How we approach this generally intractable problem computationally is detailed in Section 4 and our experimental results are reported in Section 5. Additional related work is discussed in Section 6 after presenting ours, to facilitate comparison.

2 Background on CDNs

We briefly review background on CDNs, whose full technical details can be found in [10, 11]. CDNs are motivated by collaborative industrial design in supply chains. An agent responsible for the design of a component encodes its design knowledge and preference into a *design network* (DN) $S = (V, G, P)$. The *domain* is a set of discrete variables $V = D \cup T \cup M \cup U$, where D, T, M, U are disjoint. D is a set of *design parameters*. T is a set of *environmental factors* (working conditions) of the product under design. M is a set of objective *performance measures* and U is a set of subjective *utility functions* of the agent's principal.

The dependence *structure* $G = (V, E)$ is a directed acyclic graph (DAG) whose nodes are mapped to elements of V and whose set E of arcs is from the following legal types: Arc (d, d') $(d, d' \in D)$ signifies a design constraint. Arc (d, m) $(m \in M)$ represents dependency of performance on design. Arc (t, t') $(t, t' \in T)$ represents dependency between environmental factors. Arc (t, m) signifies dependency of performance on environment. Arc (m, m') defines a composite performance measure. Arc (m, u) $(u \in U)$ signifies dependency of utility on performance. Fig. 1 (a) shows a trivial DN.

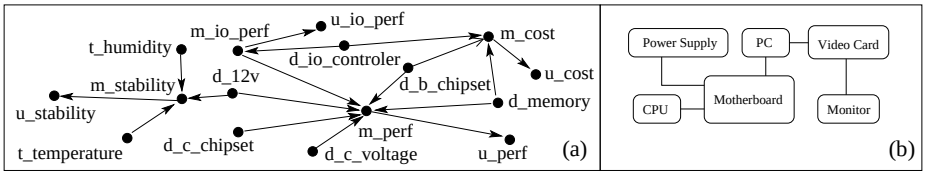

Fig. 1. (a) A trivial DN for PC motherboard. First letter of a node label indicates its type. (b) Hypertree of a simple CDN.

P is a set of potentials, one for each node x, and each is formulated as a probability distribution $P(x|\pi(x))$, where $\pi(x)$ is the parent nodes of x, but its semantics depends on x. $P(d|\pi(d))$ encodes a design constraint. $P(t|\pi(t))$ and $P(m|\pi(m))$ are typical probability distributions. Each utility variable has the domain $\{y, n\}$. $P(u = y|\pi(u))$ is a utility function $u(\pi(u))$ whose values range in $[0, 1]$. $P(u = n|\pi(u))$ is assigned $1 - P(u = y|\pi(u))$. Each node u is assigned a weight $k \in [0, 1]$ such that weights of all utility nodes sum to one.

With P thus defined, $\prod_{x \in V \setminus U} P(x|\pi(x))$ is a joint probability distribution over $D \cup T \cup M$. With the assumption of additive independence among utility variables, the expected utility of a design $\mathbf{d}$ is $EU(\mathbf{d}) = \sum_i k_i \, (\sum_{\mathbf{m}} u_i(\mathbf{m}) \, P(\mathbf{m}|\mathbf{d}))$, where $\mathbf{d}$ (bold) is a configuration of D, i indexes utility nodes in U, $\mathbf{m}$ (bold) is a configuration of parents of u_i, and k_i is the weight of u_i.

Each supplier in a supply chain is a designer of the supplied component. Agents, one per supplier, form a collaborative design system. Each agent embodies a design network called a design *subnet* and agents are organized into a *hypertree*: Each hypernode corresponds to an agent and its subnet. Each hyperlink (called *agent interface*) corresponds to design parameters shared by the two subnets (referred to as *public* variables). Hypertree organization specifies to whom an agent can communicate directly. Each subnet is assigned a weight w_i, representing a compromise of preferences among agents, and $\sum_i w_i = 1$. The collection of subnets $\{S_i = (V_i, G_i, P_i)\}$ forms a CDN. Fig. 1 (b) shows the hypertree of a simple CDN for customized PC design, where the subnet on motherboard is shown in (a).

The product $\prod_{x \in V \setminus \cup_i U_i} P(x|\pi(x))$ is a joint probability distribution over $\cup_i (D_i \cup T_i \cup M_i)$, where $P(x|\pi(x))$ is associated with node x in a subnet. The expected utility of a design $\mathbf{d}$ is $EU(\mathbf{d}) = \sum_i w_i \, (\sum_j k_{ij} \, (\sum_{\mathbf{m}} u_{ij}(\mathbf{m}) \, P(\mathbf{m}|\mathbf{d})))$, where $\mathbf{d}$ is a configuration of $\cup_i D_i$, i indexes subnets, j indexes utility nodes $\{u_{ij}\}$ in ith subnet, $\mathbf{m}$ is a configuration of parents of u_{ij}, and k_{ij} is the weight associated with u_{ij}. Through communication along the hypertree, agents can determine the optimal design $\mathbf{d}^*$ that has the maximum $EU(\mathbf{d})$ [11].

3 The Multiagent Expedition Testbed

The following instance of multiagent expedition is used in our investigation: The area is abstracted as a grid of cells. At any cell, an agent has five possible actions: moving to an adjacent cell along one of four directions (referred to as *north*, *south*, *east*, *west*) or remaining in the current cell (referred to as *halt*). The effect of an action is, however, uncertain. That is, the action *north* may cause the agent to land on each of four unintended cells.

The desirability of an object (located at a cell) is indicated by a numerical *reward*. For simplicity, we abstract *away* the object and associate the reward with the cell. A cell that is neither interesting nor harmful has a reward of a base value. The reward at a harmful cell is lower than the base value. The reward at an interesting cell is higher than the base value and can be further increased through agent cooperation. When a physical object at a given location is to be manipulated (e.g., digging, lifting, pushing, etc.), cooperation is often most effective when a certain number of agents are involved, and the per-agent productivity is reduced when less or more agents are involved. We set the most effective level at 2, although other levels can also be used. For instance, the reward that can be collected by a single agent from a given cell may be 0.3. However, if two agents cooperate and *meet* at the cell, each receives 0.4. If three or more agents meet at the cell, two of them each receives 0.4 and the other

agents receive the base value. This feature promotes effective cooperations and discourages unproductive ones. It is encoded by associating each cell with a reward pair (r_1, r_2), where r_1 is the reward collected by a single agent and r_2 is the total reward collected by two cooperating agents. Hence, the above mentioned cell has the reward $(0.3, 0.8)$. After a cell has been visited by any agent, its reward is decreased to the base value. As a result of this feature, wandering within a neighborhood will not be productive and agents must move around strategically.

Agents have no prior knowledge about the area on how the rewards are distributed. Instead, at any cell, an agent can perceive the cell's absolute location (e.g., through GPS on Earth or triangulation with two base stations on Mars). It can also perceive the reward distribution in its neighborhood. We set the neighborhood to be the 13 cells shown in Fig. 2 (a), although different settings are also possible. An agent can also perceive the location of another agent if the latter is within a 10 step radius. It can communicate with agents within this radius as well. The objective of agents is to move around the area, cooperate as needed, and maximize the team reward over a finite horizon. They must do so based on local observations and limited interagent communication.

This instance of multiagent expedition is a DEC-POMDP. The state of the environment is described by the location of all agents as well as the distribution of rewards. It is *stochastic* since the effect of actions are uncertain. It is *Markovian* as the new state is conditionally independent of the history given the current state and the joint action of agents. It is *partially observable* because each agent can only perceive its neighborhood, but not the distribution of rewards and agents beyond.

Because an agent can perceive its own location and agents nearby, a significant amont of relevant information in the environmental state is obtained through observation. On the other hand, not all relevant information has been obtained, because knowing the reward distribution beyond the neighborhood will allow the agent to plan better. To capture this difference from the case where agents cannot perceive its own location reliably, we refer to the stochastic process as a decentralized weakly partially observable Markov decision process (DEC-W-POMDP).

4 A Decision-Theoretic Graphical Models Approach

Consider the general case of the problem instance with n agents and horizon k. Given the current positions of agents, each agent has five possible actions. Hence, there are 5^n joint actions and 5^{nk} joint plans of horizon k. Since each action has five possible effects, a joint action has 5^n possible effects, a joint plan has 5^{nk} possible effects, and the 5^{nk} joint plans have a total of 5^{2nk} possible effects. Each automonous agent needs to evaluate these effects, identify the optimal joint plan, and obtain its own optimal action sequence. For six agents and horizon 2, each agent needs to evaluate $5^{24} \approx 6 \times 10^{16}$ possible effects. To carry out the computation more efficiently, we take the following measures:

Splitting Agent Team into Groups We divide n agents into smaller groups to allow high inner-group interaction and low inter-group interaction. Grouping has no negative effect on scaling up. It allows group members to stay closely so that they can cooperate effectively, as long as the group size is no smaller than the number of agents to be involved in a most effective cooperation. It allows different groups to stay away from each other, which not only allows the team to explore the area more effectively, but also allows reduction of computation by reducing inter-group interaction. In this work, we consider group size of 3, although larger sizes can also be used. We present inner-group interaction first and inter-group interaction later. We also limit horizon to 2. These two measures allow the per-agent evaluation to be reduced to $5^{12} \approx 2.4 \times 10^8$ possible effects.

Graphical Modeling These effects are evaluated by agents using a CDN. We denote the three agents in a group by A, B and C. The subnet dependence structure for B is shown in Fig. 2 (b). The subnets for A and C have the same structure. In the subnet, each decision variable $mv^{x,i}$ has 5 possible values. Each

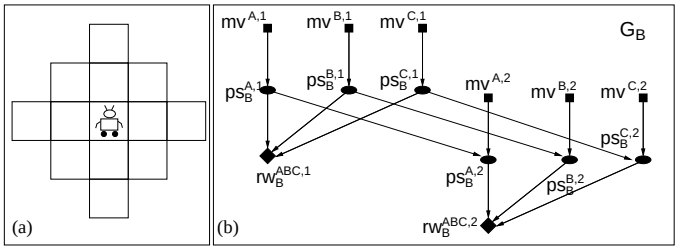

Fig. 2. (a) The 13 neighborhood cells whose rewards are perceivable by the agent. (b) Subnet structure for agent B. Design parameter $mv^{A,1}$ denotes first movement decision of agent A and is a public variable. Performance measure $ps_B^{A,1}$ denotes position of A after first movement and is a private variable in B. The utility variable $rw_B^{ABC,2}$ denotes reward received by B after second movement due to interaction with A and C.

position variable $ps^{x,1}$ has five possible values and each position variable $ps^{x,2}$ has 13 possible values. Each reward variable $rw_B^{ABC,i}$ is binary. After compilation, its runtime representation is a cluster tree of 8 clusters (see Fig. 5 (a) for an alternative cluster tree). The size of total state space of this cluster tree (the total number of probability potential values) is 619244. For each round of planing, agent B must process this state space once for each of the $5^6 = 15625$ joint plans. Agents A and C incur the same amount of computation. The entire round of group planing takes 7380 seconds (2 hours and 3 minutes) running in IBM ThinkPad 2GHz Core Duo (Java implementation without runtime optimization).

Restricting Inner-Group Interaction To speed up the computation, we restrict inner-group agent interaction to pairwise. We only allow direct cooperation between A and B and between B and C. This essentially imposes an organizational structure $A - B - C$ for the group. This restriction will not jeopardize

the effectiveness of agent cooperation because the number of agents who can cooperate equals the most effective level of cooperation of the environment (see Section 3).

The subnets for agents A and B in the resultant CDN are shown in Fig. 3. The subnet for C is similar to that of A. From (a), it can be seen that variables

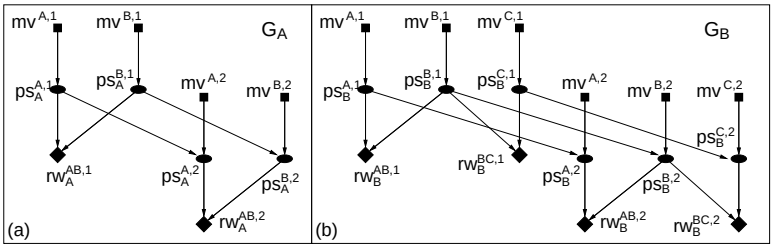

Fig. 3. (a) Subnet for agent A. (b) Subnet for agent B.

corresponding to agent C have disappeared from G_A (the similar occurs with G_C). In (b), the rewards due to B's cooperations with the other agents have now been decomposed.

The group organization $A - B - C$ does not, by itself, prevent interactions between A and C as they could still meet even though the meeting is not planed. In fact, the group could behave such that all three are trying to meet: an unproductive cooperation. To prevent such behavior, the planning computation steers A and C away from each other. How to achieve this is presented below.

Guiding Agents with Group Direction To reduce the state space of agent cluster trees, we require that agents' movements be guided by a *group direction*. If the current group direction is *north*, then A and C are not allowed to attempt *south*.

Adopting the group direction allows reduction of the space for variables $mv^{A,i}$ and $mv^{C,i}$ by one alternative action, thus reducing the state space of agent cluster trees. Furthermore, this restriction allows the agent group to move less randomly and more strategically because the movements of A and C are better coordinated. Note that even though A and C are not allowed to attempt *south* in the above scenario, they may still move to south due to the uncertain effect of their movement actions.

Because group direction affects the spaces of public variables $mv^{A,i}$ and $mv^{C,i}$, at any time, the three agents must agree on what is the current group direction. This is achieved by two measures: First of all, group members are required to stay within the 10 step radius to each other. We elaborate later how agents can achieve this through planning with CDN. The consequence is that it allows group members to perceive each other's position. Second, a common algorithm is used by group members to compute the group direction based on their positions.

The algorithm handles a given situation depending on whether there is a meeting. As mentioned above, the planning computation prevents meeting of the group. Hence, a meeting can occur only between A and B or between B and C. In such a case, the group direction is pointing from A to C. If there is no meeting, the group direction is determined according to the maximum angle in the agent triangle, as illustrated in Fig. 4 (a). Given positions of group members,

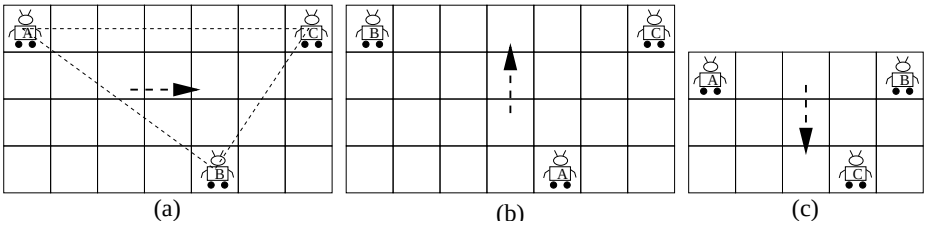

Fig. 4. Determine group directions based on agent positions.

a triangle is formed with angles, $\angle A$, $\angle B$ and $\angle C$. If a maximum angle exists, it must be either $\angle B$ as in (a), or $\angle A$ as in (b), or $\angle C$ as in (c). The dashed arrow indicates the group direction in each case. If a maximum angle does not exist, a default direction can be used. We omit computational details here due to space limitations.

The method enforces the following properties: First, it steers the group to move in formation $A - B - C$. Agent B is positioned between A and C, and three of them tend to arrange into a straight line. This helps prevent unwanted direct interaction between A and C. Second, it steers the group to move in the direction pointing from A to C. Therefore, the group direction will not change dramatically from move to move, promoting strategic group movement and avoiding wandering around in a small confined region. Third, the group direction does not dictate individual agent movement rigidly. Each agent still has enough flexibility to choose its action. For instance, suppose that bottom right cell in Fig. 4 (c) has a high r_2 value. Then, B can plan to go *south* twice and C can plan to *halt* first and then go *east*.

To further reduce the state space of agent cluster trees, we do not allow agents to attempt *halt* in the second movement. Action *halt* is necessary for two agents to meet when they are in certain relative positions such as that of Fig. 4 (c). Our stipulation will force C to halt in the first step. This requirement, combined with the previous measures, reduces the space of variables $mv^{A,2}$ and $mv^{C,2}$ to size 3 and that of $mv^{B,2}$ to size 4.

The resultant cluster trees for agents A and B are shown in Fig. 5. The cluster tree for C is similar to T_A. The size of total state space of T_B is 48169: about 12-fold reduction. The size of total state space of T_A is 10152: about 61-fold reduction. One round of group planing takes 135 sec (Java implementation without runtime optimization): about a 55-fold speed up.

Enforcing Desirable Behavior Through Utility As mentioned above, within a group, we expect cooperation between A and B and between B and C, but

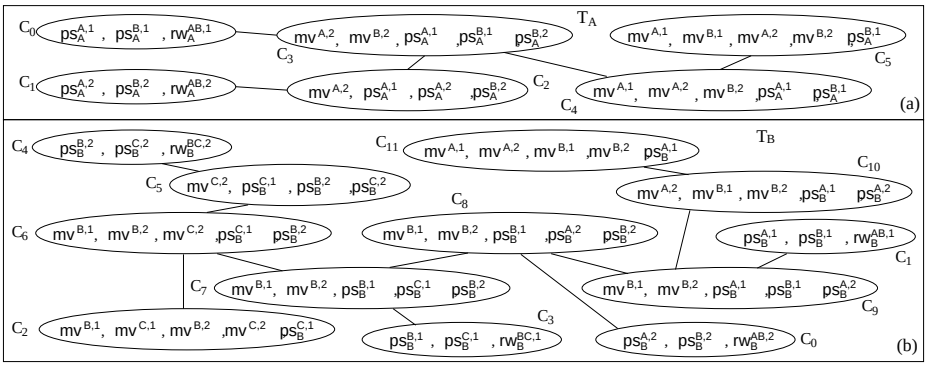

Fig. 5. (a) Cluster tree T_A for agent A. (b) Cluster tree T_B for agent B.

avoidance of direct interaction between A and C. We also expect different groups to avoid each other. We would like to achieve such desirable behavior of agents through reasoning within CDN. To do so, we replace each reward variable $rw_y^{x,i}$ by a new utility variable $ut_y^{x,i}$. Recall that $rw_y^{x,i}$ is a binary variable and distribution $P(rw_y^{x,i} = y|\pi(rw_y^{x,i}))$ associated with $rw_y^{x,i}$ encodes utility function $u(\pi(rw_y^{x,i}))$, where $\pi(rw_y^{x,i})$ is the parent nodes of $rw_y^{x,i}$ and consists of position variables $ps_y^{z,i}$. The distribution $P(ut_y^{x,i} = y|\pi(rw_y^{x,i}))$ associated with $ut_y^{x,i}$ is obtained by modifying $u(\pi(rw_y^{x,i}))$ as follows:

If $\pi(rw_y^{x,i})$ corresponds to a group configuration where either A and B are too far away (according to a distance threshold), or B and C are too far away, or A and C are too close, or agent y is too close to any other agent outside the group, $P(ut_y^{x,i} = y|\pi(rw_y^{x,i}))$ is set to 0. Otherwise, $P(ut_y^{x,i} = y|\pi(rw_y^{x,i})) = u(\pi(rw_y^{x,i}))$.

Jumping in Barren Area When a group moves into a *barren* area where all cells have the reward at or below base value, any movement not causing danger will be regarded by the agents just as good as any other. The group movement will then be dominated by danger avoidance and the group could wander around the barren area forever: an unproductive behavior.

To avoid being so trapped, agents could follow an exception rule outside CDN-based planning and move in the group direction for a number of steps. However, such *blind jump*, combined with the uncertain effect of actions, may violate intra and inter-group formation or enter dangerous cells.

We have instead let agents plan their *jump* through CDN. Jumping is triggered by the inference computation with CDN when the optimal group plan has a utility at or below the base value. In the next step, each agent checks the observed rewards for the next two cells in the group direction. If their reward values do not correspond to danger, the cells will be treated as if their reward values are 1 (the highest value) during planning with CDN.

This solution has the following advantages: First, the jumping behavior is produced in the uniform computational framework of normal movement, which facilitates analysis and implementation. Second, danger avoidance is enabled.

Third, if the target cells do not correspond to danger, the high reward values will be used in the distributions associated with variables $rw_y^{x,i}$. Because these distributions are subject to the modification into those associated with $ut_y^{x,i}$, as mentioned above, desirable behavior regarding intra and inter-group formation will be enforced during jumping. In short, this solution implements jumping along group direction as a soft guideline and combines it with other criteria naturally.

5 Experimental Results

To empirically verify the effectiveness of our method, an *Environment Simulator* is implemented as well as the agents. The Simulator models the grid, reward distribution and effect of agent actions (the probability of intended effect of an action is set at 0.9). It feeds agents with observations and updates environmental state according to agent actions. Each agent communicates with group members according to CDN hypertree and with Simulator on observations and action decisions. Agent performance is measured by rewards collected over a finite horizon.

To test the robustness of our method under different environments, three types of reward distributions were used. In a *dense* distribution, every 10×10 region has at least one high reward cell. In a *barren* distribution, high reward cells are located in clusters. Each cluster is no larger than a 6×6 region and two clusters are at least 20 cells apart. In a *path* distribution, some high reward cells form a pathway and, along the pathway, a high reward cell is no less than two cells away from another one. Examples distributions are shown in Fig. 6. For each distribution type, the initial locations of agents are identical in all runs.

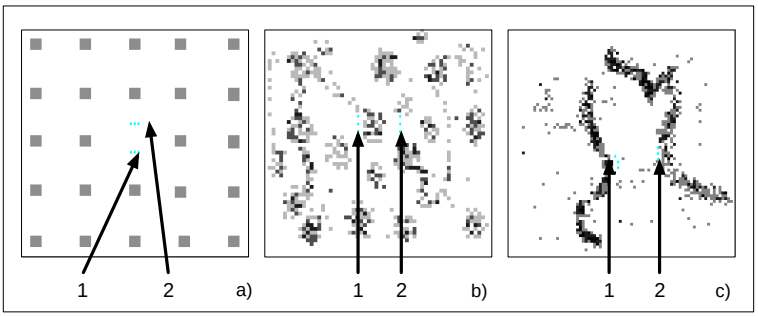

Fig. 6. Example distributions. Left: barren (120 by 120); middle: dense (60 by 60); right: path (80 by 80). Each arrow indicates the starting locations of three agents.

To compare our CDN agents with simpler but non-trivial alternatives, we implemented two types of greedy agents. The domain of decisions mv^1 and mv^2 is $\{north, south, east, west, halt\}$. The first type (GRD) is based on unilateral reward rw_u and maximizes $rw_u(mv^1)+rw_u(mv^2)$, where $rw_u(mv^i)$ is the unilateral reward of the i'th movement. The second (GRDB) considers bilateral reward rw_b as well and maximizes $rw_u(mv^1) + rw_u(mv^2) + rw_b(mv^1) + rw_b(mv^2)$. For

both types, each agent acts independently and there is no direct communication between agents.

Table 1 shows the experimental results. A team of six agents of the same type is used for each run which lasts for 80 time-steps. Each type of agents have five runs for each type of distribution.

Table 1. Experimental results. Mean rewards μ, as well as max and min, are obtained from 5 runs. Highest means for each distribution is shown as bold.

	CDN			GRD			GRDB		
	μ	max	min	μ	max	min	μ	max	min
barren	**51.82**	53.9	48.3	49.42	49.8	48.8	49.28	49.8	48.4
dense	**72.46**	81.5	59.8	60.52	62.4	58.4	68.56	76.5	58.1
path	**96.48**	100.5	91.1	72.56	78.2	65.2	74.44	84.6	68.7

On average, CDN agents collected highest rewards across all three types of distributions. Results for barren distribution were the closest among the three types of agents. As high reward clusters were far apart, agents' observation range is far below minimal distance between such clusters, and agents were started far from any such clusters, performance of a single run is more subject to chance. As a result, the minimal reward collected by CDN agents were lower than the minimal collected by greedy agents, even though on average CDN agents still outperform. For dense and path distributions, CDN agents outperform greedy agents on every run. For path distribution, the lowest reward collected by CDN agents was higher than the highest reward collected by greedy agents. The superior CDN performance may be attributed to at least two properties. First, CDN agents can plan bilateral visiting of a cell, whereas greedy agents have no direct coordination. Second, CDN agents keep groups apart and thus can explore different regions of the environment and avoid revisiting cells. Of two types of greedy agents, GRDB type outperforms GRD in dense and path distributions since it considered bilateral actions whereas GRD did not.

6 Other Related Work

The space precludes an extensive literature review and we discuss only a small subset which is considered the most relevant.

Mazes have been abstracted from office delivery applications and used in empirical study of centralized POMDP algorithms, e.g. [5]. A typical maze consists of walls, hallways, rooms and a single agent. The agent must travel to a goal location through a long sequence of movements. The agent knows the topology of the maze but may not know its starting location. Its sensors can perceive nearby walls but are noisy. In multiagent expedition, at any time, multiple alternative goals (of different reward) exist for each agent and each requires a short sequence of movements. The objective of planning is to choose among these goals wisely. Agents have no prior knowledge of the environment and the environment is multiagent.

Also abstracting from office delivery applications, Pollack and Ringuette [7] proposed Tileworld multiagent testbed, where agents' goals are to push tiles into

holes. As multiagent expedition, a Tileworld agent can pursue one of multiple alternative goals at any time; but unlike multiagent expedition, each goal requires a long sequence of movements. The environment is fully observable (agents can perceive tiles, holes and other agents) and deterministic (actions have intended effects). In contrast, the environment of multiagent expedition is weakly partially observable (agents cannot perceive beyond local region) and stochastic (effects of actions are uncertain).

Tiles and holes in Tileworld dynamically appear and disappear. The feature is reasonable in office delivery applications but insensible in the expedition environment. In multiagent expedition, after being visited by any agent, a cell's reward is reduced to the base value. As a consequence, wandering in the same neighborhood is unproductive and agents must move around strategically.

Work on independent DEC-MDPs [1] shares some features with multiagent expedition. It assumes that actions of one agent cannot affect other's observation and state, and an agent cannot observe other agent's state and communicate with them. In multiagent expedition, agents can observe the state of others if they are close by, they must plan to meet to maximize reward, and CDN utilizes limited inter-agent communication to achieve optimal joint plan.

Noh and Gmytrasiewicz [6] applied the recursive modeling method (RMM) to agents cooperating in anti-missile defense. In their environment, incoming missiles are fully observable. Uncertainty originates from the unknown state of other agents as well as the effect of intercept action.

Artificial birds [8] display formation behavior somewhat similar to the group formation presented in this work. The formation can be viewed as the ends of their birds and the behavior is generated by following simple rules. The formation of agents in expedition is the means to serve the ends. It is generated as part of the desirable behavior through decision-theoretical planning.

Multiagent expedition differs from *exploration*. As commonly referred, e.g., [4, 9], the main task of exploration is to produce a map in an unknown environment by moving around and sensing. The map produced can then be used for navigation as in mazes described above. Multiagent expedition, as we presented, does not require a map.

In posing the challenge of Mars rover operations [3], the need to take resource constraints and concurrent actions into account in planning is emphasized. CDNs encode constraints explicitly through design parameters and address concurrent actions through multiagent planning. Hence, CDN based planning provides a promising research direction towards meeting the challenge.

7 Conclusion

Multiagent expedition forms a class of DEC-W-POMDPs and captures a number of practical applications. In this work, we solve one instance of DEC-W-POMDP. The knowledge representation is based on CDN, a formal decision theoretic graphical model that supports autonomous, optimal, and efficient multiagent decision making. The generality of CDN allows incorporation of grouping, group direction, jumping and comprehensive preference encoding to achieve efficiency while maintaining optimality. Experiment shows superior performance of CDN

agents over greedy agents. Our method can easily scale up to a larger number of agents by employing more groups as well as a larger group size. The addition of groups essentially has no impact on complexity. Large group size with the hyperchain group organization maintained will cause increase of complexity linear on the group size.

Theoretical and experimental comparison of our approach to RMM [6] is underway, through which we hope to gain a better understanding of the pros and cons of our tightly-coupled agent architecture and their loosely-coupled one.

Acknowledgement

Financial support to the first author through Discovery Grant from NSERC, Canada and that to the second author through OGS from MTCU, Ontario are acknowledged.

References

1. R. Becker, S. Zilberstein, V. Lesser, and C.V. Goldman. Solving transition independent decentralized Markov decision processes. *J. Artificial Intelligence Research*, 22:423–455, 2004.
2. D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *Proc. 16th Conf. on Uncertainty in Artificial Intelligence*, pages 32–37, Stanford, 2000.
3. J. Bresina, R. Dearden, N. Meuleau, S. Ramkrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: a challenge for ai. In *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, pages 77–84, San Francisco, CA, 2002. Morgan Kaufmann.
4. S. Carpin, H. Kenn, and A. Birk. Autonomous mapping in the real robot rescue league. In D. Polani, B. Browning, A. Bonarini, and K. Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII, Lecture Notes in Artificial Intelligence (LNAI) 3020*. Springer, 2004.
5. M.L. Littman, A.R. Cassandra, and L.P. Kaelbling. Learning policies for partially observable environments: scaling up. In A. Prieditis and S. Russell, editors, *Proc. 12th Inter. Conf. on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann.
6. S. Noh and P.J. Gmytrasiewicz. Coordination and belief update in a distributed anti-air environment. In *Proc. 31st Annual Hawaii Inter. Conf. on System Sciences*, pages 142–151, 1998.
7. M. Pollack and M. Ringuette. Introducing the Tileworld: experimentally evaluating agent architectures. In T. Dietterich and W. Swartout, editors, *Proc. 8th National Conf. on Artificial Intelligence*, pages 183–189, Menlo Park, CA, 1990. AAAI Press.
8. C.W. Reynolds. Flocks, herds, and schools. In *Computer Graphics, 21, Proc. SIGGRAPH'87*, pages 25–34, 1987.
9. S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.
10. Y. Xiang, J. Chen, and A. Deshmukh. A decision-theoretic graphical model for collaborative design on supply chains. In A.Y. Tawfik and S.D. Goodwin, editors, *Advances in Artificial Intelligence, LNAI 3060*, pages 355–369. Springer, 2004.
11. Y. Xiang, J. Chen, and W.S. Havens. Optimal design in collaborative design network. In *Proc. 4th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'05)*, pages 241–248, 2005.