2003s-40

Performance of Software Agents in Non-Transferable Payoff Group Buying

Frederick Asselin, Brahim Chaib-draa

Série Scientifique
Scientific Series

Montréal
Juin 2003

CIRANO
Centre interuniversitaire de recherche en analyse des organisations

CIRANO

Le CIRANO est un organisme sans but lucratif constitué en vertu de la Loi des compagnies du Québec. Le financement de son infrastructure et de ses activités de recherche provient des cotisations de ses organisations-membres, d'une subvention d'infrastructure du ministère de la Recherche, de la Science et de la Technologie, de même que des subventions et mandats obtenus par ses équipes de recherche.

CIRANO is a private non-profit organization incorporated under the Québec Companies Act. Its infrastructure and research activities are funded through fees paid by member organizations, an infrastructure grant from the Ministère de la Recherche, de la Science et de la Technologie, and grants and research mandates obtained by its research teams.

Les organisations-partenaires / The Partner Organizations

PARTENAIRE MAJEUR
. Ministère du développement économique et régional [MDER]

PARTENAIRES
. Alcan inc.
. Axa Canada
. Banque du Canada
. Banque Laurentienne du Canada
. Banque Nationale du Canada
. Banque Royale du Canada
. Bell Canada
. Bombardier
. Bourse de Montréal
. Développement des ressources humaines Canada [DRHC]
. Fédération des caisses Desjardins du Québec
. Gaz Métropolitain
. Hydro-Québec
. Industrie Canada
. Ministère des Finances [MF]
. Pratt & Whitney Canada Inc.
. Raymond Chabot Grant Thornton
. Ville de Montréal

. École Polytechnique de Montréal
. HEC Montréal
. Université Concordia
. Université de Montréal
. Université du Québec à Montréal
. Université Laval
. Université McGill

ASSOCIÉ AU :
. Institut de Finance Mathématique de Montréal (IFM^2)
. Laboratoires universitaires Bell Canada
. Réseau de calcul et de modélisation mathématique [RCM^2]
. Réseau de centres d'excellence MITACS (Les mathématiques des technologies de l'information et des systèmes complexes)

ISSN 1198-8177

Performance of Software Agents in Non-Transferable Payoff Group Buying *

Frederick Asselin,† *Brahim Chaib-draa*‡

Résumé / *Abstract*

Les agents logiciels peuvent être utiles pour la formation de groupes d'acheteurs puisque les humains ont beaucoup de difficultés à trouver des transactions Pareto-optimales (aucun acheteur ne peut être mieux sans qu'un autre soit pire) dans les situations de négociation. Alors, quelles sont les performances informatiques et économiques des agents logiciels pour un problème de négociation particulier? Nous donnons une première réponse à cette question pour un problème de groupement d'acheteurs. Du point de vue de la théorie des jeux, ce problème est équivalent à la formation de coalitions avec gain non-transférable (le cas général). La recherche antérieure sur la formation de coalitions permettait le transfert de gain entre agents ce qui est un cas spécial. Nous argumentons que la Pareto-optimalité est un bon concept de solution pour ce problème. La formation de coalitions peut être décomposée en deux parties ayant une grande complexité de calcul : déterminer l'ordre de préférence parmi tous les groupes d'acheteurs possibles de chaque agent et trouver la meilleure structure de coalitions. Pour la première partie, nous avons trouvé une restriction raisonnable au problème permettant une réduction du nombre de groupes d'acheteurs à ordonner d'un facteur exponentiel à un facteur linéaire en fonction du nombre d'acheteurs. Pour la deuxième partie, nous cherchons à savoir par une évaluation empirique si les incitatifs à se regrouper (un gros groupe obtient un prix unitaire inférieur à un petit groupe) créent une structure spéciale rendant possiblement le problème plus facile sur le plan calculatoire. Nous évaluons un protocole de négociation pour agents logiciels que nous avons développé pour voir si le problème est difficile en moyenne et pourquoi. Ce protocole trouve assurément une solution Pareto-optimale qui minimise la pire distance à l'idéal parmi tous les agents étant donné des listes de préférences sans égalité. Notre évaluation démontre que la consommation de l'espace en mémoire et non le temps d'exécution limite les performances des agents logiciels dans ce problème de groupement d'acheteurs. De plus, nous cherchons à savoir si les agents logiciels suivant ce protocole ont le même comportement d'acheteurs que des humains dans la même situation. Les résultats démontrent que les agents logiciels ont un comportement plus différent (et meilleur car ils peuvent toujours simuler le comportement habituel des humains qui est d'acheter seul leur produit préféré) lorsqu'ils ont des préférences similaires dans l'espace des produits disponibles. Nous discutons également du type de la différence de comportement et de sa fréquence en fonction de la situation.

Mots clés : Systèmes multiagents, formation de coalitions, groupement d'acheteurs.

---

* This article presents complete results of our evaluation. Prior versions with partial results of this article were accepted at ICECR'2002 [Asselin and Chaib-draa 2002] and AAMAS'2003 [Asselin and Chaib-draa 2003].

† Ph.D. student at the Département d'informatique et de genie logiciel, Université Laval, Québec (Québec), Canada, G1K 7P4. Email: fasselin@iad.ift.ulaval.ca.

‡ Professor at the Département d'informatique et de génie logiciel, Université Laval, Québec (Québec), Canada, G1K 7P4 and at CIRANO, 2020 rue University, 25e étage, Montréal, QC, Canada, H3A 2A5. Email: chaib@iad.ift.ulaval.ca.

Software agents (SA) could be useful in forming buyers' groups since humans have considerable difficulty in finding Pareto-optimal deals (no buyer can be better without another being worse) in negotiation situations. Then what are the computational and economical performances of SA for a particular negotiation problem? In this article, we try to give a first answer to this question for a group buying problem. From the game theory point of view, the problem is equivalent to coalition formation (CF) with non-transferable payoff (the general case). Prior research in CF has allowed payoff to be transferred between agents, which is a special case. We argue that Pareto-optimality is a good solution concept to this problem. CF can be decomposed into two computationally difficult components : determining a preference ordering among all possible buying groups for each SA and finding the best coalition structure. For the first component, we have found a reasonable restriction to the group buying problem allowing the reduction of the number of possible buying groups to be ordered from a exponential to a linear factor in function of the number of buyers. For the second component, we try to investigate by an empirical evaluation if incentives to regroup (a bigger group pays less than a smaller one) create a special structure which possibly makes the problem computationally easier. We evaluate a negotiation protocol for SA that we developed to see if the problem is difficult on the average and why. This protocol provably finds a Pareto-optimal solution and furthermore, minimizes the worst distance to ideal among all SA given preference ordering without equality. This evaluation demonstrates that memory requirements and not execution time complexity limit the performance of SA in this group buying problem. Furthermore, we investigated if SA following the developed protocol had a different buying behaviour than the customer they represented would have had in the same situation. Results show that SA have a greater difference of behaviour (and a better behaviour since they can always simulate the obvious customer behaviour of buying alone their preferred product) when they have similar preferences over the space of available products. We also discuss the type of behaviour changes and their frequencies based on the situation.

Keywords: *Multiagent Systems, Coalition Formation, Group Buying.*

1 Introduction

The Internet has reduced the cost of communication, information search and more generally, computing through meta-computing (using the computational power of idle processors connected to the Internet). This reduction of cost has enabled the automation of some electronic commerce activities like online auctions [Guo, 2002; Anthony and Jennings, 2003; He and Jennings, 2003] and group buying negotiation that seeks a greater economical performance by economies of scale. We mean by negotiation the "process by which agents communicate with one another to try and come to a mutually acceptable agreement on some matter" [Lomuscio *et al.*, 2001]. Group buying is a natural application domain for research on coalition formation in multi-agent systems (MAS). Consumers have an incentive to regroup with the unit price reduction in function of the number of units bought by the group. But as more and more consumers become members of the same group, there is an increase in the number of compromises that each consumer must make in order to agree on the product bought by the group. In one extreme case, all the consumers can regroup in the same group and buy the same product. In the other extreme case, all the consumers could buy alone a different product forming thus as many groups as consumers. 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 would be a desired solution to this problem.

The use of software agents is required since they perform better than humans in finding a Pareto-optimal outcome in reasonably complex negotiations [Sandholm, 1999]. Not only this affirmation is a widely accepted conjecture, but empirical evidence tends to confirm it. Rangaswamy and Shell [1997] have conducted an experiment where two people needed to agree on an outcome among 256 possible outcomes based on preferences that were imposed to the participants. Results of this study showed that in 11.1% of the situations, humans agreed on a Pareto-optimal outcome. With the help of a negotiation support system, that percentage rose to 42.6%. So in both cases, humans were not able to find a Pareto-optimal outcome in more than half of the situations (88.9% and 57.4%). Software agents differ from negotiation support systems because they do not help humans to negotiate. Instead, they negotiate on their behalf.

When there is a set of consumers wishing for the same type of product, group buying consists in partitioning the set of such consumers into buying

groups with respect to the preferences of each consumer over all the possible buying groups. Among all the possible partitions, we would also like to find a Pareto-optimal one. Defined in that way, the computational problem is equivalent to the generation of exact set covers known to be $\mathcal{NP}$-hard [Garey and Johnson, 1979]. Incentives to regroup (a bigger group pays less than a smaller one) create a special structure possibly making the problem computationally easier. An investigation on whether this is the case or not is conducted. It turns out that the average execution time is *less* than exponential but the memory requirements limit the number of software agents in the system.

We have developed a protocol and associated algorithms that software agents can use to find a Pareto-optimal partition that minimizes the worst distance to ideal among all agents given a preference ordering without equality for each agent. This paper presents the results of the empirical evaluation of the computational and economical performances of the protocol for different quantities of agents (with random preferences from consumers) and products. It is organized as follows. In the next section, we briefly review related work. An argumentation follows in section 3 about the choice of Pareto-optimality as the solution to this coalition formation problem. Then, in section 4, we present our protocol and in section 5, we detail some results and discuss them. Finally, we conclude in section 6.

2 Prior Work on Coalition Formation in MAS

2.1 General Research Work

Research in coalition formation in MAS [Sandholm, 1996; Sandholm *et al.*, 1999; Sen and Dutta, 2000; Shehory and Kraus, 1999] has mainly focused on transferable payoff when we analyze the process from the game theory point of view. This case, which is specific, is defined by a payoff attributed to each possible coalition. Members of a coalition must agree on a division of that payoff among themselves. However, the general case in cooperative game theory uses non-transferable payoff [Osborne and Rubinstein, 1994]. Non-transferable payoff means that, for each coalition, each of its members receives an individual payoff which does not come from the payoff of the coalition because no such payoff exists. While in the transferable payoff case, the sum of members' payoff must equal the coalition payoff, in the non-

transferable payoff case, the sum of members' payoff can be anything. This is the reason why it is the general case. In group buying, each consumer has a private evaluation for each possible buying group. This evaluation depends only on the preferences and constraints of the consumer. The evaluation of a particular buying group by a consumer approximates the utility this consumer receives in belonging to this buying group. One could think that utility could be transferred from one consumer to another by an appropriate transfer of money. But in game theory, "we say that utility is transferable if the increment to the payoff of an agent caused by a transfer of money is proportional to the amount of money transferred" [Aumann, 1960]. Since humans seem to a have a utility function towards money that is of logarithmic form [Grayson, 1960], the increment to the agent's payoff is *not* proportional to the amount of money transferred. The same amount of money transferred could incur a different increment to the payoff of the agent based on the current value of the payoff since the utility function towards money is *not* linear. Hence, group buying as defined in this article is a non-transferable payoff case.

From the computer science point of view, research on coalition formation with transferable payoff has been related to optimization problems. Generally, they have searched for the partition whose sum of its coalitions' payoff is optimal. In our case, we want to generate a Pareto-optimal partition which is a generation problem (every Pareto-optimal partition is considered a solution). While optimization problems' solutions could sometimes be approximated by deterministic or non-deterministic algorithms, solution to generation problems cannot be approximated in the same way. So, less computational tools exist for generation problems in comparison with optimization ones.

2.2 Specific Research Work for Group Buying

The use of coalition formation for buying groups has recently attracted the attention of researchers. Thus, Tsvetovat *et al.* [2001] have demonstrated the economic incentives of buying groups for consumers as well as manufacturers and provided models of coalition formation in that context. Lerman and Shehory [2000] have described the macroscopic behaviour of coalitions with differential equations when agents are allowed to leave and join a coalition to finally reach a steady state. For their part, Yamamoto and Sycara [2001] have separated agents into coalitions and divided the profit generated by a coalition among its members in an efficient and stable way using transferable

payoff. Another interesting issue investigated by Ito *et al.* [2001], has been on how we can allow sellers to cooperate when a coalition requires more units than a single seller can offer. Another facet has been investigated by Lin and Yuan [2001] and concerns the reputation in the choice of a coalition's manager which represents the other members of a coalition in the negotiation with sellers. Li and Sycara [2002] have studied how to combine coalition formation with combinatorial auctions for a more efficient marketplace. Breban and Vassileva [2002] have studied the concept of trust in long term coalitions of buyers and sellers over repeated transactions. However, these approaches has studied group buying with transferable payoff and therefore, did not empirically evaluated an implemented system with non-transferable payoff.

Recently, Sarne and Kraus [2003] have proposed a non-transferable payoff model for group buying. In this model, each buying group is represented by an agent who incurs a cost for searching another buying group to merge with in addition to the cost of the internal coordination of the group's members. The buying groups evolve as agents must decide if they continue the search for potential merging partners or settle. The problem we have studied is different because it is composed of a static group of consumers of which we want a Pareto-optimal partition into buying groups.

3 Choice of the solution concept

In game theory, even determining what should be the solution to a problem is a valid research question beside finding it. This is because solutions in that domain are equilibria that are more or less stable (or stable in different ways) and there is no dominating equilibrium. Game theorists do not even communicate with the word "solution". Instead, they use the expression "solution concept". Several solution concepts exist for cooperative game theory with non-transferable payoff along with general solution concepts that could also apply to this case. Among those which are the most useful:

Social welfare It is the summation of the utility of each consumer towards its payoff. Since "each agent's utility function can only be specified up to positive affine transformation" [Mas-Colell *et al.*, 1995] in [Sandholm, 1999], this solution concept is not applicable to the group buying problem because it sums quantities (the utilities) with an arbitrary scale of value;

Pareto-optimality A solution is Pareto-optimal if no other solution gives more to a consumer without giving less to at least another one. With this solution concept, there is no comparison of utility between consumers. Instead, a payoff that a consumer can receive is compared with the other payoffs this consumer can receive. By definition, there is always at least one Pareto-optimal solution;

Individual rationality A solution is individually rational if each consumer receives a payoff at least as great as the payoff he would have received had he acted alone. The partition composed of each consumer buying alone has the vector of minimum payoffs each consumer could get. A Pareto-optimal solution is always individually rational but the inverse is not always true;

Core It is the most stable of all solution concepts in cooperative game theory since no subgroup of consumers has an advantage in leaving a solution in the core. The problem is that the set of solutions in the core can be empty [Osborne and Rubinstein, 1994]. Even determining if it is empty or not is an $\mathcal{NP}$-complete problem [Conitzer and Sandholm, 2002]. So why design and implement a protocol that tries to find a solution that does not always exist and futhermore, whose existence is computationally hard to determine? Hence, this is not an appropriate solution concept for the group buying problem;

Stable set This solution concept is a relaxation of the constraints of the core. Every stable set has the core as a subset. Unfortunately, the stable set has the problem of the core: it can be empty [Osborne and Rubinstein, 1994];

Bargaining set This solution concept uses objections and counterobjections. A solution belongs to the bargaining set if for every objection against it there is a counterobjection. This set is always non-empty and the core is always a subset of it [Sandholm, 1996]. One downside of this solution concept is that every partition of the set of buyers has its bargaining set. So it does not help to find a solution when the best partition is unknown. Furthermore, consumers need to know the payoff of the other consumers for making objections and counterobjections. Since payoffs are private information that should remain secret, this solution concept is not applicable to the group buying problem.

The solution concept chosen for the group buying problem must have some basic properties. Firstly, it must always exist (not be sometimes an empty set) unlike the core and the stable set. Secondly, it must not require summing utilities with an arbitrary scale of value like the social welfare or having knowledge of other consumers' payoff like the bargaining set. Since a Pareto-optimal solution is also individually rational, we think that it is the most appropriate solution concept for the group buying problem.

4 Overview of the Protocol

In the developed protocol, consumers tell their agent the type of product they want as well as their preferences over the possible instances of the chosen product type. Agents are then able to find a buyers' group which suits their consumer's preferences.

4.1 Reduction of the possibilities space

The group buying problem can be decomposed into two computationally difficult components : determining a preference ordering among all possible buying groups for each software agent and finding the best coalition structure. For the first component, we have found a reasonable restriction allowing the reduction of the number of possible buying groups to be ordered from a exponential to a linear factor in function of the number of buyers. For N consumers and P products, the number of possible buying groups is $(2^N - 1) \times P$. Indeed, the number of possible groups of consumers from a set of N consumers is in relation to the number of possible subsets of a set. There is 2^N subsets of a set of N elements including the empty set. Since we do not consider the empty set as a valid buying group, there is $2^N - 1$ possible groups of consumers. Each of them could buy one of P available products. Hence, there is $(2^N - 1) \times P$ possible buying groups.

This is a large number of possibilities even for a small number of consumers and products. But if we restrict the number of units each consumer can buy to only one, the number of possibilities is greatly reduced. Consider a set with 10 consumers and a product named A. The number of subsets of 3 consumers buying product A is equal to $\binom{10}{3} = 120$. If we limit the number of units each consumer can buy to only one, each group of the 120 groups of 3 consumers buys 3 units. So, they all pay the same unit price since they

all buy the same quantity of units. From the consumer's point of view, they are all the same because they buy the same product at the same unit price. Consumers can now consider only one group of 3 consumers instead of 120 different groups. For N consumers there is N non-empty groups of different cardinality. Hence, there is $N \times P$ possible buying groups when we restrict the number of units each consumer can buy to only one. This is the number of buying groups that an agent must consider when creating its list of possible and different buying groups of which it could be a member. It is not the number of all buying groups that the grouping agent of Figure 1 considers when searching for a Pareto-optimal partition (the second component of the group buying problem) which remains equal to $(2^N - 1) \times P$.

If a consumer wants multiple units of the same product, he can delegate multiple agents, one for each unit he wants. But there is a downside to this restriction when a consumer wants multiple units. If a consumer desires 50 units of a product, then he will pay the same unit price as a consumer desiring only 1 unit while contributing much more to its decrease than its counterparts. Since the restriction permits a reduction of the number of possibilities from an exponential to a linear factor in function of the quantity of consumers and that consumers wishing for multiple units are a special case, we think the restriction is a good compromise. Furthermore, we expect group buying to be used for high value luxury products where the gain from group buying seems the greatest. A typical consumer usually buys only one unit of such products.

4.2 Sequence diagram of the protocol

Figure 1 shows a sequence diagram of the developed protocol. There are two types of agents in the protocol. The consumer agent represents a consumer wishing for one unit of a product. The grouping agent tries to find a partition of the set of consumer agents from buying groups that are effective. Since all the consumer agents have the same behaviour, there is only one such agent in Figure 1.

When the agent has received the type of product its consumer wishes and the preferences over instances of that product type, it registers itself to the grouping agent in charge of that type of product by the Register() operation. After a registration time has elapsed, the grouping agent ends the registration period (shown by the WaitForRegister() operation). It then sends information to the registered consumer agents with the GiveMember-

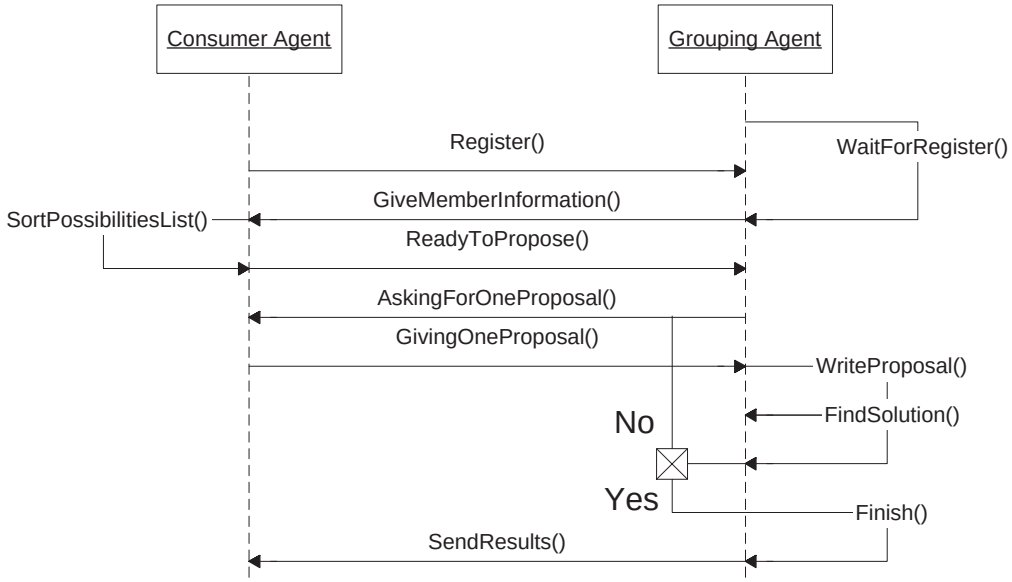

Figure 1: Coalition formation protocol followed by all agents.

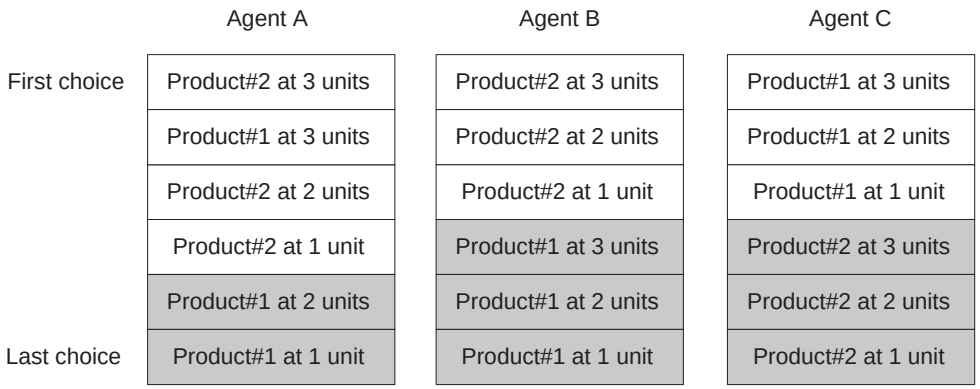

Figure 2: Sorted lists of possibilities for three agents with two available products.

Information() operation. The transmitted information consists of the number of registered consumer agents, the specification of available products and their price schedules which return the unit price of the product in function of the number of units bought by the buying group. With this information and with the preferences of its consumer, each consumer agent can construct and sort a possibilities list from the most preferred to the least. Figure 2 shows such lists for 3 consumer agents and 2 products. Agent A prefers to buy Product#2 with the two other agents (which means 3 units bought). Agent B second choice consists of buying Product#2 with one of the other agents (which means 2 units bought). Agent C third choice consists of buying alone Product#1 (which means 1 unit bought). The grey boxes indicate possibilities that are *not* individually rational. They are situations giving less payoff than the possibility of buying alone the preferred product. For Agent A in Figure 2, buying with another agent Product#1 (the Product#1 at 2 units grey box) is *not* individually rational because it is *less* preferred than buying alone Product#2 (the Product#2 at 1 unit white box). Buying alone Product#1 is also *not* individually rational because it is also *less* preferred than buying alone Product#2. Such situations are inadmissible here since each agent can buy a product alone.

If a consumer agent does not have an individually rational buying group in which it is alone, it tells the grouping agent it is not ready to propose, by the ReadyToPropose() operation with an attribute assigned to false, and it quits the coalition formation protocol. Or else, it tells the grouping agent it

is ready to propose by the same operation but with an attribute assigned to true. We impose that each consumer agent must have a buying group in its list of possibilities in which it is alone in order to assure that a partition exists. In this case, the partition in which each consumer agent is alone in its group is a solution to the problem of group buying. Otherwise, consumers could wait hours and even days for other consumers to join them to form buying groups and at the end, no such group could be created because no partition existed for the given set of consumers. We do not think that this restriction is limiting because in the present situation where everybody is buying alone, someone for whom buying alone any available products is not individually rational has the same behaviour than the consumer agent quitting the protocol (it does not buy the product).

When all registered consumer agents have told the grouping agent if it is ready or not to propose, the grouping agent asks each consumer agent who is ready for their preferred buying group with the operation AskingForOneProposal(). The consumer agents answer by giving the most preferred buying group in their list not already proposed with the operation GivingOneProposal(). Having no information about the preferences of the other consumer agents and without knowing when the grouping agent will stop asking for proposals because it finds a partition, we cannot think of a better strategy for each consumer agent than to propose their individually rational buying groups in decreasing order of preferences.

When all consumer agents have given their proposal, 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 proposes that group, it becomes effective. This process is equivalent to the generation of K-subsets of a N-set which is a combinatorial problem [Kreher and Stinson, 1998]. It consists in giving all subsets of K elements from a set of N elements. We used a successor algorithm [Kreher and Stinson, 1998] that takes in input a valid subset and gives as output the next subset in the lexicographic order as shown by algorithm 1. 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.

Furthermore, since the number of consumers that will propose a given

buying group is not known in advance, it would be valuable to generate incrementally the proposed groups. For example, if a twelfth consumer proposes a group of 10 consumers, we would like to generate only the newly created groups by this consumer and not regenerate also the groups created by the tenth and eleventh consumers which are already known. In other words, we would like an algorithm that computes solutions to partial input without knowing future pieces of input. Such an algorithm is called an online algorithm and performance comparison with a corresponding off-line algorithm (algorithm with the full input) is named competitive analysis. It turns out that algorithm 1 can be transformed into an online algorithm with only a simple change in its input. Since its structure is unchanged, there is no loss of performance between the online and the off-line version which is surprising. As an example, if a tenth consumer proposes a group of 10 buyers, then the group created is generated with the tenth consumer and 9 $(10 - 1 = 9)$ consumers chosen from the 9 preceding consumers. Since there is only one way to choose 9 elements from 9 elements, there is only one group generated. When an eleventh consumer proposes the same group of 10 buyers, then the groups created are added to the previously generated group. These newly created groups are generated with the eleventh consumer and 9 $(10 - 1 = 9)$ consumers chosen from the 10 preceding consumers. Since there are 10 ways to choose 9 elements from 10 elements, 10 new groups are created and added to the lone group generated by the tenth consumer. The same scheme continues as more consumers propose the same group of 10 using algorithm 1 but with preceding consumers as input. There are two exceptions to this scheme. When a consumer has proposed a buying group in which he is alone and when all consumers have proposed the same buying group including all of them, then these groups are generated without algorithm 1.

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 [Garey and Johnson, 1979]. We used a backtracking algorithm [Knuth, 2000] that takes advantage of an heuristic to efficiently prune the search tree to find and generate partitions as shown by algorithm 2.

Before giving explanations for algorithm 2, the data structures used must be examined. Each consumer is represented with an object called "column object" within a double-linked circular list which has a root object named

Algorithm 1 Successor algorithm giving the next K-subset of N-set in lexicographic order (from [Kreher and Stinson, 1998]).

Require: $\vec{T} = [t_1, t_2, ..., t_K]$ (a K-subset of a N-set where t_x, for $1 \leq x \leq k$, is an element of the N-set), K (the cardinality of the K-subset) and N (the cardinality of the N-set).
  $\vec{U} \leftarrow \vec{T}$
  $i \leftarrow K$
  while $i \geq K$ and $t_i = N - K + i$ **do**
    $i \leftarrow i - 1$
  end while
  if $i = 0$ **then**
    Terminate
  else
    for $j \leftarrow i$ to K **do**
      $u_j \leftarrow t_i + 1 + j - i$
    end for
    Return $\vec{U}$
  end if

h. Each buying group is represented also by a double-linked circular list of objects representing each a consumer that is a member of the buying group. The lists of buying groups stack up as in Figure 3 and objects representing the same consumer are double-linked circularly among themselve with the "column object" representing that consumer. Of course, buying groups including all customers are not represented in the data structures used to find a partition because we already know that they form a partition in itself.

The main operations of algorithm 2 are $R[x]$ (returns the neighbour to the right of object x in the horizontal list), $L[x]$ (returns the neighbour to the left of object x in the horizontal list), $D[x]$ (returns the neighbour under object x in the vertical list), $U[x]$ (returns the neighbour over object x in the vertical list), $C[x]$ (returns the "column object" linked to object x), $S[y]$ (returns the number of buying groups containing the consumer represented by "column object" y), Cover a "column object" (remove the "column object" from its horizontal list and the objects of the buying groups containing the consumer corresponding to this "column object" from their vertical lists.) and Uncover a "column object" (undo the operations of Cover for the same "column object").

Algorithm 2 Recursive algorithm "Search" for the exact set cover problem (from [Knuth, 2000]).

Require: h (the root of the list of "column objects"), O_k (an array for keeping in memory the $k-1$ buying groups already chosen) and data structures like the ones of Figure 3. At the first call, $k = 0$.

```
if R[h] = h then
    Print the solution
    Return
else
    s ← ∞ {Start of the heuristic of Golomb and Baumert}
    for all j ← R[h],R[R[h]],..., while j ≠ h do
        if S[j] < s then
            c ← j
            s ← S[j]
        end if
    end for{End of the heuristic of Golomb and Baumert}
    Cover column c {The Cover subroutine is presented in algorithm 3}
    for all r ← D[c],D[D[c]],..., while r ≠ c do
        O_k ← r
        for all j ← R[r],R[R[r]],..., while j ≠ r do
            Cover column C[j] {The Cover subroutine is presented in algorithm 3}
        end for
        Search(k + 1) {Recursive call to the Search routine}
        r ← O_k
        c ← C[r]
        for all j ← L[r],L[L[r]],..., while j ≠ r do
            Uncover column C[j] {The Uncover subroutine is presented in algorithm 4}
        end for
    end for
    Uncover column c {The Uncover subroutine is presented in algorithm 4}
    Return
end if
```

Algorithm 3 Cover algorithm for "column object" covering (from [Knuth, 2000]).

Require: c ("column object" to cover) and data structures like the ones of Figure 3.

```
L[R[c]] ← L[c]
R[L[c]] ← R[c]
for all i ← D[c],D[D[c]],..., while i ≠ c do
  for all j ← R[i],R[R[i]],..., while j ≠ i do
    U[D[j]] ← U[j]
    D[U[j]] ← D[j]
    S[C[j]] ← S[C[j]] − 1
  end for
end for
```

Algorithm 4 Uncover algorithm that undoes the changes made by the Cover algorithm to the data structures (from [Knuth, 2000]).

Require: c ("column object" to uncover) and data structures like the ones of Figure 3.

```
for all i ← U[c],U[U[c]],..., while i ≠ c do
  for all j ← L[i],L[L[i]],..., while j ≠ i do
    S[C[j]] ← S[C[j]] + 1
    U[D[j]] ← j
    D[U[j]] ← j
  end for
end for
L[R[c]] ← c
R[L[c]] ← c
```

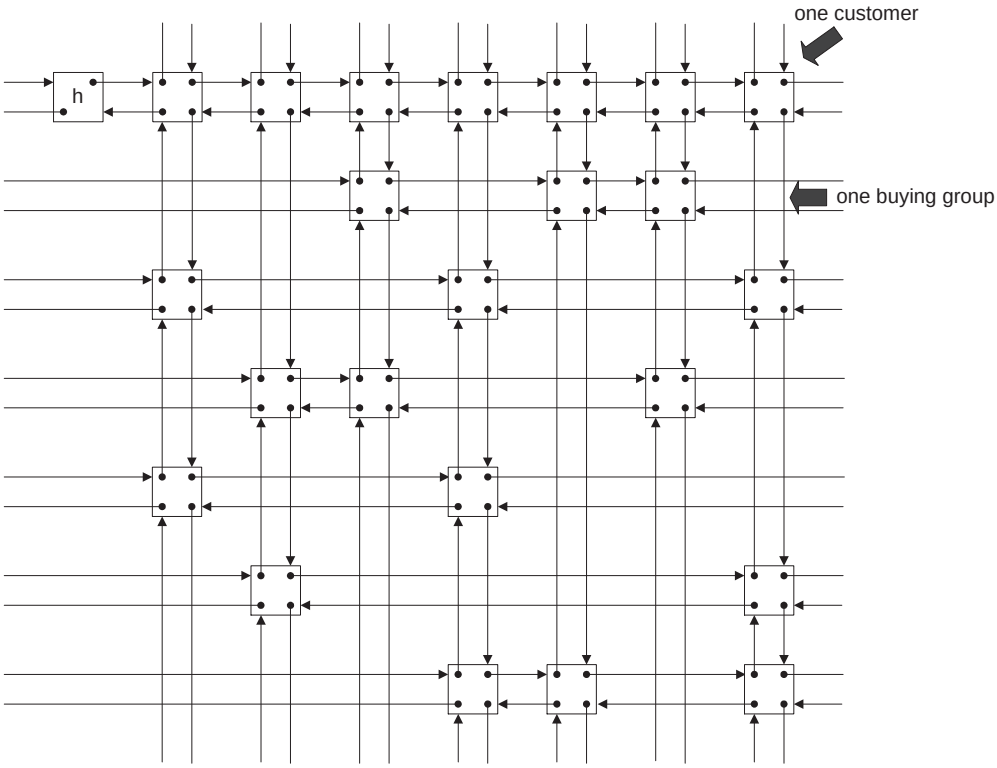

Figure 3: Data structures of an exact set cover problem with 7 consumers and 6 buying groups (from [Knuth, 2000]).

Using algorithm 2, we generate all partitions of the set of consumers from a set of buying groups by recursively choosing a buying group ($O_k \leftarrow r$) that does not contain a consumer already in a previously chosen buying group. If no such buying group exists and there are still consumers who are not in a chosen buying group, we backtrack by replacing the group last chosen (Uncover column c and Return which causes the execution to continue at the $r \leftarrow O_k$ statement) by another one. If we find a partition ($R[h] = h$ which means that all customers are in one and only one group), we print it and return which also causes the execution to continue at the $r \leftarrow O_k$ statement that begins the replacement of the group last chosen by another one to resume the search of partitions. When all cases have been tried, the algorithm stops. Because an exhaustive search could be long, Knuth [2000] used an heuristic [Golomb and Baumert, 1965] to make the search tree as

narrow as possible such that the pruning of this tree eliminates as many search paths as possible.

If no partition exists among the effective buying groups, the grouping agent launches other rounds of proposals until it finds one. One could ask why the consumer agents do not send their complete list of individually rational buying groups the first time the grouping agent asks for a proposal. As a principle, private information should remain secret as long as possible to make manipulation of the protocol to unwanted ends harder. Since the communication burden of the entire protocol (see table 1) is negligible compared to the computational burden of the grouping agent (one combinatorial problem and one $\mathcal{NP}$-hard problem), we do not think that the multiple rounds of proposals add significantly much to the execution time of the protocol.

The protocol always terminates since there is always a partition composed of all the buying groups in which each consumer agent is alone. It is also sound and complete since it searches exhaustively the partitions. If the grouping agent finds at least one partition, it terminates the protocol (Finish() operation) and sends to each consumer agent that was ready to propose the buying group in which it belongs in the partition among all the ones found that minimizes the worst distance to ideal among all agents (SendResults() operation).

Here are some definitions and analytical results about the developed protocol:

Definition 1. A partition p is dominated by a partition p' if no consumer prefers p to p' and there exists at least one consumer that prefers p' to p.

Definition 2. A partition is feasible if each consumer is willing to propose the buying group containing him in this partition.

Definition 3. A feasible partition p is Pareto-optimal if no other feasible partition dominates it.

Definition 4. The distance to ideal for a consumer agent A in relation to a partition p is defined as the number of buying groups that were proposed by agent A before the one that includes that agent in the partition p. As an example, for Agent A in Figure 2, the group that buys 2 units of Product#2 has a distance to ideal of 2 since 2 buying groups were proposed before it by Agent A.

Proposition 1. *A partition p found in the first round where such partition exists and that minimizes the sum of the distance to ideal of the agents in comparison with other partitions found in that round is Pareto-optimal given that the lists of possibilities do not have buying groups which are equally preferred by a consumer.*

The proof of proposition 1 will be conducted in three steps: two lemmas and the proposition.

Lemma 1. *All the partitions that are found in a later round than the first round where we find a partition cannot dominate a partition found in this first round given that the lists of possibilities do not have buying groups which are equally preferred by a consumer.*

Proof of Lemma 1. If a partition is found in such a later round, there exists an agent that proposed its buying group in this partition in that later round. Clearly, it would prefer to be in its buying group of the partition found first than in the group it proposed later which is part of the partition found in the later round. Since the first condition of definition 1 is not met because at least one agent prefers the first partition found to the one found in a later round, then the first partition is *not* dominated by partitions found in a later round. □

Lemma 2. *A partition p found in the first round where such partition exists and that minimizes the sum of the distance to ideal of the agents among partitions found in that round cannot be dominated by such partitions given that the lists of possibilities do not have buying groups which are equally preferred by a consumer.*

Proof of Lemma 2. Suppose the existence of a partition p' which dominates the partition p. This means that there exists at least an agent A whose distance to ideal is shorter in p' than in p. All the other agents do not have a greater distance to ideal in p' than in p. So the sum of the distance to ideal of the agents in p' should be less than the same sum in p. Thus, there exists a sum which is less than the minimal sum of p. By this contradiction, we prove that what we supposed true is false. No such partition p' dominates the partition p. □

Proof of Proposition 1. Such a partition p is not dominated by partitions found in the later rounds by lemma 1. Furthermore, such a partition p is

not dominated by partitions found in the same round by lemma 2. Since no partition exists in preceding rounds of the round when we first found a partition, we can say that a partition p is dominated by no other feasible partition. By definition 3, we have proven that the partition p is Pareto-optimal. □

The protocol finds a Pareto-optimal solution given that the lists of possibilities do not have buying groups which are equally preferred by a consumer. If sellers reduce the unit price, although minimally, for each additional member in the buying group to incite regrouping, then we suppose that equalities will be rare because groups will be differentiated by the unit price. The consumer agent could be given a set of rules or ask its consumer to settle equalities. Further research will include the study of the burden equalities impose on agents, consumers and sellers.

Proposition 2. *The partition found by the proposed protocol and its associated algorithms minimizes the worst distance to ideal among all consumer agents that were ready to propose in comparison with all other feasible partitions given that the lists of possibilities do not have buying groups which are equally preferred by a consumer.*

Proof of Proposition 2. The worst distance to ideal among all consumer agents in a partition is always related to the round where we find that partition. It is the distance to ideal of one of the consumer agents that proposed a buying group that became effective with its proposal and that permitted the partition to exist. Since no partition exists before the first round where we find one and partitions found in later rounds return a greater worst distance to ideal, then the worst distance to ideal is minimized when we choose a partition that is found in the first round where a partition exists. The protocol returns such a partition as a solution to the problem of group buying. □

5 Results and Discussion

To the best of our knowledge, this is one of the first multiagent coalition formation protocols with non-transferable payoff which is the general case. The problem of group buying with non-transferable payoff belongs to this case and this is the main reason why we studied it. Prior research has developed specific protocols for the transferable payoff case. Unfortunately, these

Attributes	Proposed protocol
Efficiency	Pareto-optimal and minimizes worst distance to ideal
Stability	Pareto-optimal
Simplicity	$\Theta(PN^2)$ messages
Distribution	Centralized
Symmetry	Yes
Money transfer	No

Table 1: Qualitative evaluation of the proposed protocol.

protocols cannot be used in group buying as it is defined in this article because they solve an optimization problem while we have a generation problem (of exact set covers). An optimization problem is defined as the search for the feasible solution that optimizes the objective function of the problem while a generation problem wants to search all feasible solutions [Kreher and Stinson, 1998]. For the same reason, we cannot use our protocol (and associated algorithms) to solve their optimization problems. Therefore, comparison with other protocols loses its relevance. Instead, this research is an evaluation of the performance of software agents in coalition formation with non-transferable payoff studied in the context of a real world application: group buying.

5.1 Qualitative Results

Rosenschein and Zlotkin [1994] have elaborated attributes for negotiation mechanisms which were slightly extended by Kraus [2001]. Table 1 summarizes those attributes as well as the value they have in the proposed protocol. The efficiency attribute evaluates whether or not the mechanism squanders payoff in the solution returned. The proposed protocol returns a Pareto-optimal solution, hence it does not squander payoff because no consumer could have more without another consumer having less. Furthermore, the solution minimizes the worst distance to ideal among all software agents such that an individual agent does not get a really bad payoff from the solution.

The stability attribute refers to the ways an agent or a group of agents could prefer another solution and abandons the proposed solution. Pareto-optimality is not the most stable solution among all the solution concepts from cooperative game theory as seen in section 3 but it is the most ap-

propriate since it always exists, does not require summing utilities with an arbitrary scale of value and having knowledge of other consumers' payoff.

The simplicity of a mechanism refers to its communication and computational complexity.

Proposition 3. *The number of messages exchanged in the theoretical worst case is in* $\Theta(PN^2)$ *for* P *products and* N *consumer agents.*

Proof of Proposition 3. Considering Figure 1, for N consumer agents, there are N messages for Register(). The grouping agent responds with N messages for GiveMemberInformation(). N messages come from the ReadyToPropose() operation. In the worst case, there is $2N$ messages for each proposal round, an AskingForOneProposal() and a GivingOneProposal() for each consumer agent. The number of proposal rounds is bounded by the longest list of possibilities among all consumer agents. Since there are P groups where the agent buys alone a product, then at least $P-1$ buying groups are not individually rational because one of the P product is preferred to the $P-1$. Thus, subtracting those $P-1$ groups from the $N \times P$ possible ones gives at most $PN-P+1$ individually rational groups and consequently, the same number of proposal rounds. Adding the N messages from the SendResults() operation, we now have $2PN^2-2PN+6N$ $(= N+N+N+2N(PN-P+1)+N)$ messages exchanged in the theoretical worst case. Thus, the communication complexity is $\Theta(PN^2)$. □

The exact set cover generation problem makes the worst case execution time complexity of the whole protocol to be *more* than polynomial. But simulation results presented in section 5.3 tend to indicate that in practice, the average execution time is polynomial at least for 15 software agents or less. 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. The simulation results of section 5.3 indicates that in practice, the average memory requirement complexity of the whole protocol is at *least* polynomial for 15 software agents or less. So, incentives to regroup (a bigger group pays less than a smaller one) could have created a special structure making the group buying problem computationally easier from the execution time complexity point of view but not from the memory requirement complexity point of view.

The distribution attribute indicates if the solution is computed in a distributed way or not. A distributed computation is preferred to a centralized one since it avoids bottleneck and is more robust to failure. The solution in the proposed protocol is computed in a centralized way because of the inherent computational complexity of the problem. While on some occasions we could benefit from the parallelism of distribution, it is much more difficult to design a computationally efficient distributed system than a centralized one because we need to consider the additional communication burden of the distributed system.

Symmetry refers to the property of a mechanism to treat its participants with impartiality. The grouping agent interacts with all the consumer agents in the same way and in this sense, we could say that the proposed protocol has the symmetry property.

Finally, the money transfer attribute which was added by Kraus [2001] indicates if money is transferred to resolve conflicts between agents. Since money transfer demands resources and can be manipulated, one would prefer not to be obligated to have it. Because the studied problem is a non-transferable payoff case, the proposed protocol does not use money transfer.

5.2 Parameters of the evaluation

Multiple parameters have to be randomly generated for the empirical evaluation of the proposed protocol. Here is a list of them.

- number of consumers;
- quantity of available products;
- specifications of available products;
- preferences of consumers over available products specifications.

The proposed protocol has been evaluated for several numbers of consumer agents (2, 3, 4, 5, 10, 15 and 20) and quantities of products (2, 10, 100 and 1000). The specifications of available products were represented with 10 attribute-value couples. This kind of representation has already been used by RosettaNet [2003] for electronic components selling and buying. Each attribute could be assigned one of 10 discrete values. The price of each available product was set by a randomly generated price schedule which returned a reduced price with an increase of the quantity of units bought. Preferences

of consumers were generated by allocating a weight to each attribute and the price (based on an uniform distribution) determining its relevance. The possible values for each attribute were divided into two sets: the acceptable ones and the inacceptable ones. Acceptable values got a weight in relation with its rank in the preference order among acceptable values for the related attribute. The reserve price of each consumer was chosen from a uniform distribution such that no consumer agent would not be ready to propose to the grouping agent. We did that to assure that results for N agents were actually computed for N agents and not N minus the number of consumer agents that were not ready.

5.3 Quantitative Results

We tested the previous protocol 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 was developed using Java JDK 1.4 and JACK Intelligent Agents^{TM} 3.5 [Agent Oriented Software Pty. Ltd., 2002], 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 aforementioned computer platform and focus our attention on the cases with 15 agents or less. After the evaluation was completed, we ran 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.

Figure 4 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. As expected, the execution time increases with the quantity of available products but it remains sublinear on a logarithmic scale meaning that the 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 several

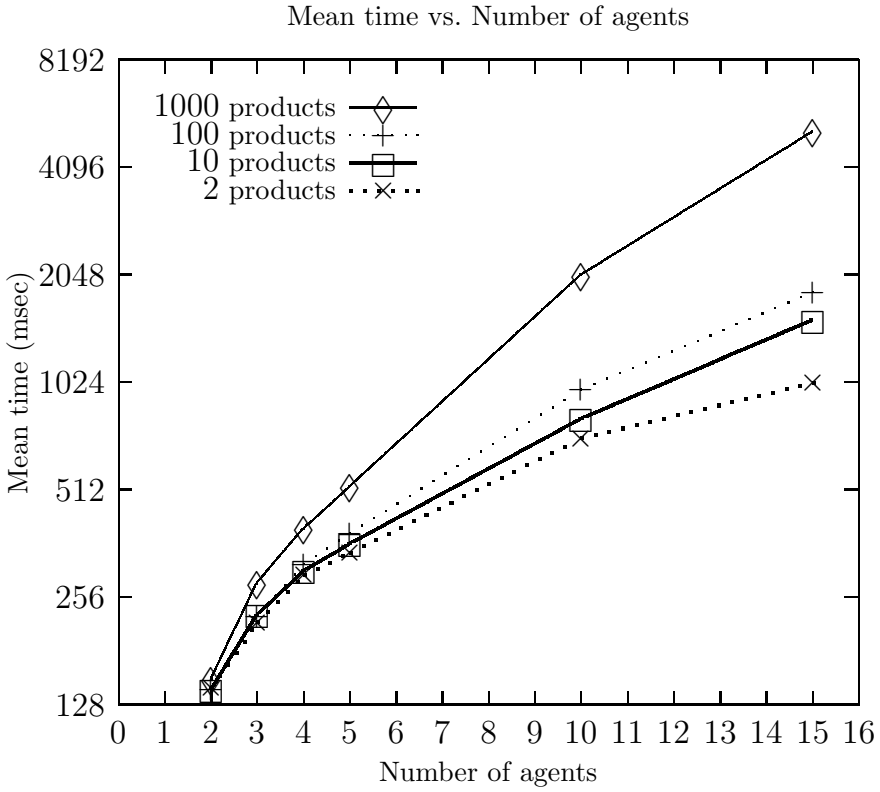

Figure 4: Mean time of execution versus the number of consumers' agents in the protocol for different quantities of available products.

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 possibilities of each agent is bound by individually rational buying groups thus limiting greatly the number of proposal rounds.

Figure 5 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. 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. With an increased density, agents are more likely to form effective buying groups stored in the implementation of the protocol as linked nodes which take memory space. 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). If this ratio is low, the agents are most sparse in the product space and they form less effective buying groups (resulting in less memory space used) because their preferences are farther from one another. 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.

If we consider a computer with 450 Mega-bytes of memory space allocated to the protocol, we can compute cases with 20 agents. With more memory space, we can compute cases with more than 20 agents but not much more than this number of agents since the memory consumption grows rapidly in function of the number of agents as Figure 5 shows. If more than 20 agents wish to form buying groups, the grouping agent could divide the set of consumers into subsets containing 20 agents or less. The division could be based on the first few buying groups in the preferences list of each consumer agent. Subsets would then contain consumer agents with similar preferences. After finding a Pareto-optimal partition for each subset, the grouping agent could merge buying groups with the same product. It would result in a partition of the initial set of consumer agents. We cannot assure that this partition is Pareto-optimal among the set of all feasible partitions. The quality of this partition will be investigated in future work.

Figure 6 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;
- he buys another product.

We can see in Figure 6 that with fewer products, the proposed protocol

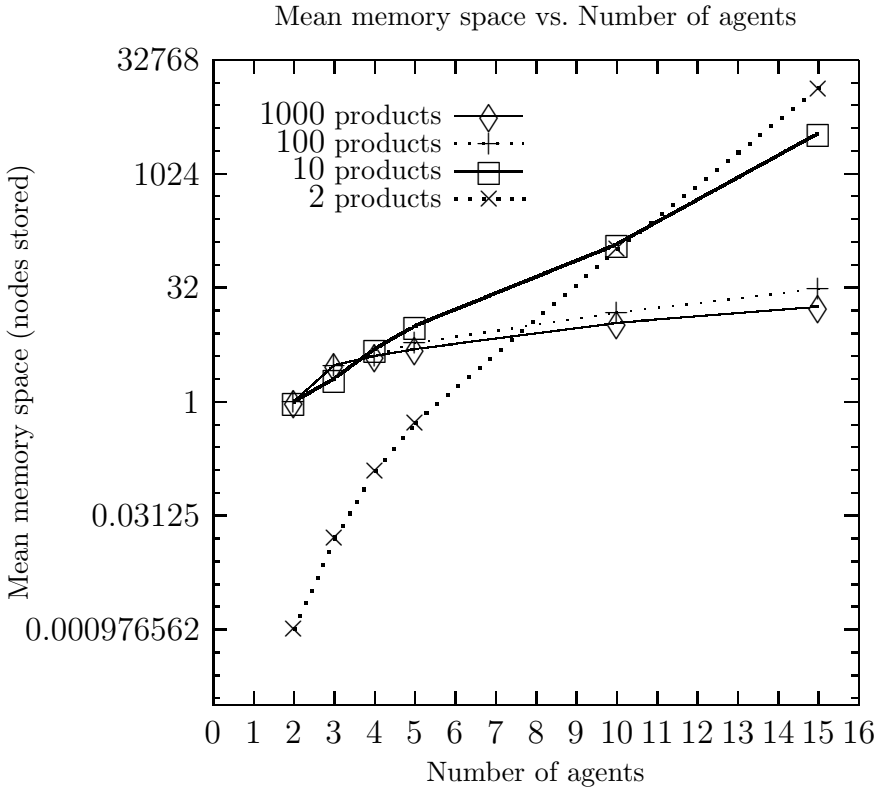

Figure 5: Mean memory space used versus the number of consumers' agents in the protocol for different quantities of available products.

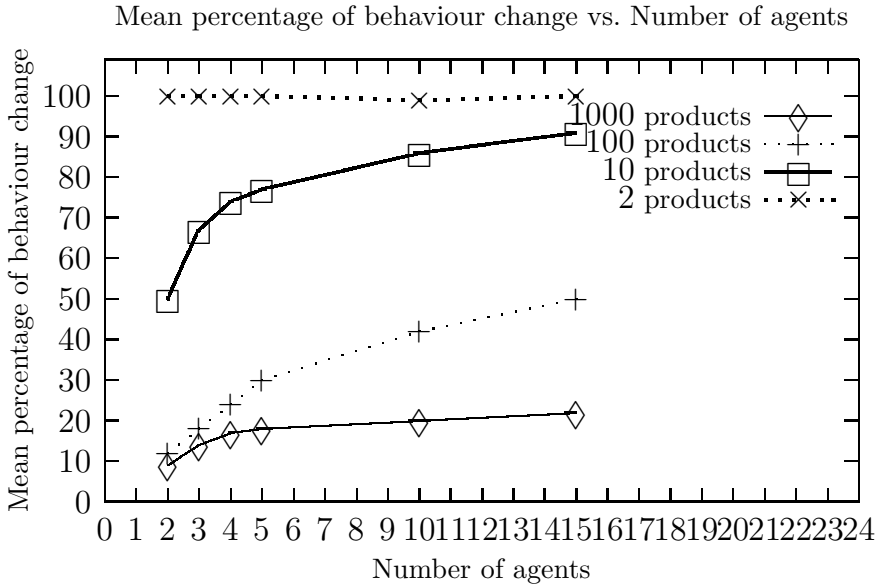

Figure 6: Mean percentage of behaviour change versus the number of consumers' agents in the protocol for different quantities of available products.

has a better chance of changing the behaviour of the consumers. With a fixed number of products, Figure 6 shows also that with more consumer agents, 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 already had the maximum (100%) of behaviour change. We explain both observations by the fact that consumers agents are more dense in the space of possible products and it is easier to aggregate into buying groups and to change their behaviour in that way. The 99.9% of behaviour changes for the 10 agents and 2 products case was caused by one execution where 9 of the 10 agents desired a product and not the second and the other agent wanted the second product and not the first so it ended up buying its preferred product alone (and only one acceptable).

There are 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 Figure 7. This figure shows that only the number of available products influences the type of change of con-

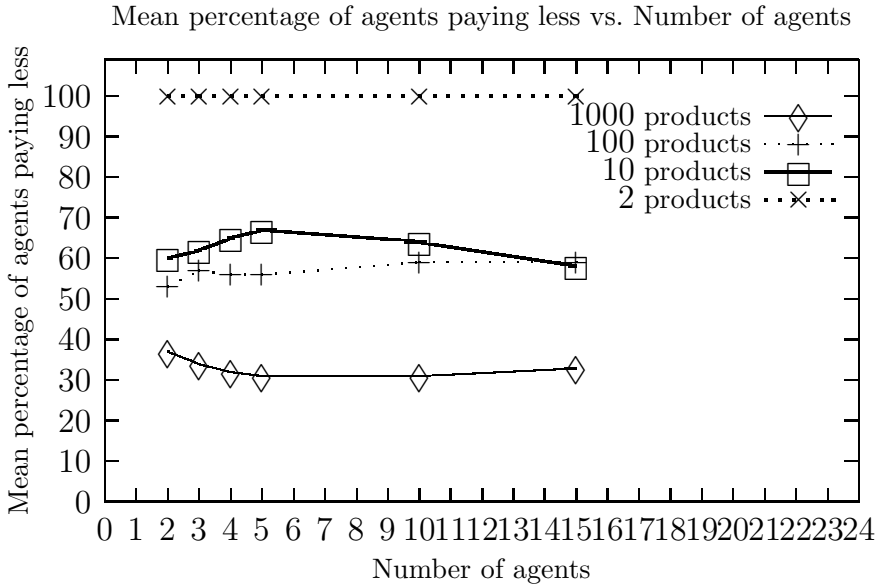

Figure 7: Mean percentage of agents paying less the same product they would have bought alone versus the number of consumers' agents in the protocol for different quantities of available products.

sumers' behaviour. The percentage is relatively the same for a fixed number of products as the four cases with different numbers of products indicate. This percentage changes for different numbers of available products although the cases with 10 and 100 products show similar percentages. When there is little choice of products, few products will be in the preferences list of consumers because some of the buying groups 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 a margin for consumer agents to join groups buying another product than the one their consumer would have bought alone.

6 Conclusions and Future Work

In this paper, we have presented a centralized symmetric protocol without money transfer for group buying that returns a Pareto optimal solution that minimizes the worst distance to ideal among all agents given that the lists of possibilities do not have buying groups which are equally preferred by a consumer. We have found that limiting to one the number of units of a product each agent can buy allows the reduction of the number of possible buying groups to be ordered from an exponential to a linear factor in function of the number of buyers. The protocol changes the buying behaviour of consumers from the normal habit of buying the most preferred product alone at the local store. More changes of buying behaviour occur as the agents become more dense in the space of available products. If few products are available, consumer agents buy the same product as they would have bought alone but at a reduced price. When there is a huge choice of products, consumer agents take the opportunity to join a group that buys another product than the one they would have bought alone. Its execution time complexity is *less* than exponential on average for the studied range (15 agents or fewer) meaning that incentives to regroup could have created a special structure making the group buying problem computationally easier from the execution time complexity point of view. But its memory requirements limit its use to no more than 15 agents for 130 Meg of RAM allocated (no more than 20 agents for 450 Meg of RAM) to the protocol in cases of a high ratio (around 10) of number of agents to quantity of products on the computer platform used for evaluation. Its communication complexity is $\Theta(PN^2)$ messages for P products and N consumers agents in the theoretical worst case.

Future work includes research on the quality of the partition of the set of consumer agents obtained by merging together buying groups with the same product from partitions of the subsets limited to 20 consumer agents. We will also do research on the burden imposed by possible equalities in the preferences list of consumer agent on agents, consumers and sellers. Finally, further evaluation of the protocol will be conducted with different statistical distributions of agents' preferences on products, of available products' specifications and of price schedules.

Acknowledgments

We would like to thank M. Donald E. Knuth for making available by the Web the source code for solving the exact set cover generation problem and M. Donald L. Kreher and M. Douglas R. Stinson for making available by the Web the source code for solving the generation of the K-subsets of a N-set problem. We made some modifications to these source codes for our purpose and translated them into the Java programming language.

References

Agent Oriented Software Pty. Ltd. (2002). JACK Intelligent Agents^{TM} 3.5. Software Agents Development Framework.

Anthony, P. and Jennings, N. R. (2003). Developing a Bidding Agent for Multiple Heterogeneous Auctions. *ACM Transactions on Internet Technology*, **3**(3). To appear.

Asselin, F. and Chaib-draa, B. (2002). Toward a Protocol for the Formation of Coalitions of Buyers. In T. G. Crainic and B. Gavish, editors, *Proceedings of the 5th International Conference on Electronic Commerce Research (ICECR-5)*.

Asselin, F. and Chaib-draa, B. (2003). Coalition Formation with Non-Transferable Payoff for Group Buying. In T. Sandholm and M. Yokoo, editors, *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*. ACM Press. To appear.

Aumann, R. J. (1960). Linearity of Unrestrictedly Transferable Utilities. *Naval Research Logistics Quarterly*, **7**, 281–284.

Breban, S. and Vassileva, J. (2002). A Coalition Formation Mechanism based on Inter-Agent Trust Relationships. In C. Castelfranchi and W. L. Johnson, editors, *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, volume 1, pages 306–307. ACM Press.

Conitzer, V. and Sandholm, T. W. (2002). Complexity of Determining Nonemptiness of the Core. Technical Report CMU-CS-02-137, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, San Francisco, CA, USA.

Golomb, S. W. and Baumert, L. D. (1965). Backtrack Programming. *Journal of the ACM*, **12**(4), 516–524.

Grayson, C. J. (1960). Decisions under Uncertainty: Drilling Decisions by Oil and Gas Operators. Technical report, Division of Research, Harvard Business School, Boston, MA, USA.

Guo, X. (2002). An Optimal Strategy for Sellers in an Online Auction. *ACM Transactions on Internet Technology*, **2**(1), 1–13.

He, M. and Jennings, N. R. (2003). SouthamptonTAC: An Adaptive Autonomous Trading Agent. *ACM Transactions on Internet Technology*, **3**(3). To appear.

Ito, T., Ochi, H., and Shintani, T. (2001). A Group Buy Protocol based on Coalition Formation for Agent-Mediated E-Commerce. In *Proceedings of the 2nd International Conference on Software Engineering, Artificial Intelligence, Networking & Parallel/Distributed Computing (SNPD-01)*.

Knuth, D. E. (2000). Dancing Links. In J. Davies, B. Roscoe, and J. Woodcock, editors, *Millennial Perspectives in Computer Science*, pages 187–214. Palgrave. Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare.

Kraus, S. (2001). *Multi-Agent Systems and Applications*, volume 2086 of *Lecture Notes in Artificial Intelligence*, chapter Automated Negotiation and Decision Making in Multiagent Environments, pages 150–172. Springer-Verlag.

Kreher, D. L. and Stinson, D. R. (1998). *Combinatorial Algorithms : Generation, Enumeration and Search*. CRC Press.

Lerman, K. and Shehory, O. (2000). Coalition Formation for Large-Scale Electronic Markets. In *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS-2000)*, pages 167–174. IEEE Computer Society.

Li, C. and Sycara, K. (2002). Algorithm for Combinatorial Coalition Formation and Payoff Division in an Electronic Marketplace. In C. Castelfranchi and W. L. Johnson, editors, *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, volume 1, pages 120–127. ACM Press.

Lin, Y.-H. and Yuan, S.-T. (2001). Negotiation-Credit Driven Coalition Formation in E-Markets. In S.-T. Yuan and M. Yokoo, editors, *Proceedings of the 4th Pacific Rim International Workshop on Multi-Agents (PRIMA-2001)*, volume 2132 of *Lecture Notes in Artificial Intelligence*, pages 122–138. Springer-Verlag.

Lomuscio, A. R., Wooldridge, M., and Jennings, N. R. (2001). A Classification Scheme for Negotiation in Electronic Commerce. In F. Dignum and C. Sierra, editors, *Agent Mediated Electronic Commerce : The European AgentLink Perspective*, volume 1991 of *Lecture Notes in Artificial Intelligence*, pages 19–33, Berlin, Germany. Springer-Verlag.

Mas-Colell, A., Whinston, M., and Green, J. R. (1995). *Microeconomic Theory*. Oxford University Press.

Osborne, M. J. and Rubinstein, A. (1994). *A Course in Game Theory*. MIT Press, Cambridge, MA, USA.

Rangaswamy, A. and Shell, G. R. (1997). Using Computers to Realize Joint Gains in Negotiations : Toward an Electronic Bargaining Table. *Management Science*, **43**(8), 1147–1163.

Rosenschein, J. S. and Zlotkin, G. (1994). *Rules of Encounter : Designing Conventions for Automated Negotiation among Computers*. MIT Press, Cambridge, MA, USA.

RosettaNet (2003). [Online]. http://www.rosettanet.org/, (Visited on june 17, 2003).

Sandholm, T. W. (1996). *Negotiation among Self-Interested Computationally Limited Agents*. Ph.D. thesis, University of Massachusetts at Amherst.

Sandholm, T. W. (1999). *Multiagent Systems : A Modern Approach to Distributed Artificial Intelligence*, chapter Distributed Rational Decision Making, pages 201–258. MIT Press, Cambridge, MA, USA.

Sandholm, T. W., Larson, K. S., Andersson, M., Shehory, O., and Tohmé, F. (1999). Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence*, **111**(1–2), 209–238.

Sarne, D. and Kraus, S. (2003). The Search for Coalition Formation in Costly Environments. In A. Omicini and S. Ossowski, editors, *Proceedings of the 7th International Workshop on Cooperative Information Agents (CIA'2003)*, Lecture Notes in Artificial Intelligence, Berlin, Germany. Springer-Verlag. To appear.

Sen, S. and Dutta, P. S. (2000). Searching for Optimal Coalition Structures. In *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS-2000)*, pages 287–292. IEEE Computer Society.

Shehory, O. and Kraus, S. (1999). Feasible Formation of Coalitions among Autonomous Agents in Nonsuperadditive Environments. *Computational Intelligence*, **15**(3), 218–251.

Tsvetovat, M., Sycara, K., Chen, Y., and Ying, J. (2001). Customer Coalitions in the Electronic Markets. In F. Dignum and U. Cortès, editors, *Proceedings of the Workshop on Agent-Mediated Electronic Commerce III*, volume 2003 of *Lecture Notes in Artificial Intelligence*, pages 121–138. Springer-Verlag.

Yamamoto, J. and Sycara, K. (2001). A Stable and Efficient Buyer Coalition Formation Scheme. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS-2001)*, pages 576–583. ACM Press.