Task and Result Sharing in Multi-Agent Systems

John Cardente
CS525m: Multi-Agent Systems, Spring 2002
Worcester Polytechnic Institute
cardente_john@emc.com
Higher Context: Distributed Problem Solving

- Motivations:
  - Speed up through parallelization
  - Distribution of expertise
  - Distribution of Data
  - Distribution of Results

- General Steps
  - Task decomposition
  - Task allocation
  - Task accomplishment
  - Results Synthesis

Task Sharing

Results Sharing
Task Decomposition

• Partitioning of a task into sub-tasks for possible allocation to another agent
• Goal is to make sub-tasks as independent as possible to:
  – Minimize coordination
  – Minimize shared data
  – Minimize share resources
• Task decomposition is a hard problem and generally performed a priori by system designers.
Task Allocation

• Homogenous Systems
  – Agents identical, allocation simple since each is equally qualified to work on sub-tasks

• Heterogeneous Systems
  – Sub-task requirements must be matched to agent skills
  – Potentially difficult problem
Which kind of system to build?

- Homogenous systems are simpler
  - Only one kind of agent to build
  - Don’t have to consider agent skills when distributing sub-tasks
- Homogenous systems considered **unsuitable** for *complex* problems
  - Low overall utilization of skills and resources
- In general heterogeneous systems preferred for complex problems
  - Therefore need to solve the problem of matching tasks to skills
Agent Roles in Task Allocation

• Agents can assume two roles
  – **Servers**: Agents capable of providing a service
  – **Clients**: Agents requiring a service to be performed
• Agents can be both clients and servers simultaneously
  – I.e. An agent may use the services of other agents to complete a service is to providing to another agent
• Task allocation systems must provide a way to match clients with servers
Centralized Allocation Systems

- Centralized
  - 3rd party manages client-server matching
  - Hierarchical Subordination
    - Superior agents order subordinates to carry out task.
    - Typically a static, pre-defined agent organization
  - “Egalitarian”
    - No subordination, all agents considered “equal”
    - Requires special “broker” or “trader” agents to manage client requests and server bids
    - Allows centralized allocation techniques to be applied in dynamic agent organizations
Example Hierarchical Allocation System

In this case the sub-agents represent different functions, not parallel resources. As a result calling them is like calling a procedure in a typical program.
Example Egalitarian Allocation System

Client

Trader

Servers

Request_A

Accept_D

Reject_C

Accept_D

A

B

C

D

Request_A
Distributed Allocation Systems

- Each agent individually attempts to obtain required services
- Acquaintance Network
  - Direct Allocation
    - Agents can only use the services of the agents it knows about
      - Potentially serious scalability issues
  - Delegated Allocation
    - Agents can ask other agents to use their acquaintances to find an agent capable of providing a particular service
      - Requires strongly connect acquaintance network
  - Both methods require accurate knowledge of agent skills (coherence)
    - May use various “caching” strategies to maintain and age acquaintance information
Distributed Allocation Systems
(cont’d)

• Contract Net
  – “Market Place” approach
    • Clients issue description of tasks
    • Servers reply with bids
    • Client chooses the best bidder
    • Server affirms its commitment
  – Proven approach from other disciplines
  – Relatively simple to implement
  – Well suited for dynamic environments
  – Concurrent and many-to-many nature of the protocol creates challenging race conditions
Example: Direct Allocation System

A is aware of

Set of Agents A is aware of

Set of Agents Skill

Request

Reject

Accept

Requests and Rejections

A

B

C

D

E

F

G
Example: Delegated Allocation System

Request$_A$ → B
Reject$_B$
Request$_A$ → C
Accept$_X$

A → D
E → F

J. Cardente  CS525m Hw1 Presentation  2/11/2002  Page 13
Example: Contract Net Allocation System

Race Condition
Taxonomy of Allocation Modes

- Allocation of Tasks
  - Centralised
    - Hierarchical (Imposed)
  - Distributed
    - Egalitarian (Trader/Broker)
    - Acquaintance
    - Contract Net
### Task Allocation System Tradeoffs

<table>
<thead>
<tr>
<th>Centralized Trader</th>
<th>Distributed Acquaintance</th>
<th>Contract Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Benefits</td>
<td>• Benefits</td>
<td>• Benefits</td>
</tr>
<tr>
<td>– Coherence</td>
<td>– No Bottleneck</td>
<td>– Proven/Simple</td>
</tr>
<tr>
<td>• Drawbacks</td>
<td>– Fault tolerance</td>
<td>– Flexibility</td>
</tr>
<tr>
<td>– Bottleneck</td>
<td></td>
<td>– Drawbacks</td>
</tr>
<tr>
<td>– Fault Intolerance</td>
<td></td>
<td>– Message volume</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Temporal &amp; Spatial Ignorance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Types of Tasks

- **Independent**
  - Tasks are self-contained in that they contain all the information needed for their solution
  - Can be performed in any order and concurrently
  - Example: Tower of Hanoi example

- **Interdependent**
  - The solutions of some sub-tasks are required for the solution of other sub-tasks
  - Coordination possible if dependencies known before hand
  - Possible for dependencies to only become apparent at runtime
  - A *Results Sharing* mechanism is needed to solve these dependencies
  - Example: component design problem
Motivations for Results Sharing

- **Confidence:**
  - Independent derivations affirm/challenge previous results leading to more confidence

- **Completeness:**
  - Combination of partial results leads to a larger set of results

- **Precision:**
  - Sharing of results allows for iterative refinement

- **Timeliness:**
  - Obvious performance benefits via parallel processing
Functionally Accurate Cooperation

- Agents eventually converge on correct result
- Iterative exchanging of results is used to:
  - Refine partial results
  - Refute incorrect partial results
- All partial results must be considered tentative
- Standard representations of results and metrics simplify results sharing
- Potential Issues:
  - Communication requirements
  - Forward progress
  - Distraction

Message frequency must be tuned to avoid BW saturation, and distraction but allow for continued forward progress.
Shared Repositories and Negotiated Search

• Single, shared repository used to communicate partial results
  – “Blackboard” approach
• Shared repository avoids need to broad/multi-cast results to other agents
• Agents with different criteria can review, critique and revise results
  – Agents allowed to relax constraints if necessary
• Agents do not need to track agents that may be impacted by results
Distributed Constrained Heuristic Search

• Used mainly in distributed resource allocation problems
• Agents manage all requests for “resources” that are assigned to them
  – “Resources” may be partial solutions
• Agents may solve contention based on
  – Real time demands
  – Some knowledge of aggregate resource demands provided by agents at system start time
• Resource contention focuses communications and solves the “difficult” problems first
• Belief is that resolving contention leads to more informed searches and less wasted effort
Organizational Structuring

- Organizes agents into an organization
  – May be based on how the task was decomposed
- Agents use knowledge of the organization to
  – Determine with whom to communicate
  – Prioritize tasks
- Agents only need to know about the local organizational structure (coherence)
- Choosing an organization structure can, itself, be a difficult problem!
Example: Shared Repository

Partial Results

- PResult_A
- PResult_A1
- PResult_B
- PResult_B1

Hypothesis

- Hypothesis_F
- Hypothesis_F1
- Hypothesis_D
- Hypothesis_D1

Poses

- Pose
- Pose_F
- Pose_F1

Refutations

- Refute
- Refute_D
- Refute_D1
Example: Organizational Structuring

Geographically distributed “cells”
## Results Sharing System

### Tradeoffs

<table>
<thead>
<tr>
<th>Shared Rep.</th>
<th>DCHS</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Benefits</strong></td>
<td><strong>Benefits</strong></td>
<td><strong>Benefits</strong></td>
</tr>
<tr>
<td>– Limited Messaging</td>
<td>– More efficient searches</td>
<td>– Reduced messaging</td>
</tr>
<tr>
<td>– Ignorant of dependencies</td>
<td>– Requires “resource” based paradigm</td>
<td>– Relaxed coherency</td>
</tr>
<tr>
<td><strong>Drawbacks</strong></td>
<td><strong>Drawbacks</strong></td>
<td><strong>Drawbacks</strong></td>
</tr>
<tr>
<td>– Potential Bottleneck</td>
<td>– Formulating proper organization can be very difficult</td>
<td>– Fault Intolerance</td>
</tr>
<tr>
<td>– Fault Intolerance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
References


- **Problem Structure and Subproblem Sharing in MultiAgent Systems.** Dorothy Mammen & Victor Lesser. Umass Amherst
  - http://dis.cs.umass.edu/~mammen/scholar.html

- **Methods for Task Allocation via Agent Coalition Formation.** Onn Shehory, Carnegie-Mellon University

  - http://citeseer.nj.nec.com/walsh98market.html