# Bilateral multi-issue negotiation of execution contexts by proactive document agents

Jerzy Kaczorek

The University of Computer Science and Skills,
Lodz, Poland
Email: jkaczorek@gmail.com

Bogdan Wiszniewski*

Department of Intelligent Interactive Systems,
Faculty of Electronics, Telecommunications and Informatics,
Gdansk University of Technology,
Gdansk, Poland
Fax: +48-58-347-22-22
Email: Bogdan.Wiszniewski@eti.pg.edu.pl
*Corresponding author

**Abstract:** A proactive document can react to its actual environment by autonomously selecting and performing actions integrated into its body and interact with its user. When migrating over a network of execution devices it may encounter diverse execution contexts, each one set up according to temporal characteristics of a receiving device and preferences of its owner. A concept to augment proactive documents with negotiation capability is proposed – to make them responsive to such dynamically changing contexts, and implemented in a system, where they can migrate as attachments to e-mail messages, owing to a dedicated e-mail client capable of handling them. Negotiation is based on a simple game-theoretic mechanism to minimise computation load on execution devices. Four negotiation algorithms are proposed and two of them evaluated in more detail in a series of experiments, when respectively, negotiating parties do not or do have knowledge on past encounters and negotiated contracts.

**Keywords:** proactive documents; dynamic execution contexts; mobile agents; ad hoc collaborative processes.

**Reference** to this paper should be made as follows: Kaczorek, J. and Wiszniewski, B. (2019) 'Bilateral multi-issue negotiation of execution contexts by proactive document agents', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 32, No. 3, pp.180–196.

**Biographical notes:** Jerzy Kaczorek obtained his MSc in Computer Science in 2006, and PhD in 2015, both in Computer Science from the Gdansk University of Technology (GUT), Poland. He is currently employed as a CIO in a private company in Grudziadz, Poland, and as an Assistant Professor at the University of Computer Science and Skills in Bydgoszcz, Poland. His research interests include document engineering, intelligent agents, automated negotiation and game theory.

Bogdan Wiszniewski is graduated from the Gdansk University of Technology in 1977 and was awarded his MSc in Computer Science and Engineering with honours. In 1984 and 1998, respectively, he completed his PhD and DSc. In 2006 he was awarded a Professor title by the President of Poland. He was a Lecturer in the universities in Canada, USA and UK. He was also the Principal Investigator or coordinator in many national and international R&D projects with the significant industrial content. He is an author or co-author of many publications in national and international journals and conferences, including several national and international monographs. He was awarded twice by the Polish Minister of Higher Education and Science for his scientific merits. His current research focus on distributed processing, in particular open multi-agent systems and eCollaboration using methods and tools of document engineering.

This paper is a revised and expanded version of a paper entitled 'Bilateral multi-issue negotiation between active documents and execution devices' presented at 9th Int. Conf. on Digital Society ICDS'15, Lisbon, Portugal, 22–27 February 2015.

## 1 Introduction

Knowledge workers, who collaborate in a network organisation, may interact ad-hoc, and in a loosely-coupled manner by exchanging electronic documents that constitute units of *information*, and at the same time, units of *interaction* (Glushko and McGrath, 2008). This dichotomy has become apparent with the advent of *proactive documents*, implemented as autonomous *software agents* with built-in execution capability. Examples of technical implementation of such documents include *placeless documents* (Dourish et al., 2000), *living documents* based on an agent framework (Schimkat and Küchlin, 2002) or *interactive documents* implemented by us as mobile agents – which instead of a real agent platform rely on commonly available standard e-mail services to migrate over a network of loosely coupled mobile devices. Our document-agents can be proactive in three ways: by embedding pieces of code or scripts that may be executed directly on the receiving device, scripts that can test for availability and request activation of various local tools installed on the device, or scripts requesting the local worker's system to call specific remote services the document would like to have access to.

When migrating over the network, the proactive document can carry both, the *content* to be worked on, and the specification of its migration path including *activities* and *transitions* of the workflow process it implements. Each activity represents a piece of work to be performed on the visited device, whereas the relevant transition indicates where the outgoing document (constituting a result of the activity) should migrate next. Such a proactive document-agent can combine a passive content with active services to explore information resources, to enable interaction of users with its logical structure elements, and to automatically update its content. Our implementation allows proactive documents not to subscribe to any particular vendor, tool or format, and could be attached to e-mail messages as any MIME conformant content (Freed and Borenstein, 1996).

One advantage of the proactive document implemented as an agent is its autonomy in achieving goals predefined by its author. Based on the functionality integrated into its body, its knowledge on the implemented business process, its ability to migrate to other nodes of a distributed system and to learn to make optimal decisions under uncertainty, it can adapt to various conditions provided by dynamically changing execution contexts. On the other hand, its major disadvantage is total dependence on computational resources provided by the current execution device used to perform the relevant activity of its workflow process. In that sense an execution context may often be considered adverse by a document-agent, as its expectations or needs may be in conflict to what a device would be willing to offer. In consequence, a document should be able to adequately react and adjust to that context at some acceptable cost.

A special workflow engine in a form of a dedicated e-mail client to be installed on a device to make it capable of handling such documents was implemented by us on several mobile platforms during the research reported in this paper, including Windows Phone, iOS and Android, and capable of detaching and unmarshaling proactive attachments, activating them to perform a respective activity, and upon its conclusion, marshaling and sending them further as e-mail attachments to other collaborators indicated in their migration paths. In general, functionality of our workflow engine can also support negotiation between documents it handles and the device it is installed on, as described in the paper.

A single activity of a workflow process implemented by a proactive document could be performed in diverse execution contexts – each one set up dynamically, according to specific characteristics of the receiving device, its current location and preferences of its owner (worker).

A preferences file, prepared by the worker, enables the workflow engine to switch a device between modes, according to its current location or preoccupation. Besides the *airplane mode*, when certainly using any network by a device would not be allowed, other possible modes may include: a *business mode*, assuming access to the corporate, trusted network from inside of the worker's organisation, a *travel mode*, when access to available networks is possible but unstable or not recommended for security reasons, a *private mode*, when some computations a document may be willing to perform would not be possible, and so on. Modes could further be combined with specific features of a particular device class (for instance, a laptop or a smartphone), its current state (when getting low on the battery or approaching the allowed monthly data transfer limit) and personal preferences of the worker (in particular accessibility options for impaired users). Further in the paper we distinguish several classes of devices and define various options for them.

Motivation for this work has been to augment functionality of proactive documents with negotiation capability to make them responsive to such execution contexts. Each single activity could be then performed without undue delay and in an optimal way, helping to rationalise document flows in the related business process. A negotiation mechanism combined with a built in migration path would enable a process originator to set up ad-hoc a temporary virtual organisation of collaborators, and in a loosely coupled manner – without a need to configure any specific distributed system infrastructure in advance.

Examples of configuration needs include opening FTP, HTTP/HTTPS ports in local systems of the prospective participants of the planned process, configuring firewalls protecting their respective sites and servers, or when some 'fully loaded' agent framework is used, like JADE for example, also setting-up the agent platform by adding and connecting special software objects (containers) before running it (Bellifemine et al., 2007). When compared to the above, running a temporary virtual organisation with e-mail services would not require workers to be competent system administrators. Since e-mail is commonly used across organisations to exchange documents, one may reasonably assume that all the required services are already set up

properly and related SMTP, IMAP and POP3 ports are opened (Cotton et al., 2011).

Our negotiation mechanism is bilateral and has been designed to be extremely simple and easy to implement – to minimise computation load on execution devices, as very often no external negotiation service, such as the one described in Bala et al. (2013), could be provided for performance and security reasons in a dynamically changing execution context of a mobile device.

It involves two rational parties – a document-agent and a device-agent. They are rational in the economic sense, i.e., during negotiation each one attempts to maximise its preferences, modelled as a utility function (Faratin et al., 1998). Preferences of a document author (originator) are implemented by the document's code, whereas preferences of a device owner (user) are specified in the preferences file and interpreted by the local workflow engine handling the received document. Both agents are autonomous, i.e., they negotiate one with another in the name of their human 'perpetrators'.

The rest of the paper is organised as follows. Section 2 defines structure of multi-issue offers, which are exchanged between parties during negotiation, and specify various characteristics of execution contexts. Section 3 introduces a simple bargaining game (SBG) as an underlying negotiation model and discusses its several variants for resolving conflicts between the document and its execution device during their single encounter, i.e., without any knowledge on prior encounters and previously negotiated contracts. Section 4 expands the proposed negotiation model by a machine learning mechanism, to train documents to play the proposed game during repeated encounters, when historical data on past negotiations with various devices exist. Section 5 evaluates performance of two basic algorithms to play it, for single and repeated encounters respectively. Section 6 reviews related work and Section 7 concludes the paper.

## 2 Multi-issue offers

Reaching agreement on the particular execution context by conflicting parties requires them to have the same understanding of all essential issues making that context. Further in the paper we have assumed five such issues, enabling nondiscriminating representation of a possibly broad range of negotiated options in a single offer. We model them as attributes $A_1, ..., A_5$, which values are represented by specific combinations of their relevant option flags, with '0', '1' and '–' indicating respectively that a given option 'is not' or 'is' present in the offered/required execution context, or may be 'any'. Moreover, the order in which these attributes have been defined is intended to reflect hierarchical dependency of negotiated issues. Below we just outline our coding scheme of attribute values; their detailed description could be found in Kaczorek (2015).

### 2.1 $A_1$: performer

Each workflow activity could be performed by a document or a worker, acting alone or jointly. We distinguish three groups of options related to this issue, labelled respectively by $D$, $W$, and $J$. They are further split in more specific option values, which are represented with binary flags. Each flag indicates if a given option should be considered within the related issue: whether a performer could be the worker (*Wkr*) operating a device, a service (executable code) embedded in the document (*EmS*), an external service (*ExS*) the document would like to call via the device, or a local tool (*LoT*) or service (*LoS*) it may want to access when executing on the device. For example, by combining these flags as specified by formula (1) one would set the 'performer' issue of the negotiated execution context of group $D$ to $D3$, indicating that: the document would like to perform its activity on its own by using its embedded code and some external service but without any specific tool and independently of any service installed on the device.

$$\begin{array}{lccccc} & Wkr & EmS & ExS & LoT & LoS \\ D3 = & 0 & 1 & 1 & 0 & - \end{array} \tag{1}$$

### 2.2 $A_2$: availability of resources

Completion of the activity may depend on what kind of network the execution device could provide, what browsers and possibly other tools are installed on it, their compatibility to the ones required by a document, the input method of user data, and so on. We distinguish three groups of options related to this issue: when no network connection is possible, when it can be made from inside of the worker's organisation, or from outside of it, labelled respectively by $S$, $I$ and $E$. They are further split in more specific option values (flags): *eIP* and *lIP* indicating respectively if a device could be exclusively or alternatively connected from inside or outside of the worker's organisation, whether a specific browser (*SpB*) or tool (*SpT*) is required, or their proposed substitutes (*AnB*, *SuT*) are possible, and a full-size keyboard (*FKb*) is connected or just a smaller set of selection buttons (*SeB*) should be used instead. For example, by combining these flags as specified by formula (2) one would set the 'availability' issue of the negotiated execution context of group $E$ to $E1$, indicating that: the device could be connected from outside of its host organisation and the document would not care about which browser to use nor seek any other support from the device.

$$\begin{array}{lcccccccc} & locIP & extIP & SpB & AnB & SpT & SuT & FKb & SeB \\ E1 = & 0 & 1 & - & - & 0 & 0 & 0 & 0 \end{array} \tag{2}$$

### 2.3 $A_3$: performance characteristics

Performance of the document during the activity depends on the quality of the link providing access of the execution device to the internet and physical characteristics of its hardware. We distinguish five groups of options related to this issue: when no connectivity exist (or is not

recommended by the device), or either WiFi, a slower 3G or the faster 4G/LTE modem or the classic (twisted pair) Ethernet connectivity could be provided. They are labelled respectively by $U$, $R$, $M$, $A$ and $N$, and further split in more specific option values, indicating whether the connection may be wired (*Wre*), would use a TV cable or a telephone line (*C/L*), a cellular modem (*TMo*) or a wireless (WiFi) network (*Wrs*) card. Moreover, a device may have a processor (*CPU*) above or below some average power required by a document, and provide more or less memory (*RAM*) than some optimum required by a document. For example, by combining these flags as specified by formula (3) one would set the 'performance' issue of the negotiated execution context of group $R$ to $R2$, indicating that: the device could access only a wireless network and the document would accept less powerful CPU but with more RAM instead.

$$\begin{array}{lcccccc} & Wre & C/L & TMo & Wrs & CPU & RAM \\ R2 = & 0 & 0 & 0 & 1 & 0 & 1 \end{array} \tag{3}$$

### 2.4 $A_4$: security mechanisms

During each activity a minimum security of both parties must be provided, in particular when the document content is processed, and when the execution device agrees to connect to the external service requested by the document. We distinguish four groups of options related to this issue: when the connection has no security mechanisms involved, when the wireless network is protected by the access key, when the secure data transfer protocol is used, or both, the access key and the secure data transfer protocol are used. They are labelled respectively by $P$, $K$, $T$, and $C$, and further split in more specific option values, indicating whether a related remote site provides HTTPS (*SeT*), the involved wireless connection is protected by the access key (*AcK*), a document is digitally signed (*DSg*), and a device has an anti-virus tool installed (*AnV*). For example, by combining these flags as specified by formula (4) one would set the 'security' issue of the negotiated execution context of group $K$ to $K4$, indicating that: the document would accept insecure data transfer but require a wireless network to be protected by the access key, its content would be digitally signed and scanned with the device's antivirus software.

$$\begin{array}{lcccc} & SeT & AcK & DSg & AnV \\ K4 = & 0 & 1 & 1 & 1 \end{array} \tag{4}$$

### 2.5 $A_5$: eeliability of worker-document interaction

During each activity the appropriate reliability level of human performance should be supported by the execution device. We distinguish four groups of options related to this issue: when the device's system does not provide any support to protect the document content from being lost because of user mistake, when some elementary back-up mechanisms for the document content are provided, when the document content is fail-safe, and when the execution device could provide a maximum possible level of reliability of man-machine interaction. They are labelled respectively by $L$, $B$, $F$, and $H$, and further split in more specific option values, indicating whether the worker can decide on permanence of document content modifications with an acceptance button (*AcB*), the autosave mode can be set to prevent loosing accidentally the content entered so far by mistake (*ASv*), correctness of the content being entered may be improved by an automatic check of correctness (*ACh*) facility, and an undo button (*UdB*) is provided to improve comfort of work of users and further reduce the rate of errors when modifying document content. For example, by combining these flags as specified by formula (5) one would set the 'reliability' issue of the negotiated execution context of group $F$ to $F1$, indicating that: the document would perform the activity on its own, so no action from the user operating a device would be required, except of accepting the result by pressing the acceptance button.

$$\begin{array}{lcccc} & AcB & ASv & ACh & UdB \\ F1 = & 1 & 0 & 0 & 0 \end{array} \tag{5}$$

## 3 The negotiation process model

Negotiating parties exchange offers modelled as 5-tuples of items, which are values selected from the corresponding sets $A_i = \{a_{i_1}, a_{i_2}, \ldots, a_{i_n}\}$, $i = 1, \ldots, 5$. Each set $A_i$ represents the $i^{\text{th}}$ negotiated attribute and $i_n = |A_i|$. Operator $|\ |$ denotes cardinality of its argument set. We call set $A_T = \times_{i=1}^{5} A_i$ of all 5-tuples a *space of offers*. Based on that, offer $o \in A_T$ is defined as $o = \langle v_1, v_2, \ldots, v_5 \rangle$, where $v_i \in A_i$. Each single attribute value $v_i$ has an integer number assigned to reflect its utility to the negotiating party, calculated by function $u_i : A_i \to N$. Negotiating parties have their own sets of functions $\{u_1, \ldots, u_5\}$ to calculate utility of each respective item in the offer. Utility of each offer $o \in A_T$ is calculated by either party in a simple multi-criterial fashion, as weighted sum $U(o) = \sum_{i=1}^{5} w_i \cdot u_i(v_i)$, where $w_i$ are positive numbers such that $\sum_{i=1}^{5} w_i = 1$, and denote contribution of each respective item $v_i$ to the overall offer utility (Triantaphyllou, 2013). Neither $w_i$ nor $u_i$ are considered to be time dependent, so utility of each exchanged offer remains constant throughout negotiation and afterwards, until parties complete execution of the negotiated service (consume the agreed contract). Throughout the rest of the paper all items of a multi-issue offer are assumed to contribute equally to its respective utility, which we normalise against $\max_{o \in A_T}(U(o))$, so that $U : A_T \to (0, 1]$.

The left-open interval models the non-zero property of function $U$ to indicate that each offer is acceptable to negotiating parties. Formally, their ultimate objective is to find offer $o_c \in A_T$ that is acceptable to execution device $P_1$ and proactive document $P_2$, such that

$$o_c = \arg \max_{o \in A_T} U_1(o) U_2(o), \tag{6}$$

where $U_1$ and $U_2$ are their respective utility functions (Nash, 1950). Offer $o_c$ is called a *contract* between $P_1$ and $P_2$. One problem with solving equation (6) is that neither party knows its opponent's utility function, nor is willing to reveal its own. In consequence, parties have to exchange offers and counter-offers to systematically search space $A_T$.

Nash proposed a unique solution to equation (6), which multiplicative form represents the concern for equity – the product of the value gains is maximised more for more *equal* individual gains and each party is motivated by *proportionate cooperation* (MacCrimmon and Messick, 1976). We have implemented this proportionate cooperation principle in Subsection 3.1 with a *SBG*.

### 3.1 Simple bargaining game

Formally, devices and documents are players bargaining over finite set $C_B \subset A_T$, called a *bargaining set*. In each round, players $P_1$ and $P_2$ alternately make their moves by selecting offers from $C_B$. In each $k^{\text{th}}$ move player $P_i$ evaluates selected offer with its payoff function $\pi_i : A \times N \to (0, 1]$, calculated as $\pi_i(o, k+1) = \delta_i \cdot \pi_i(o, k)$, where $\delta_i < 1$ is a *discount factor*, and $\pi_i(o, 0) = U_i(o)$. Discounting models the cost of delay in submitting by $P_i$ an offer that could conclude the game in the next move. Let denote opponent of player $P_i$ by $P_{-i}$, their respective utility functions by $U_i$ and $U_{-i}$, payoff functions by $\pi_i$ and $\pi_{-i}$, and discount factors by $\delta_i$ and $\delta_{-i}$.

With both players motivated by the proportionate cooperation principle according to equation (6), rules of the SBG game are the following:

1 The game is started by player $P_1$.

2 Players do not reveal their $U_i$, $\delta_i$ and $\pi_i$, but share knowledge on $C_B$.

3 Utility values of players' offers are discounted upon transition to the next round.

4 Players exchange offers until the game is concluded by $P_i$, who:

   a quits the game by repeating one of its previous offers (negotiation failure), or

   b accepts one of $P_{-i}$'s offers received before as the contract (negotiation success).

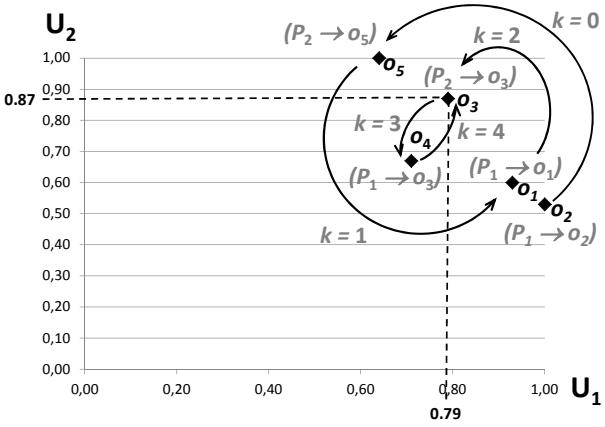
The rules of the SBG game constitute a mechanism designed to allow each party to adopt freely any algorithm for selecting offers in step 3, but at the same time to make them rational to play it in a socially beneficial way (Binmore, 1994). Since in general the required proportionate cooperation principle mentioned before is not self-enforcing, discounting introduced in step 2 has been intended by us as the incentive for players to explore new offers and to discourage them from unnecessarily prolonging the game by repeating the old ones, when searching of the bargaining set for the Nash solution.

Consider the following example. Let players $P_1$ and $P_2$ bargain over $C_B = \{o_1, o_2, o_3, o_4, o_5\}$, with the respective values of their utility functions specified by formula (7).

$$\begin{array}{lccccc} & o_1 & o_2 & o_3 & o_4 & o_5 \\ U_1: & 0.93 & 1.00 & 0.79 & 0.71 & 0.64 \\ U_2: & 0.60 & 0.53 & 0.87 & 0.67 & 1.00 \end{array} \tag{7}$$

By playing the SBG game players would search the example $C_B$ set for the solution of equation (6) as shown in Figure 1.

**Figure 1** Searching $C_B$ for the Nash solution




Player $P_1$ starts the game in move $k$ = 0 by submitting its most valued offer $o_2$. Instead of accepting it, player $P_2$ attempts $o_5$ in the next move $k$ = 1, as utility of $o_2$ is of the least value to it. Clearly, in this round neither player could reasonably accept its opponent's offer. The next attempt by $P_1$, now in move $k$ = 2, is offer $o_1$. Since $P_2$ values it less than $o_3$, in move $k$ = 3 it submits offer $o_3$ to $P_1$ as the more preferred one. Now $P_1$ has to make a decision what to do in move $k$ = 4: withdraw from the negotiation, submit $o_4$ as its counter offer, or accept any previous offer of its opponent. It may be seen in Figure 1 that a withdrawal would not be reasonable, as at least one of the offers $o_5$ and $o_3$ (submitted by $P_2$ before) is better than $o_4$ (the last one not yet submitted by any player). But submitting $o_4$ would make $P_1$ loosing more than when accepting $o_3$. On the other hand, its previous offers $o_2$ and $o_1$ were rejected by $P_2$, so there would be no point to insist on accepting them by $P_2$. Finally, neither accepting $o_5$, which is of lesser utility to $P_1$ than offers $o_4$ (the only one left not submitted yet) and $o_3$ (that $P_2$ offered in move $k$ = 3), nor submitting $o_4$ as a counter offer to $o_3$ valued higher, would be reasonable to $P_1$. Therefore $o_3$ seems to be the best option to $P_1$, and could be accepted in move $k$ = 4. Indeed, given the values listed in formula 7, the value of $U_1(o_3) \cdot U_2(o_3) = 0.87 \cdot 0.79 = 0.687$ is the maximum one. Note, that depending on the actual values of discount factors used by each player the contract may be agreed faster – for lower values of $\delta_i$ player $P_i$ would be more keen to accept counter offers submitted by its opponent $P_{-i}$. So with $\delta_1$ and $\delta_2$ values close to 1 players $P_1$ and $P_2$ would be more patient to explore the solution space and would finally agree on $o_c = o_3$ in move $k$ = 4 (as shown in Figure 1), whereas with $\delta_1$ and $\delta_2$ close

to 0 players would be more desperate to consider more the first ever offers submitted, in the 'take it or exit' manner, and less interested in exploring deeper the solution space. In the experiments reported in Section 5 we have assumed players to be moderately patient, with their discount factors $\delta_1 = \delta_2 = 0,8$.

The game is implemented by Algorithm 1. When exchanging offers both parties count their moves in variable $k$ and share bargaining set $C_B$. Player $P_i$ collects all offers received so far from $P_{-i}$ in its working set $C_{Ri}$ of received offers, and respectively, all offers submitted to $P_{-i}$ in its working set $C_{Si}$. Working sets of each player are initially empty and counter $k$ of moves is set to 0.

**Algorithm 1** A generic SBG algorithm

**Input:** bargaining set ($C_B$).
**Output:** agreed contract ($o_c$).
*Initialisation of working sets:*
1: $C_{R1} \leftarrow \emptyset$; $C_{S1} \leftarrow \emptyset$; $C_{R2} \leftarrow \emptyset$; $C_{S2} \leftarrow \emptyset$;
2: $k \leftarrow 0$; $o_2 \leftarrow null$;
3: **while** $o_1 \neq o_2$ **do**
*Device's move:*
4: $o_1 \leftarrow$ submitOffer($C_B$,$C_{R1}$,$C_{S1}$,$o_2$,$k$); $k \leftarrow k+1$;
*Document's move:*
5: $o_2 \leftarrow$ submitOffer($C_B$,$C_{R2}$,$C_{S2}$,$o_1$,$k$); $k \leftarrow k+1$;
6: **if** $o_1 = o_2$ **then** $\{o_c \leftarrow o_1$; **break**;$\}$;
7: **end while**
*Contract agreed:*
8: **return** $o_c$;

It is interesting to see how good the solution returned by Algorithm 1 could be and how well it implements the proportionate cooperation principle mentioned before. In other words we expect that upon concluding the game both parties will simultaneously attain their possibly highest individual gains. Note that the discounting mechanism incorporated in the game makes each move precious to either player $P_i$, since payoffs for any submitted offer in each successive round decrease geometrically of a common fractional ratio (discounting factor) $\delta_i$. Because of that a rational player would not risk submitting offers of less utility before submitting offers of higher utility, for otherwise offers of higher utility could be wasted. In consequence of that each player would like to submit offers in the order of its own preference hoping that any of the offers submitted so far could be accepted by the opponent as soon as possible. It also assumes that its opponent is equally rational and knows that its opponent would submit offers in the same manner. The following theorem holds:

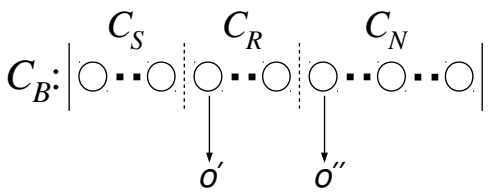
*Theorem 1:* Given the bargaining set $C_B$ of offers shared by players, the SBG game terminates with the solution (contract offer) $o_c \in C_B$ such that payoffs for each player attain simultaneously their highest possible values.

*Proof:* Since $U : A_T \rightarrow (0,1]$ each offer in $C_B$ can be accepted by both parties, but either player would like to know the order in which its opponent would prefer them. When selecting offers from $C_B$ in the order of its individual preferences each player $P_i$ collects gradually knowledge on the preferences of its opponent $P_{-i}$ round by round in its respective $C_{Ri}$ set of offers received so far. Similarly, in each round each player $P_i$ knows that its opponent knows preferences of offers submitted by it so far in its respective $C_{Si}$ set. However, both players have no knowledge on each other preferences of offers in the set $C_N = C_B - C_{Si} \cup C_{Ri}$, i.e., offers that have not been submitted yet by any of them. In the first round the offer made by the player starting the game may be accepted right away by its opponent. It will be the case when the first offer ever made by the starting player happened to be the best offer for both parties. Otherwise $P_{-i}$ would submit new offer $o_n \in C_N$. Each time $P_{-i}$ does that $P_i$ knows that utility of such a new offer is of lesser value to $P_{-i}$ than any offer it received so far in $C_{Ri}$. Its utility is also of lesser value to any offer in $C_{Si}$, for otherwise it would have been submitted by $P_i$ before. When exchange of offers comes to the point when all remaining offers in $C_N$ have utility of lesser value to $P_i$ than any new offer $o_n$ that may yet be received it knows that nothing better could be submitted by $P_{-i}$ and the best possible offer in $C_R$ should be accepted. If however, any offer $o'_n \in C_N$ is still better than the last recently received counteroffer $o_n$ player $P_i$ submits $o'_n$ hoping that the opponent would eventually accept it, if not any of its previously submitted offers in $C_{Si}$. The exchange of offers continues until either party is not able to find in $C_N$ any offer better than any of the received so far. In that case the first party who discovers that concludes the game by selecting offer $o_c \in C_{Ri}$ received so far from its opponent that is of the highest utility value to $P_i$. When the concluding player $P_i$ finally chooses $o_c$ it will be of the highest possible utility value it could attain in this game, i.e., such that maximises its individual gain but can simultaneously be accepted by the opponent. On the other hand there is no offer in $C_{Si}$ that would be of higher utility value to $P_{-i}$, for otherwise it would mean that $P_{-i}$ selected its last new counteroffer from $C_N$ despite of having already a better offer to be accepted in its $C_{R_{-i}}$ set. But as assumed before both players are rational and submit offers in the order of their individual preferences, so it would not be possible and $o_c$ is of the highest possible utility value to attain simultaneously by the player and its opponent. □

### *3.2 Best offer selection*

Based on the above proof it may be seen that when considering its next move, player $P_i$ analyzes current content of its respective working sets and makes a decision whether to accept any of the received offers in $C_R$ or to submit a new one from $C_N$. A straightforward approach for doing that by $P_i$ would be to compare in each move $k$ payoff of the most valued offer from $C_R$ (that would conclude the game) with a discounted payoff of the most valued counteroffer in $C_N$ (that would continue it). This concept is outlined schematically in Figure 2, where offers in $C_R$ and $C_N$ are sorted in a descending order of their utility values, and implemented formally by Algorithm 2.

**Figure 2** Naive best offer selection




**Algorithm 2** `submitOffer()` – naive selection

**Input:** bargaining set ($C_B$), past received ($C_R$) and submitted ($C_S$) offers, incoming offer ($o_r$), player's discount factor ($\delta_i$), move number ($k$).
**Output:** offer to be submitted in return ($o_s$).

1: $C_R \leftarrow C_R \cup \{o_r\}$;
2: $C_N \leftarrow C_B - (C_R \cup C_S)$;
3: $o' \leftarrow \arg\max_{o \in C_R} \pi_i(o,k)$;
4: $o'' \leftarrow \arg\max_{o \in C_N} \pi_i(o,k)$;
5: **if** $\pi_i(o',k) > \delta_i \cdot \pi_i(o'',k)$ **then** $o_s \leftarrow o'$ **else** $o_s \leftarrow o''$;
6: $C_S \leftarrow C_S \cup \{o_s\}$;
7: **return** $o_s$;

This implementation is considered *naive*, since whenever a player decides to submit a new offer it does not care whether one could be accepted by its opponent in the next move. For bargaining sets with a small number of offers and reasonable high discount factors the contract may be then agreed in a relatively small number of rounds, even when players would have to be searching for the contract in $C_B$ until the very last element. With larger sets, however, a naive approach may lead to an excessive number of rounds, up to the $\lceil \frac{|C_B|}{2} \rceil$ maximum. Therefore a more clever player may want to consider first which would be a future counteroffer it may receive from its opponent in response to the currently considered one. It would imply a recursive implementation of the *submitOffer()* procedure.

Let us define strategy function $s_i : N \rightarrow C_B$ to associate moves of player $P_i$ with the relevant offers from the bargaining set. A decision made by player $P_i$ in move $k$, whether to accept one of the received offers $o'_i \in C_{Ri}$, or continue the game by selecting new offer $o''_i \in C_N$, would be $s_i(k) = \arg\max_{o \in \{o'_i, o''_i\}}(\pi_i(o'_i,k), \delta_i \cdot \pi_i(o''_i,k))$, where $P_i$'s candidate offer to conclude the game $o'_i = \arg\max_{o \in C_{Ri}} \pi_i(o,k)$, and to continue the game $o''_i = \arg_{o \in C_N}[\pi_i(o,k) \cong \max(1-\delta_{-i}) \cdot \pi_i(o,k) + \delta_{-i} \cdot \pi_i(s_{-i}(k+1),k)]$. Offer $o''_i$ is chosen from $C_N$ by $P_i$ as the one providing a payoff value closest to the weighted sum of payoffs, what is denoted by $\cong$. The first component of this sum represents a payoff portion related to the 'concluding' offer that could eventually be accepted by the opponent, whereas the second component represents a payoff portion of the 'continuation' counteroffer that would be submitted in return by the opponent if the 'concluding' offer could not be accepted. The rationale for choosing the respective weights $1-\delta_{-i}$ and $\delta_{-i}$ is that with $\delta_{-i}$ closer to 1 the opponent would prefer to continue the game with a new counteroffer submitted by it in the next move, for otherwise with $\delta_{-i}$ closer to 0 the opponent may be more willing to accept an offer submitted by $P_i$ in the current move. For that reason, $1-\delta_{-i}$ and $\delta_{-i}$ used in the expression for calculating $o''_i$ are called respectively *conclusion* and *continuation* factors of $P_i$.

A recursive part of the formula for $o''_i$ requires $P_i$ to calculate $s_{-i}(k+1)$ in order to assess which counteroffer $P_{-i}$ would eventually submit in move $k+1$. Besides knowing $\delta_{-i}$ by $P_i$ (which for a time being could be assumed $\delta_{-i} \approx \delta_i$), assessment of the prospective counteroffer would require $P_i$ to know also its opponent's utility function $U_{-i}$. Player $P_i$ may cope with it by assuming that its opponent evaluates offers in the respective $C_{Si}$ and $C_N$ subsets within the same (0, 1] range, and utility values it calculates for offers in each respective subset are distributed evenly. Then payoff of a counteroffer selected by $P_{-i}$ may be approximated by $P_i$ as the average of payoffs the latter would get with its $\pi_i$ function. It is denoted formally as $\bar{\pi}(C,k) = 1/|C| \sum_{o \in C} \pi_i(o,k)$, where $C \subset C_B$. Therefore a decision made by player $P_{-i}$ in move $k$ may be modelled by $P_i$ as $s_{-i}(k) = \arg_{o \in C_B - C_{Ri}}[\pi_i(o,k) \cong 0.5 \cdot (\pi_i(o'_{-i},k) + \delta_{-i} \cdot \pi_i(o''_{-i},k))]$. Offer $o'_{-i} \in C_{Si}$ is in $P_i$'s view the one that $P_{-i}$ would consider acceptable to $P_i$ and concluding the game; $P_i$ models it as $o'_{-i} = \arg_{o \in C_{Si}}[\pi_i(o,k) \cong \bar{\pi}(C_{Si},k)]$. Offer $o''_{-i} \in C_N$ is in $P_i$'s view a new offer that $P_{-i}$ may choose to continue the game; $P_i$ models it as $o''_{-i} = \arg_{o \in C_N}[\pi_i(o,k) \cong (1-\delta_i) \cdot \bar{\pi}(C_N,k) + \delta_i \cdot \pi_i(s_i(k+1),k))]$. The respective conclusion and continuation factors for $P_{-i}$ are $1-\delta_i$ and $\delta_i$. Recursive calculation of offers $o''_i$ and $o''_{-i}$ that imply continuing the game will eventually terminate in the last possible move $k_{max} = |C_B| + 1$, when $C_N$ gets empty. If $P_i$ concludes the game it selects $s_i(k_{max}) = \arg\max_{o \in C_{Ri}} \pi_i(o,k_{max})$, otherwise $P_{-i}$ concludes the game by selecting $s_{-i}(k_{max}) = \arg_{o \in C_{Si}}[\pi_i(o,k_{max}) = \bar{\pi}(C_{Si},k_{max})]$.

Continuation counteroffers $o''_i$ and $o''_{-i}$ are calculated, respectively for $P_i$ and $P_{-i}$, in a recursive manner. Instead of two formulas calling one another recursively, and using discount factors $\delta_i$ and $\delta_{-i}$ alternately, just one unified formula may be provided. Consider $P_i$ submitting offer $o''_i$ in the last move $k = k_{max}$ of the game, when $o''_i$ will be immediately accepted by $P_{-i}$. According to the formula for calculating $o''_i$ proposed before, the respective conclusion factor of $P_i$ will be $1-\delta_{-i}$. In order of that to happen conclusion factor of $P_{-i}$ in the penultimate move of the game $k = k_{max} - 1$ would be $1 - \delta_i \cdot (1-\delta_{-i})$, in an intermediate move $k = k_{max} - 2$ of $P_i$ would be $1 - \delta_{-i} \cdot (1 - \delta_i \cdot (1-\delta_{-i}))$, and so on. That pattern may be described with function $\lambda(k,\delta_i,\delta_{-i})$, called a *cumulative conclusion factor* and calculated as:

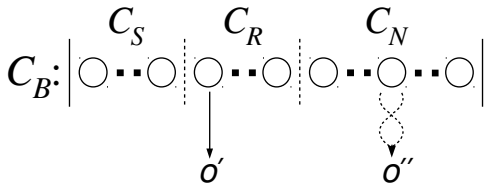
$$\lambda(k,\delta_i,\delta_{-i}) = \begin{cases} (1-\delta_{-i}) & k = k_{max} \\ (1-\delta_{-i} \cdot \lambda(k+1,\delta_{-i},\delta_i)) & 0 \le k < k_{max} \end{cases} \quad (8)$$

Factor $\lambda(k,\delta_i,\delta_{-i})$ measures belief of $P_i$ that its new offer selected from $C_N$ in move $k$ would be accepted by $P_{-i}$ and concluding the game. With that in mind consider two hypothetical offers in $C_N$:

one $\tilde{o}' = arg_{o \in C_N} max(1 - \delta_{-i}) \cdot \pi_i(o, k)$, assessed by $P_i$ as concluding the game, and respectively $\tilde{o}'' = arg_{o \in C_N - \{\tilde{o}'\}}[\pi_i(o) \cong \bar{\pi}(C_N - \{\tilde{o}'\}, k)]$, as continuing the game, i.e., a counteroffer of $P_{-i}$ when rejecting $\tilde{o}'$. By substituting $\tilde{o}'$ and $\tilde{o}''$ in the respective parts of the weighted sum for calculating $o_i''$ specified before, one gets $o_i'' = \text{arg}_{o \in C_N}[\pi_i(o, k) \cong \lambda(k, \delta_i, \delta_{-i}) \cdot \pi_i(\tilde{o}', k) + (1 - \lambda(k, \delta_i, \delta_{-i})) \cdot \pi_i(\tilde{o}'', k)]$. By assuming $\delta_{-i} \approx \delta_i$, a recursive calculation of $\lambda$ may be further approximated by iterative form $\lambda(k, \delta_i, \delta_i) = 1 + \sum_{n=1}^{k_{max}-k+1} (-\delta_i)^n$. The closer values of $\delta_{-i}$ and $\delta_i$ are in reality, the more accurate estimation of the concluding offer could be.

The concept of an estimated selection of a concluding offer is outlined schematically in Figure 3 and implemented formally by Algorithm 3.

**Figure 3** Estimated best offer selection




**Algorithm 3** `submitOffer()` – estimated selection

**Input:** bargaining set ($C_B$), past received ($C_R$) and submitted ($C_S$) offers, incoming offer ($o_r$), players' discount factor ($\delta_i$), move number ($k$).
**Output:** offer to be submitted in return ($o_s$).

$C_R \leftarrow C_R \cup \{o_r\}$;
$C_N \leftarrow C_B - (C_R \cup C_S)$;
$o' \leftarrow \arg\max_{o \in C_R} \pi_i(o, k)$;
$\tilde{o}' = arg_{o \in C_N} max(1 - \delta_{-i}) \cdot \pi_i(o, k)$;
$\tilde{o}'' = arg_{o \in C_N - \{\tilde{o}'\}}[\pi_i(o) \cong \bar{\pi}(C_N - \{\tilde{o}'\}, k)]$;
$\lambda \leftarrow 1 + \sum_{n=1}^{k_{max}-k+1} (-\delta_i)^n$;
$o_i'' = \text{arg}_{o \in C_N}[\pi_i(o, k) \cong \lambda \cdot \pi_i(\tilde{o}', k) + (1 - \lambda) \cdot \pi_i(\tilde{o}'', k)]$;
**if** $\pi_i(o', k) > \delta_i \cdot \pi_i(o'', k)$ **then** $o_s \leftarrow o'$ **else** $o_s \leftarrow o''$;
$C_S \leftarrow C_S \cup \{o_s\}$;
**return** $o_s$;

By eliminating recursion from the player's decision in each phase which new offer to select from $C_N$ to be accepted by its opponent in the next phase, we get the less computationally demanding implementation of the *submitOffer()* procedure than its recursive counterpart, and which still provides a considerably better chance for players to find a contract before exploring the entire bargaining set $C_B$. The estimated version of *submitOffer()* was used in our experiments reported further in the paper.

## 4 Repeated Encounters

To this end only single encounters of a proactive document-agent with execution devices have been assumed, i.e., when it has no knowledge (or does not care) about its past encounters with devices of specific classes. Recall from Section 3 that upon concluding negotiation successfully after playing the game using Algorithm 3, along with the agreed contract each agent may collect two other pieces of information: the actual bargaining set shared with its opponent, and the sequence of offers submitted by the latter. When brought back by each document agent to the agency (a sever where they rest idle before migrating again to implement another workflow), this knowledge could be collected and used to train them in reaching agreements in a yet smaller number of rounds; in other words, it will make them more knowledgeable before the next encounter with a device of the class they could recognise. Note that training data needed for that could be accumulated quite fast, given that agents may have to negotiate execution contexts for dozens of activities in a single workflow process. So with several workflow processes originated by the agency (even with just a few document-agents migrating in the system), it yields a collection of hundreds of bargaining sets and their related negotiation histories.

The question we would like to address now is whether repeated (multiple) encounters of a document agent with execution devices of some class could aid the former in recognising that class, and predicting next – for each properly recognised device class – a feasible solution of the game. By 'feasible' we mean a solution that would be negotiated anyway by using Algorithm 3, but when knowledge on past encounters is used – in a significantly shorter series of moves.

Formally, we have distinguished two classification subproblems, i.e., one may want a document to be trained:

1 to recognise class $x_i$ of its opposing device, out of the possible set of classes $\{x_1, x_2, ..., x_n\}$

2 to discover precedence of offers in the bargaining set that a device of the recognised class would most likely prefer.

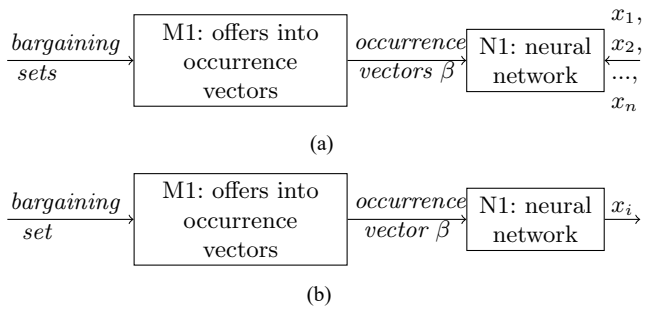
A rationale behind this distinction is the following. Firstly, combinations of specific options and their values, which are present in offers in bargaining sets of devices from the same class, could implicitly identify that class. Secondly, preferences of devices belonging to the same class may indirectly affect partial ordering of counteroffers submitted to the document by that device during negotiation. As demonstrated further in Subsection 5.1 four generic device classes such as workstations, laptops, tablets and smartphones could differ in subtle ways from one another, both in terms of certain option values present or missing for selected attributes, as well as their order of preference. In other words, although each single device may have its bargaining set and preferences defined individually by its user, bargaining sets and preferences of all devices belonging to the same class will have some common features enabling its distinction from other classes.

We have decided to classify these features with neural networks, for their ability to generalise, enabling handling of unseen data, and their relatively simple implementation.

### 4.1 Classification of device types

A solution to the first classification problem is outlined in Figure 4. Occurrence vectors $\beta$ collect flags representing content of each respective multi-issue offer in $C_B$, as defined in Section 2; They are constructed by module M1 as bit-words $\beta = \omega_1\omega_2 \ldots \omega_5$, with each bit field $\omega_i$ of length $|A_i|$ indicating which value option of the respective attribute $A_i$ occurs in any offer of $C_B$. Thus generated bit-words provide input to neural network N1. In the training phase [Figure 4(a)], a collection of bargaining sets negotiated by a document during its past encounters, is converted by M1 to occurrence vectors, which are associated next with their proper device class labels $x_i$ by network N1 to train it. In the classification phase [Figure 4(b)], a negotiated bargaining set is converted by M1 into the occurrence vector and used next to classify the respective device by the already trained network N1.

**Figure 4** Device class recognition modules, (a) training (b) classification

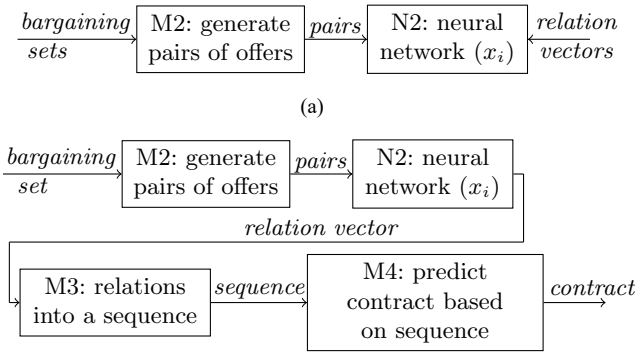



### 4.2 Classification of device preferences

A solution to the second classification problem is outlined in Figure 5. Upon recognising device class $x_i$, a document is ready to recognise preferences of offers in the bargaining set shared with the actual device of that class. We base this analysis on recognising binary relations between each pair of offers in a sequence of offers which devices of the class known to be $x_i$ had submitted in the past. Each device class $x_i$ requires a separate neural network N2 for that – of the same number of layers and neurones, but with different weights.

In the training phase [Figure 5(a)] module M2 generates all possible pairs of offers in $C_B$ to train network N2 to recognise their precedence relations. A relation vector consists of '1' or '0' labels, each one indicating whether for each respective pair of offers $(o_i, o_j)$ they preceded or succeeded one another in sequences recorded during past encounters with a device of class $x_i$. In the classification phase [Figure 5(b)] module M2 generates all possible pairs of offers for the currently negotiated $C_B$ and provides them to the already trained network N2 to extract their precedence relations. Floats returned by the network are discretised with a threshold set to 0.5 to distinguish '1' from '0' labels marking respectively the recognised precedence relation. Next, module M3 reconstructs a sequence of offers based on the extracted relations; the sequence indicates in what order the opposing device would like to select offers from the bargaining set. This knowledge is utilised by module M4 to assess a possible concluding offer in Algorithm 4, as explained in the next subsection.

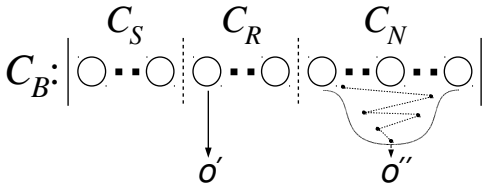
**Figure 5** Device preferences recognition modules, (a) training (b) classification




### 4.3 Intelligent offer selection

Based on the content of a bargaining set, a document equipped with functional modules outlined in Figure 4(b) and Figure 5(b), should be able to recognise a device class before starting the actual negotiation, as well as predict preferred ordering of all offers a device may want to submit. This knowledge is utilised by a document to make a decision, in each respective round of the negotiation, which offer not submitted yet would be the most promising one to conclude negotiation with a device in the next round. Recall that in Algorithm 3 this decision was based on estimation how much the opposing device may be willing to concede by accepting a new offer submitted by a document, based on the belief of the latter – modelled with the $\lambda$ factor defined by formula (8). Instead of that, Algorithm 4 takes advantage of knowing the preferred sequence of offers reconstructed by module M3 to imitate continuation of the game using two sequences made of offers that remained in $C_N$: the opponent's one reconstructed by $P_i$, and its own, sorted according to its utility function $U_i$; this concept is outlined schematically in Figure 6 and implemented formally by Algorithm 4.

**Figure 6** Intelligent best offer selection




Imitation of the game implemented by the 'repeat-until' loop in lines 5–11 assumes conclusion factor $\lambda \approx 1$ in

each move $n$ and fine stepping through offers. With lower values of $\lambda$, reflecting less confidence of $P_i$ that offer $\widetilde{o}_i$ would be concluding, slightly coarser stepping might be considered, e.g., by assuming $\Delta(\lambda, C_N) = \lceil (1-\lambda)(|C_N|-1) \rceil$ and modifying line 10 of Algorithm 4 to $n \leftarrow n + 1 + \Delta(\lambda, C_N)$. With that modification, the closer conclusion factor $\lambda$ gets to 0 in each iteration of the loop, the more candidate offers from $C_N$ would be skipped by $P_i$ when imitating the game. It however does not preclude the skipped offers from being selected in a next negotiation round. In the experiments, reported further in the paper fine stepping ($\Delta = 0$) through the imitation loop of Algorithm 4 was assumed.

**Algorithm 4** `submitOffer()` – intelligent selection

**Input:** bargaining set ($C_B$), reconstructed sequence of opponent's offers ($\widetilde{C}_{-i}$), past received ($C_R$) and submitted ($C_S$) offers, incoming offer ($o_r$), players' discount factor ($\delta_i$), move number ($k$).
**Output:** offer to be submitted in return ($o_s$).
1: $C_R \leftarrow C_R \cup \{o_r\}$;
2: $C_N \leftarrow C_B - (C_R \cup C_S)$;
3: $\widetilde{C}_i \leftarrow sort(C_N, U_i)$;
4: $o' \leftarrow \arg\max_{o \in C_R} \pi_i(o, k)$;
5: **repeat**
*Imitate the game using player's and opponent's sequences of offers [module M4 in Figure 5(b)]:*
6: $\widetilde{C}_{Si} \leftarrow \emptyset$; $\widetilde{C}_{Ri} \leftarrow \emptyset$; $n \leftarrow 0$;
7: $\widetilde{o}_{-i} \leftarrow \widetilde{C}_{-i}[n]$;
8: $\widetilde{o}_i \leftarrow \widetilde{C}_i[n]$;
9: $\widetilde{C}_{Ri} \leftarrow \widetilde{C}_{Ri} \cup \{\widetilde{o}_{-i}\}$; $\widetilde{C}_{Si} \leftarrow \widetilde{C}_{Si} \cup \{\widetilde{o}_i\}$;
10: $n \leftarrow n + 1$;
11: **until** $\widetilde{o}_i \in \widetilde{C}_{Ri}$ || $\widetilde{o}_{-i} \in \widetilde{C}_{Si}$
12: $o'' \leftarrow \widetilde{o}_i$;
13: $\widetilde{C}_{-i} \leftarrow \widetilde{C}_{-i} / \{o_r, o''\}$
14: **if** $\pi_i(o', k) > \delta_i \cdot \pi_i(o'', k)$ **then** $o_s \leftarrow o'$ **else** $o_s \leftarrow o''$;
15: $C_S \leftarrow C_S \cup \{o_s\}$;
16: **return** $o_s$;

## 5 Experiments

As mentioned at the beginning of Section 4, training of document-agents to recognise device classes and predict feasible solutions of the current encounter, given the properly recognised class of the actual device and the bargaining set it would like to share with the document, could be based on data which represent recorded histories of games. Training data may come from two sources: document agents collecting *ex post* historical data on their activities during repeatedly performed workflow processes in a virtual organisation they belong to (when all agents negotiated with their opponents using only Algorithm 3), or *ex ante* generation of training data based on some predefined set of generic device classes and their proximate preference characteristics (to enable them to negotiate using Algorithm 4).

In our original research we used the second approach, and considered both stationary computers (workstations) and mobile devices including laptops, tablets, smartphones and cellphones, working in two modes (connected and not connected to the network), i.e., ten generic classes in total (Kaczorek, 2015). Owing to that we were able to assess how effective SBG could be in resolving conflicts between proactive documents and devices during repeated encounters compared to single ones, and whether acceptable payoff levels could be observed, well before actually implementing workflow processes and virtual organisations for concrete target domains. We were also able to assess how well, and at what cost, proactive document agents may be trained to take advantage of Algorithm 4. In the experiments reported further in this section the neural network toolbox of MATLAB 2012b was used to experiment with alternative network architectures. All algorithms presented before were also implemented in MATLAB, whereas training and test data generation procedures were implemented in C++.

For brevity we present below results of our experiments with just four classes of devices (workstations, laptops, tablets and smartphones, by default always connected to the network), which were representative enough to evaluate Algorithms 3 and 4.

### 5.1 Preference of offers

As argued before, the content of a multi-issue offer implies a concrete execution context that the receiving device could provide to the arriving proactive document. Depending on its purpose and design, each device would exhibit various combinations of option flags for issues considered in Section 2, as well as would have various preferences of offers embedding these combinations.

Workstations (class $x_1$ of execution devices) provide potentially the most powerful execution context for proactive documents. For that reason some preference may be given to documents with embedded functionality supporting advanced interaction with workers and services, i.e., with options from the $J$ and $D$ groups, less often with options from the $W$ group. Because of their immobility, workstations may either be located in the worker's office or at home. When in office, they are usually wired to the organisation's intranet (groups $I$ and $N$ of options), or less often, connected directly to other networks (groups $E$ and $N$ of options). When located at home, they may still be wired to the collaborator's organisation intranet through a VPN tunnel (groups $I$ and $N$ of options), otherwise they may use modem connections (options of the $I$ or $E$ group combined with options of the $A$ or $M$ group). Security would often involve options from the $T$ group, rarely options form the $P$ group. Finally, reliability options may be from any $H$, $F$, $B$, or $L$ group, depending on the particular software used to handle active documents.

Laptops (class $x_2$ of execution devices) are not less powerful than stationary workstations, and because of their improved mobility are more flexible, owing to the 'performance' attribute ($A_3$) options related to the WiFi capability; in fact their preference rules are the

same as the workstation ones, extended by rules involving options from the $R$ group, but with preference given to options from the $N$ group before the $R$ group. Another difference is that laptops are in general more 'personal' than workstations, so their users (owners) may want to have more control on what proactive document-agents can do when performing their activities. Therefore laptops may more prefer options of the $W$ group before options of the $J$ and $D$ groups.

Tablets (class $x_3$ of execution devices) lack the most classic components of a personal computer, namely keyboard and mouse, but due to the advanced touch-up screen technology, they can easily compensate for that with other interaction paradigms. They also can, due to their sufficient screen width, emulate a keyboard of a size compared to a small laptop. In general they are very 'personal' devices, so their software systems may be customised by users to the point that some local tools or services commonly installed on workstations or laptops may be unavailable to the arriving proactive document-agent. Dominant contexts with regard to the 'performer' attribute would than have options of the $W$ group, giving the worker full control over a document content. If, however, proactive documents could reduce their expectations to the less demanding set of services, or can complete a simpler activity on their own, options of the $D$ and $J$ groups would probably be accepted by the tablet device. Because of their highly customised local systems, tablets can provide most often only substitutes of specific tools required by the document, therefore a smaller subset of combinations of option flags from the $I$ and $E$ groups (compared to their workstation and laptop counterparts) should be considered. Another issue may be the available networking hardware; for tablets we have assumed in the experiments only the WiFi and 4G/LTE connectivity, i.e., combinations of option flags from the $R$ and $A$ groups.

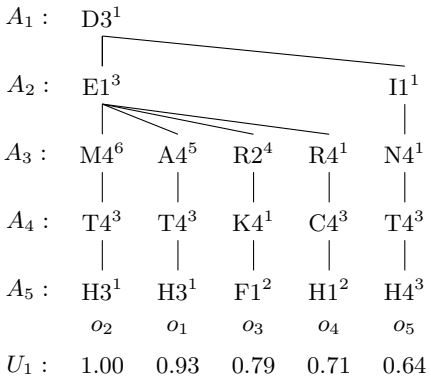
Contexts provided by smartphones (class $x_4$ of execution devices) differ significantly when compared to their tablet counterparts. Their touch-up screens can still enable interaction almost to the same extent as the former (including keyboard emulation), but due to their yet more customised functionality, many tools and services may in practice never be provided to the worker or to the document-agent when performing a specific activity. Therefore even less combinations of option flags from the $I$ and $E$ groups are possible compared to tablets, whereas preference of options related to the 'performer' attribute could be the same as for tablets, i.e., combinations of option flags from the $W$ group would be more preferred than the ones from the $D$ and $J$ groups. For the same reason options related to interaction reliability would realistically be limited to combinations of option flags from the $L$ group only. Networking hardware of smartphone devices enable access to wireless and cellular networks alike, however preference shall be given to the ones from the $R$ group before the $M$ group, due to the relatively higher cost of using cellular networks.

### 5.2 Option trees

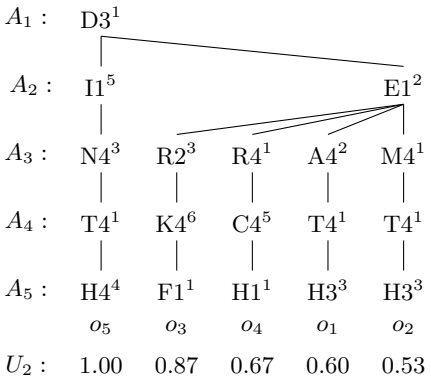
Preference rules of each class of agents have been formally specified by us as *policies* (Kaczorek, 2015). However, each agent may implement the policy of its class individually, with specific combinations of option flags for each issue discussed in Section 2. In consequence, each party may view the shared bargaining set $C_B$ as a differently sorted set of offers. Each particular view can be represented with an *option tree*, which constitutes specific implementation of the policy. As explained further in Subsection 5.3, we used the technique of implementing policies with option trees to generate training and test datasets for modules shown in Figures 4 and 5.

Document and option trees for the bargaining set $C_B = \{o_1, o_2, o_3, o_4, o_5\}$ negotiated in the example shown in Figure 1 are shown in Figure 7. Nodes of the option tree are labelled with option values defined before, while utility of each single attribute value for the given party is indicated by the integer superscript.

**Figure 7** Option trees of negotiating partners, (a) device options (b) document options




(a)




(b)

### 5.3 Negotiation effectiveness

Two network architectures `N1` and `N2` were tested to determine how many layers and neurones would suffice, and would not consume too much memory of a proactive

document agent with all modules shown in Figure 4(b) and Figure 5(b) embedded in its code. For each device class, a significant number of random bargaining sets was created, each one with multi-issue offers generated by a procedure specially implemented for that; the uniform distribution of attribute values was assumed, so each single bargaining set could be considered a statistically valid representative of all bargaining sets of the given execution device class. A resource of 2,000 such datasets was created in total. Another generation procedure combined possible options for each respective device class policy and generated a rich set of their implementations (option trees). Each negotiation experiment involved both, estimated (Algorithm 3) and intelligent (Algorithm 4) selection of offers and the total number of exercises reached 1,600.

We compared payoffs reached by proactive documents that did not take advantage of machine learning with those that did. They were also compared with payoff values that could be calculated directly from formula (6), when $U_1$ and $U_2$ were known beforehand – considered an ideal case and called a *fair* payoff; it represented a situation, when parties could reach agreement in the first negotiation round.

Moreover, four conceptual types of proactive document agents with reasonably diversified preference rules were introduced. The rationale behind that was to check if the lack of specific options in the bargaining set may affect a number of negotiation rounds. In that regard *protected* documents have been distinguished from *open* ones – by taking into account whether they need any secure connection to perform their activity or not, and *heavy* documents from *light* ones – by considering whether they require CPU power and the amount of RAM above or below some average levels. In consequence, a protected document would not consider combinations of option flags of the $P$ group for the 'security' issue, whereas a heavy document would consider only a few selected combinations of option flags of the $U$, $R$, $M$, $A$ and $N$ groups for the 'performance' issue.

Results of experiments, when negotiations between documents of four different types and execution devices of four different classes were simulated are summarised in Figure 8(a). Each party used Algorithm 3 for estimated selection of offers in each round. It may be seen that payoff levels for each respective pair of players are strongly diversified, most notably for laptop devices. This is because offers negotiated by laptops involved a much richer set of options than tablets or smartphones.

Results obtained for untrained documents have been contrasted next to payoff levels that were got by trained documents. In this series of simulation experiments trained documents used Algorithm 4 for intelligent selection of offers in each round, whereas devices used estimated selection as before. Payoff levels obtained by just one out of all 16 tested pairs (protected and heavy type trained documents negotiating with laptops) are illustrated in Figure 8(b). It also illustrates how quality of training documents to classify devices can affect their capability to reach agreements in a lesser number of rounds. A *fitness level* value indicates the ratio of devices of a given class correctly recognised by a document. So, the 0% fitness level in Figure 8(b) indicates a document that is not trained at all to recognise devices [like all documents shown in Figure 8(a)], whereas respectively the 100% level indicates that all laptop devices were correctly recognised by documents during all tests. It may be seen that payoffs got by even incompletely trained documents were significantly better than the ones got by documents not trained at all.

**Figure 8** Payoff levels for selected device and document classes, (a) document not trained (b) documents trained

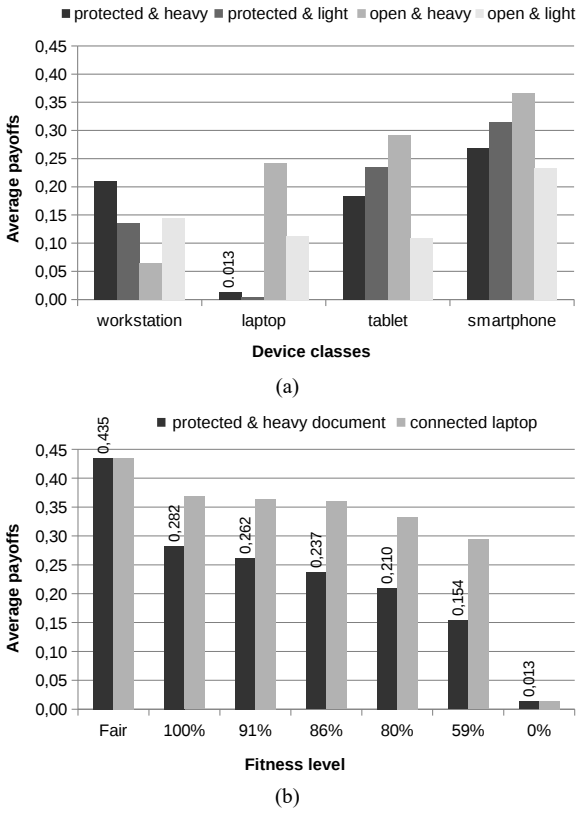



Figure 9 illustrates results of negotiation experiments with documents of all four types trained adequately, i.e., each with the 100% fitness level, to recognise the relevant device classes and their sequences of offers. In that series of experiments bargaining sets of a moderate size were used, for which negotiation involved 20–25 rounds, and option trees of the respective negotiating parties were sorted opposite to one another to possibly maximise a number of rounds in each exercise. Documents implemented alternately Algorithms 3 and 4, whereas devices used only Algorithm 3. For each device class, documents that could recognise sequences of offers returned by devices were also able to get close to the fair result in a significantly smaller number of rounds.

### *5.4 Training effectiveness*

Recognition of device preferences by a proactive document enables it to guess a contract faster than by negotiating a

larger content of the respective bargaining set. As described before in Figure 4(b) and Figure 5(b), it is a two-step process: first a device class is classified by neural network N1, and next a sequence of offers from that set is generated by module M3, based on relations between offers in $C_B$, recognised by neural network N2. A proactive document would have rather limited computational resources when executing on the device, therefore experiments were aimed at finding acceptably small-sized network architectures that could effectively classify devices and sequences within domains determined by attribute value options specified before.

In a series of experiments it was found that a 3-layer network N1, with 20 neurones in each of the first two layers and nine neurones in the output layer sufficed to recognise device classes, whereas a 2-layer network N2, with ten neurones in the first layer and 1 neurone in the output layer sufficed to recognise preferences of a device from a single class. Networks N1 and N2 had respectively 82 and 174 inputs.

Two characteristics of intelligent proactive documents were determined in these experiments: *fitness level* mentioned before, and *hit rate*, measuring the ratio of sequences of offers reconstructed correctly from the negotiated bargaining sets. Results are shown in Figure 10.

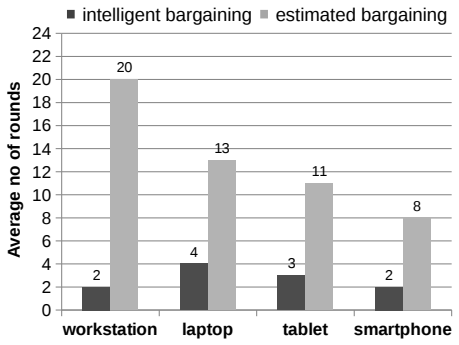
For the experiments with networks $N_2$ trained to recognise preferences of devices, output produced only by adequately trained networks $N_1$ was used, i.e., recognising device classes with a 100% fitness level (called *perfect* for brevity). It allowed us to eliminate unnecessary cumulation of errors.

It may be seen in Figure 10(a) that $N_1$, trained with at least 50 example sequences containing all offers specified by the respective option tree, was able to reach a perfect fitness level. In other words, a proactive document using such a network would properly recognise each device class defined in Subsection 5.1. Sequences generated from the experimental bargaining sets were used next to train each respective $N_2$ network capable of recognising device preferences. It may be seen in Figure 10(b) that for the workstation and laptop classes, already ten sequences were sufficient to train the relevant $N_2$ network to recognise device preferences, whereas above 20 sequences preferences of devices from each class considered in the paper could be recognised.

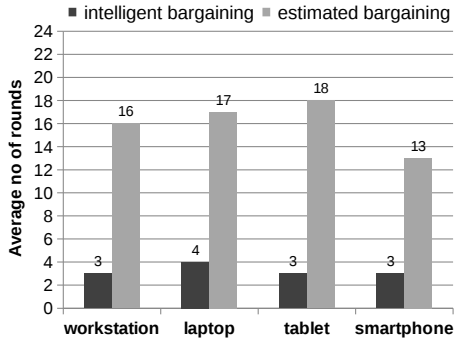
### 5.5 Performance testing

The experiments reported above allowed us to realistically assess costs of implementing the proposed intelligent bargaining mechanism in terms of the *time* needed to train a document and the additional *memory load* that it will have to carry to support five neural networks needed to recognise four device classes and their preferences. Training a document to recognise device classes varied from 1.27 to 3.36 seconds, when performed on a moderately equipped laptop. Furthermore, training a document to recognise preferences of a given device class varied from 3.03 to 12.45 seconds on the same laptop, respective to the lengths of example sequences used. It implies that actual training may be performed directly on one of the available execution devices in the system.

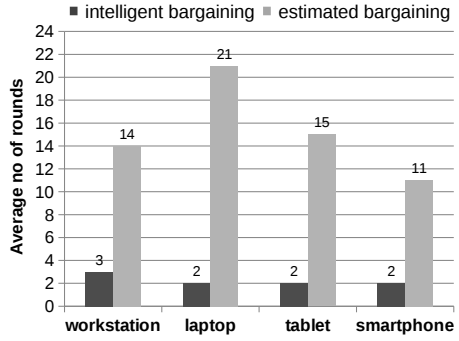
**Figure 9** Intelligent vs. estimated offer selection in four document classes, (a) protected and heavy (b) protected and light (c) open and heavy (d) open and light



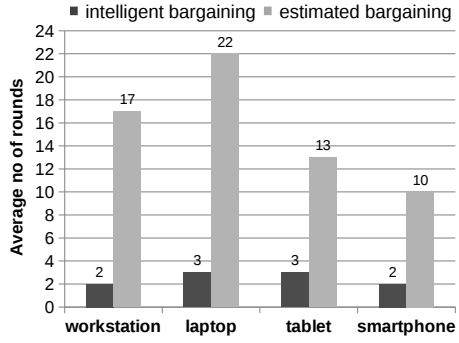

(a)




(b)

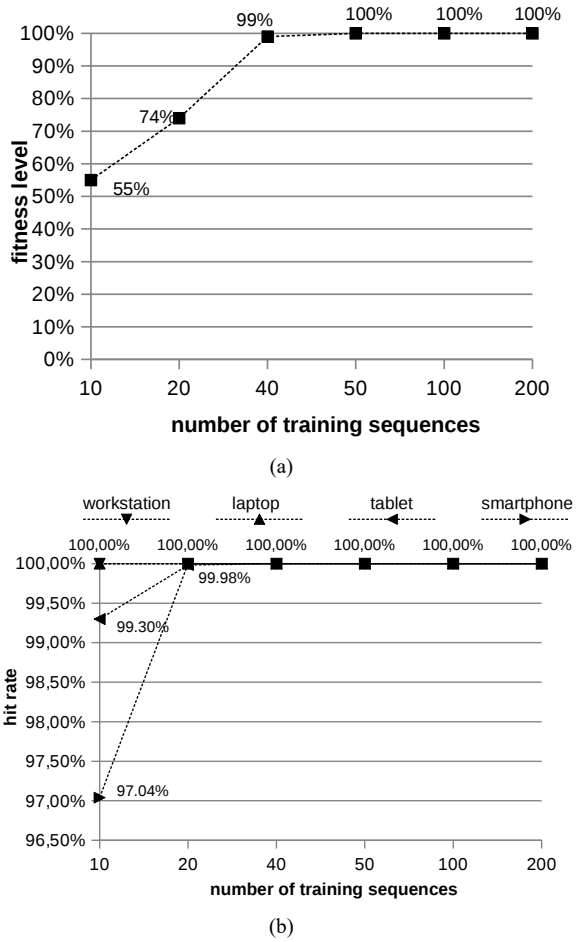



(c)




(d)

**Figure 10** Document training capability characteristics, (a) network $N_1$ for device classes (b) network $N_2$ for device preferences




## 5.6 *Volume testing*

Moderately sized bargaining sets, which were used in the experiments reported before, resulted in relatively short training sequences; after converting them to the relational form, as explained in Subsection 4.2, they provided a teaching material of a lesser volume. Such a setting could make training exercises more difficult compared to a richer set of relations. However, larger bargaining sets (of about 1,000 offers) were also tested, over which various documents negotiated with a laptop device. With estimated selection of offers (Algorithm 3), a proactive document was able to conclude negotiation with its execution device in about 100 rounds, with respective average payoffs for each party $\pi_1 = 0.95 \cdot 10^{-10}$ and $\pi_2 = 0.75 \cdot 10^{-10}$, and $\delta_1 = \delta_2 = 0.8$. Network $N_2$ trained with a set of precedence relations generated on the basis of this exercise was able to recognise about 85% pairs, which upon reconstruction of sequences of offers in the order preferred by a device, allowed documents to reach agreements with devices in about three rounds, like in the experiments with smaller bargaining sets described before. The average payoffs were respectively $\pi_1 = 0.26$ and $\pi_2 = 0.22$. Average time of training $N_2$ was, however, nearly eight hours – on the same laptop used earlier during experiments with bargaining sets of a lesser size. That may be considered a long time, but as mentioned at the beginning of Section 4, training of the network can be performed offline in the agency, in between document-agent missions. Moreover, the agency may use a more powerful server provided by the organisation implementing its workflow processes with document-agents proposed in this paper.

## 5.7 *Implementability considerations*

Intelligent selection of offers implemented by Algorithm 4 requires each proactive document to carry code enabling it to perform five different classification tasks. These tasks, however, may be accomplished with just two network architectures $N_1$ and $N_2$ specified before.

In our implementation a document would have to carry five different sets of weights to set up just two networks to work, whereas execution of the network would be delegated to the dedicated e-mail client installed on a device, as mentioned in Section 1.

Additional load of documents capable of recognising device classes and device preferences could be estimated for three layers of network `N1` as:

1 20 neurones, 82 inputs each; 82 weights and 1 bias for each neurone requires $20 \cdot (82 + 1) = 1{,}660$ floating point numbers

2 20 neurones, 20 inputs each; 20 weights and 1 bias for each neurone requires $20 \cdot (20 + 1) = 420$ floating point numbers

3 9 neurones, 20 inputs each; 20 weights and 1 bias for each neurone requires $9 \cdot (20 + 1) = 189$ floating point numbers.

and two layers of `N2` as

1 10 neurones, 174 inputs each; 174 weights and 1 bias for each neurone requires $10 \cdot (174 + 1) = 1{,}750$ floating point numbers

2 1 neurone, 10 inputs; 10 weights and 1 bias requires $1 \cdot (10 + 1) = 11$ floating point numbers.

Based on the above it can be calculated that networks `N1` and `N2` required respectively 2,269 and 1,761 floating point numbers to set them up, therefore for handling four classes considered in the paper the total of 9,313 floating point numbers were required. In Java single-precision 32-bit IEEE 754 floating point representation (IEEE, 2008) that amount of data required a document to carry just 37.2KB of extra load.

These figures are very promising and provide a good indicator for our current implementation effort aimed at incorporating `N1` and `N2` network architectures in the dedicated e-mail client mentioned before.

## 6 Related work

A negotiation model considered in the paper involves exchange of multi-issue offers by two players, with no information about preferences known to one another, nor any assumption made on the class of functions used to calculate utility of offers. The only assumption on the negotiation process has been that parties start with the offer of their highest utility and make gradual concessions by selecting consecutive offers from their individually sorted option trees. One advantage of the simple negotiation model proposed in the paper is that SBG can be played by proactive documents and execution devices without support of any external negotiation service. Of course, solutions based on external services may be considered to aid the negotiating party in that respect, such as negotiation-as-a-service (NaaS) proposed in Bala et al. (2013), but it may work only when the proactive document-agent has (or the device's options file allows it to) access the network from its current execution device.

### 6.1 Multi-issue negotiation

Although a generic approach to the problem of multi-issue negotiation with no information about the opponent has been proposed in the literature (Lai and Sycara, 2009), a formal mathematical proof of the convergence of the monotonic concession strategy, which SBG implements with option trees, was not provided until (Zheng et al., 2013). The only property of utility functions of negotiating parties assumed there was that they had to be *concave*, due to a specific geometric interpretation of the space of offers $A_T$. It was shown that the distance between offers of the two negotiating parties decreases in each round until a contract could be agreed. This geometrical interpretation may be applied to SBG played by proactive documents and devices negotiating over option trees. Recall the example negotiation in Figure 1 and note that utility values of offers and counteroffers selected by each party according to the amount of concession, which either one can accept in the current round, are getting closer in the space of offers until contract $o_3$ is agreed. Two observations could be made on the model proposed in this paper with regard to the above. One is that bargaining sets are discrete and option trees provide strictly monotonic ordering of complete offers for each respective negotiating party. Another is that the utility values assigned to each issue in the option tree may be calculated by parties arbitrarily. Therefore no specific assumptions have to be made on the class nor properties of utility functions used by players to evaluate offers in their respective option trees – except to be injective, to enable sorting of the tree and ensure monotonicity of preferences.

### 6.2 Intelligent negotiation

Augmenting proactive documents with learning capability to speed-up bargaining, is an important feature for negotiation with mobile execution devices – as their capabilities may dynamically degrade between encounters because of unstable network connections or low battery load, among others.

A major problem for negotiation parties is the lack of a detailed knowledge on the opponent and its tactics, making negotiating behaviour of the latter hard to predict. That knowledge could be discovered by analysing historical data on previous encounters, or when no negotiation history is available – by implementing mechanisms enabling dynamic adaptation of the negotiation agent to its opponent behaviour during the actual encounter (Baarslag et al., 2016).

Modelling specific opponent's characteristics by a negotiation agent to enable it to adapt to the behaviour of the former, and generate offers leading to agreements faster, may be represented as the optimisation problem. For example, in Wang and Wang (2013), the best offer in each round is searched in the set of possible offers using the particle swarm optimisation (PSO) method (Kennedy and Eberhart, 1995). PSO was used to maximise a specially defined objective function parametrised by *time pressure* and *eagerness* factors; the latter aggregated history success-or-failure of the last few transactions. Another example of modelling generation of offers as the optimisation problem could be found in Brzostowski and Kowalczyk (2006), where negotiation was modelled as a multi-state control process, so the intelligent agent's task was to determine a sequence of optimal controls, i.e., to predict future offers.

Using neural networks to aggregate history of past offers and assist a negotiation agent in selecting appropriate tactics have been demonstrated in Roussaki et al. (2011). Authors assumed that no a-priori knowledge on opponents existed, thus proper offline training of the network with historical data was not possible; instead they proposed online training during the actual encounter. In consequence, their research concentrated on predicting offers that would faster lead to the contract – based on the offers submitted by the opponent during some initial series of rounds of the actual encounter. Two problems with this approach exist, namely how many rounds are needed to train a network during the current encounter in order to enable it to generate a concluding offer, and what are computational costs of achieving that, i.e., what minimum size of the network would suffice. Some hints were given in Roussaki et al. (2006), presenting results of experiments with neural networks of various sizes in a single issue bilateral negotiation; encounters of 100–200 rounds required just three neurones in its hidden layer and one in its output layer to predict a contract.

When compared to the above, our approach also relies on neural networks, but prefers training of document agents before, rather then during the encounter. In general, neural networks are supposed to perform better when they are trained offline, owing to a much richer set of offers that could be used to train a network (Wilson and Martinez, 2003). Two observations could be made on the model proposed in this paper with regard to the above. One is that content of option trees is predetermined by a set of classes of execution devices that a proactive document

could encounter during its workflow. Another is that each execution device has some inherent preferences imposed by rules specific to its class. With our experiments we were able to show that if only the training set includes sufficiently many offers from bargaining sets of each respective device class, the network may be effectively trained to recognise device classes based on the offers in the respective bargaining sets they share with their actual oponnents. Moreover, solution of the problem of predicting the exact sequence of offers leading to the agreed contract by a negotiation agent, addressed before in Brzostowski and Kowalczyk (2006), may be based on generalising orderings of offers observed during its past encounters with devices of the same, properly recognised class.

## 7 Conclusions

Main contribution of the paper is threefold. One is document engineering, which uses a document-centric modelling perspective to view documents as both interaction and information units (Glushko and McGrath, 2008). Owing to the negotiation mechanism, a proactive document's code brought to the receiving device can be seamlessly adjusted and incorporated in the actual execution context to become an integral component of its system. Another is knowledge driven BPM systems, which could be set-up ad hoc by collaborators to support human decision-making by leveraging knowledge about processes and their contexts in an automated and proactive manner (Motahari Nezhad and Akkiraju, 2015). Finally, augmenting e-mail messages with proactive document attachments, which can combine their passive content with active services, and interact on their own with devices and their users, allowed converting an ordinary e-mail system into a sort of multi-agent system, introducing to e-mail based exchange of documents self-organisation and self-steering (Bellotti et al., 2005).

The primary goal of the research reported in the paper has been to show that conflicting requirements on execution contexts can be resolved by a proactive document-agent and a device-agent in a responsive way in a game-theoretic fashion and without any external service support. This could be particularly useful when devices operate in uncertain or insecure network environments or under pressure of time or low battery load, among others.

Another goal has been to assure concluding negotiation in a possibly small number of rounds. One solution for that could be just reducing a document's discount factor. It, however, would weaken negotiation position of a document, as devices could use discount factors of higher values, leading to contracts of lesser payoffs to the document. Instead, an estimated selection of offers was used, which allowed parties to explore bargaining sets for a concluding offer in fewer number of rounds, compared to the naive version, not suitable for larger bargaining sets. Experiments indicated that estimated selection lead to reasonable payoffs, close to the fair one.

Finally, it has been demonstrated that if negotiation histories from previous encounters with devices exist, i.e., documents had a chance to meet representatives of certain device classes during their past missions, machine learning using really simple neural networks could be adopted by a document in a cost effective way to improve selection of offers during negotiation. Such an improvement would not imply that documents might be willing to accept offers submitted by the opposing device faster, but by discovering its preferences to save a number of rounds for negotiating a contract that would be agreed anyway.

As the extension of this work, we plan to investigate further how proactive document-agents could learn classes of execution devices and their preferences not included initially in the classification proposed in Section 2 – such as smartwatches, devices which already exist but their functionality evolves in time, or devices yet to be invented for the growing IoT ecosystem. Besides, a dedicated application involving negotiating proactive document-agents may use a more specialised definition of its space of offers – to narrow the target specific options and fit better to the given application class or document semantics, e.g., *security* for banking documents, *performance* for authoring documents with multimedia content, or *reliability* for documents exchanged in crisis management systems. Definitions of specific spaces of offers could be based on existing specification standards, e.g., the common information model (CIM) (DMTF, 2005). These applications may, however, require more advanced machine learning schemes, e.g., a social learning approach proposed in Hao et al. (2014) – where documents could collect more information about the population of devices by interacting with their peers rather then working alone.

## Acknowledgements

This work was supported in part by the National Science Center in Poland under grant DEC1-2011/01/B/ST6/06500

## References

Baarslag, T., Hendrikx, M.J.C., Hindriks, K.V. and Jonker, C.M. (2016) 'Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques', *Autonomous Agents and Multi-Agent Systems*, September, Vol. 30, No. 5, pp.849–898, DOI: 10.1007/s10458-015-9309-1.

Bala, M.I., Vij, S. and Mukhopadhyay, D. (2013) 'Intelligent agent for prediction in e-negotiation: an approach', in *Proceedings of the International Conference on Cloud Ubiquitous Computing Emerging Technologies, CUBE 2013*, November, pp.183–187, DOI: 10.1109/CUBE.2013.41.

Bellifemine, F.L., Caire, G. and Greenwood, D. (2007) *Developing Multi-Agent Systems with JADE*, Wiley.

Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. and Grinter, R.E. (2005) 'Quality versus quantity: e-mail-centric task management and its relation with overload', *Human–Computer Interaction*, Vol. 20, Nos. 1–2, pp.89–138.

Binmore, K. (1994) *Game Theory and the Social Contract*, Playing Fair Edition, Vol. 1, MIT Press, Cambridge, MA.

Brzostowski, J. and Kowalczyk, R. (2006) 'Predicting partner's behaviour in agent negotiation', in *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS'06*, ACM, New York, NY, USA, May, pp.355–361, DOI: 10.1145/1160633.1160697.

Cotton, D., Eggert, L., Touch, J., Westerlund, M. and Cheshire, S. (2011) *Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry*, RFC 6335, RFC Editor, August [online] https://www.rfc-editor.org/info/rfc6335 (accessed 19 Oct 2018).

DMTF (2005) *Common Information Model* [online] http://www.dmtf.org/standards/cim (accessed 19 Oct 2018).

Dourish, P., Edwards, W.K., LaMarca, A., Lamping, J., Petersen, K., Salisbury, M., Terry, D.B. and Thornton, J. (2000) 'Extending document management systems with user-specific active properties', *ACM Trans. Inf. Syst.*, April, Vol. 18, No. 2, pp.140–170, DOI: 10.1145/348751.348758.

Faratin, P., Sierra, C. and Jennings, N.R. (1998) 'Negotiation decision functions for autonomous agents', *Robotics and Autonomous Systems*, Vol. 24, No. 3, pp.159–182 [online] https://doi.org/10.1016/S0921-8890(98)00029-3 (accessed 19 Oct 2018).

Freed, N. and Borenstein, N. (1996) *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC 2045, RFC Editor, November [online] https://www.rfc-editor.org/info/rfc2045 (accessed 19 Oct 2018).

Glushko, R.J. and McGrath, T. (2008) *Document Engineering – Analyzing and Designing Documents for Business Informatics and Web Services*, MIT Press.

Hao, J., Leung, H-F. and Ming, Z. (2014) 'Multiagent reinforcement social learning toward coordination in cooperative multiagent systems', *ACM Trans. Auton. Adapt. Syst.*, December, Vol. 9, No. 4, pp.20:1–20:20, DOI: 10.1145/2644819.

IEEE (2008) *IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2008, August, pp.1–70, DOI: 10.1109/IEEESTD.2008.4610935.

Kaczorek, J. (2015) *Automated Negotiations over Collaboration Protocol Agreemens*, PhD thesis, Gdańsk University of Technology [online] http://pbc.gda.pl/Content/55656/phd_kaczorek_jerzy.pdf.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', in *Proceedings of the IEEE International Conference on Neural Networks*, IEEE, Los Alamitos, CA, USA, November, Vol. 4, pp.1942–1948, DOI: 10.1109/ICNN.1995.488968.

Lai, G. and Sycara, K. (2009) 'A generic framework for automated multi-attribute negotiation', *Group Decision and Negotiation*, Vol. 18, No. 2, pp.169–187, DOI: 10.1007/s10726-008-9119-9.

MacCrimmon, K.R. and Messick, D.M. (1976) 'A framework for social motives', *Behavioral Science*, Vol. 21, No. 2, pp.86–100, DOI: 10.1002/bs.3830210203.

Motahari Nezhad, H.R. and Akkiraju, R. (2015) *Towards Cognitive BPM as the Next Generation BPM Platform for Analytics-Driven Business Processes*, pp.158–164, Springer International Publishing.

Nash, J. (1950) 'The bargaining problem', *Econometrica*, Vol. 18, No. 2, pp.155–162.

Roussaki, I., Papaioannou, I.V. and Anagnostou, M.E. (2006) 'Employing neural networks to assist negotiating intelligent agents', in *Proceedings of the 2nd IET International Conference on Intelligent Environments, IE'06*, IEEE, Los Alamitos, CA, USA, July, Vol. 1, pp.101–110, DOI: 10.1145/1160633.1160697.

Roussaki, I., Papaioannou, I.V. and Anagnostou, M.E. (2011) 'Using neural networks for early detection of unsuccessful negotiation threads', *International Journal on Artificial Intelligence Tools*, Vol. 20, No. 3, pp.457–487, DOI: 10.1142/S0218213011000231.

Schimkat, R. and Küchlin, W. (2002) 'Living documents – micro servers for documents', in *XML-Based Data Management and Multimedia Engineering – EDBT 2002 Workshops, EDBT 2002 Workshops XMLDM, MDDE, and YRWS*, Prague, Czech Republic, 24–28 March, pp.512–525.

Triantaphyllou, E. (2013) 'Multi-criteria decision making methods: a comparative study', *Applied Optimization*, Springer, USA, ISBN: 9781475731576.

Wang, Z. and Wang, L. (2013) 'Adaptive negotiation agent for facilitating bi-directional energy trading between smart building and utility grid', *IEEE Transactions on Smart Grid*, Vol. 4, No. 2, pp.702–710, DOI: 10.1109/TSG.2013.2237794.

Wilson, D.R. and Martinez, T.R. (2003) 'The general inefficiency of batch training for gradient descent learning', *Neural Networks*, Vol. 16, No. 10, pp.1429–1451, DOI: 10.1016/S0893-6080(03)00138-2.

Zheng, R., Chakraborty, N., Dai, T., Sycara, K. and Lewis, M. (2013) 'Automated bilateral multiple-issue negotiation with no informatio about opponent', in *Proceedings of the 47th Hawaii International Conference o System Sciences, HICSS 2013*, IEEE, Los Alamitos, CA, USA, pp.520–527, DOI: 10.1109/HICSS.2013.626.