Journal of Experimental & Theoretical Artificial Intelligence
Vol. 18, No. 1, March 2006, 17–48

Performance of software agents in non-transferable payoff group buying

FREDERICK ASSELIN* and BRAHIM CHAIB-DRAA

Computer Science and Software Engineering Department, Laval University, Quebec City, QC, G1K 7P4, Canada

(*Received 4 April 2005; 14 February 2006*)

Software agents can be useful in forming buyers' groups since humans have considerable difficulties in finding Pareto-optimal deals (no buyer can be better without another being worse) in negotiation situations. What are the computational and economical performances of software agents for a group buying problem? We have developed a negotiation protocol for software agents which we have evaluated to see if the problem is difficult on average and why. This protocol probably finds a Pareto-optimal solution and, furthermore, minimizes the worst distance to ideal among all software agents given strict preference ordering. This evaluation demonstrated that the performance of software agents in this group buying problem is limited by memory requirements (and not execution time complexity). We have also investigated whether software agents following the developed protocol have a different buying behaviour from that which the customer they represented would have had in the same situation. Results show that software agents have a greater difference of behaviour (and 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: Software agents; Coalition formation; Cooperative game theory; Non-transferable payoff; Group buying

1. Introduction

The Internet has reduced the cost of communication and information search. This reduction of cost has enabled the automation of some electronic commerce activities

*Corresponding author. Email: fasselin@damas.ift.ulaval.ca

Journal of Experimental & Theoretical Artificial Intelligence
ISSN 0952–813X print/ISSN 1362–3079 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/09528130500448252

such as group buying negotiations which seek a better economic performance by economies of scale. By negotiation we mean 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 a multi-agent system (MAS). Consumers have an incentive to regroup with the unit price reduction as a function of the number of units bought by the group. However, 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, thus forming as many groups as consumers. Finding a Pareto-optimal partition (no other partition gives more to one consumer without giving less to another) 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). This is not only 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 on the participants. The results of this study showed that humans agreed on a Pareto-optimal outcome in 11.1% of the situations. With the help of a negotiation support system, that percentage rose to 42.6%. Therefore 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 desiring a product (e.g. a television set), group buying consists of partitioning the set of such consumers into buying groups with respect to the preferences of each consumer across 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 NP-hard (Garey and Johnson 1979). Incentives to regroup (a larger group pays less per unit than a smaller one) create a special structure, possibly making the problem computationally easier. An investigation of whether this is the case or not has been 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.

Some useful definitions to help in understanding this paper are given below

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 who 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 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. For example, for agent A in figure 2, the group that buys two units of Product#2 has a distance to ideal of 2 since two buying groups were proposed before it by agent A.

We have developed a protocol and associated algorithms which software agents can use to find a Pareto-optimal partition that minimizes the worst distance to ideal (see Definition 4) among all agents given a strict preference ordering (no two buying groups are equally preferred) for each agent. In this paper we present the results of the empirical evaluation of the computational and economical performances of the protocol for different numbers of agents (with random preferences from consumers) and products. It is organized as follows. In section 2 we briefly review related work. A formal definition of the group buying problem with the assumptions used is presented in section 3. In section 4 we present our protocol, and in section 5 we detail some results and discuss them. Finally, we conclude in section 6. The choice of Pareto-optimality as the solution to this coalition formation problem is discussed in appendix A. In appendix B, we give a detailed presentation of the application of the algorithms used in our protocol to the group buying problem.

2. Prior work on coalition formation in a multi-agent system

2.1 *General research work*

Research in coalition formation in MASs (Sandholm *et al.* 1999, Shehory and Kraus 1999, Larson and Sandholm 2000, Sen and Dutta 2000, Dang and Jennings 2004, Tombuş and Bilgiç 2004) (the list of references is *not* exhaustive) has mainly focused on transferable payoff; like Osborne and Rubinstein (1994), we consider the term 'payoff' to be synonymous with the term 'utility' when we analyse 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. In group buying, transferable payoff is exemplified by the work of Yamamoto and Sycara (2001). In their work, a set of agents forms a buying group purchasing a certain quantity of an item at a certain total price. The problem consists in determining which part of the total price each agent of the buying group must pay.

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 the members receives an individual payoff which does not come from the payoff of the coalition because no such payoff exists. Whereas 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. As we can see, the non-transferable payoff case is the general case and the case of transferable payoff is a specific case. In our version of group buying, each consumer has a private evaluation for each possible buying group. This evaluation depends only on the consumer's preferences and constraints. The evaluation of a particular

buying group by a consumer approximates the utility that this consumer receives in belonging to this buying group.

We could allow agents to exchange money even in the non-transferable payoff case. This means that each agent still has an individual payoff that depends only on its preferences and constraints but it could give (or receive) a payment to (from) another agent that influences both its payoff and the payoff of this other agent. That could allow for a more efficient outcome but at the cost of a more complex problem. Indeed, we would have to compute the payment for each agent as well as truthfully eliciting the relationship between money and the payoff of each agent. As stated by Kraus (2001), money exchange demands resources (and infrastructure) and can be manipulated. However, the main reason we decided not to permit it was for simplicity for the consumers. The common consumer might not understand and accept that he must give a payment to another consumer buying the same item. This could be worse if the payment is made to a consumer of a different buying group. If the payment is included in the buying price, the same consumer would still not understand or accept why he pays a greater price for the same item than another consumer in his buying group. In short, money exchange goes beyond the scope of our present research, but could be studied later.

Formally, we define coalition formation with transferable and non-transferable payoff from cooperative game theory as follows (Osborne and Rubinstein 1994):

Definition 5: A coalition formation game with transferable payoff $\langle M, v\rangle$ consists of:

- a finite set M (the set of agents);
- a function v that associates with every non-empty subset S of M (a coalition) a real number $v(S)$ (the worth of S).

Definition 6: A coalition formation game with non-transferable payoff $\langle M, X, V, (\succeq_i)_{i\in M}\rangle$ consists of:

- a finite set M (the set of agents);
- a set X (the set of consequences);
- a function V that assigns to every non-empty subset S of M (a coalition) a set $V(S) \subseteq X$;
- for each agent $i \in M$ a preference relation $\succeq_i$ on X (formally written as $(\succeq_i)_{i\in M}$). In the transferable payoff case, the preference relation of an agent is implicit in the part of the worth $v(S)$ of S received by the agent belonging to S (the larger the real number, the better).

A coalition formation game with transferable payoff can be associated with an equivalent coalition formation game with non-transferable payoff as follows: $X = \Re^{\mathrm{M}}$, $V(S) = \{x \in \Re^{\mathrm{M}}: \sum_{i\in S} x_i = v(S)$ and $x_j = 0$ if $j \in M\backslash S\}$ for each coalition S, and $x \succeq_i y$ if and only if $x_i \geq \mathrm{y}_i$ (the element i of the vector x is greater than or equal to the element i of the vector y) (Osborne and Rubinstein 1994). Therefore every transferable payoff case can be transformed into a non-transferable case but the inverse is *not* true. Hence non-transferable payoff is more general than transferable payoff.

Caillou *et al.* (2002) have proposed a coalition formation protocol which finds a Pareto-optimal solution with agents using non-transferable payoff. However, their protocol is *not* symmetric, i.e. it does not treat all agents with impartiality

(our protocol is symmetric). The first agent to participate in their protocol chooses its preferred partitions, the second agent chooses its preferred partitions among those chosen by the first agent, the third agent chooses its preferred partitions among those chosen by the second, etc., until the last agent chooses its preferred partitions among those chosen by the next-to-last agent. Clearly, the first agent has an advantage over the second agent, who has an advantage over the third agent, etc. Therefore every agent is treated differently by the protocol, with each agent having an advantage over the successive agents. In the general context of competitive agents in electronic commerce, this is unacceptable. Competition requires that all participants receive equal consideration.

Furthermore, in the worst case, all partitions, even those that are unacceptable to the other agents, must be evaluated by the first agent in their protocol. In contrast, our protocol makes agents compute only 'number of agents × number of different products of the same type' coalitions (with a limit of one unit bought per agent). This quantity is only a small fraction of the total number of all coalitions which, in turn, is only a small fraction of the total number of all partitions. Finally, and more significantly, the problem resolved by their protocol is a matching problem between a group of professors and a group of classes in a course schedule, which is different from our problem.

From the computer science point of view, research on coalition formation with transferable payoff has been related to optimization problems. Generally, researchers in this context have searched for the partition whose sum of the payoffs of its coalitions is optimal. Theoretically, we could also formulate our problem as an optimization problem by searching for the solution that minimizes the sum of the distance to ideal (see Definition 4) of the agents. If we could find the optimal solution to this formulation of our problem, then this optimal solution would also be Pareto-optimal. However, that is theory; in practice the number of coalitions is so large, even for small instances, that they cannot all fit in the memory. For example, our protocol uses only a subset of all coalitions and yet it still went out of memory for larger cases. In order to find an optimal solution, we must know many, perhaps all, coalitions. In contrast, our protocol finds coalitions on an 'as needed' basis, saving memory space. In doing this, it stops after the first coalition structure has been detected which generally happens well before all coalitions are known. Therefore our approach clearly saves memory space so that instances which are *not* computable in the optimization approach are computable in ours.

2.2 *Specific research work for group buying*

The use of coalition formation for buying groups has recently attracted the attention of researchers, mainly in the context of transferable payoff. Thus Tsvetovat *et al.* (2001) have demonstrated the economical incentives of buying groups for both consumers and manufacturers, and have provided models of coalition formation in that context. Lerman and Shehory (2000) have used differential equations to describe the macroscopic behaviour of coalitions when agents are allowed to leave and join a coalition and finally reach a steady state in the distribution of the number of coalitions of different sizes. In their model, agents are mobile and go from vendor site to vendor site where they randomly encounter other lone agents (with whom they form a buying group of two agents on the vendor site of their meeting) or a buying

group already located at a particular vendor site (with whom they join to increase the number of agents in the group by one). In the same model, buying groups stay at their particular vendor site and, in cases where the agents are allowed to leave a buying group to become alone again, the agents resume their random search for lone agents or buying groups. Note that in the model of Lerman and Shehory (2000) all agents have the same goal, i.e. to purchase a specific product at the lowest price (the transferable payoff case). Since agents have no preference among vendors, the optimal solution to their problem is to regroup all agents desiring the same product in any vendor site. If all agents want the same product, then all agents should regroup themselves in one buying group in any one of the vendor sites to attain the optimal solution. Because of the random encounters and the fact that each agent has only local information (such as the size of the buying group at the vendor site where the agent is actually located), the optimal solution is rarely found in the Lerman–Shehory approach, although agents with minimal communication skills could find it (all agents tell a central mechanism which product they want, and the mechanism regroups agents with the same product into a buying group; the agents could even broadcast the product they want to each other, which would be feasible considering the small quantity of communicated information). Furthermore, the solution found using the Lerman–Shehory model is *not* provably Pareto-optimal. The primary interest of their work (which is quite interesting) is the macroscopic model of coalition formation which explains the distribution of the number of coalitions with different sizes at different times until the steady state of the distribution is obtained. It is *not* a practical protocol for group buying because of the absence of a guarantee of the quality of the solution found. It does not guarantee Pareto-optimality.

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 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 reputation in the choice of a coalition manager who 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. Vassileva *et al.* (2002) have studied the concept of trust in long-term coalitions of buyers and sellers over repeated transactions. However, these approaches studied group buying with transferable payoff and therefore did not empirically evaluate an implemented system with non-transferable payoff. Hyodo *et al.* (2003) applied a genetic algorithm to the group buying problem. He and Ioerger (2004) have considered the problem of group buying in conjunction with bundle search where a consumer needs to buy different goods as a bundle. Matsuo *et al.* (2004) used the multi-attribute utility theory, in particular the Analytic Hierarchy Process method, to form buying groups.

Sarne and Kraus (2003) have proposed a model for non-transferable payoff group buying. In this model, each buying group is represented by an agent who incurs a cost for searching for another buying group with which to merge in addition to the cost of the internal coordination of the group members. The buying groups change when agents must decide whether 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 for which we want a Pareto-optimal partition into buying groups.

Leyton-Brown *et al.* (2002) have studied the concept of bidding clubs, as they relate to buying groups, modelling collusion in auctions. However, their work is different from ours because we do not use auctions for group buying. In particular, the unit price paid by a member of a buying group is determined by a price schedule and the number of units bought by the buying group, and is not subject to negotiation as in auctions. Furthermore, agents in group buying are expected to regroup, but collusion of agents is considered undesirable behaviour in an auction, at least from the seller's point of view.

3. Formal definition of the group buying problem

The group buying problem studied in this paper comprises the following.

- A finite and fixed set of consumer agents M with $|M| = m$.
- A finite and fixed set of different available products R of the same type (e.g. a set of different television sets from different manufacturers). Each product $r \in R$ has a price schedule giving the unit price of r as a function of the number of units bought by a buying group.
- The set of possible buying groups O from the consumer agent point of view is determined by the function $f: [1, \ldots, m] \times R \rightarrow O$, meaning that a buying group with one consumer agent and up to m consumer agents can buy any product $r \in R$.
- For a particular consumer agent i, not all buying groups $o \in O$ are desirable: the unit price paid by the group could be higher than the maximum price the consumer is willing to pay, some of the characteristics of the product bought by the group could be unacceptable to the consumer, or the buying group could be *not* individually rational (see appendix A for a definition). Hence we denote by $O_i \subseteq O$ the set of buying groups that are acceptable to consumer agent i.
- Each consumer agent i has a payoff function $u_i: O_i \rightarrow \Re$ which attributes to all acceptable buying groups for i a payoff representing the general level of satisfaction of i for a particular buying group $o \in O_i$. This payoff depends not only on the unit price paid by the buying group o, but also on the other characteristics of the product bought by the group.
- The set of possible buying groups S from the protocol point of view is determined by the function $g: 2^M \backslash \emptyset \times R \rightarrow S$, meaning that a buying group in the solution of the problem can be any non-empty subset of the set of consumer agents M (the power set of M minus the empty set) buying any product $r \in R$.
- A solution to the group buying problem is a collection of buying groups $S' \subseteq S$ so that each consumer agent $i \in M$ is in one and only one set $s \in S'$ (a partition, or exact set cover, of the set of consumer agents M into buying groups $\in S'$). Furthermore, for all buying groups s', $s'' \in S'$, s' and s'' buy different products r' and r'' ($r' \neq r''$) or else they would form together a unique buying group. The partition must be Pareto-optimal.

- The protocol is initiated by a grouping agent that directs its execution. Based on the terminology of game theory, we could say that the grouping agent implements the 'social choice' function h: $[u_1(O_1), \ldots, u_m(O_m)] \rightarrow S'$, meaning that the grouping agent takes the preferences u_i of each consumer agent $i \in M$ on their set of acceptable buying groups O_i and returns a collection of buying groups $S' \subseteq S$ with the properties explained in the previous item. The grouping agent is what is called, in game theory, a mechanism (Mas-Colell *et al.* 1995).

We also make the following assumptions.

- All consumer agents are interested only in their own payoff and not in the payoff of other consumer agents.
- The payoff of each consumer agent is non-transferable.
- Each consumer agent $i \in M$ has in its set of acceptable buying groups O_i a group in which it buys alone a product. This is to ensure that at least one partition (i.e. a solution) exists for the problem.
- No consumer agent can join the protocol after the grouping agent has closed the registration phase. Once a consumer agent has told the grouping agent it is ready to participate in the protocol after having seen the number of registered consumer agents and the set of different available products R, that consumer agent is bound to the solution found by the protocol.
- Every consumer agent $i \in M$ buys only one unit of the product. This assumption is only to restrict the cardinality of the set O of possible buying groups from the point of view of the consumer agent i. The protocol could support multiple units, although with a little unfairness for the agent who bought multiple units (explained in section 4.1).
- The payoff function u_i of each consumer agent must represent strict preferences. This means that for all o', $o'' \in O_i$, $o' \neq o''$ if and only if $u_i(o') \neq u_i(o'')$. Note that if a large buying group o' pays the same unit price as a small buying group o'' for the same product, then $u_i(o') = u_i(o'')$ with $o' \neq o''$, but in that case the grouping agent will consider that the consumer agent i prefers the larger group to the smaller group so that strict preferences are maintained. The protocol could be executed without this assumption, but it helps in obtaining the properties described in propositions 1 and 2.

4. Overview of the protocol

In the protocol developed below, consumers tell their software agent which product they want (e.g. a television set) as well as their preferences across the possible instances of the chosen product (different television sets). 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 parts: (1) determining a preference ordering among all possible buying groups for each software agent; (2) 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 an exponential to a linear factor as a 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 are 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 are $2^N - 1$ possible groups of consumers. Each of them could buy one of P available products (e.g. different television sets not just different units of the same television set). Hence there are $(2^N - 1) \times P$ possible buying groups.

This is a large number of possibilities even for a small number of consumers and products. However, if we restrict the quantity 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 A. The number of subsets of three consumers buying product A is equal to $C_3^{10} = 120$. If we limit the quantity of units each consumer can buy to only one, each group of the 120 groups of three consumers buys three units. Therefore 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 (i.e. groups of the same size buying the same product A are equally desirable). Consumers can now consider only one group of three consumers instead of 120 different groups. For N consumers there are N non-empty groups of different cardinality. Hence there are $N \times P$ possible buying groups when we restrict the quantity 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 a special agent. This special agent will be treated by the protocol as a number of different

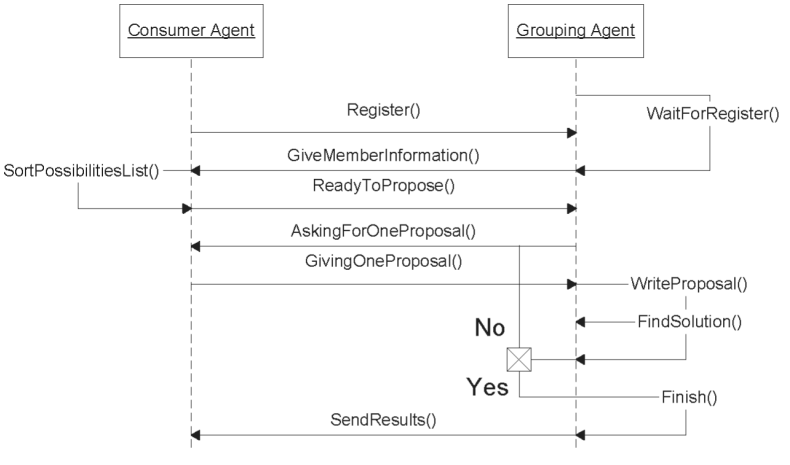

Figure 1. Coalition formation protocol followed by all agents.

agents equal to the quantity of units he wants (one agent per unit). For example, if the consumer tells his agent to buy 50 units, then every buying group proposed by this agent will be treated by the protocol as if the group was proposed simultaneously by 50 different agents. The effects of these special agents on the scalability of the protocol are part of future work. However, there is a disadvantage to this restriction when a consumer wants multiple units of a product. If a consumer wants 50 units, he will pay the same unit price as a consumer who wants only one 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 as a function of the number of consumers, we believe that the restriction is a good compromise. Also, when it comes to buying a high-priced product like a dishwasher, a television, or home tools, consumers usually buy only one unit at a time. High-priced products are exactly the kind of product we see as being well suited for group buying. The purpose of the protocol is to offer to consumers who want only one unit of a product the same price discount as if they wanted many units of a product.

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: many consumer agents and a grouping agent. The consumer agent represents a consumer who desires to buy 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 (i.e. a buying group for which the number of agents who proposed it is at least as great as the number of agents/units in that group). Since all the consumer agents have the same behaviour, there is only one such agent in figure 1.

When the agent has received the product its consumer wants and the preferences over instances of that product, it registers itself to the grouping agent in charge of that 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 via the GiveMemberInformation() operation. The transmitted information consists of the number of registered consumer agents, the specification of the different available products of the chosen type, and their price schedules which return the unit price of the product as a 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 three consumer agents and two products.

Agent A prefers to buy Product#2 with the two other agents (which means three units bought). Agent B's second choice consists of buying Product#2 with one of the other agents (which means two units bought). Agent C's third choice consists of buying alone Product#1 (which means one unit bought). The shaded 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 Product#1 with another agent (the shaded box labelled 'Product#1 at two units') is *not* individually rational because it is *less* preferred than buying alone Product#2 (the unshaded box labelled 'Product#2 at one unit'). Buying alone Product#1 is also *not* individually rational because it is also *less* preferred than

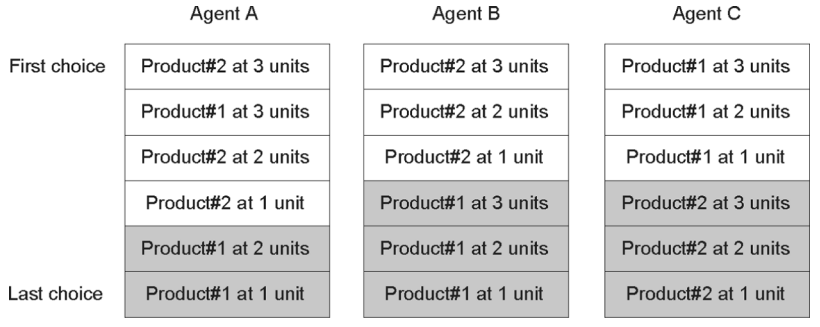

Figure 2. Sorted lists of possibilities for three agents with two different available products.

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 that it is not ready to propose by using the ReadyToPropose() operation with an attribute assigned to false, and it quits the coalition formation protocol; alternatively, it tells the grouping agent that it is ready to propose by using the same operation but with an attribute assigned to true. We impose the condition that each consumer agent must have a buying group in its list of possibilities in which it is alone in order to ensure that a partition exists. In this case, a partition in which each consumer agent is alone in its group is a solution to the problem of group buying. This solution also is individually rational (see appendix A for a definition), and therefore we are assured that an individually rational solution exists for our protocol to find. Otherwise, consumers could wait hours and even days for other consumers to join them to form buying groups and, in 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 product is not individually rational has the same behaviour as the consumer agent quitting the protocol (it does not buy the product).

When all registered consumer agents have told the grouping agent whether they are ready or not to propose, the grouping agent asks each consumer agent who is ready for its preferred buying group using the operation AskingForOneProposal(). The grouping agent also sends to each consumer agent who is ready, the number of consumer agents who have declared themselves ready. In this way, each consumer agent can only propose buying groups which are no larger than the number of ready consumer agents. The consumer agents answer by giving the most preferred buying group (i.e. the description of the product the group buys and the quantity of units bought by the group) in their list which is not already proposed using the operation GivingOneProposal(). Note that, for a given product, a consumer agent prefers a larger buying group to a smaller one because of the possible unit price reduction (the unit price in a larger group is never greater, and is sometimes less, than in a smaller group). Knowing that, each consumer agent will propose, as its most preferred group, a group with the maximum number of buyers (i.e. the total number of

consumer agents that were ready to propose) and a larger group that buys a product A will always be proposed before a smaller group that buys the same product A.

In the context of the Internet, consumer agents will not know the other consumer agents with whom they try to form buying groups or their preferences. In other words, the grouping agent will act as a trusted third party keeping secret the identity and propositions of any consumer agent from the other consumer agents. Without this information, the consumer agent has nothing with which to counter-speculate. We could assume that each consumer agent has distributional information about the preferences of other consumer agents. However, in practice, where do we find accurate probabilities about the preferences of other consumers? Even if we do have this information, we are not sure that the consumers with whom we are interacting are a representative sample of the population on which the probabilities were obtained. Furthermore, the advantage of using distributional information for a consumer agent is only assured over a large number of executions of the protocol. The only sure information available to the consumer agents is that the grouping agent will stop asking for a proposal immediately after it has found a solution. The group containing a particular consumer agent in the solution will have been proposed by it. Therefore, if the preferences of the other agents and the time when the grouping agent will find a solution are not known, a strategy for each consumer agent is to reveal the individually rational buying groups in decreasing order of preference. If a consumer agent does not propose the next most preferred buying group (say BG1) but another one (say BG2), and the grouping agent finds a solution with the consumer agent belonging to the BG2, then the grouping agent could have found a solution with the consumer agent in the more preferred BG1 group instead of the one with the BG2 group if BG1 was proposed before BG2. Nevertheless, if the consumer had proposed BG2 in its right place in the preferences ordering, the grouping agent would still have found the solution with the consumer agent belonging to BG2 if this is Pareto-optimal. Nothing is lost, and sometimes something is gained, by proposing buying groups in decreasing order of preference. Remember also that if the grouping agent is a trusted third party, it has no interest in the outcome of the protocol which is solely executed in the interest of consumers, in order to save consumers money by forming buying groups. In other words, lying without knowing the preferences of the other consumer agents but only the rules of the protocol is damaging to the lying agent. Collusion is also difficult between consumer agents since they do not know each other and therefore cannot communicate with one another.

When all consumer agents have given their proposal, the grouping agent tries to find buying groups that were created with the operation WriteProposal(). This operation works as follows. 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 (out of, say, 15 consumers who where ready to propose), if only nine consumer agents propose it, then it is not created. However, if a tenth consumer agent proposes that group, it becomes effective. This process is known as generation of K-subsets of a N-set which is a combinatorial problem (Kreher and Stinson 1998). It consists of giving all subsets of K elements from a set of N elements. We used a successor algorithm (Kreher and Stinson 1998) which takes as an input a valid subset and gives as output the next subset in the lexicographic order as shown by algorithm 3 (see appendix B). 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 the memory for consideration in later rounds.

Furthermore, since the number of consumers who will propose a given buying group is not known in advance, it would be valuable to generate the proposed groups incrementally. 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 also regenerate 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 a partial input without knowing future items of input. This is known as an online algorithm, in contrast with an offline algorithm (an algorithm with full input from the start). It turns out that algorithm 3 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 offline versions, which is surprising. For example, if a tenth consumer proposes a group of 10 buyers, then the group created is generated with the tenth consumer and nine $(10-1=9)$ consumers chosen from the nine preceding consumers. Since there is only one way to choose nine elements from nine elements, only one group is 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 nine $(10-1=9)$ consumers chosen from the 10 preceding consumers. Since there are 10 ways to choose nine 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 3 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, these groups are generated without algorithm 3.

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 using the FindSolution() operation. This problem is equivalent to the generation of exact set covers which is known to be NP-hard (Garey and Johnson 1979). We used a backtracking algorithm (Knuth 2000) that takes advantage of a heuristic to prune the search tree efficiently to find and generate partitions as shown by algorithm 4 (see appendix B). The purpose of the FindSolution() operation is (i) to establish whether a solution exists after a round of the protocol and then, after one solution is known, (ii) to find a Pareto-optimal solution generated in the round. Algorithm 4 is specialized to decide whether a solution exists but it can also find a Pareto-optimal solution. In all but the most trivial instances of the problem, the first solution will be found after many rounds, and hence many executions of algorithm 4. Therefore it is important to chose an algorithm that performs well in deciding whether a solution exists because this algorithm will be executed often. Once we know that a solution exists, we need to find a Pareto-optimal solution by searching for the solution that minimizes the sum of the distance to ideal (see Definition 4). This is an optimization problem over all solutions generated in the last round and not over all possible solutions. As explained at the end of section 2.1, we cannot generally solve the optimization problem over all possible solutions because of the amount of memory

Table 1. Qualitative evaluation of the proposed protocol.

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

needed, but solving it on only the solutions generated in the last round reduces the memory requirement. Also, an optimization algorithm would be more efficient than algorithm 4 in finding the solution in the last round once we know that a solution exists. However, algorithm 4 is more efficient than an optimization algorithm would be in deciding whether a solution exists in all but the last round. Therefore, in the best situation, we would use algorithm 4 in all but the last round and an optimization algorithm in the last round only. However, we would still have to translate the information in the data structures of algorithm 4 into data structures suitable for the optimization algorithm. The translation process would take additional execution time and memory space. Since the first hurdle of our problem was the consumption of memory space, we leave for future work any improvement of the execution time of the algorithm used in the last round of the protocol.

If no partition exists among the effective buying groups, the grouping agent launches other rounds of proposals until it finds one. We could ask why the consumer agents do not send their complete list of individually rational buying groups the first time that the grouping agent asks for a proposal. It is only in order to reveal the private information of consumer agents on an as-needed basis. However, the protocol would essentially be the same if consumer agents revealed their complete list at once. The grouping agent would consider the consumer agents' buying groups proposals in decreasing order of preference in order to use minimal memory space to store the buying groups that are actually created. Since the communication burden of the entire protocol (see table 1 in section 5.1) is negligible compared with the computational burden of the grouping agent (one combinatorial problem and one NP-hard problem), we do not think that the multiple rounds of proposals add significantly 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 the partitions exhaustively. 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 its buying group in the partition found (SendResults() operation). This partition is one that minimizes the worst distance to ideal among all agents and among all partitions already found.

4.3 *Formal description of the protocol*

Algorithm 1 gives a formal description of the protocol from the point of view of the grouping agent without going into too much detail. The data structure

Algorithm 1 The protocol from the point of view of the grouping agent.

```
repeat
  if a particular consumer_agent wants to register then
    Add that consumer_agent to RegisteredAgents[];
until the registration period has ended
for each consumer_agent in RegisteredAgents[] do
  Send to consumer_agent the number of agents in RegisteredAgents[] and the specification of each
  different available product with its price schedule;
repeat
  if a particular consumer_agent is ready to propose then
    Add that consumer_agent to ReadyAgents[];
until each consumer_agent in RegisteredAgents[] has answered or a time limit has been reached
for each consumer_agent in ReadyAgents[] do
  Send to consumer_agent the number of agents in ReadyAgents[];
StillProposingAgents[] ← ReadyAgents[]; {We make a copy of ReadyAgents[] into StillProposingA-
gents[].}
newBuyingGroup ← false;
roundNumber ← 1;
solution.sumOfDistanceToIdeal ← ∞;
repeat
  for each consumer_agent in StillProposingAgents[] do
    Add the proposal of consumer_agent to BuyingGroupsProposed[];
    if the proposal creates at least an effective buying_group then
      newBuyingGroup ← true;
      if the proposal contains a buying_group with only one consumer_agent then
        Remove consumer_agent from StillProposingAgents[]; {This is the last proposal for this
        consumer agent because all following proposals are not individually rational.}
        Add buying_group to Data Structure of Figure 7;
      else if the proposal contains a buying_group with all consumer_agent in ReadyAgents[] then
        if buying_group.sumOfDistanceToIdeal < solution.sumOfDistanceToIdeal then
          solution ← buying_group;
      else
        Uses Algorithm 3 to generate all buying_group created and place them in Data Structure of
        Figure 7;
  if newBuyingGroup = true then
    newBuyingGroup ← false;
    for each solution_found (if any) by using Algorithm 4 with Data Structure of Figure 7 do
      if solution_found.sumOfDistanceToIdeal < solution.sumOfDistanceToIdeal then
        solution ← solution_found;
    if solution.sumOfDistanceToIdeal ≠ ∞ then
      for each consumer_agent in ReadyAgents[] do
        Send to consumer_agent the solution;
  roundNumber ← roundNumber + 1;
until solution.sumOfDistanceToIdeal ≠ ∞ {Until a solution is found.}
```

RegisteredAgents[] is used to store consumer agents that registered. The data structure ReadyAgents[] is used to store consumer agents that are ready to propose buying groups to the grouping agent. The data structure StillProposingAgents[] is used to store consumer agents that will propose another buying group in the next round of the protocol. The variable newBuyingGroup is a flag used to indicate that at least one new buying group has been created from the proposal in the current round and therefore algorithm 4 must be called to try to find a solution. The variable roundNumber indicates the number of the current round of the protocol. It is used to

Algorithm 2 The protocol from the point of view of a consumer agent.

```
if the consumer agent is interested in a particular product type then
    Send to the grouping agent in charge of that product type a registration message before the end
    of the registration period;
    Receive the number of agents in RegisteredAgents[] and the specification (and price schedule) of
    each different available product of the chosen type;
    With the previous information, sort in decreasing order of preference (prefer the bigger group to the
    smaller one in case of equality of preference between two groups) the N × P possible buying groups
    ignoring the ones that are not individually rational and the ones where the unit price is greater than
    the maximal amount of money the consumer is willing to pay as well as the ones buying a product
    unacceptable to the consumer for any reason;
    if in the sorted list of buying groups (listOfPreference[]) there is a group where the consumer agent
    buys a product alone then
        Send a message to the grouping agent indicating that the consumer agent is ready to propose
        before the time limit;
        Receive the number of agents in ReadyAgents[];
        repeat
            if the grouping agent wants a proposal then
                while the most preferred buying group in listOfPreference[] has more consumer agents than
                the number of agents in ReadyAgents[] do
                    Extract the most preferred buying group from listOfPreference[];
                Extract the most preferred buying group from listOfPreference[];
                Communicate the extracted buying group to the grouping agent;
        until the grouping agent communicates the solution OR no more buying groups are left in listOf-
        Preference[]
        Receive the solution from the grouping agent;
    else
        Send a message to the grouping agent indicating that the consumer agent is not ready to propose
        before the time limit if possible;
```

indicate at which round a given consumer agent has proposed a specific buying group in order to calculate the sum of the distance to ideal of a particular solution (solution.sumOfDistanceToIdeal, i.e. the sum of the round number where each consumer agent proposed the buying group in which it is in that solution). Proposed buying groups are stored in the data structure BuyingGroupsProposed[] and buying groups actually created (synonymous with effective buying groups) are stored in a data structure similar to the one shown in figure 7 (see below). If a consumer agent proposes a buying group in which it is alone, then we know that it will not propose further buying groups in the next rounds because they are *not* individually rational. If all consumer agents in ReadyAgents[] proposed a buying group including all of them, then this buying group is a partition of the consumer agents and therefore it is a solution. Since many solutions can be generated in a given round, we choose the one that minimizes the sum of the distance to ideal between all the solutions found in a given round. If solution.sumOfDistanceToIdeal $\neq \infty$ (its initial value), this means that a solution was found and the protocol communicates it to all consumer agents in the data structure ReadyAgents[]. Otherwise, the number of the current round is incremented and a new round begins.

Similarly, algorithm 2 gives a formal description of the protocol from the point of view of a consumer agent without going into too much detail. With information from the grouping agent, each registered consumer agent can construct a list of preferences

listOfPreference[] for buying groups in the data structure. If the consumer agent is ready to propose, then it will extract the groups stored in listOfPreference[] one by one in decreasing order of preference, ignoring the buying groups containing more consumer agents than the number of agents that are ready to propose. The ready consumer agent will either run out of groups in its listOfPreference[] or the grouping agent will communicate the solution before the consumer agent has time to reach the end of it. If the consumer agent is *not* ready to propose, then it sends an appropriate message to the grouping agent and does nothing more.

5. Results and discussion

To the best of our knowledge, this is one of the first multi-agent 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 protocols cannot be used in group buying with non-transferable payoff as it is defined here because transferable payoff is a special case. Our protocol solves the general case and hence we could use it to solve the specific case which is now solved by the existing algorithms for transferable payoff. Because those algorithms are specialized, they would evidently outperform our protocol on the specific case but they cannot be applied to the general case. Therefore comparison with other protocols loses its relevance. Instead, this research is a 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. We now present some analytical results obtained using our protocol.

5.1 *Analytical results*

Proposition 1: *A partition p found in the first round where such a partition exists and which minimizes the sum of the distance to ideal of the agents compared 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* (*strict preferences*).

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 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, 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 a partition exists and which 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 Suppose that there exists a partition p' which dominates partition p. This means that there exists at least an agent A whose distance to ideal is shorter in p' than in p. None of the other agents have a greater distance to ideal in p' than in p. Therefore 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 assumed to be true is false. No such partition p' dominates the partition p. □

We now give the proof of Proposition 1.

Proof of Proposition 1: By Lemma 1, such a partition p is not dominated by partitions found in the later rounds. Furthermore, by Lemma 2, such a partition p is not dominated by partitions found in the same round. Since no partition exists in rounds preceding 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 proved 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, even minimally, for each additional member in the buying group to incite regrouping, then we assume 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 that 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 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 which 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, 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. □

Rosenschein and Zlotkin (1994) have developed attributes for negotiation mechanisms which were slightly extended by Kraus (2001). Table 1 summarizes these 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 so that an individual agent does not receive a really poor payoff from the solution.

The stability attribute refers to the ways that 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 (see appendix A), but it is the most appropriate since it always exists and does not require summing (or comparing) utilities with an arbitrary scale of values and knowing the payoffs of other consumers. Although theoretically unstable, the Pareto-optimal solution found by the proposed protocol has some practical stability since each consumer agent knows only its payoff. Therefore none of the consumer agents can find another effective buying group in which all agents, including itself, are at least as good as in the partition found by the protocol (their payoff in the buying group is greater or equal to their payoff in the partition found) because of their lack of information (i.e. the payoff of the other consumer agents). The only buying group for which the consumer agents can know all the payoffs of its members are those where they are alone, and they cannot be better than the buying groups found since the solution is individually rational. Consumer agents who still want to find a better buying group for themselves could be worse off than in the partition found since Pareto-optimality means that if at least one agent is better off, at least one other agent is worse off, and they could be that latter agent. Furthermore, the solution found by the proposed protocol has some fairness in the sense that it minimizes the worst distance to ideal among all consumer agents that were ready to propose given strict preferences for all consumer agents.

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: In 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 are $2N$ 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 products is preferred to the $P-1$. Thus, subtracting these $P-1$ groups from the $N \times P$ possible groups 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 *more* than polynomial. However, simulation results presented in section 5.3 indicate that, in practice, the average execution time is at least polynomial for 15 or less software agents. The generation of K-subsets of an N-set problem makes the worst-case memory requirement complexity of the whole protocol *more* than polynomial because the number of K-subsets of an N-set increases exponentially with N increasing for a constant K. The simulation results of section 5.3 indicate that, in practice, the average memory requirement complexity of the whole protocol is at least polynomial for 15 or less software agents. Therefore incentives to regroup (a larger group pays less per unit than a smaller one) could have created a special structure, making the group buying problem computationally easier from the point of view of execution time complexity but *not* from the point of view of memory requirement complexity.

The distribution attribute indicates whether or not the solution is computed in a distributed way. A distributed computation is preferred to a centralized computation because 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. Although we could benefit from the parallelism of distribution on some occasions, it is much more difficult to design a computationally efficient distributed system than a centralized system because we need to consider the additional communication burden of the distributed system.

Symmetry refers to the property of a mechanism of treating 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 whether money is transferred to resolve conflicts between agents. Since money transfer demands resources and can be manipulated, it would be preferable not to be obliged to have it. Also, money transfer could generate discriminatory prices (different agents buying the same product pay different prices) which might not be well accepted or understood by the consumers. We decided to restrict the research reported in this paper by not allowing money transfer.

5.2 *Parameters of the evaluation*

The following parameters have to be randomly generated for the empirical evaluation of the proposed protocol:

- number of consumers
- number of different 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 several numbers of different available 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 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. The unit price for one unit of each available product was chosen with the uniform distribution on the set {\$1700, \$1740, \$1780, . . . , \$2500}. The difference between the worst unit price (for only one unit) and the best (for 1000 units) was chosen with the uniform distribution on the set {\$800, \$840, \$880, . . . , \$1200}. This difference was equally divided into 39 classes of unit prices for the price schedule depending on the number of units bought by the buying group (at most 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 125, 150, 175, 200, 225, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, and 1000 units bought). The preferences of consumer agents were generated by allocating a weight to each attribute and the price (based on a uniform distribution) determining its relevance for a particular consumer. We decided according to a uniform distribution on the set {0,1, . . . ,10} how many non-price attributes received a non-zero weight for a particular available product and consumer agent. Therefore, in one out of 11 chances, the price received 100% of the weight, meaning that the consumer was only interested by a low price. Otherwise, the weight of the price was chosen according to the uniform distribution on the set {10%, 11%, 12%, . . . , 50%}. The non-price attributes receiving non-zero weight were picked randomly (also using a uniform distribution) and their weight was sequentially chosen according to the uniform distribution on the set {17%, 18%, 19%, . . . , 22%} of the remaining unallocated weight except for the last one which received all the remaining unallocated weight. The possible values for each attribute were divided into two sets: the acceptable values and the unacceptable values. However, in the experiment, we decided that all values would be acceptable for all consumers in order to avoid the easy case where some agents have a short list of buying groups to propose. In the evaluated case, all agents have a long list of buying groups to propose, making the number of partitions of the set of agents very large and therefore making the problem difficult. For the evaluation of a particular value of an attribute for a consumer agent, values for an attribute received a weight relative to its rank in the preference order among other values for the related attribute. Values were ranked randomly, again using a uniform distribution for each consumer agent. The first value picked received a weight of 100 points, the second a weight of 90 points, . . . , and the last value received only 10 points. The reserve price of each consumer was chosen from a uniform distribution on the set {\$1500, \$1501, \$1502, . . . , \$2500} so that all consumer agents would be ready to propose to the grouping agent. This was done to ensure that the results for N agents were actually computed for N agents and not for N minus the number of consumer agents that were not ready because no buying group respected their reserve price.

5.3 *Empirical results*

We tested our protocol on a Pentium 4 with a 1.4 GHz processor and 256 MB of RAM of which 130 MB was dedicated to the execution of the protocol. Our protocol was developed using Java Development Kit (JDK) 1.4 and JACK Intelligent Agents^{TM} 3.5 (Agent Oriented Software Pty Ltd 2002), a framework for

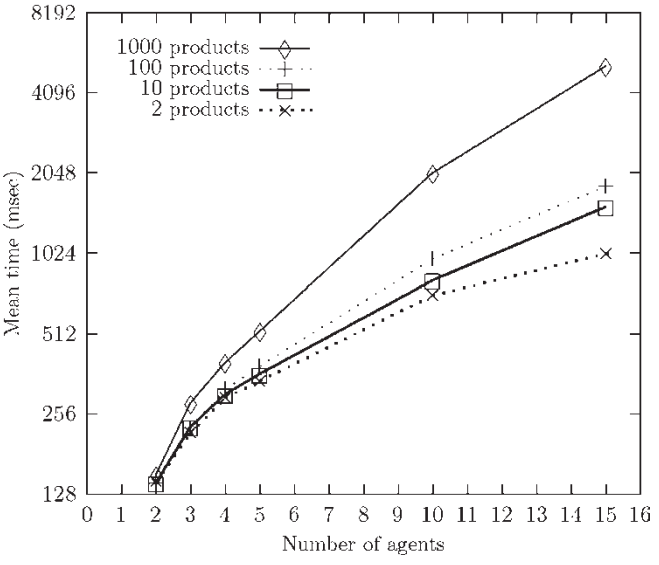

Figure 3. Mean time of execution versus the number of consumer agents in the protocol for different numbers of available products.

programming software agents. We have precisely executed the protocol 1000 times for each number of agents–number of products couple (e.g. 1000 times for two agents with two available products) with random preferences for the agents and always the same list of available products with their price schedule. Each of the five times we executed the protocol for the case of 20 agents and two products, the program went out of memory (even with 130 MB of available memory). Therefore we consider the case with 20 agents to be the limit of the protocol on this computer platform and focus our attention on cases with 15 agents or less. After the evaluation was completed, we ran the protocol on a different platform with more memory but less processing speed. With 450 MB of dedicated memory, we had enough memory for 20 agents but not for 25 agents. Although group buying is more profitable as the number of units bought increases, we often see real-life situations where it is beneficial for groups as small as only two consumers regrouping themselves to buy two units (one for each) and split the savings.

Figure 3 shows the mean time (in milliseconds) of the 1000 executions of the protocol for each couple between the time the grouping agent sends information to registered consumer agents and the time it sends the solution to them. As expected, the execution time increases with the number of available products, but it remains sublinear on a logarithmic scale meaning that the complexity is *less* than exponential for the range studied (2–15 agents). This result is a rather surprising in view of the presence of two combinatorial problems (generation of K-subsets of an N-set and generation of exact set covers). We can explain this by the incentive of there being several members of 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 for each agent is bound by individually rational buying groups, thus greatly limiting the number of proposal rounds.

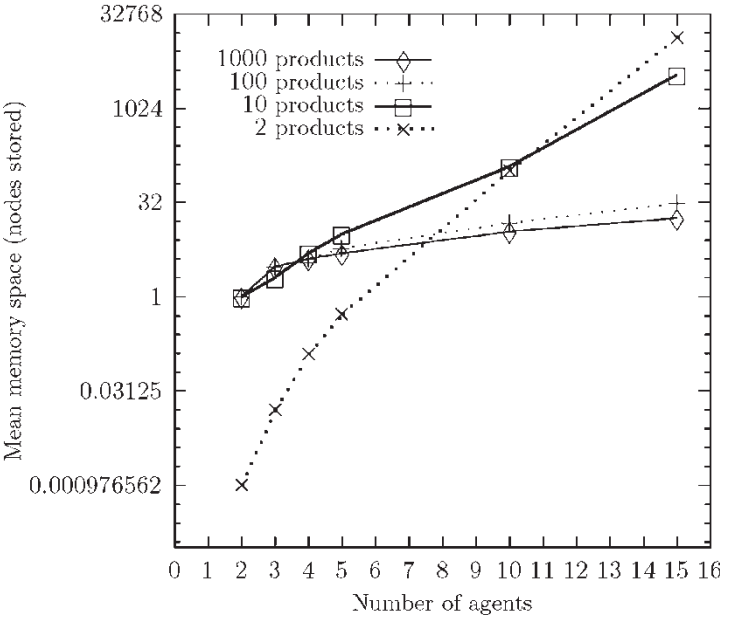

Figure 4. Mean memory space used versus the number of consumer agents in the protocol for different numbers of available products.

Figure 4 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 number of agents–number of products couple. Surprisingly, a larger number of available products results in the use of less memory. This is due to 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 why the case of 20 agents and two products (a ratio of $20/2 = 10$) exceeds the memory capacity of the test platform. The case of 20 agents and 100 products uses much less memory but we cannot ensure that it would not become a case of 20 agents and two 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 the use of less memory space) because their preferences are more widely separated. The cases with two products and between two and five agents used almost no memory because the buying groups formed were already partitions of the set of consumers and therefore included all consumers. The protocol did not store these groups in the memory for later use because it terminated after finding these partitions.

If we use a computer with 450 MB 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 many more because memory consumption increases rapidly as a function of the number of agents (figure 4). If more than 20 agents desire to form buying groups, the grouping agent could divide the set of

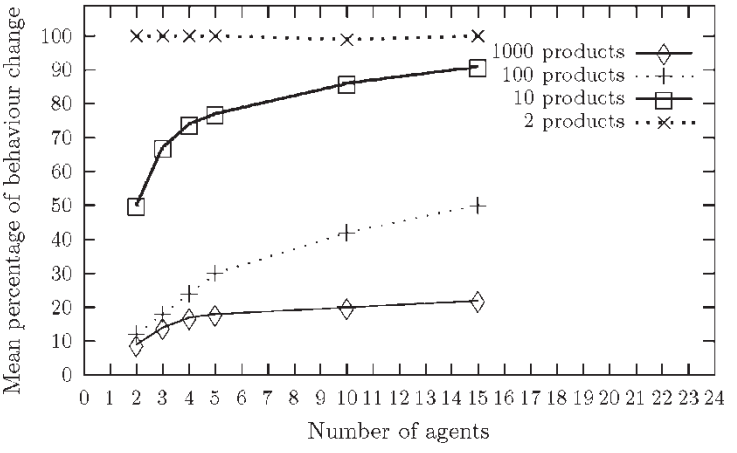

Figure 5. Mean percentage of behaviour change versus the number of consumer agents in the protocol for different numbers of available products.

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. However, we cannot ensure 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 5 shows the percentage change (more than 1000 executions for each number of agents–number of available products couple) 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 5 that with fewer products, the proposed protocol has a better chance of changing the behaviour of the consumers. For a fixed number of products, figure 5 also shows that, with more consumer agents, there is an increase in the percentage of consumers changing their behaviour for three of the four different numbers of products. The cases with two products already had the maximum behaviour change (100%). We can explain both observations by the fact that consumer agents are denser 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–two products case was caused by one execution where nine of the 10 agents desired one product and not the second, and the other agent wanted the second product and not the first so that it ended up buying its preferred product alone (and only one acceptable).

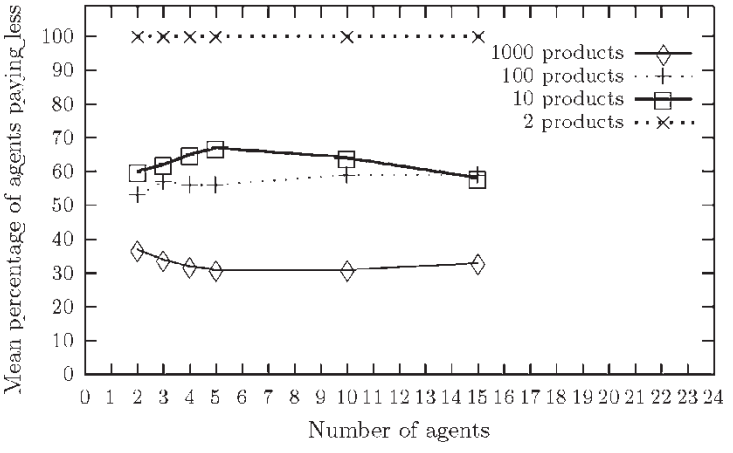

Figure 6. Mean percentage of agents paying less for the same product they would have bought alone versus the number of consumer agents in the protocol for different numbers of available products.

There are only two types of change of behaviour and therefore their percentages complement themselves to 100%. Therefore in figure 6 we show only the percentage of consumers who change behaviour by buying the same product that they would have bought alone but at a lower price. This figure shows that only the number of available products influences the type of change of consumer behaviour. The percentage is relatively the same for a fixed number of products as indicated by the almost flat aspect of the curves for the four cases with different numbers of products. 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 rational and not proposed by the consumer agents. Therefore consumer agents will regroup with others having the same preferred product. However, if there is a very large choice of products, although some buying groups will still be non-individually rational, there will be enough buying groups with different products left to create a margin for consumer agents to join groups buying a product other 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 which 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 as a function of the number of buyers. The protocol

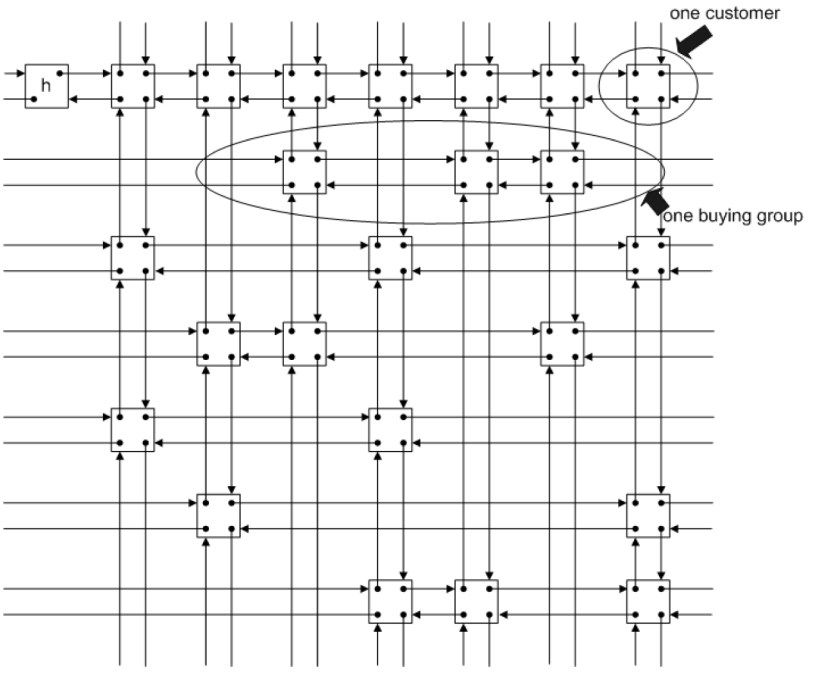

Figure 7. Data structures of an exact set cover problem with seven consumers and six buying groups (from Knuth 2000).

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 very large choice of products, consumer agents take the opportunity to join a group that buys a product other than the one they would have bought alone. The execution time complexity of the protocol is *less* than exponential on average for the range studied (15 agents or fewer), meaning that incentives to regroup could have created a special structure making the group buying problem computationally easier from the point of view of execution time complexity. However, the memory requirements limit its use to no more than 15 agents for 130 MB of RAM allocated to the protocol (no more than 20 agents for 450 MB of RAM) in cases of a high ratio (around 10) of number of agents to number of products on the computer platform used for evaluation. Its communication complexity is $\Theta(PN^2)$ messages for P products and N consumer agents in the theoretical worst case.

Future work will include investigation of the quality of the partition of the set of consumer agents obtained by merging buying groups with the same product from partitions of the subsets limited to 20 consumer agents. We will also investigate the burden imposed by possible equalities in the preferences list of consumer agents, consumers, and sellers. The effects of special consumer agents (those buying multiple units of a product) on the scalability of the protocol also require study. Finally, further evaluation of the protocol will be conducted with different statistical

distributions of agents' preferences of products, specifications of available products, and price schedules.

Acknowledgements

The authors would like to thank Mr D.E. Knuth (source code for solving the exact set cover generation problem) and Mr D.L. Kreher and Mr D.R. Stinson (source code for solving the generation of the K-subsets of an N-set problem) for making the source codes available on the Internet. We made some modifications to these source codes for our purpose and translated them into the Java programming language. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and in part by the Social Sciences and Humanities Research Council of Canada (SSHRC).

Appendix A. Choice of solution

In game theory, not only is finding the solution to a problem a valid research agenda, but also determining what could be the solution to the problem. 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 using the word 'solution'. Instead, they use the expression 'solution concept'. Several solution concepts exist for cooperative game theory with non-transferable payoff as well as general solution concepts which could also apply to this case. We consider in this list of solution concepts properties of solutions (optimal social welfare, Pareto-optimality, and individual rationality) as well as equilibria (core, stable set, bargaining set, and kernel), although it is an abuse of language. The most useful of these solution concepts are as follows.

Social welfare: this is the summation of the utility of each consumer. Since 'each agent's utility function can only be specified up to positive affine transformation' [$a \times u(\bullet) + b$ such that $a > 0$ and $b \in \Re$ is such a transformation on $u(\bullet)$] (Mas-Colell *et al*. 1995, cited by 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 one other. 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 that 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 if he had acted alone. The partition composed of each consumer buying alone has the vector of the minimum payoffs that each consumer could receive. If an individually rational solution exists then there is also an individually rational Pareto-optimal solution. Either an individually rational solution is Pareto-optimal or it is eventually

dominated by a Pareto-optimal solution that is better and therefore also individually rational itself.

Core: this 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 whether it is empty or not is an *NP*-complete problem (Conitzer and Sandholm 2003). So why design and implement a protocol that tries to find a solution that does not always exist and, furthermore, 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 same problem as the core; it can be empty (Osborne and Rubinstein 1994).

Bargaining set: This solution concept uses objections and counter-objections. A solution belongs to the bargaining set if for every objection there is a counter-objection. This set is always non-empty and the core is always a subset of it (Sandholm 1996). One disadvantage of this solution concept is that every partition of the set of buyers has its bargaining set. Therefore 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 in order to make objections and counter-objections, which is rarely the case in real situations. For all these reasons, this solution concept is not applicable to the group buying problem.

Kernel: As is the case for the bargaining set, this solution concept uses objections and counter-objections, although differently defined. Unlike the bargaining set, the kernel requires that we compare the utilities of different agents which can have an arbitrary scale of value, thus resulting in the same inapplicability previously seen in the social welfare. Therefore 'the kernel is an appropriate solution concept only in situations in which the utilities of different players can be meaningfully compared' (Osborne and Rubinstein 1994) which is not the case for competitive agents with privately defined utilities as in this group buying problem.

The solution concept chosen for the group buying problem must have some basic properties. First, it must always exist (not sometimes be an empty set), unlike the core and the stable set. Secondly, it must not require to sum (or compare) utilities with an arbitrary scale of value like social welfare (or the kernel) or have knowledge of other consumers' payoffs like the bargaining set. Since there is an individually rational Pareto-optimal solution if an individually rational solution exists, and we ensure that such a solution exists in our protocol (as explained in section 4.2), we believe that individually rational Pareto-optimality is the most appropriate solution concept for the group buying problem consideredin this paper.

Appendix B. Algorithms used

Algorithm 3 is used in the WriteProposal() operation of the protocol. It simply generates all newly created buying groups.

Before explaining algorithm 4, the data structures used must be examined. Each consumer is represented by an object called a 'column object' within a double-linked circular list which has a root object named h. Each buying group is also represented by a double-linked circular list of objects, each representing a consumer that is a member of the buying group. The lists of buying groups stack up as in figure 7 and objects representing the same consumer are double-linked circularly among themselves with the 'column object' representing that consumer. Of course, buying

Algorithm 3 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**
    $i \leftarrow i - 1$
  if $i = 0$ **then**
    Terminate
  else
    for $j \leftarrow i$ to K **do**
      $u_j \leftarrow t_i - 1 + j - i$
    Return $\vec{U}$

Algorithm 4 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 7. At the first call, $k = 0$.
  if $R[h] = h$ **then**
    Print the solution
    Return
  else
    $s \leftarrow \infty$ {Start of the heuristic of Golomb and Baumert}
    for all $j \leftarrow R[h], R[R[h]], ...,$ **while** $j \neq h$ **do**
      if $S[j] < s$ **then**
        $c \leftarrow j$
        $s \leftarrow S[j]$
    {End of the heuristic of Golomb and Baumert}
    Cover column c {The Cover subroutine is presented in algorithm 5}
    for all $r \leftarrow D[c], D[D[c]], ...,$ **while** $r \neq c$ **do**
      $O_k \leftarrow r$
      for all $j \leftarrow R[r], R[R[r]], ...,$ **while** $j \neq r$ **do**
        Cover column $C[j]$ {The Cover subroutine is presented in algorithm 5}
      Search$(k+1)$ {Recursive call to the Search routine}
      $r \leftarrow O_k$
      $c \leftarrow C[r]$
      for all $j \leftarrow L[r], L[L[r]], ...,$ **while** $j \neq r$ **do**
        Uncover column $C[j]$ {The Uncover subroutine is presented in algorithm 6}
    Uncover column c {The Uncover subroutine is presented in algorithm 6}
    Return

Algorithm 5 Cover algorithm for 'column object' covering (from (Knuth 2000)).

Require: c ('column object' to cover) and data structures like the ones of Figure 7.
  $L[R[c]] \leftarrow L[c]$
  $R[L[c]] \leftarrow R[c]$
  for all $i \leftarrow D[c], D[D[c]], \ldots$, **while** $i \neq c$ **do**
    for all $j \leftarrow R[i], R[R[i]], \ldots$, **while** $j \neq i$ **do**
      $U[D[j]] \leftarrow U[j]$
      $D[U[j]] \leftarrow D[j]$
      $S[C[j]] \leftarrow S[C[j]] - 1$

Algorithm 6 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 7.
  for all $i \leftarrow U[c], U[U[c]], \ldots$, **while** $i \neq c$ **do**
    for all $j \leftarrow L[i], L[L[i]], \ldots$, **while** $j \neq i$ **do**
      $S[C[j]] \leftarrow S[C[j]] + 1$
      $U[D[j]] \leftarrow j$
      $D[U[j]] \leftarrow j$
  $L[R[c]] \leftarrow c$
  $R[L[c]] \leftarrow c$

groups including all customers are not represented in the data structures used to find a partition because we already know that they are a partition.

The main operations of algorithm 4 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').

Using algorithm 4, we generate all partitions of the set of consumers from a set of buying groups by recursively choosing a buying group ($O_k \leftarrow r$) which 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) with 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 which 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 a heuristic (Golomb and Baumert 1965) to make the search tree as narrow as possible so that the pruning of this tree eliminates as many search paths as possible.

References

Agent Oriented Software Pty Ltd, JACK intelligent agents™ 3.5, Software Agents Development Framework, 2002.

P. Caillou, S. Aknine, and S. Pinson "A multi-agent method for forming and dynamic restructuring of pareto optimal coalitions", in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, Bologna, Italy, Vol. 1, 2002, pp. 1074–1081.

V. Conitzer and T. Sandholm, 2003, "Complexity of determining nonemptiness of the core", in *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003*), Acapulco, Mexico, 2003, pp. 613–618.

V.D. Dang and N.R. Jennings, "Generating coalition structures with finite bound from the optimal guarantees", in *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, New York, 2004, pp. 564–571.

M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco, CA: W.H. Freeman, 1979.

S.W. Golomb and L.D. Baumert, "Backtrack programming", *J. ACM*, 12, pp. 516–524, 1965.

L. He and T.R. Ioerger, "Combining bundle search with buyer coalition formation in electronic markets: a distributed approach through negotiation", in *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, New York, 2004, pp. 1438–1439.

M. Hyodo, T. Matsuo and T. Ito, "An optimal coalition formation among buyer agents based on a genetic algorithm", in *International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE'03)*, Loughborough, UK, 2003, pp. 759–767.

T. Ito, H. Ochi, and T. Shintani, "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)*, Nagoya, Japan, 2001.

D.E. Knuth, "Dancing links", in *Millennial Perspectives in Computer Science: Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare*, Oxford, 2000, pp. 187–214.

S. Kraus, "Automated negotiation and decision making in multiagent environments", in *Multi-Agent Systems and Applications*, M. Luck, V. Marik, O. Stepankova and R. Trappl, Eds, Berlin, Springer-Verlag, 2001, pp. 150–172.

D.L. Kreher and D.R. Stinson, *Combinatorial Algorithm, Generation, Enumeration and Search*, Boca Raton, FL, CRC Press, 1998.

K.S. Larson and T.W. Sandholm, "Anytime coalition structure generation: an average case study", *J. Exp. Theor. Artif. Intell.*, 12, pp. 23–42, 2000.

K. Lerman and O. Shehory, "Coalition formation for large-scale electronic markets", in *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS-2000)*, Boston, MA, 2000, pp. 167–174.

K. Leyton-Brown, Y. Shoham, and M. Tennenholtz, "Bidding clubs in first-price auctions", in *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Canada, 2002, pp. 373–378.

C. Li and K. Sycara, "Algorithm for combinatorial coalition formation and payoff division in an electronic marketplace", in *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, Bologna, Italy, 2002, Vol. 1, pp. 120–127.

Y.-H. Lin and S.-T. Yuan, "Negotiation-credit driven coalition formation in e-markets", in *Proceedings of the 4th Pacific Rim International Workshop on Multi-Agents (PRIMA-2001)*, Taipei, Taiwan, 2001, pp. 122–138.

A.R. Lomuscio, M. Wooldridge and N.R. Jennings, "A classification scheme for negotiation in electronic commerce", in *Agent Mediated Electronic Commerce: The European AgentLink Perspective*, F. Dignum and C. Sierra, Eds, Berlin, Springer-Verlag, 2001, pp. 19–33.

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

T. Matsuo, T. Ito, and T. Shintani, "A buyers integration support system in group buying", in *Proceedings of the IEEE International Conference on E-Commerce Technology (CEC'04)*, San Diego, CA, 2004, pp. 111–118.

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

A. Rangaswamy and G.R. Shell, "Using computers to realize joint gains in negotiations: toward an electronic bargaining table", *Manage. Sci.*, 43, pp. 1147–1163, 1997.

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

RosettaNet. Available online at: http://www.rosettanet.org/(accessed 13 January 2006).

T.W. Sandholm, "Negotiation among self-interested computationally limited agents". PhD thesis, University of Massachusetts at Amherst (1996).

T.W. Sandholm, ''Distributed rational decision making'', in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed., Cambridge, MA, MIT Press, 1999, pp. 201–258.
T.W. Sandholm, K.S. Larson, M. Andersson, O. Shehory and F. Tohmé, ''Coalition structure generation with worst case guarantees'', *Artif. Intell.*, 111, pp. 209–238, 1999.
D. Sarne and S. Kraus, ''The search for coalition formation in costly environments,'' in *Proceedings of the 7th International Workshop on Cooperative Information Agents (CIA'2003)*, Helsinki, Finland, 2003, pp. 117–136.
S. Sen and P.S. Dutta, ''Searching for optimal coalition structures'', in *Proceedings of the 4th International Conference on MultiAgent Systems (ICMAS-2000)*, Boston, MA, 2000, pp. 287–292.
O. Shehory and S. Kraus, ''Feasible formation of coalitions among autonomous agents in nonsuper-additive environments'', *Comput. Intell.*, 15, pp. 218–251, 1999.
Ö. Tombuş and T. Bilgiç, ''A column generation approach to the coalition formation problem in multi-agent systems'', *Comput. Oper. Res.*, 31, pp. 1635–1653, 2004.
M. Tsvetovat, K. Sycara, Y. Chen, and J. Ying, 2001, ''Customer coalitions in the electronic markets'', in *Proceedings of the Workshop on Agent-Mediated Electronic Commerce III*, Barcelona, Spain, 2001, pp. 121–138.
J. Vassileva, S. Breban and M.C. Horsch, ''Agent reasoning mechanism for long-term coalitions based on decision making and trust'', *Comput. Intell.*, 18, pp. 583–595, 2002.
J. Yamamoto and K. Sycara, ''A stable and efficient buyer coalition formation scheme'', in *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS-2001)*, Montréal, Canada, 2001, pp. 576–583.