Coalition Formation with Non-Transferable Payoff for Group Buying

Frederick Asselin
Département d'Informatique et de Génie Logiciel
Université Laval, Québec, QC, Canada, G1K 7P4
frederick.asselin@ift.ulaval.ca

Brahim Chaib-draa
Département d'Informatique et de Génie Logiciel
Université Laval, Québec, QC, Canada, G1K 7P4
chaib@ift.ulaval.ca

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Performance, Experimentation, Algorithms

Keywords

Multiagent systems, coalition formation, group buying

1. INTRODUCTION

Group buying is a natural application domain for research on coalition formation. Finding a Pareto-optimal partition (no other partition gives more to a consumer without giving less to another one) of the set of consumers in buying groups is equivalent to the generation of exact set covers known to be $\mathcal{NP}$-hard [1]. Since humans have difficulties in finding Pareto-optimal deals in reasonably complex situations, the use of software agents is justified. In this short paper, an investigation on the computational and economic performances of software agents in non-transferable payoff coalition formation (the general and less studied case) applied to group buying is conducted.

2. OVERVIEW OF THE PROTOCOL

In the developed coalition formation protocol shown in figure 1, consumers tell their agent the type of product they want as well as their preferences over the possible instances of the choosen product type. Agents are then able to find a buyers' group which suits their consumer's preferences. The number of possible buying groups for N agents and P products is $(2^N - 1) \times P$ which is a large number even for a limited number of agents and products. By restricting the number of units each agent can buy to only one, the quantity of buying groups to be considered by the agents is reduced to $N \times P$ because all groups buying the same product instance with the same number of members, buy the same number of units and therefore, pay the same unit price. In the protocol of figure 1, agents send

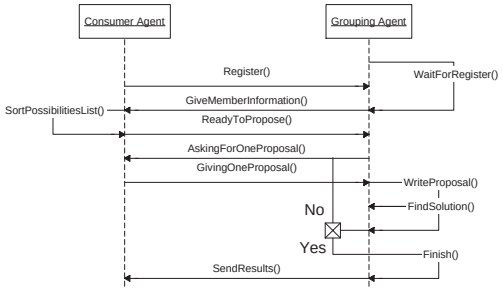

Figure 1: Coalition formation protocol followed by all agents.

the most preferred buying group of their customer not already proposed in each proposal round until there is a partition of the set of customers among the buying groups created by the proposals.

When all consumer agents have given their proposal for a given round, the grouping agent tries to find buying groups that were created with the operation WriteProposal(). A buying group is created when the number of consumer agents that proposed it is at least as high as the number of members of the proposed buying group. For a particular group of 10 consumers, if only 9 consumer agents propose it, then it is not created. But if a tenth consumer agent propose that group, it becomes effective. This process is equivalent to the generation of K-subsets of a N-set which is a combinatorial problem [3]. It consists in giving all subsets of K elements from a set of N elements. We used a successor algorithm [3] that takes as input a valid subset and gives as output the next subset in the lexicographic order. Having the first subset in this order and recursively calling this algorithm, we can generate all valid subsets. Buying groups created in a round are stored in memory for consideration in later rounds.

If new buying groups become effective in a proposal round, the grouping agent tries to find a partition of the set of consumer agents that were ready to propose among all the effective buying groups created since the beginning of the protocol with the FindSolution() operation. This problem is equivalent to the generation of exact set covers which is known to be $\mathcal{NP}$-hard [1]. We used a backtracking algorithm [2] that takes advantage of an heuristic to efficiently prune the search tree to find and generate partitions. Since every agent is obligated to have a buying group in which it buys alone a product, the partition consisting in every such buying groups is a solution that the backtracking algorithm will eventually find and thus, the protocol always terminates given sufficient time and memory space.

AAMAS'03, July 14–18, 2003, Melbourne, Australia.
ACM 1-58113-6683-8/03/0007.

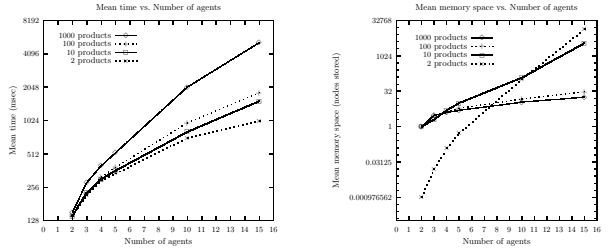

Figure 2: Computational performances.

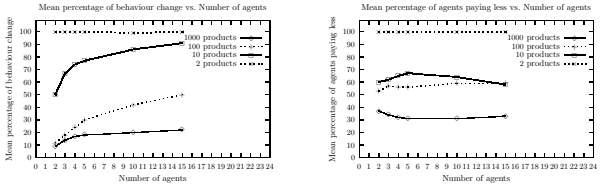

Figure 3: Economic performances.

3. RESULTS AND DISCUSSION

We tested the protocol of figure 1 on a Pentium 4 with 1.4 GHz processor, 256 Meg of RAM of which 130 Meg was dedicated to the execution of the protocol. Our protocol has been developed using Java JDK 1.4 and Jack 3.5, a framework for programming software agents. We have precisely executed the protocol a thousand times for each number of agents – quantity of products couple (a thousand times of 2 agents with 2 available products for example) with random preferences for the agents and always the same list of available products with their price schedule. Each of the 5 times we executed the protocol for the case of 20 agents and 2 products, the program went out of memory (even with 130 Meg of available memory). So we consider the case with 20 agents to be the limit of the protocol on the fore-mentioned computer platform and focus our attention on the cases with 15 agents or less. After the evaluation was completed, we have run the protocol on a different platform having more memory but less processing speed. With 450 Meg of dedicated memory this time, we had enough memory for 20 agents but not for 25 agents.

The graphic to the left of figure 2 demonstrates the mean time (in milliseconds) of the 1000 executions of the protocol for each couple between the moment the grouping agent sends information to registered consumer agents to the time it sends the solution to them. The exact set cover generation problem makes the worst case execution time complexity of the whole protocol to be *more* than polynomial. As expected, the execution time increases with the quantity of available product but it remains sublinear on a logarithmic scale meaning that the average complexity is *less* than exponential for the studied range (2 to 15 agents). This result is a little surprising due to the presence of two combinatorial problems (generation of K-subsets of a N-set and generation of exact set covers). We can explain this by the incentive of being a lot in a buying group in order to benefit from a price reduction which pushes agents to aggregate quickly and by the fact that the list of proposals of each agent is bounded to individually rational buying groups thus limiting greatly the number of proposal rounds.

The graphic to the right of figure 2 demonstrates the mean quantity of memory used (in nodes which are the data structure representing an agent in an effective buying group) of the 1000 executions of the protocol for each agents-products couple. The generation of K-subsets of a N-set problem makes the worst case memory requirement complexity of the whole protocol to be *more* than polynomial because the number of K-subsets of a N-set increases exponentially with N increasing for a constant K. Surprisingly, a greater quantity of available products leads to less used memory. This comes from the ratio of agents to products. The greater this ratio is, the more dense the agents are in the products space and the more likely they are of forming buying groups stored in memory. This is the reason why the case of 20 agents and 2 products (a ratio of $20/2 = 10$) exceeds the memory capacity of the test platform. The case of 20 agents and 100 products used much less memory but we cannot assure that it would not become a case of 20 agents and 2 products if all the agents decide that the same 98 products of the 100 are inadmissible (reserve price exceeded or inadmissible value for an attribute). The cases with 2 products and a number of agents between 2 and 5 used practically no memory because the formed buying groups were already partitions of the set of consumers and therefore, included all consumers. The protocol did not store these groups in memory for later use because it terminated after finding these partitions.

The graphic to the left of figure 3 shows the percentage (over a thousand executions for each "number of agents – quantity of available products" couple) of change in the behaviour of the consumers participating in the protocol in relation to the normal habit of buying the most preferred product alone at the local store. A change of behaviour occurs in two situations : the consumer buys his preferred product at a lower price than the one paid for one unit or he buys another product. We can see in the graphic to the left of figure 3 that with fewer products, the proposed protocol has a better chance of changing the behaviour of the consumers. With a fixed number of products, this graphic shows also that with more consumers agent, there is an increase in the percentage of consumers changing their behaviour for three of the four different quantities of products. The cases with 2 products had already the maximum (100%) of behaviour change. We explain both observations by the fact that they are more dense in the space of possible products and it is easier to aggregate in buying group and change that way their behaviour.

There is only two types of change of behaviour and therefore, their percentages complement themselves to 100%. So we only show the percentage of consumers that change behaviour by buying the same product they would have bought alone but at a lower price in the graphic to the right of figure 3. This graphic shows that only the number of available products influences the type of change of consumers' behaviour. When there is a little choice of products, few products will be in the preferences' list of consumers because some of the buying group for those products will be considered non-individually rationals and not proposed by the consumer agents. So consumer agents will regroup with others having the same preferred product. But if there is a huge choice of products, although some buying groups will still be non-individually rationals, there will be enough buying groups with different products left to create margin for consumer agents to join groups buying another product than the one their consumer would have bought alone.

4. REFERENCES

[1] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[2] D. E. Knuth. Dancing links. In *Millennial Perspectives in Computer Science*, pages 187–214. Palgrave, 2000.

[3] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration and Search*. CRC Press, 1998.