Supply Chain Management World

A Benchmark Environment for Situated Negotiations

Yasser Mohammad[1,4](✉), Enrique Areyan Viqueira[3], Nahum Alvarez Ayerza[1], Amy Greenwald[3], Shinji Nakadai[1,2], and Satoshi Morinaga[1,2]

[1] AIST, Tokyo, Japan
y.mohammad@aist.go.jp
[2] NEC Inc., Tokyo, Japan
[3] Brown University, Providence, USA
[4] Assiut University, Asyut, Egypt

Abstract. In the very near future, we anticipate that more and more artificially intelligent agents will be deployed to represent individuals and institutions. Automated negotiation environments are a mechanism by which to coordinate the behavior of such agents. Most existing work on automated negotiation assumes a context that is predefined, and hence, static. This paper focuses on the dynamic case, which we call *situated negotiation*, where agents need to decide not only how to negotiate, but with whom, and about what. We describe a common benchmark simulation environment for evaluating situated negotiation strategies, and evaluate several baseline strategies in the proposed environment.

1 Introduction

Negotiation is a process by which **self-interested** parties aim to reach an agreement. Self-interestedness implies a partial ordering over different possible outcomes, which in turn implies the existence of a continuous **utility function** that assigns a real value to all outcomes [6].

In **automated negotiation**, one or more of the negotiating parties is an artificially intelligent (AI) agent. Interest in automated negotiation is increasing, because of the growing use of AI to automate business operations [17], and the understanding that these agents must be capable of reaching agreements, if the businesses they represent are to be successful.

We refer to an instance of automated negotiation as a **negotiation thread**. A negotiation thread involves at least two agents (often called negotiators), each with its own utility function and **strategy**, negotiating about some **agenda**. A negotiation strategy is a mapping from the state of the negotiation, as understood by an agent, to the actions allowed by the negotiation **protocol** (sometimes called a **mechanism**). A negotiation agenda is the space of issues under consideration: e.g., in the context of supply chain management, the possible prices, quantities, and delivery dates.

Traditionally, automated negotiation research has focused on **context-free** negotiations. Such a negotiation is characterized by a single thread, in which agents endowed with static utility functions negotiate about a fixed agenda [3]. Here, the key research questions usually pertain to the design of an effective negotiation strategy [7].

To apply automated negotiation technology in realistic business settings, however, agents will need to decide not only *how* to negotiate, but with *whom* and about *what*. Furthermore, when an agent is simultaneously negotiating with multiple other agents, their utility in one negotiation is necessarily dynamic, as it depends on the success or failure of other negotiations [2]. We call such scenarios **situated negotiations** to emphasize the role of the context, or the *situation*, in the negotiation process.

In many settings, it may not be optimal, or even possible, to decompose a situated negotiation neatly. For example, consider an agent A that is negotiating with another agent B. If A receives an offer from a third agent C, it should be reluctant to accept any worse offer from B. In general, the availability of a third agent C as a potential or actual negotiation partner will affect the offers that A places and is willing to accept from B. Dividing a situated negotiation into a set of independent negotiations may lead to suboptimal behavior in all of them.

Different aspects of situated negotiations have been studied in the literature under different names, including *negotiation with outside options* [9], *one-to-many negotiation* [11], *negotiation in distributed environments* [10], *concurrent negotiations* [20], and was applied to complex multiagent resource allocation [2], distributed task allocation [8], cloud computing [2], and smart grids [1].

The first contribution of this paper is to present a common benchmark simulation environment called the Supply Chain Management (SCM) world that is rich enough to help illuminate the challenges faced by situated negotiators, while at the same time simple enough to focus the research effort on core problems. Availability of similar benchmark problems in other domains has proved useful in stimulating research and generating new ideas. Examples include the Face Recognition Grand Challenge (FRGC) [16], the Trading Agent Competition [18], The Robot World Cup Initiative (RoboCup), and the Automated Negotiation Agents Competition (ANAC) [7]. The second contribution of this paper is three baseline strategies and their evaluation in the proposed environment.

This paper is organized as follows. Section 2 defines situated negotiations in more detail. Section 3 outlines the objectives we believe a simulation should attain in order to serve as a useful benchmark for current and future research on situated negotiations. Section 4 describes the proposed benchmark problem that was designed to achieve these objectives. Section 5 describes the annual automated agent negotiation competition (ANAC) 2019 supply chain management league (SCML), an instantiation of these ideas. Section 6 introduces three strategies for this problem, and Sect. 7 evaluates the proposed strategies.

2 Situated Negotiations

Problems in automatic negotiation are usually studied without regard to the environment in which the negotiation takes place. From an engineering perspective, this abstraction is justifiable; it can make an otherwise intractable problem tractable. For example, external pressures to reach agreement quickly can be modeled by a negotiation *deadline*, an exponential discount factor on a *utility function*, as part of the *opponent model*, or in a *reservation value* (i.e., the value for failure to reach agreement). But in many negotiation scenarios, it is not so simple to encode the effect of the environment on the negotiation. Informally, the environment creates what we call **situated** negotiations. An agent is engaged in a situated negotiation if the utility function it uses to guide its negotiation is dynamic, and varies with the context in which the negotiation is situated.

A primary example of a situated negotiation is a negotiation under uncertainty. For example, when an agent is not endowed with perfect knowledge of the utility function of the entity it represents, and consequently engages in *preference elicitation* during the negotiation [14] to refine its estimate of its utility function, its current estimate is, in general, situation-dependent.

An agent's utility function is also situation-dependent when it is negotiating in the presence of an *outside option*, i.e., a **substitute**, whose value is either unknown or subject to change. For example, if an agent is negotiating about the price of a plane ticket from Tokyo to California, and in the midst of the negotiation there is an earthquake in Tokyo, the agent's utility function—specifically, its reservation value—may suddenly need to be updated.

An important type of situated negotiation is an **embedded** negotiation. In such a negotiation, an agent's utility function heavily depends on contextual information in that it depends on the collective outcome of multiple negotiations. We call such a utility function **global**. A key task of the agent, then, is to figure out a way to decompose this global utility function into **local** utility functions to be farmed out to the separate negotiation threads. This task is known to be notoriously difficult for autonomous bidders in simultaneous and sequential auctions [5], a special case of many-to-one automated negotiation in which the "one's" (i.e., the auctioneer's) strategy is public, but can be done effectively when integrated with an appropriate bidding strategy [19].

For example, imagine an agent that engages in two *concurrent* negotiations on behalf of someone planning to attend the Tokyo Olympics—one about plane tickets and the other about hotel reservations. The agent's global utility function may ascribe non-zero value only to both travel goods together, implying that the goods are **complements**. Regardless of how this global utility function is decomposed into local utility functions and then farmed out to the two separate negotiation threads, the negotiations are embedded because the conclusion of either would impact the agent's utility function in the other.

The matching market in the U.S. Navy detailing system, which allocates sailors to job vacancies, is an example of an embedded negotiation that marries concurrent negotiations with outside options [9]. In this system, vacancies are published and sailors apply to fill them. Commanders then choose among the applicants via concurrent bilateral negotiations. (Likewise, one can imagine a sequential version in which negotiations are conducted consecutively instead of concurrently, and where the utility function of each subsequent negotiation is affected by past outcomes and predictions about future outcomes.) Li *et al.* [9]

argue that relying on fixed reservation values in each negotiation thread for the duration of the concurrent negotiations is sub-optimal. On the contrary, the reservation value (and hence utility function) in one thread must be updated based on how negotiations unfold in the others.

What these scenarios have in common is that factors external to a negotiation thread itself affect aspects of that negotiation, which entail changes to the utility function. These scenarios are called **situated negotiations** in this paper, and are characterized by *dynamic utility functions that emerge endogenously during possibly concurrent and/or possibly consecutive negotiations*.

3 Design Objectives

The goal of this work is to advance the state-of-the-art in situated negotiation. To achieve this goal, we propose that researchers *benchmark* their progress using a *common* simulation environment. The primary advantage of a common environment is that it facilitates the comparison of agent negotiation strategies. The alternative would involve the arduous task of reimplementing strategies across domains. Moreover, when multiple research teams develop competing approaches, running them all on a common benchmark environment more closely resembles real-world negotiations among disparate parties.

We believe that any common benchmark environment that is intended to further research in autonomous agents and multi-agent systems (AAMAS) should satisfy three design objectives. First, it should model a real-world scenario, thereby increasing its relevance, and enabling researchers to jump start the (strategic) design process using existing intuitions. Second, it should be easy for researchers to run experiments to compare different mechanisms, different agent strategies/designs within a given mechanism, etc. Finally, it should support a canonical design and implementation, to facilitate collaboration among researchers and reproducibility of results.

For the special case of situated negotiation, the environment should model a negotiation scenario that involves one or more of the sub-problems depicted in Sect. 2; and if the scenario involves more than of these sub-problems, it should be relatively straightforward to isolate and study specific ones.

4 The SCM World: A Common Benchmark Environment

A **supply chain** is a sequence of processes by which raw materials are converted into finished goods. A supply chain is usually managed by multiple independent entities, whose coordination is called **supply chain management** (SCM). SCM exemplifies situated negotiation. The SCM world was built on top of an open-source automated negotiation platform called NegMAS [13] to serve as a common benchmark environment for the study of situated negotiation.

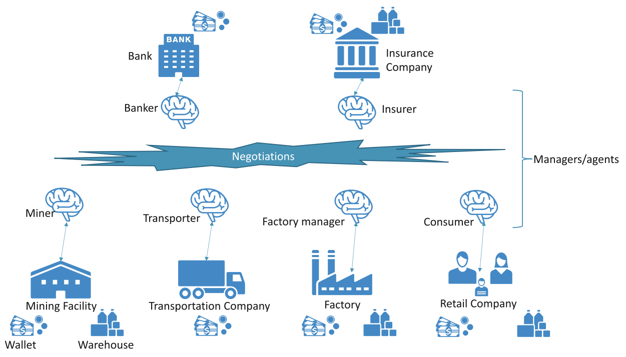

Fig. 1. The main entities and their managers (agents) in the SCM world simulation.

Entities. SCM consists of six types of entities and their corresponding **managers** (See Fig. 1): factories, mining facilities, retail companies, transportation companies, banks, and insurance companies. The relationship between these entities and their managers is one-to-one. All entities have accompanying **wallets** that store their cash. Moreover, factories, mining facilities, retail companies, and insurance companies have accompanying storage **warehouses**. In more detail:

Factories convert raw materials and intermediate products into intermediate and final products by running their manufacturing processes for some time, assuming all inputs, enough funds, and enough time are available to run the processes. They are managed by **factory managers**.

Mining facilities are capable of mining raw materials, which they do to satisfy their negotiated contracts. They are managed by **miners** that act only as sellers in the SCM world.

Retail companies are interested in consuming a subset of the final products to satisfy some predefined consumption schedule. They are managed by **consumers** that act only as buyers.

Transportation companies transport materials between warehouses. They are managed by **transporters** that represent service providers.

Banks provide loans to potential buyers.

Insurance companies insure managers against **breaches of contract** committed by other managers (e.g., failure of a seller to deliver promised products on time, insufficient funds in the buyer's wallet at the time of delivery, transportation delay by a transporter, etc).

Agents. In the SCM world, agents represent managers. The goal of each agent is to accrue as much profit as possible.

All trade in the SCM world is conducted through negotiations. Negotiations can be bilateral or multilateral, and can use any negotiation protocol—synchronous or asynchronous—to reach an agreement. As a special case, some (or all) agreements may be arrived at using auction protocols, allowing for direct comparison between the auction mechanisms and other negotiation protocols.

When an agreement is signed, it is converted into a contract. When a contract comes due, the simulator attempts to execute it. For a contract between a buyer and a seller, it moves the agreed upon quantity of that product from the seller's inventory to the buyer's, and the agreed upon price from the buyer's wallet to the seller's. For a transportation contract, it moves the products from the source to the destination (after any agreed upon transportation delay), and moves the transportation cost to the wallet of the transporter. If any of these executions fail, a breach of contract can occur. Breaches can also occur if either party decides not to honor the contract. In cases of potential breaches, the simulator may offer the agents involved an opportunity to renegotiate.

To find negotiation partners, agents may **request-a-negotiation** with potential trading partners directly, or publish their interest in negotiating on a public **bulletin board** that lists **call-for-proposals** (CFPs). Each such CFP specifies the publisher and the proposed negotiation issues. Interested agents then respond to the publisher with a request to negotiate. Requesting such a negotiation implies acceptance of the negotiation agenda.

Simulation. Before the start of the simulation, an initial balance is deposited in each agent's wallet, and catalog prices are posted for all products. In addition, each agent is assigned a private profile, which characterizes its production capabilities and/or its consumption preferences. Each SCM world simulation runs for multiple (say, 100) steps. During each step:

1. Agents make any outstanding loan payments, all contracts that come due are executed, and any breaches that arise are handled.
2. Agents then engage in negotiations for multiple steps (say, 10). During this time, they are also free to read the bulletin board, post CFPs, and respond to CFPs.
3. Finally, all production lines in all factories advance one time step, meaning required inputs are removed from inventory, generated outputs are stored in inventory, and production costs are subtracted from the factories' wallets. Moreover, transportation advancement is simulated.

Utility Functions. The SCM world does not endow agents with utility functions. On the contrary, all utility functions are endogenous, meaning they are engendered by the simulator's dynamics and agents' interactions. Endogenous utility functions that arise as the market evolves are a distinguishing feature of situated negotiations. In the SCM world, a major determiner of an agent's profits is its ability to position itself well in the market via successful negotiations, which in turn depends on the utility functions it uses to guide its negotiations.

Desiderata. The SCM world satisfies the generic AAMAS design objectives outlined in Sect. 3, as well as the ones that are specific to situated negotiations. First, it is possible to instantiate all the example situated negotiation scenarios described in Sect. 2. For example, by disabling banks, insurance companies, transportation companies, and factory managers, so that only miners and consumers negotiate about the price of a ready-made product to be delivered at a fixed time, it is possible to model *negotiation with outside options* [9], where the outside options are other agents trading the same product. Second, the environment is a simulation of a real-world marketplace in which business intuitions can be applied to generate and test automated negotiation strategies. Finally, a canonical implementation of the SCM world simulation is available as an open source library [13], to enhance reproducibility and provide a common platform to advance the state-of-the-art in situated negotiations.

5 ANAC 2019 SCM League

One way to expedite the widespread use of a common benchmark environment throughout a research community is to sponsor a competition in the environment. To this end, in 2019, the SCM league (SCML), based on the SCM world design was organized as part of the Automated Negotiation Agents Competition (ANAC) [7], held at the International Joint Conference on AI.

SCML '19 is one relatively simple instantiation of the SCM world. The simplifications were design choices aimed at reducing any complexity in the SCM world that did not immediately pertain to situated negotiations, so as to provide a relatively straightforward setting in which to develop innovative negotiation strategies, while at the same time ensuring a sufficient level of activity. Specifically, in SCML '19, activity was measured via **business size**, defined as the total monetary value of all successfully executed contracts. The design was then optimized in attempt to avoid **market blockage**, namely a business size of zero.

SCML '19 ignored logistics (i.e. no transportation companies were simulated). Instead, all products were transported between all entities free of charge, after a predefined constant delay (which was set to zero). In addition, warehouse capacity was infinite. The bank was disabled and all agents were initialized with large balances to avoid the need for loans. These simplifications, which sidestepped cash flow, storage limitations, and logistic complications, were intended to lower the barrier to entry in the initial year of the competition.

The insurance company was not removed from the simulation. Agents interacted with the insurance company via the ultimatum mechanism: i.e., the latter made a single final offer of an insurance policy, which the agent could accept or reject, without any possibility of haggling. All other agreements were reached via bilateral negotiations, using the **alternating offers protocol** [3], in which agents exchange offers and counteroffers.

The production graph used in SCML '19 was organized as a single chain, with a single raw material, a single finished good, and a set of intermediate products. To manufacture each product, there was but a single process that consumed one item of the product just before it in the chain.

The SCML development team designed the miners, the consumers, and the insurance company. The job of the participants was to develop a **factory manager**. The development team also provided a baseline factory manager, whose strategy is described in Sect. 6. This agent was an eager business partner, and thus participated in the competition to ensure sufficiently many trading opportunities, thereby increasing the business size metric.

The behavior of the built-in agents make SCML '19 a **pull economy**, meaning it is demand driven. Proactive consumers drive demand by posting buy CFPs. Baseline factory manager agents react by responding to the consumers' buy CFPs (offering to sell), and then post their own buy CFPs further down the chain. Miners at the far end of the chain are similarly reactive.

Consumers. Consumers in SCML '19 are proactive. They post buy CFPs, which drive the supply chain. The negotiation agendas that characterize these CFPs reflect the consumers' utility functions, which in turn are characterized by consumption schedules that usually cannot be fulfilled via a single factory during a single time step, but instead require multiple of one or the other or both, and hence create a situated negotiation scenario.

A consumer c's utility of consuming a finished good is determined by its profile π_c. This profile includes a predefined consumption schedule S_c that defines, for each step, a preferred quantity to consume, as well as overconsumption and underconsumption penalties, $\hat{O}_c$ and $\hat{U}_c$, respectively. Thus, the utility functions reward consumers who follow their schedules closely, and penalize deviations from them. These assumptions lead to the form of consumers' utility functions shown in Eq. 1.

Given an outcome (u, q, t), denoting unit price, quantity, and execution time, respectively, consumer c's utility is given by

$$U_c\left(u,q,t\right) = \begin{cases} 0, & u<0 \text{ or } q<0 \text{ or } t<0 \\ \alpha_u h_u^{\tau_u,\beta_u}\left(u\right) + \alpha_q h_q^{\tau_q,U,O}\left(q, S\left(t\right)\right), & \text{otherwise} \end{cases} \tag{1}$$

The parameters α_*, where $*$ is the issue name (i.e., $* \in \{u,q\}$), are values in $(0,1)$ drawn from a Dirichlet distribution that varies with the consumer. The parameters $\beta_u, \tau_u, \tau_q, U$, and O, are drawn from a normal distribution that likewise varies with the consumer.

The function $h_u^{\tau_u,\beta_u}$ is monotonic in the unit price, $x \in \mathbf{R}_0^+$: $h_u^{\tau_u,\beta_u}\left(x\right) = -\left({}^{x}/_{\beta_u}\right)^{\tau_u}$. The function $h_q^{\tau_q,U,O}$ takes as input two quantities; the first is specified by the outcome, and the second, by the consumer's schedule at time t. This function has the following form:

$$h_q^{\tau_q,U,O}\left(x,y\right) = \begin{cases} e^{-U\left(\frac{y-x}{y}\right)^{\tau_q}} & x \le y \wedge y \ne 0 \\ e^{-O\left(\frac{x-y}{y}\right)^{\tau_q}} & x \ge y \wedge y \ne 0 \end{cases} \tag{2}$$

With every negotiation opportunity a fresh utility function is created based on the consumer's profile. Consequently, even if a consumer already engaged in a failed negotiated with another agent about an existing CFP, it will behave differently the next time, so their negotiation may as yet succeed.

Miners. Miners in SCML '19 are purely reactive. They wait for buy CFPs for the raw material to be posted, and respond, based on their utility functions, to all whose negotiation agendas are consistent with their mining abilities. Note that miners' utilities are not coupled across negotiations in the same way that consumers' are, because a miner's total profit across negotiations is simply the sum of its profits in its individual negotiations.

A miner m's utility of mining (i.e., generating) any quantity of a raw material is determined by its profile π_m. At a high-level, miners should prefer to mine fewer raw materials, as late as possible, which it should then aim to sell the highest possible prices. However, in an attempt to increase business size, miners preferred to mine more, rather than fewer, raw materials. These assumptions lead to the form of the miners' utility functions, described in Eq. 3, and generated in an analogous way to consumers'.

Given an outcome (u, q, t), denoting unit price, quantity, and execution time, respectively, miner m's utility is given by

$$U_m(u, q, t) = \begin{cases} 0, & u < 0 \text{ or } q < 0 \text{ or } t < 0 \\ \alpha_u g_u(u) + \alpha_q g_q(q) + \alpha_t g_t(t), & \text{otherwise} \end{cases} \tag{3}$$

The parameters α_*, where $*$ is the issue name (i.e., $* \in \{u, q, t\}$), are values in $(0, 1)$ drawn from a Dirichlet distribution that varies with the miner. The parameters τ_* and β_*, where $*$ is again the issue name, are drawn from a normal distribution that likewise varies with the miner. The functions g_* are monotonic in the issue value, $x \in \mathbf{R}_0^+$: $g_*(x) = (x/\beta_*)^{\tau_*}$.

With the goal in mind of optimizing business size, the following design choices were made for SCML '19: Baseline factory managers *always* bought insurance. The insurance premium was relatively cheap (10% of the outcome's total value), and did not increase all that much with breaches, and breach penalties were minimal (2%). These choices effectively prevented market blockage, and favored larger business sizes, as shown in Sect. 7.

6 Strategies

There are inherent difficulties in building a realistic simulation environment. Figuring out how to best trade off time and/or space complexity for realism, for example, can be challenging.

SCM factory managers face multiple challenges, including: (1) strategic placement of CFPs (i.e., proactively initiating negotiation opportunities), (2) reacting to negotiation requests from others, (3) creating utility functions for negotiation threads, (4) negotiation strategies for each thread, (5) inventory control, and (6) production scheduling. An SCM agent strategy encompasses all the heuristics a factory manager uses to address these six challenges.

In this section, we describe three agents strategies we developed for the SCM world, as instantiated in SCML '19. The first was designed as a baseline strategy, upon which participating teams could base their design. This strategy tackled the embedded negotiation aspect of SCML (see Sect. 2), albeit heuristically. The second strategy focuses on procurement, and draws inspiration from the newsvendor model [15], by formulating a discrete optimization problem whose decision variables are the quantity of inputs to buy. The solution to this problem is useful in deciding what buy CFPs to post, and what sell CFPs to respond to. The third strategy tries to find a negotiation agenda—specifically, a price—that is both profitable from its point of view and, at the same time, acceptable to other agents. By working to artificially inflate prices, this strategy aims at altering the trading environment in which the agent is situated to promote itself.

Greedy Factory Manager: A Baseline. The **Greedy Factory Manager** (GFM) was designed to showcase all the components needed to design a factory manager for the SCM world. GFM was also intended to be run in all simulations so that it could ensure sufficient business size, even at the expense of being profitable. GFM's strategy overcontracts, which avoids starving factory managers at earlier levels in the supply chain, but results in many breaches of contract.

The GFM agent employs a reactive-seller, proactive-buyer strategy, much like consumers. It is *reactive* in that it requests negotiations with the publishers of all buy CFPs about the product it produces, as long as it can schedule the desired quantity of the product of interest to be manufactured within the proposed delivery time. When such a negotiation request is accepted, GFM calculates the utility of the potential sell contract as the marginal utility of its outcome, given all existing (buy and sell) contracts, pessimistically assuming that any ongoing negotiations will fail.In this way, the controller decomposes the agent's global utility function, which values the potential outcomes of multiple negotiations, into local utility functions, which values only one outcome. GFM then spawns a negotiator, endowed with the corresponding marginal utility as its utility function. These negotiators embody embedded negotiations, in the sense of Sect. 2.

After a sell contract is signed, the consumption schedules of the necessary inputs are increased accordingly, and GFM then *proactively* places buy CFPs, using the same placement strategy as consumers (Sect. 5). The utility of each potential buy contract is calculated using Eq. 1, taking as the target consumption schedule the production demands of all existing sell contracts. When it accepts another agent's request to negotiate, GFM spawns an internal consumer agent, which in turn spawns a negotiator with this utility function. Similar to consumers, the GFM controller couples these negotiators through utility functions that depend on a shared consumption schedule. Whenever a contract is signed or executes successfully, the utility functions of all ongoing negotiations are updated to reflect a change in production demands and production line occupancy. Likewise, GFM recalculates the marginal utilities of all potential sell contracts whenever a contract is signed or executes successfully.

GFM uses a simple time-based negotiation strategy [4]. At time step t, it offers an outcome with the minimum utility above the so-called **aspiration** level a, which deceases over time as follows: $a(t) = 1 - ({}^{t}/_{T})^{4}$. Here T is the maximum number of negotiation steps, a value specified by the protocol. GFM accepts an offers if its utility is at least the utility of its own ensuing offer at the current aspiration level.

The GFM agent is so called because it uses a greedy heuristic to scheduling production. This heuristic aims to produce outputs as late as possible, in attempt to increase the negotiation power of the agent when buying inputs.

Newsvendor Model Agent. The **Newsvendor Model Agent** (NVM) takes inspiration from the newsvendor model [15], a classic model in operations research used to model the choice of an optimal inventory level for a perishable product (e.g., a newspaper). The NVM agent plans for some finite horizon, assuming that unsold inputs and outputs at the end of that horizon will have no value. Analogous to newsvendor models, an agent implementing this strategy tries not to over- or under-produce during its planning horizon. They do not want to stock too many products, as any excess (whatever does not sell) will go to waste; but they also do not want to stock too few, as any shortage will result in lost sales.

At each time step t, an SCML agent faces (at least) four decisions: the quantity of inputs to buy, y^t_{IN}; the price at which to buy those inputs, x^t_{IN}; the quantity of outputs to sell, y^t_{OUT}; and the price at which to sell those outputs, x^t_{OUT}. The goal of the NVM agent is to maximize its total expected profits over a finite time horizon, in the face of uncertain and non-stationary elastic demand.

The NVM agent models the uncertainty it faces at time step t by a joint distribution $G^t \doteq G^t_{P_{\text{IN}},Q_{\text{IN}},P_{\text{OUT}},Q_{\text{OUT}}}$, where $G^t_{P_{\text{IN}},Q_{\text{IN}},P_{\text{OUT}},Q_{\text{OUT}}}(P^t_{\text{IN}} \leq p^t_{\text{IN}}, Q^t_{\text{IN}} \leq q^t_{\text{IN}}, P^t_{\text{OUT}} \leq p^t_{\text{OUT}}, Q^t_{\text{OUT}} \leq q^t_{\text{OUT}})$ is the cumulative probability that, at time t, q^t_{IN} units of the input IN will be sold at price p^t_{IN} per-unit, and q^t_{OUT} units of the output OUT will be sold at price p^t_{OUT}. We denote by $G_{P^t_{\text{OUT}}}$ (respectively, $G_{Q^t_{\text{OUT}}}, G_{P^t_{\text{IN}}}$, and $G_{Q^t_{\text{IN}}}$) the marginal distribution over output prices (respectively, output quantities, input prices, and input quantities). $G_{P_{\text{IN}},Q_{\text{IN}},P_{\text{OUT}},Q_{\text{OUT}}}$ was estimated by a histogram, which was constructed from data obtained offline, via repeated simulations between one NVM agent and one GFM at each of the other levels in the production chain. For SCML '19, a histogram was a sufficient representation because of the small number of trading quantities entertainined by GFM agents.

Given a fixed time horizon T, a **plan of action** is defined as a collection of tuples $P = \{(x_t, y_t, z_t)\}^T_{t=1}$. This plan completely specifies for each time period $t = 1, \ldots, T$, the number x_t of inputs to buy, y_t of outputs to sell, and z_t of inputs to turn into outputs. A plan is **feasible** if it can be executed, i.e., if at every time step there are enough inputs to be bought, enough outputs to be sold, and enough inputs to be converted into outputs.

More formally, the goal of the NVM agent is to find a feasible plan that maximizes its total expected profits over the time horizon T:

$$\begin{aligned}
\max_{\mathbf{x},\mathbf{y},\mathbf{z}} \; & \mathbb{E}_{Q^t_{\text{IN}},Q^t_{\text{OUT}}}\left[\textstyle\sum_{t=1}^{T} p^t_{\text{OUT}} \min(y_t, Q^t_{\text{OUT}}) - p^t_{\text{IN}} \min(x_t, Q^t_{\text{IN}}) - \textsc{Cost} \cdot z_t\right] \\
\text{s.t.} \quad & z_t \leq \textsc{Capacity} \\
& y_t \leq O_t = \textstyle\sum_{k=1}^{t-1} z_k - y_k \\
& z_t \leq I_t = \textstyle\sum_{k=1}^{t-1} x_k - z_k \\
& x_t, y_t, z_t \geq 0
\end{aligned} \tag{4}$$

All these constraints must hold for all time steps $t \in \{1, \ldots, T\}$, with initial conditions $O_1 = I_1 = y_1 = z_1 = 0$. Variables O_t and I_t are auxiliary variables representing the output, respectively the input, inventory levels at time t. The initial conditions specify that, at the beginning of the planning horizon, the agent has no inputs nor outputs in storage, and hence, cannot produce or sell outputs. Note that these initial conditions can easily be changed; thus, the agent can plan differently given non-zero storage. COST is the agent's private, per-unit production cost, while CAPACITY is the maximum number of inputs that can be converted into outputs during a single time step. The current version of NVM sets $p^t_{\text{IN}} = \mathbb{E}[P^t_{\text{IN}}]$ and $p^t_{\text{OUT}} = \mathbb{E}[P^t_{\text{OUT}}]$.

At each time t, NVM solves for an optimal plan of action.[1] Given this plan, the agent posts a single buy CFP with quantity range $(\max(1, y_1 - \delta_q), y_1 + \delta_q)$, price range between 0 and the expected catalog price $p^t_{\text{IN}} = \mathbb{E}[P^t_{\text{IN}}]$, and time range $(t+\underline{\delta}_t, t+\overline{\delta}_t)$.[2] Additionally, NVM requests negotiations with publishers of sell CFPs. With sufficient (e.g., unlimited) negotiation resources, it can conduct negotiations that are consistent with its optimal plan of action with any agent who is interested in negotiating about anything.

To estimate the utility of a potential buy contract, NVM uses an ad hoc function defined solely in terms of price, namely $u(p) = 1 - p$, which means the agent prefers lower prices, at all possible values of quantity and time. The utility of a potential sell contract is calculated in terms of both price and quantity, as $u(p, q) = e^{(p-1.5)}q$, if $p > 0$ and $-\infty$ otherwise. In other words, NVM prefers to sell many outputs at higher prices, provided the price is not zero. An independent copy of the relevant (buy or sell) utility function is used in all concurrent negotiations.

Like GFM, NVM operates as a reactive seller (requesting negotiations with all publishers of buy CFPs) and uses the built-in aspiration-level negotiator. Unlike GFM, upon receiving a delivery of inputs, it immediately sends the inputs to one of its production lines, where they are scheduled in a FIFO fashion.

Self-Adjustable Heuristic Agent (SAHA). Rather than redesign the various components of an agent (negotiators, utility functions, scheduler, etc.), the **self-adjustable heuristic agent** (SAHA) implements a high-level behavior on top of GFM. Specifically, SAHA imports the aspiration-level negotiator, the utility functions, and the baseline scheduler from GFM. The main focus of SAHA is then on strategic placement of CFPs with the intent of achieving a high profit margin. Moreover, the interaction of multiple SAHA agents, all aiming for higher profit margins, artificially inflates (deflates) the prices of its sell (buy) contracts.

When selling (buying) products, SAHA posts CFPs with progressively higher (lower) prices until the other agents start rejecting their proposals outright. SAHA then decreases (increases) prices until it enters into negotiations again, always seeking to post CFPs with prices near the highest (lowest) observed

[1] Details of the dynamic program we used to efficiently solve (4) for optimal plans are left for a longer version of this paper.

[2] These parameters were manually tuned to $\underline{\delta}_t = 5, \overline{\delta}_t = 15$, and $\delta_q = 5$.

acceptable price. We observed in our experiments that over time, the agent's buying and selling price ranges seem to stabilize. In more detail:

1. SAHA requests a negotiation with the publisher of a buy CFP for its outputs, or it counters with a modified negotiation agenda in its desired price range.
2. SAHA requests a negotiation with the publisher of a sell CFP for its inputs, up to a stock limit, again within its desired price range.
3. SAHA posts sell (buy) CFPs for all the outputs in inventory (inputs needed).
4. If SAHA enters into a negotiation and it fails, it reverts the desired price range for that product to its previous value.

The SAHA agent maintains a set of records based on past and current CFPs containing each product's minimum and maximum prices. Whenever a new CFP is posted, or the agent reaches an agreement, the records are updated with the new information, and the product price ranges for that product are recalculated, adding or subtracting an increment as follows: Buying Range $= [0, \mathrm{CP} + \Delta_1 \mathrm{CP}]$, where CP is the catalog price for the product and Δ_1 is the buy increment; ans Selling Range $= [M - \Delta_1 M, M + \Delta_2 M]$, where M is the maximum price observed for that product, Δ_1 is the buy increment and Δ_2 is the sell increment. The agent will create a set of 20 prices for the negotiation between those ranges in order to avoid a negotiation fail due to a timeout.

We tuned the agent's behavior by optimizing three hyperparameters: the minimum elapsed time until entering a negotiation; the maximum inventory level at any time; and the buy and sell increments used to create price ranges.

7 Experiments

This section describes a series of experiments that we ran to evaluate the three aforementioned agent strategies for managing a factory in the SCM world. The following round-robin design was employed. A set of N random world configurations were generated. For each configuration, two sets of factories, each of cardinality F, were selected. For each of the three possible combinations of agents (i.e., GFM vs. NVM, GFM vs. SAHA, and SAHA vs. NVM), two simulations were conducted, one with each of the two sets of factories managed by each of the two competing agents. In total, each of the N world configurations was simulated $3 \times 2 = 6$ times. An agent's score in a single simulation is the profit it

Table 1. Results of a Comparative Study using the SCML '19 settings.

Strategy	Median	Mean (±Std.)	Kolmogorov-Smirov Test Statistic (p-value)		
			NVM	SAHA	GFM
NVM	**0.315**	0.221 (±0.636)	–	0.213 (0.046)	0.625 (1.359×10^{-14})
SAHA	0.168	**0.401** (±0.628)		–	0.588 (6.059×10^{-13})
GFM	−0.055	−0.107 (±0.154)			–

achieves as a fraction of its initial wallet balance (which was set to 1000 for these experiments, for all agents). All agents' scores in all simulations were collected and analyzed, as described in the following subsections.

In the first experiment, the settings used in the ANAC SCML '19 standard track league (Sect. 5) were used [12]. Twenty different world configurations were employed, with one factory per strategy per simulation ($F = 1$), leading to 120 world simulations and 240 scores per agent. A summary of the results of this experiment is presented in Table 1. SAHA achieved the highest average score while NVM achieved the highest median score. The difference in score distributions between SAHA and NVM was not statistically significant, according to a factorial two-sided Kolmogorov–Smirnov test with a Bonferroni multiple-comparisons correction ($t = 0.213, p = 0.046 > 0.05/3$). Both agents achieved higher scores than the baseline GFM agent ($p < 1.4 \times 10^{-14}$ for NVM, and $p < 6.1 \times 10^{-13}$ for SAHA).

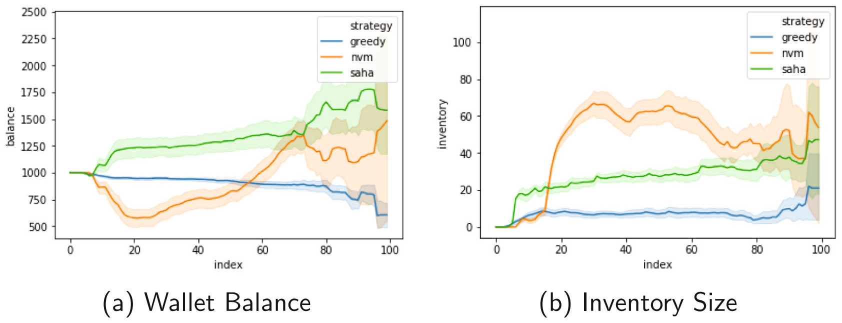

Fig. 2. Evolution of wallet balance and inventory size over time.

Figure 2a shows the evolution of the three agents' wallet balances and inventory sizes over time. NVM's evolving wallet balances and inventory sizes accurately reflects its strategy: its wallet balance initially decreases while its inventory size increases, as it accrues inputs to manufacture into outputs; its wallet balance then begins to recover (around time step 20), when it starts to do more selling than buying. SAHA, in contrast, tries to create favorable market conditions from the beginning, and achieves a nearly monotonic increase in both its wallet balance and its inventory size. By the end of the simulations, NVM and SAHA tend to achieve similar wallet balances, and similar inventory sizes, thought

Table 2. Effect of the insurance company on the market.

Condition	Negotiations	Agreements	Contracts	Executed	Business size
Without Insurer	5867.5	3559	**1068.5**	397.5	2379.25
With Insurer	5299	**3781**	983	**472.5**	**3927.58**

NVM, because of its lookahead, does a better job of unloading excess inventory at the very end than SAHA. GFM's balance, on the other hand, *decreases* almost monotonically, due to its tendency to overcontract.

The insurance company was introduced into the SCM world to increase business size. To assess whether it was successful in achieving this goal, we reran the experimental design used in the comparative study, but without the insurance company. The results of this experiment are presented in Tables 2 and 3.

In Table 2, we see that although there were more agreements reached in the presence of the insurance company, there was also an 8% reduction in the number of contracts signed, likely because of the cost of insurance. (GFM and SAHA always buy insurance; NVM never does.) Nevertheless, there was also a 16% reduction in the number of breached contracts, because breaches at lower levels of the production chain did not automatically cause breaches at higher levels. This in turn led to a 65% increase in business size, which demonstrates that the insurance company did provide the benefits for which it was designed.

Table 3. Results of the Comparative Study without the insurance company.

Strategy	Median	Mean (±Std.)	Kolmogorov-Smirov Test Statistic (p-value)		
			NVM	SAHA	GFM
NVM	0.077	−0.044 (±0.524)	–	−0.37 (1.22×10^{-24})	0.448 (7.612×10^{-36})
SAHA	**0.257**	**0.421** (±0.477)		–	0.603 (1.247×10^{-64})
GFM	−0.050	−0.071 (±0.112)			–

We now briefly investigate how heavily each of the three agents relied on the insurance company (Table 3). SAHA appears to be least dependent, with its median profit increasing by 16.8%, and with almost no change in its mean profit. This robustness allowed SAHA to outperform both NVM and GFM, and the difference is statistically significant after a Bonferroni multiple-comparisons correction ($p < 1.3 \times 10^{-64}$ for NVM) and ($p < 7.7 \times 10^{-36}$ for GFM).

Our experimental results suggest that NVM and SAHA are more successful factory managers in the SCM world than the baseline GFM. NVM's performance has lower variance in the presence of the insurance company, while SAHA has better average performance and is especially robust to the omission of the insurance company. It remains to be seen, however, whether GFM might be more competitive if it were not parameterized to maximize business size.

Conclusion

This paper described a common benchmark simulation environment, which is available as an open-source library, and can thus serve as a sandbox to advance research on situated negotiations. We presented a set of desiderata we believe this kind of simulator should satisfy in order to be a useful model of real-world negotiation scenarios, and argued that the proposed benchmark satisfies them. We then described the SCM world, as well as SCML, an automated negotiation competition, that was run in 2019 using this benchmark. A baseline strategy for this competition, along with two other competitive entrants, were also described and evaluated. In future renditions of SCML, we expect to alter the SCM world simulation in light of the lessons learned in 2019. Ultimately, our goal is to design and build environments that isolate various aspects of situated negotiations to promote the development of automated negotiation strategies.

References

1. Adabi, S., Movaghar, A., Rahmani, A.M., Beigy, H., Dastmalchy-Tabrizi, H.: A new fuzzy negotiation protocol for grid resource allocation. J. Network Comput. Appl. **37**, 89–126 (2014)
2. An, B., Lesser, V., Irwin, D., Zink, M.: Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In: Proceedings of the 9th AAMAS, pp. 981–988 (2010)
3. Aydoğan, R., Festen, D., Hindriks, K.V., Jonker, C.M.: Alternating offers protocols for multilateral negotiation. In: Fujita, K., et al. (eds.) Modern Approaches to Agent-based Complex Automated Negotiation. SCI, vol. 674, pp. 153–167. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51563-2_10
4. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robot. Auton. Syst. **24**(3–4), 159–182 (1998)
5. Greenwald, A., Boyan, J.: Bidding algorithms for simultaneous auctions. In: Proceedings of the 3rd ACM Conference on Electronic Commerce, pp. 115–124 (2001)
6. Jaffray, J.Y.: Existence of a continuous utility function: an elementary proof. Econometrica **43**(5/6), 981–983 (1975)
7. Jonker, C.M., Aydogan, R., Baarslag, T., Fujita, K., Ito, T., Hindriks, K.V.: Automated negotiating agents competition (ANAC). In: AAAI, pp. 5070–5072 (2017)
8. Krainin, M., An, B., Lesser, V.: An application of automated negotiation to distributed task allocation. In: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 138–145 (2007)
9. Li, C., Giampapa, J., Sycara, K.: Bilateral negotiation decisions with uncertain dynamic outside options. IEEE Trans. Syst. Man. Cybern. Part C (Appl. Rev.) **36**(1), 31–44 (2006)
10. Li, M., Vo, Q.B., Kowalczyk, R., Ossowski, S., Kersten, G.: Automated negotiation in open and distributed environments. Expert Syst. Appl. **40**(15), 6195–6212 (2013)
11. Mansour, K., Kowalczyk, R.: A meta-strategy for coordinating of one-to-many negotiation over multiple issues. In: Wang, Y., Li, T. (eds.) Foundations of Intelligent Systems, pp. 343–353. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25664-6_40
12. Mohammad, Y., Fujita, K., Greenwald, A., Klein, M., Morinaga, S., Nakadai, S.: ANAC 2019 SCML (2019). http://tiny.cc/f8sv9y
13. Mohammad, Y., Greenwald, A., Nakadai, S.: Negmas: a platform for situated negotiations. In: Twelfth International Workshop on Agent-Based Complex Automated Negotiations (ACAN2019) in Conjunction with IJCAI (2019)
14. Mohammad, Y., Nakadai, S.: Optimal value of information based elicitation during negotiation. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019, pp. 242–250. International Foundation for Autonomous Agents and Multiagent Systems (2019)

15. Petruzzi, N.C., Dada, M.: Pricing and the newsvendor problem: a review with extensions. Oper. Res. **47**(2), 183–194 (1999)
16. Phillips, P.J., Flynn, P.J., Scruggs, T., Bowyer, K.W., Worek, W.: Preliminary face recognition grand challenge results. In: 7th International Conference on Automatic Face and Gesture Recognition (FGR06), pp. 15–24. IEEE (2006)
17. PRNewswire: digital process automation market by component, business function, deployment type, organization size, industry vertical and region - global forecast to 2023 (2019). http://tiny.cc/573o9y
18. Wellman, M.P., Greenwald, A., Stone, P., Wurman, P.R.: The 2001 trading agent competition. Electron. Markets **13**(1), 4–12 (2003)
19. Wellman, M.P., Sodomka, E., Greenwald, A.: Self-confirming price-prediction strategies for simultaneous one-shot auctions. Games Econ. Behav. **102**, 339–372 (2017)
20. Williams, C., Robu, V., Gerding, E., Jennings, N.R.: Negotiating concurrently with unknown opponents in complex, real-time domains. Front. Artif. Intell. Appl. **242**, 834–839 (2012)