Negotiating Complex Contracts

Mark Klein
Massachusetts Institute of Technology
Cambridge MA 02139
m_klein@mit.edu

Peyman Faratin
Massachusetts Institute of Technology
Cambridge MA 02139
peyman@mit.edu

Hiroki Sayama
New England Complex Systems Institute
Cambridge MA 02138
sayama@necsi.org

Yaneer Bar-Yam
New England Complex Systems Institute
Cambridge MA 02138
yaneer@necsi.org

Abstract

Work to date on computational models of negotiation has focused almost exclusively on defining contracts consisting of one or a few independent issues, linear utility functions, and tractable contract spaces. Many real-world contracts, by contrast, are much more complex, consisting of multiple inter-dependent issues, nonlinear utility functions, and intractably large contract spaces. This paper describes a simulated annealing based approach appropriate for negotiating such complex contracts, evaluates its efficacy, and suggests potentially promising avenues for defining more efficient algorithms for negotiating complex contracts.

Introduction

Work on computational models of negotiation has focused almost exclusively on defining contracts consisting of one or a few independent issues (Faratin, Sierra et al. 2000) (Ehtamo, Ketteunen et al. 2001). They work as follows:

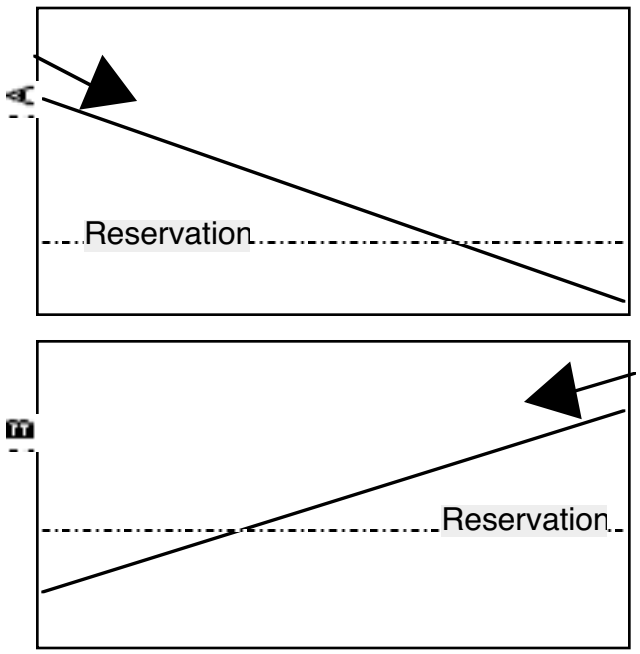

Figure 1: The standard approach to negotiation.

Each point on the X axis represents a candidate contract. For simplicity of exposition we show only one dimension in these figures, but in general there will be one dimension for every issue negotiated over. The Y axes represents the utility of each contract to each agent. Both agents have a reservation utility value: only contracts whose utility is above that agent's reservation value will be accepted. Since relative few issues are involved, the space of all possible contracts can be explored exhaustively, and since the issues are independent, the utility functions mapping a candidate contract to its utility for an agent are *linear* (Bar-Yam 1997). In such a context, the reasonable strategy is for each agent to start at its own ideal contract, and concede, through iterative proposal exchange, just enough to get the other party to accept the contract. Since the utility functions are simple, it is feasible for one agent to infer enough about the opponent's utility function through observation to make concessions likely to increase the opponent's utility.

Real-world contracts, by contrast, are generally much more complex, consisting of a large number of inter-dependent issues. A typical contract may have tens to hundreds of distinct issues. Even with only 50 issues and two alternatives per issue, we encounter a search space of roughly 10^15 possible contracts, too large to be explored exhaustively. The value of one issue selection to an agent, moreover, will often depend on the selection made for another issue. The value to me of a given DVD player, for example, depends on whether it is a good match with the tuner and speakers I plan to purchase with it. Such issue interdependencies lead to *nonlinear* utility functions with multiple local optima (Bar-Yam 1997):

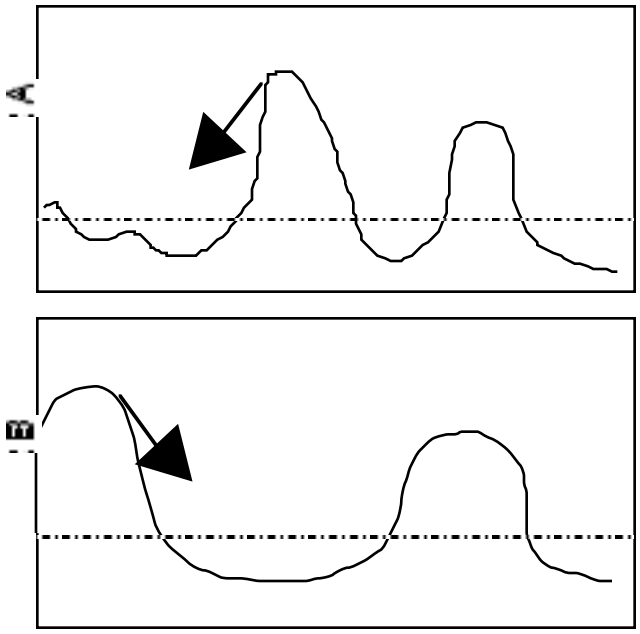

Figure 2: Negotiation with complex contracts.

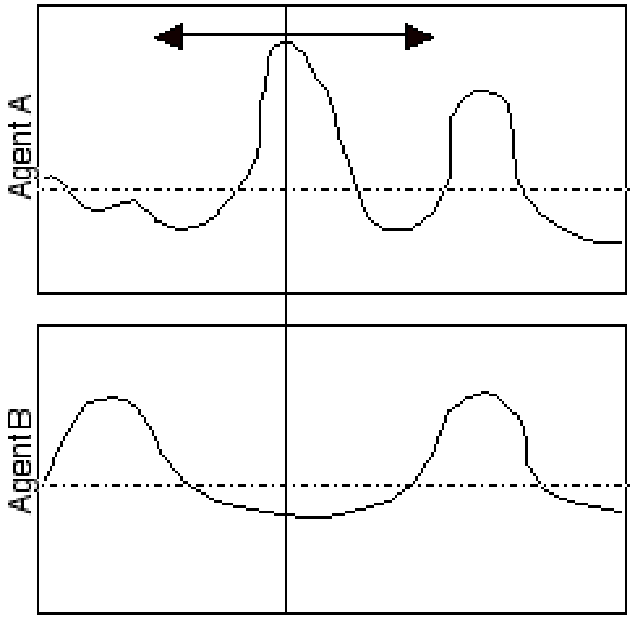

Figure 3: Single text negotiation.

In such contexts, an agent finding its own ideal contract becomes a nonlinear optimization problem, difficult in its own right. Simply conceding as slowly as possible from one's ideal can result in the agents missing contracts that would be superior from both agent's perspectives. In figure 2 above, for example, if both agents simply concede slowly from their own ideal towards the opponents' ideal, they will miss the better contracts on the right. Exhaustive search for such 'win-win' contracts, however, is impractical due to the size of the search spaces involved. Finally, since the utility functions are quite complex, it is no longer practical for one agent to learn the other's utility function in a reasonable amount of time.

Such contexts, we argue, require radically different negotiation techniques, which allow agents to find 'win-win' contracts in intractably large multi-optima search spaces in a reasonable amount of time. In the following section we describe a negotiation approach that make substantial progress towards achieving these goals.

Mediated Single Text Negotiation

A standard approach to dealing with complex negotiations in human settings is the mediated single text negotiation (Raiffa 1982). In this process, a mediator proposes a contract that is then critiqued by the parties in the negotiation. A new, hopefully better proposal is then generated by the mediator based on these responses. This process continues, generating successively better contracts, until the reservation utility value is met or exceeded for both parties. We can visualize this process as follows (Figure 3):

Here, the vertical line represents the contract currently proposed by the mediator. Each new contract moves the line to a different point on the X axis. The goal is to find a contract that is sufficiently good for both parties.

We defined a simple experiment to help us explore how this approach could be instantiated in a computational framework. In this experiment, there were two agents negotiating to find a mutually acceptable contract consisting of a vector S of 100 boolean-valued issues, each issue assigned the value 0 or 1, corresponding to the presence or absence of a given contract clause. This defined a space of 2^100, or roughly 10^30, possible contracts. Each agent had a utility function calculated using its own 100x100 influences matrix H, wherein each cell represents the utility increment or decrement caused by the presence of a given pair of issues, and the total utility of a contract is the sum of the cell values for every issue pair present in the contract:

$$U = \sum_{i=1}^{100} \sum_{j=1}^{100} H_{ij} \, S_j S_j$$

The influence matrix therefore captures the dependencies between issues, in addition to the value of any individual contract clause. For our experiments, the utility matrix was initialized to have random values between –1 and +1 in each cell. A different influences matrix was used for each simulation run, in order to ensure our results were not idiosyncratic to a particular configuration of issue inter-dependencies.

The mediator proposes a contract that is initially generated randomly. Each agent then votes to accept or reject the contract. If both vote to accept, the mediator

mutates the contract (by randomly flipping one of the issue values) and the process is repeated. If one or both agents vote to reject, a mutation of the most recent mutually acceptable contract is proposed instead. The process is continued until the utility values for both agents become stable (i.e. until none of the newly generated contract proposals offer any improvement in utility values for either agent). Note that this approach can straightforwardly be extended to a N-party (i.e. multi-lateral) negotiation, since we can have any number of parties voting on the contracts.

We defined two kinds of agents: hill climbers and simulated annealers. The hill climbers used a very simple decision function: they accepted a mutated contract only if its utility to them was greater than that of the last contract they accepted. The annealers were more complicated, implementing a Monte Carlo machine (Kalos and Whitlock 1986). Each annealer had a virtual 'temperature' T, such that it will accept contracts worse than earlier ones with the probability:

$$P(\text{accept}) = e^{-\Delta U/T}$$

where ΔU is the utility change between contracts. In other words, the higher the virtual temperature, and the smaller the utility decrement, the greater the probability that the inferior contract will be accepted. The virtual temperature of an annealer gradually declines over time so eventually it becomes indistinguishable from a hill climber. This kind of annealing has proven effective in finding near-optima in large multiple-optima utility functions, because annealers can move freely through the utility function, potentially skipping relatively small valleys on the way to higher optima (Bar-Yam 1997). This suggests that annealers will be more successful than hill-climbers in finding good contracts through the negotiation process. The reality, as we shall see, turned out to be more complicated.

Our aggregate results comparing hill-climbers with annealers can be summarized as follows:

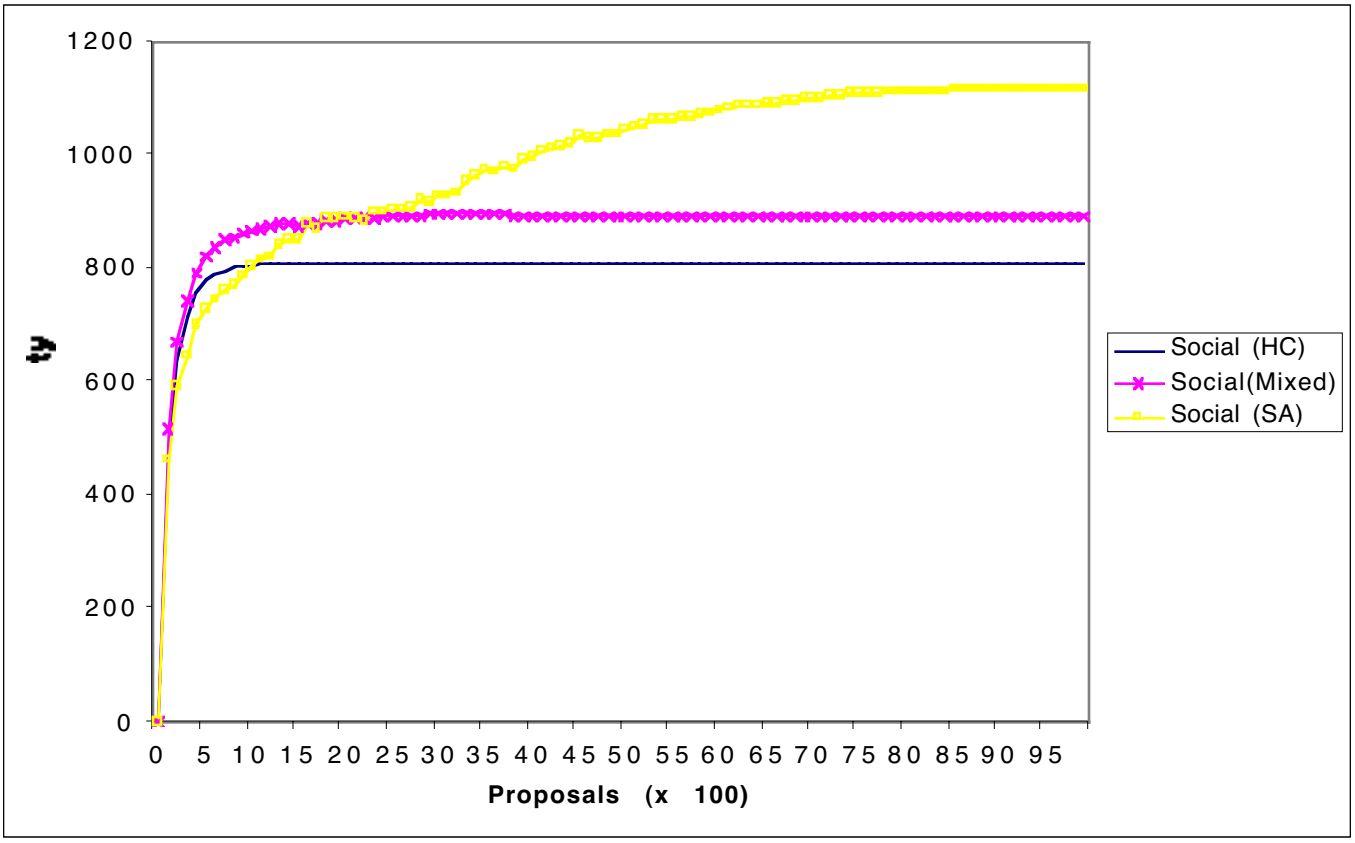

Figure 3: Social welfare values

This figure shows the social welfare (i.e. the sum of the contract utilities for the two negotiating agents) for a pair of annealers, a pair of hill-climbers, and a 'mixed' case with both an annealer with a hill-climber, averaged over 100 simulation runs.

The social welfare results revealed two interesting patterns. One is that *the presence of annealer agents always increases social welfare*. The social welfare for the two annealer case was roughly 40% greater than that of the two hill-climber case, and the mixed case produced a smaller but still statistically significant 15% improvement over the hill-climbers. This confirmed the value of the simulated annealing approach.

These results can be understood better by looking at the individual agent payoffs:

	Agent 2 anneals	Agent2 hill-climbs
Agent 1 anneals	[1100] 550/550	[880] 700/180
Agent1 hill-climbs	[880] 180/700	[800] 400/400

where the cell values are laid out as follows:

[<social welfare>]
<agent 1 utility>/<agent 2 utility>

If both agents are hill-climbers, then they both get a poor payoff, since it is difficult to find many contracts that represent an improvement for both parties. A typical negotiation for two hill-climbers looks like the following:

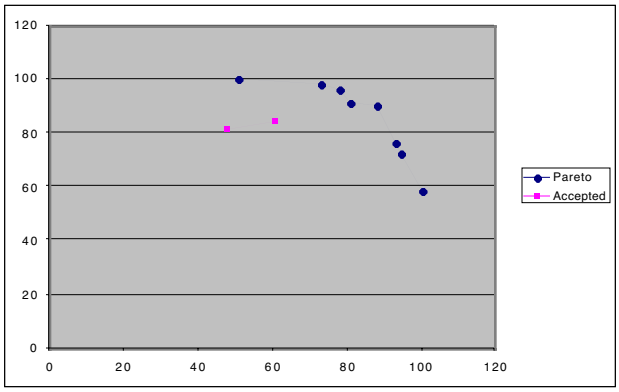

Figure 4: A typical negotiation with two hill-climbers.

This figure shows the normalized utilities of the accepted contracts for each agent, plotted next to the pareto-efficient line. As we can see, in this case the mediator was able to find only two contracts that satisfied both hill-climbers.

If one agent is a hillclimber and the other is an annealer, the hillclimber does well but the annealer fares very poorly. This pattern can be understood as follows. When an annealer is at a high virtual temperature, it becomes a chronic conceder, accepting almost anything, beneficial or not. The hill-climber thus in effect 'drags' the annealer towards its own local optimum, which is not particularly likely to also be optimal for the annealer:

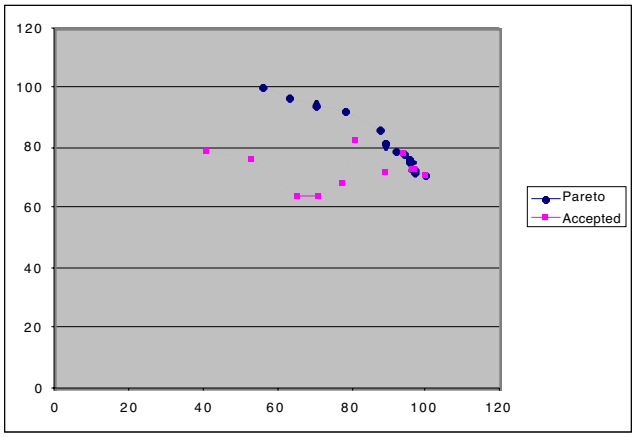

Figure 5: A typical negotiation with an annealer and hillclimber.

The highest social welfare, however, is achieved when both agents are annealers, since they both are willing to accept individually worse contracts in the beginning in the hope of finding win-win contracts later on:

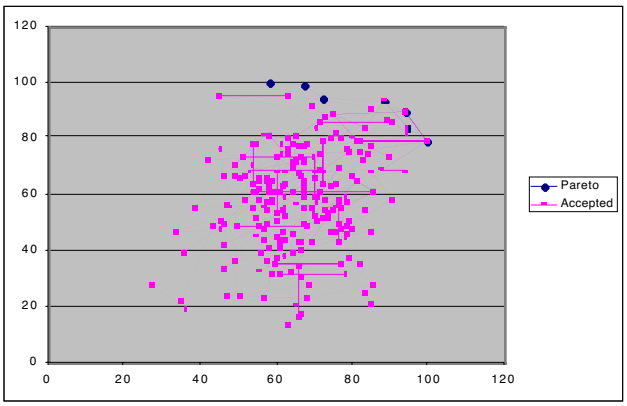

Figure 6: A typical negotiation with two annealers.

When we ran the experiments with 15 issues (producing a small enough contract space to allow calculation of the pareto-efficient line) we found that negotiations with one or more annealers almost always reached a pareto-efficient contract, but negotiations with two hillclimbers almost never found a pareto-efficient contract.

The results also show that *hill-climbers reach stability sooner than annealers*. The hill climbers typically reached stability after roughly 100 proposal exchanges, while the annealers approached stable utility values after roughly 800 proposal exchanges. This makes sense because hill climbers simply climb to the top of the closest utility optimum and then stop, while annealers can, when at a high temperature at least, 'hop' among multiple optima in the utility function.

Next Steps: Faster Negotiations

The simulated annealing approach produces better social welfares than hill-climbing but involves larger numbers of proposal exchanges. What can we do about this?

Better contract alternative generation operators. In our experiments the contract space was explored in random walk fashion, and all the 'intelligence' was in the evaluation process. One example of a domain-independent approach we are exploring is 'genetic annealing', which uses abstracted measures of the issue inter-dependency structure to cluster highly-interdependent issue sets into 'genes' that are recombined al la sexual reproduction to more quickly explore the large search spaces involved in contingent contract design.

Introducing (limited) cooperative information exchange. It is clear that if agents cooperate they can produce higher contract utilities. Imagine for example that two hill-climbers vote to accept a contract based on whether it increases the social welfare, as opposed to their individual utilities. We have found that if we compare this with two 'selfish' hill-climbers, the cooperative hill-climbers *both* benefit individually compared to the selfish case, thereby increasing social welfare as well. Other kinds of cooperation are imaginable. Agents can begin by presenting a list of locally [near-]optimal contracts, and then agree to explore alternatives around the closest matches in their two sets. Note that in the previous work with independent issues, this kind of information exchange has not been necessary because it relatively easy for agents to infer each other's utility functions from observing their negotiation behavior, but with inter-dependent issues and large multiple-optima utility functions this becomes intractable and information exchange probably must be done explicitly.

Next Steps: Contracts as Processes

Another direction we plan to pursue for our future work involves providing a systematic way of defining the space of possible contracts. Any contract can, we argue, be viewed as the specification for a *process* that spells out which actor does what when. The simplest of contracts may only specify that a good is exchanged for a given monetary consideration. The most complex contracts may spell out, in excruciating detail, what each party should do in a wide range of normal and exceptional circumstances. In all cases, however, the contract represents a mutually agreed-to process.

Previous work (Bernstein, Klein et al. 1999) has shown that process (and therefore contract) design can be treated as configuration, wherein one first identifies the abstract process one is interested in, and then customizes it by selecting specific processes for each of the substeps in the abstract process. To make this concrete, imagine that we want to subcontract out the task of purchasing goods over the Internet. The first step is to find an abstract process for this that we can customize. One way to do so is by retrieving the appropriate process from a process ontology, such as that stored in the MIT Process Handbook (Malone, Crowston et al. 1999). The Handbook ontology contains over 5000 business process models, ranging from very abstract processes such as 'allocate resources' to relatively specific ones. The Handbook model for 'buy over internet', for example, consists of the following five substeps:

- Identify needs
- Find sources via internet
- Select supplier
- Place order over internet
- Pay using credit card
- Receive good

Each one of the substeps in this abstract process model has a branch of the process ontology that captures different ways of achieving this step. The branch for the 'select supplier' substep, for example, is the following:

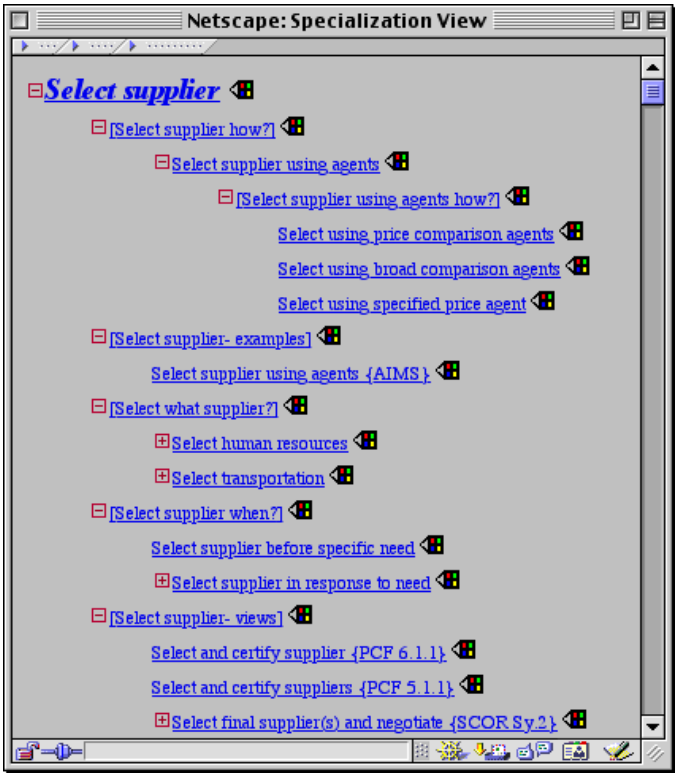

Figure 7: A fragment of the 'select supplier' ontology.

One can therefore start specifying the contract by selecting which one of the alternative 'select supplier' processes will be used. Repeating this procedure for all the substeps of the abstract 'buy over internet' process (as well as specifying any necessary attribute values for the selected substep processes) should result in a complete specification of the normative aspects of the subcontract.

The next step in defining the contract is to consider how the execution of the contract can fail (i.e. its *exceptions*) and how those exceptions can be handled. This is an important problem. MAS are increasingly being viewed as a way to rapidly connect entities that have never worked together before, for such applications as disaster recovery operations, open electronic marketplaces, virtual supply chains, and international coalition military forces (Jennings, Sycara et al. 1998) (Wooldridge, Jennings et al. 1999) (1999) (Fischer, Muller et al. 1996) (Tsvetovatyy, Gini et al. 1997). Such 'open' systems introduce a wide range of potentially devastating exceptions including infrastructure failures, agents reneging on commitments, denial of service attacks, emergent dysfunctions such as chaotic behavior (Youssefmir and Huberman 1995) (Sterman 1994) and so on (Klein and Dellarocas 1999) The vast majority of MAS work to date, however, has considered only well-behaved agents running on reliable infrastructures in relatively simple domains (Hägg 1996) (Klein, Rodriguez-Aguilar et al. 2000).

One possible approach to handling contract execution exceptions is to simply require that agents charge less for their services in proportion to how likely they are to fail at their assigned tasks. The problem with this approach is that it removes from subcontractor agents any direct incentive to avoid failures or reduce their impact on the contractor (they have already paid the penalty up-front) and it can be difficult for the contracting agent to assess the subcontractor failure likelihood and thereby the appropriate discount. An alternative approach is for the contract to specify the actions, including for example early notification or penalties, to be taken for each important exception. The appropriate penalties are easier to determine because the contractor need only estimate the impact (but not the probability) of an exception, and it provides the subcontractor with incentives to avoid high-impact failures. This thus results in contracts that increase social welfare.[1]

The Process Handbook ontology has been extended to support such contractual specification of exception handling behavior, by the addition of a taxonomy of generic exception types, such that all processes are linked to their characteristic exceptions, and all exceptions are linked in turn to processes suitable for handling (anticipating and avoiding, or detecting and resolving) them (Klein and Dellarocas 2000). Imagine, for example, that we have selected a Dutch (descending price) auction process for supplier selection. By consulting the Process Handbook ontology we can see that such auctions are prone, among other things, to the 'price collision loop' exception wherein two agents give the same bid for a good, leading the auctioneer to raise the price and try again, leading to the same bid conflict, ad infinitum. The Handbook ontology describes a range of handlers for this exception, such as randomly selecting a winner, disqualifying one or both conflicting bidders, and so on. Once handlers have been selected for all exceptions that can potentially occur with the selected business process (and any handler process attributes have been specified as necessary), we have finished specifying the exception-related aspects of the contract.

It is straightforward to map contract configuration as described above into a negotiation framework:

- for every substep S in the abstract process being specified, create the issue "how do we achieve substep S?" whose candidate values are all the possible processes suitable for achieving that substep
- for every exception E in the process being defined, create the issue "how do we handle exception E?" whose candidate values are all the possible processes suitable for handling that exception
- for every process attribute A in the abstract process, substeps and exception handlers, create the issue "what is the value for attribute A?" whose candidate values depend on the attribute being considered

The issues involved in designing contingent contracts are thus highly interdependent. The output of one substep in a process, for example, will often have to match the input of the next substep, so the utility of one choice is highly dependent on the other choice. The value of a given exception handler can be dependent upon choices made for the normative steps and for the other exception handlers.

This approach relies on treating contract formation as configuration from a pre-defined design space, but this arguably is realistic for many important real-world domains such as supply chains where the abstract processes and most commonly used alternatives are relatively stable and well-known.

Contributions

This paper presents, as far as we are aware, the first computational contract negotiation approach suited for multiple interdependent issues. We have shown that annealing produces superior social welfare, in the multiple optima search spaces implied by this problem, to the hill-climbing approach that has been the standard in negotiation work to date.

Acknowledgements

This work was supported by funding from the DARPA Control of Agent-Based Systems (CoABS) program, as well as the NSF Computation and Social Systems program.

[1] We are grateful to Benjamin Grosof of the MIT Sloan School of Management for pointing out this tradeoff.

References

(1999). Proceedings of the International Workshop on Knowledge-Based Planning for Coalition Forces. Edinburgh, Scotland.

Bar-Yam, Y. (1997). Dynamics of complex systems. Reading, Mass., Addison-Wesley.

Bernstein, A., M. Klein, et al. (1999). The Process Recombinator: A Tool for Generating New Business Process Ideas. Proceedings of the International Conference on Information Systems (ICIS-99), Charlotte, North Carolina USA.

Ehtamo, H., E. Ketteunen, et al. (2001). "Searching for Joint Gains in Multi-Party Negotiations." European Journal of Operational Research **1**(30): 54-69.

Faratin, P., C. Sierra, et al. (2000). "Using similarity criteria to make negotiation trade-offs." Proceedings Fourth International Conference on MultiAgent Systems. IEEE Comput. Soc.

Fischer, K., J. P. Muller, et al. (1996). Intelligent agents in virtual enterprises. Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'96), Blackpool, UK.

Hägg, S. (1996). A Sentinel Approach to Fault Handling in Multi-Agent Systems. Proceedings of the Second Australian Workshop on Distributed AI, in conjunction with Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia.

Jennings, N. R., K. Sycara, et al. (1998). "A Roadmap of Agent Research and Development." Autonomus Agents and Multi-Agent Systems **1**: 275-306.

Kalos, M. H. and P. A. Whitlock (1986). Monte Carlo methods. New York, J. Wiley & Sons.

Klein, M. and C. Dellarocas (1999). Exception Handling in Agent Systems. Proceedings of the Third International Conference on AUTONOMOUS AGENTS (Agents '99), Seattle, Washington.

Klein, M. and C. Dellarocas (2000). "A Knowledge-Based Approach to Handling Exceptions in Workflow Systems." Journal of Computer-Supported Collaborative Work. Special Issue on Adaptive Workflow Systems. **9**(3/4).

Klein, M., J. A. Rodriguez-Aguilar, et al. (2000). Using Domain-Independent Exception Handling Services to Enable Robust Open Multi-Agent Systems: The Case of Agent Death. Cambridge MA USA, Massachusetts Institute of Technology.

Malone, T. W., K. Crowston, et al. (1999). "Tools for inventing organizations: Toward a handbook of organizational processes." Management Science **45**(3): 425-443.

Raiffa, H. (1982). The art and science of negotiation. Cambridge, Mass., Belknap Press of Harvard University Press.

Sterman, J. D. (1994). Learning in and about complex systems. Cambridge, Mass., Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Tsvetovatyy, M. B., M. Gini, et al. (1997). "MAGMA: An agent-based virtual marketplace for electronic commerce." Applied Artificial Intelligence **11**(6): 501-524.

Wooldridge, M., N. R. Jennings, et al. (1999). A Methodology for Agent-Oriented Analysis and Design. Proceedings of the Third Annual Conference on Autonomous Agents (AA-99), Seattle WA USA, ACM Press.

Youssefmir, M. and B. Huberman (1995). Resource contention in multi-agent systems. First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, CA, USA, AAAI Press.