# Search Algorithms for Automated Negotiation in Large Domains

Thimjo Koça[1*], Dave de Jonge[2] and Tim Baarslag[1]

[1]Intelligent & Autonomous Systems Group, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands.
[2]Artificial Intelligence Research Institute, IIIA-CSIC, Bellaterra, Spain.

*Corresponding author(s). E-mail(s): thimjo.koca@cwi.nl; Contributing authors: davedejonge@iiia.csic.es; T.Baarslag@cwi.nl;

**Abstract**

This work presents several new and efficient algorithms that can be used by negotiating agents to explore very large domains. The proposed algorithms can search for bids close to a utility target or above a utility threshold, and for win-win outcomes. While doing so, these algorithms strike a careful balance between being rapid, accurate, diverse, and scalable, allowing agents to explore spaces with as many as $\mathbf{10^{250}}$ possible outcomes on very run-of-the-mill hardware. We show that our methods can be used to respond to the most common search queries employed by **87%** of all agents from the Automated Negotiating Agents Competition between **2010** and **2021**. Furthermore, we integrate our techniques into negotiation platform GeniusWeb in order to enable existing state-of-the-art agents (and future agents) to handle very large domains.

**Keywords:** automated negotiation, large domain, search algorithm

## 1 Introduction

As more and more processes and information are being digitized, the duration and complexity of business processes can be reduced in new ways. Automated negotiation is a promising example of such a technology that can bring benefits to various fields, including procurement [1], supply chain management [2], and resource allocation [3].

Such negotiations can take place over many issues at the same time; for instance, in the pharmaceutical sector it is common for a company to procure hundreds of products from other vendors. For example, suppose Bob is interested to procure 100 pharmaceutical products from Sally – each characterized by a unit price and a quantity. Even if there are just ten possible unit prices and quantities to choose from per product, the outcome space of all potential bids is already enormous with $100^{100}$ possibilities.

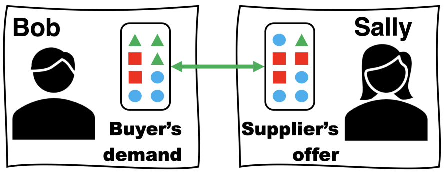

**Fig. 1** In a negotiation domain with 100 products, it is easily possible to have an outcome space of $100^{100}$ possible deals.

When Bob proposes a new offer to Sally, he will typically define a number of criteria that his next offer must fulfill (e.g. targeting a utility goal, guaranteeing certain utility bounds, or conforming to trade-offs between his own preferences and Sally's). Bob then faces the problem of searching an enormous outcome space for a bid that best fits these criteria. Moreover, Bob and Sally keep their preferences private, so to increase the chances in achieving an agreement, he needs to: (a) exchange a high number of offers with Sally; (b) propose offers that, over time, are qualitatively as diverse as possible, i.e. sample the outcome space broadly to try and chance upon an offer to Sally's liking. Hence, Bob needs a *scalable* way to search the huge outcome space in a manner that is *timely*, *accurate*, and *diverse*.

However, it is a challenging task to design a search method that satisfies these four properties. First, the search process for a satisfactory bid typically translates to a non-tractable combinatorial problem that is impossible to solve in an exact or even approximate manner. Second, there are inescapable trade-offs between the desirable properties: for instance, searching through exhaustive enumeration may be highly accurate and diverse, but will not be scalable.

Currently, search mechanisms proposed by state-of-the-art agents and automated negotiation platforms perform poorly in very large outcome spaces (as we will see in Section 5), because they assume that either: (a) the outcome space is small enough to be enumerated and explored rapidly [4, 5], which does not scale; (b) the outcome space be explored effectively by random sampling [6–8], which is not true in very large spaces (see Section 5.3.2); or (c) search goals can be defined deterministically for each individual negotiation issue in isolation [9], which leads to poor accuracy in finite domains and narrow exploration of the outcome space.

In this work, we propose several algorithms that can explore outcome spaces with as many as $10^{250}$ possible outcomes when the user's preferences are expressed by the widely-used additive utility function (i.e. with no issue interdependencies) [10–14]. In particular, for agents that seek a certain level of utility, we propose BIDS (**B**idding using **D**iversified **S**earch): an algorithm that employs a dynamic-programming approach to exploit the additive structure of the utility function. For agents that seek offers lying within a utility interval, we propose Sampling-BIDS: an algorithm that build upon BIDS to sample the outcome space rapidly. Finally, for agents that aim for preference trade-offs with their opponent, we propose IPS (**I**terative **P**areto **S**earch) which combines partial solutions iteratively to construct offers.

We show that our algorithms are *accurate*, i.e. they can identify approximate solutions with arbitrary error bounds to the associated search problems. Their search is also *diverse* by exploring the outcome space broadly, i.e. by offering a wide range of options to the opponent. Furthermore, by surveying the search queries of participants of the Automated Negotiating Agents Competition (ANAC) we show that our methods are *generally applicable*: our algorithms can serve the three most common search queries employed by 87% of ANAC agents — the utility-lookup query, the utility-sampling query, and the trade-off query. Finally, we integrate our algorithms into the negotiation platform GeniusWeb and show that state-of-the-art agents can negotiate over domains that are 25 times larger than with their original search algorithms.

## 2 Problem Setting

We propose algorithms that tackle three search queries that are commonly used by negotiating agents in a manner that is scalable, rapid, accurate, and provides diversity. To do so, we need to first formalize each of the queries and discuss the associated challenges.

### 2.1 Negotiation Model

Agents in our setting negotiate over a finite set of issues $\mathcal{I} = \{1, \ldots, n\}$ and each issue $i \in \mathcal{I}$ has an associated finite set of values $V_i$. All possible combinations of values form the *outcome space*, which is denoted by $\Omega = \prod_{i \in \mathcal{I}} V_i$. For instance, in a scenario where the dates of 100 different shipments $\mathcal{I}$ are negotiated, the value set $V_i$ for each shipment $i$ would span 365 possible delivery dates, resulting in an outcome space with $365^{100}$ possibilities. Each element $\omega \in \Omega$ is called a negotiation *outcome* and whenever it is convenient we will denote the component of $\omega$ corresponding to issue $i \in \mathcal{I}$ by $\omega_i \in V_i$.

The private preferences of each party over $\Omega$ are expressed through a utility function $u : \Omega \to [0, 1]$. We focus in this work on utility functions that are additive with respect to the utilities of each issue:

$$u(\omega) = \sum_{i \in \mathcal{I}} \lambda_i \cdot u_i(\omega_i)$$

where $\lambda_i \geq 0 \wedge \sum_{i \in \mathcal{I}} \lambda_i = 1$ are the weights defined for each issue, and $u_i : V_i \to [0, 1], \forall i \in \mathcal{I}$ are utility functions defined over each individual issue. Additive utility functions are widely used [10–14] and, as we will see in Section 4, their structure allows for some scalable, rapid, accurate, and diverse search of the outcome space. Note that additive utility functions assume no dependencies between individual issues, but they can still encode rather complex preference structures as each $u_i$ can be defined arbitrarily. In the rest of this paper $u$ denotes an agent's own utility function, while $u'$ denotes the utility function of the opponent.

Given the structure of the domain, a negotiation protocol (e.g. the Alternating Offers Protocol (AOP) [15]) regulates how the agents exchange offers during the negotiation. We consider protocols that allow in each round the communication of one or several bids, i.e. possible outcome(s) $\omega$ to agree upon, or a special message — for instance, a message that indicates acceptance of the opponent's latest offer, or a message informing a walk-away.

## 2.2 Typical Search Queries

There are many negotiating agents in literature, each with their own negotiation strategy and learning methodologies [18, 5, 12, 24, 19, 11]. However, despite the richness of possible negotiating strategies, when an agent decides on a bid to propose next, it generally complies to the following pattern: it first sets some criteria that the proposed bid needs to satisfy – for instance an appropriate utility target, a suitable utility interval, or a particular trade-off with the opponent's interests — and subsequently tries to identify the most appropriate bids that meet one of these three search queries (illustrated in Fig. 2).

The utility-lookup is the simplest among the queries. Agents define in each round a utility target $u_t \in \mathbb{R}$ and search for bid(s) with utility as close as possible to the target utility (illustrated by the blue line in Fig. 2):

$$\underset{\omega \in \Omega}{\operatorname{argmin}} \quad |u(\omega) - u_t| \tag{1}$$

The target can be determined through a time-based strategy (e.g. Agent K [16]), a behavior-based strategy (e.g. Nice-Tic-For-Tac Agent [17]), or through some other criteria (e.g. through a resource-based tactic [18]).

The second query is the sampling-utility query within predefined utility bounds. In each round, agents search for one or more bids with utility that lies within a utility interval $[u_{min}, u_{max}] \subseteq \mathbb{R}$ (illustrated by the red rectangle in Fig. 2):

$$S \subseteq \{\omega \in \Omega : u(\omega) \in [u_{min}, u_{max}]\} \tag{2}$$

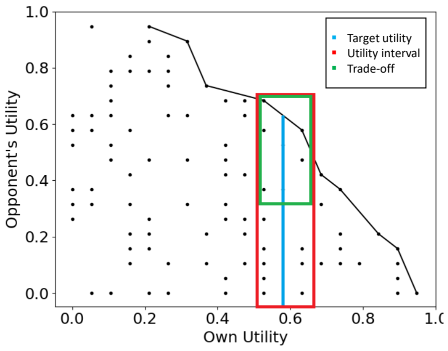

**Fig. 2** Illustration of the three most common search queries over a utility diagram. The x-axis shows the agent's own utility and the y-axis the opponent's utility. Dots represent possible bids and the continuous curve represents the Pareto-frontier. The blue line illustrates the possible picks for a target utility, the red rectangle depicts the possible options for the utility-sampling query, and the green rectangle illustrates the trade-off query of maximizing the opponent's utility within the red rectangle.

Most works in the literature determine the bounds of the interval through a time-based strategy (e.g. Agent M [7]).

The third query considers certain trade-offs with the opponent while generating a new bid. Agents search for bids that optimize some objective function $f : \Omega \to \mathbb{R}$, while aiming for at least a minimum utility $u_t$ for themselves (illustrated by the green rectangle in Fig. 2).

$$\begin{aligned} \underset{\omega \in \Omega}{\text{argmin}} \quad & f(\omega) \\ \text{subject to} \quad & u(\omega) \geq u_t \end{aligned} \tag{3}$$

The objective function $f$ can model the opponent's preferences in some way, for instance by estimating the opponent's utility function (e.g. The Fawkes Agent [19]) or by minimizing distance to the opponent's offers (e.g. Similarity-Tactic [20]). Estimations can be obtained to produce $f$, either by making use of some opponent modelling algorithms [21], or using background knowledge about the domain [22, 23].

The three queries are rather generic: in fact, when surveying the search queries used by the participants of Automated Negotiating Agents Competition (ANAC) [10] since its inception (2010-2021), we find that 87% of all participating agents use one of the three identified queries (see Table 1). Given their ubiquity, it is important to have a generic, well-founded way to produce answers to these queries, either as part of a well-known negotiation framework (for instance Genius [4], or NegMAS [6]) or as a module available to future agents. This would aid to decouple the negotiation strategies of agents from their search methods, and as a result make the comparison of negotiation strategies easier.

**Table 1** Typical search queries used by ANAC agents.

| **Search query** | **% of agents per ANAC year** | | | | | | | | | | **total** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **2021** | **2018** | **2017** | **2016** | **2015** | **2014** | **2013** | **2012** | **2011** | **2010** | |
| utility-lookup | 23% | 57% | 14% | 17% | 4% | 9% | 11% | 20% | 33% | 13% | 20% |
| utility-sampling | 33% | 19% | 52% | 78% | 70% | 50% | 45% | 40% | 56% | 29% | 50% |
| trade-off | 33% | 19% | 5% | 5% | 13% | 27% | 22% | 30% | 11% | 29% | 17% |
| other | 11% | 5% | 29% | 0% | 13% | 14% | 22% | 10% | 0% | 29% | 13% |

### 2.3 Design Requirements of Search Algorithms used by Negotiating Agents

Search algorithms that can answer the three discussed queries needs to provide a good balance between four requirements. First, the search need to be *scalable*, since agents need to be able to negotiate in realistic scenarios, where negotiation can take place over more than 100 issues (e.g. in fields such are supply-chain management and procurement). Second, algorithms need to be *accurate* so that negotiation strategies operate with minimal error. Third, since there is usually a deadline providing pressure to come to a conclusion, search algorithms need to be *rapid* so that the agents can exchange a high number of offers and therefore increase their chances in achieving an agreement. Fourth, the produced bids need to be *diverse*, so that there is a higher chance to achieve a Pareto efficient agreement (i.e. an agreement beyond which parties can improve their benefits only by harming their opponent).

*Example 1* (Importance of diversity) Suppose a buyer negotiates with the seller to obtain a TV, considering the issues of *price* (low, average, high) and *quality* (low, moderate, high). The buyer aspires for a high-quality TV at low price, while the seller seeks a high price and is indifferent about the quality. If the buyer concedes naively among the two issues he might offer an average price for a moderate-quality TV, or even concede to paying a high price for a low-quality TV. By not exploring the outcome space properly, the option of a high-quality TV for a high price is missed, which would be a better deal for the buyer and still acceptable for the seller.

Designing search algorithms that satisfy such requirements is challenging, mainly because of two reasons. First, the optimization problems associated with our three queries are hard to solve exactly and difficult to approximate rapidly. Second, guaranteeing all four requirements at once is difficult because often there are trade-offs between them. For instance, enumerating all possible outcomes of an outcome space, as in GeniusWEB [4], provides high accuracy and some diversity but is not scalable. Similarly, a search implemented through random sampling, as in NegMas [6], is scalable, diverse, and rapid but becomes rather inaccurate as the number of negotiation issues increases.

## 3 Related Work

Most works in the fields of automated negotiations focus on the design of negotiation strategies and protocols and abstract away the method of searching through the outcome space.

Jonker and Treur [9] propose the attribute-planning method, the earliest outcome-space search algorithm we are aware of. The authors in each round determine a utility target for the offer to be proposed. After that, they use the set target and a heuristic to define a utility target for each individual issue (additive utility functions are used). The method scales well and is rapid. However, it assumes continuous values for each issue and it can have accuracy

**Table 2** Comparison of search algorithms from literature with respect to the four design requirements, as well as their space and time complexity. In the table, $|\mathcal{I}|$ represents the number of negotiation issues, $|V|$ represents the highest number of possible values for one issue ($|V|= \max_i |V_i|$), and $|Im(d)|$ is a number related to the discretization operator $d$ that BIDS uses (see Section 4.1) for more details.

| | **Scalable** | **Rapid** | **Accurate** | **Diverse** | **Space** | **Time** |
|---|---|---|---|---|---|---|
| **BIDS** | ✓ | ✓ | ✓ | ✓ | $O(\lvert\mathcal{I}\rvert \cdot \lvert Im(d)\rvert)$ | $O(1)$ |
| **Attribute-Planning** [9] | ✓ | ✓ | ✓/× | × | $O(1)$ | $O(\lvert\mathcal{I}\rvert \cdot \lvert V\rvert)$ |
| **Enumeration** [4] | × | × | ✓ | ✓ | $O(\lvert\mathcal{I}\rvert \cdot \lvert V\rvert)$ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ |
| **Random Sampling** [6] | ✓ | ✓ | × | ✓ | $O(1)$ | $O(1)$ |
| **MCTS** [24] | ✓ | × | ✓ | ✓ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ |
| **NB$^3$** [25] | ✓ | × | ✓ | ✓ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ |
| **MOBANOS** [22] | ✓ | × | ✓ | ✓ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ | $O(\lvert V\rvert^{\lvert\mathcal{I}\rvert})$ |

problem when applied to discrete issues. Moreover, the heuristic used to alter the target utility for each issue provides almost no diversity.

The well-known negotiation platform GENIUS [4] provides a default search method through which all possible outcomes are enumerated during a search process. Enumerating outcomes provides high accuracy and can be done in a way that is also diverse. Nonetheless, the method is slow for moderately large outcome spaces and cannot scale to large outcomes since it has an exponential time complexity with respect to the number of negotiation issues.

In NegMas [6] the proposed search method is based on random sampling. As a consequence, the method is scalable and has high diversity. However, random sampling has an inherent trade-off between rapidness and accuracy (in the probabilistic sense). In NegMas since we are in the context of negotiations the method is tuned to be rapid and as a result it is not accurate on very large spaces.

Participants of ANAC 2014 [26] were given the task to negotiate over large outcome spaces, under nonlinear preferences. Several meta-heuristics were proposed to implement the search, including simulated annealing [7] and a genetic algorithm [8]. Similarly to random sampling, the proposed methods are scalable, provide high diversity and are tuned to be rapid. However, their accuracy deteriorates as the number of negotiation issues increases.

Buron et al. [24] propose a bidding strategy that uses Monte-Carlo Tree-Search (MCTS) to explore the outcome space. Their method is scalable, accurate and provides diversity. However, it is heavily coupled with their negotiation strategy and is also designed to operate under no time pressure.

De Jonge and Sierra [25] propose NB$^3$, a multi-objective variant of Branch and Bound, designed specifically for negotiations. It aims to search for good proposals in large non-linear domains, taking into account the utility values of multiple agents. Their algorithm generates a search tree, in which each node $\nu$ represents a partial offer. For each of these tree nodes $\nu$ and each agent $a_i$, the algorithm calculates an upper bound $ub_i(\nu)$ and a lower bound $lb_i(\nu)$. The upper bound $ub_i(\nu)$ represents the highest utility for $a_i$ among all offers that are extensions of this partial offer (i.e. among all branches that are descendants of $\nu$), while the lower bound $lb_i(\nu)$ represents the lowest utility for $a_i$ among all offers that are extensions of this partial offer. Furthermore, NB$^3$ calculates a so-called expansion heuristic $h(\nu)$, which depends on the lower and upper bounds, and which, at each iteration of the algorithm, determines which node will be expanded next. NB$^3$ can be adapted to linear domains, however, as we will see later does not scale as well as the algorithms we propose here. Moreover, because it is designed as an anytime algorithm, it can be arbitrarily inaccurate.

De Jonge et al. [22, 23] propose MOBANOS (Multi-OBjective ANd/Or Search), a method to obtain the Pareto frontier of a multi-objective combinatorial problem based on And/Or search [27] and an iterative combination of partial solutions. The algorithm is designed to work over domains with non-linear utility functions and hard constraints. It can be accurate and diverse, and scales well as long as the constraints do not involve too many variables. While this certainly applies to linear domains (which have no hard constraints at all), it turns out that it does not scale as well as our IPS algorithm, which is specifically tailored to purely linear domains.

Amini and Fathian [28] compare the performance of different stochastic search techniques in certain scenarios, with space sizes ranging from $59,049$ to $1,048,576$ possible outcomes. Lastly, there is a body of works which assumes dependencies between issues, represents the dependencies by graphs, and proposes negotiating strategies (and as a consequence search techniques) over these issues [29–31]. However, these methods do not scale to the space sizes we are interested in.

## 4 Searching Very Large Outcome Spaces

We propose BIDS (**Bi**dding using **D**iversified **S**earch), Sampling-BIDS, and IPS (**I**terative **P**areto **S**earch) — three algorithms that exploit the additive structure of utility functions to rapidly search very large outcome spaces while providing accuracy and diversity. While BIDS answers the utility-lookup query, Sampling-BIDS tackles the utility-sampling query, and IPS the trade-off query. To provide tractable solutions, BIDS discretizes the codomain of the utility functions and applies a dynamic-programming-based search to obtain an approximate solution to the associated optimization problems. With the same aim, IPS discretizes the codomain of the utility functions and identifies the solution that best answers the trade-off query by iteratively constructing the Pareto Front.

### 4.1 Looking for bid(s) that satisfy a utility target through BIDS

We recall from Section 2.2 that the utility-lookup query is defined by:

$$\underset{\omega \in \Omega}{\operatorname{argmin}} \quad |u(\omega) - u_t| \tag{4}$$

A useful property of the utility-lookup query is that its solution can be expressed through a recurrent relationship among the issues. To formalize the idea we first need to consider partial outcomes: A *partial outcome* $\omega|_I$ is an outcome defined over only some issues $I \subset \mathcal{I}$, while $\Omega^P$ is the set of all partial outcomes over all possible subsets of issues. Furthermore, given a utility function $u : \Omega \to [0,1]$, we will denote by $u^P : \Omega^P \to [0,1]$ the extension of $u$ over $\Omega^P$:

$$u^P(\omega|_I) = \sum_{i \in I} \lambda_i \cdot u_i(\omega_i) \tag{5}$$

We define also the *concatenating operator* $+$ through which a value $v \in V_j$ for an issue $j \in \mathcal{I} \setminus I$ is attached to a partial outcome $\omega|_I = (\omega_1, \ldots, \omega_i)$:

$$\omega|_I + v_j = (\omega_1, \ldots, \omega_i, v_j). \tag{6}$$

Given this and denoting by $\sigma_k(u_t) = (\omega_1^*, \dots, \omega_k^*)$ the solution of Eq. 1 for a target utility $u_t$ when the first $k$ issues of $\Omega$ are used, the recurrent equation of utility-lookup query is given by:

$$\omega_k^* = \begin{cases} \text{argmin}_{\omega_1 \in V_1} |u(\omega_1) - u_t|, & k = 1 \\ \text{argmin}_{\omega_k \in V_k} |u[\sigma_{k-1}(u_t - u^P(\omega_k)) + \omega_k] - u_t|, & \text{otherwise} \end{cases} \tag{7}$$

Intuitively, if we suppose we know the solutions of Eq. 1 for $k-1$ issues and all possible utility thresholds no larger than $u_t$, then to solve the problem for $k$ issues we have to simply pick the value of the $k^{th}$ issue that minimizes our objective function.

An algorithm that uses (7) to solve Eq. 1 will have exponential time complexity with respect to the number of negotiation issues since the utility codomain is continuous and therefore the sub-problems created while solving the original problem are almost always non-overlapping. In other words, to provide a solution to recurrence (7), exponentially many sub-problems need to be solved.

*Example 2* (Non-overlapping sub-problems) Suppose we want to find a bid close to utility target $u_t = 0.7$ in a negotiation over only three products of the example in Fig. 1. To calculate $\sigma_3(0.7)$, 100 sub-problems of calculating $\sigma_2(\cdot)$ need to be solved, each requiring yet another 100 partial solutions to $\sigma_1(\cdot)$. In general, since each issue-utility ranges over a continuous interval, there will be no overlap between the sub-problems that need to be solved, resulting in $100^3$ calculations in the worst case.

The key to a tractable solution of Eq. 1 is to discretize the utility codomain and induce optimal sub-structure to the problem. BIDS does exactly this (see Algorithm 1) and as a consequence, can apply dynamic programming to calculate an approximate solution. To discretize the codomain while preventing negative utility thresholds from arising, BIDS uses the following discretization mapping:

$$d_p(u_t) = \begin{cases} \lfloor u_t \rceil_p, u \geq 0, \\ 0, otherwise \end{cases} \tag{8}$$

where $\lfloor \rceil_p : \mathbb{R} \rightarrow \mathbb{Q}$ rounds a real number at its $p^{th}$ decimal.

The table used by dynamic programming has $|\mathcal{I}| \cdot |Im(d)|$ entries, where $|\mathcal{I}|$ is the number of issues and $|Im(d)|$ is the number of image points of $d$ (i.e. points in the grid defined over the utility codomain). As a consequence, the space computational complexity of BIDS is $O(|\mathcal{I}| \cdot |Im(d)|)$. Moreover, given that there are no more than $|V|$ possible values per issue, the time complexity of an implementation of BIDS that computes the dynamic programming table before the beginning of the negotiation and only searches the table in run-time is $O(|\mathcal{I}| \cdot |V| \cdot |Im(d)|)$ for the table-construction and $O(1)$ to search it during run-time. Lastly, there are trade-offs between the approximation accuracy of BIDS and its computational complexity.

**Algorithm 1** BIDS

```
Signature: BIDS$_k(u_t)$
1: if $k = 1$ then
2:     $\sigma_k(u_t) \leftarrow \text{argmin}_{\omega_1 \in V_1} \quad |u^P(\omega_1) - u_t|$
3: else
4:     $\omega_k^* \leftarrow \text{argmin}_{\omega_k \in V_k} |u^P[\text{BIDS}_{k-1}(d_p(u_t - u^P(\omega_k))) + \omega_k] - u_t|$
5:     $\sigma_k(u_t) = \omega_k^* + \text{BIDS}_{k-1}(d_p(u_t - u^P(\omega_k^*)))$
6: end if
7: return $\sigma_k(u_t)$
```

### 4.1.1 Trading Computational Complexity for Approximation Accuracy

For simplicity, assume we use a regular grid over the utility codomain, with each point being $10^{-p}$ apart from its closest neighbors and where $p \in \mathbb{N}$ is the precision parameter which we can tune. Then the approximation error the method introduces in each iteration is at most $10^{-p}$. Given that there are $|\mathcal{I}|$ issues, the algorithm runs $|\mathcal{I}|$ iterations for each solution. Therefore, the absolute error the method can introduce is $|\mathcal{I}|\cdot 10^{-p}$, which means the higher the precision, the smaller the introduced error. On the other hand, having grid points $10^{-p}$ apart implies that the grid is composed of $10^p$ points, which means the space complexity of the algorithm is $O(|\mathcal{I}|\cdot 10^p)$ and the time complexity of the table-construction is $O(|\mathcal{I}|\cdot|V|\cdot 10^p)$. Consequently, the more precise the algorithm is, the more space and construction time is going to require.

## 4.2 Using BIDS to implement the Sampling-Utility Query

As noted in Section 2.2, the equation of the sampling-utility query is:

$$S \subseteq \{\omega \in \Omega : u(\omega) \in [u_{min}, u_{max}]\} \tag{9}$$

BIDS can be used as a building block for an algorithm that addresses the sampling-utility. Algorithm 2 presents Sampling-BIDS, a method that provides $n_s$ samples within a specified utility interval $\mathbb{I} = [u_{min}, u_{max}]$ in scalable, rapid, accurate, and diverse manner. The algorithm uses some arbitrary distribution — typically uniform — to sample $n_s$ utility targets $U_t \subset \mathbb{I}$ (line 1 in the pseudo-code) and then uses BIDS to identify bids the utility of which is as close as possible to each of the targets. Its space complexity is the same as BIDS, i.e. $O(|\mathcal{I}|\cdot|Im(d)|)$, while the time complexity of an "offline" implementation is $O(n_s)$.

**Algorithm 2** Sampling BIDS

```
Signature: Sampling-BIDS($n_s$, $[u_{min}, u_{max}]$)
1: $U_t \leftarrow$ determineUtilSamples($n_s, u_{min}, u_{max}$)
2: $B \leftarrow \emptyset$
3: for $u_t \in U_t$ do
4:     $B \leftarrow B \cup \{\text{BIDS}_n(u_t)\}$
5: end for
6: return $B$

Signature: determineUtilSamples($n_s, u_{min}, u_{max}$)
1: $U_t \leftarrow \emptyset$
2: for $i \in \{1, \ldots, n_s\}$ do
3:     $U_t \leftarrow U_t \cup$ uniform($u_{min}, u_{max}$)
4: end for
```

## 4.3 Trading off utility with the opponent using IPS

The trade-off query can be expressed as a constrained optimization:

$$\begin{aligned} \underset{\omega \in \Omega}{\text{argmin}} \quad & f(\omega) \\ \text{subject to} \quad & u(\omega) \geq u_t \end{aligned} \tag{10}$$

We focus on the most common form of the trade-off query, in which the objective function represents a model of the opponent's utility:

$$\begin{aligned} \underset{\omega \in \Omega}{\text{argmin}} \quad & u'(\omega) \\ \text{subject to} \quad & u(\omega) \geq u_t \end{aligned} \tag{11}$$

where $u'(\omega) = \sum_{i \in \mathcal{I}} \lambda_i \cdot u'_i(\omega_i)$ and $u(\omega) = \sum_{i \in \mathcal{I}} \lambda_i \cdot u_i(\omega_i)$.

Agents that use this query, try to optimize a model of their opponent's preferences given some restrictions on their own utility.

We propose to tackle the trade-off query through IPS (see Algorithm 3), an approach that combines the discretization idea introduced in Section 4.1 with the Pareto front construction of the MOBANOS algorithm [22, 23].

*Example 3* (Application of IPS) To understand how IPS works, suppose we have a domain with two issues $\{i_1, i_2\}$, four possible values per issue $\{1, 2, 3, 4\}$, a utility function $u'$ that is to maximize, another utility function $u$ that serves as a constraint, and a utility threshold $u_t = 0.8$.

IPS first generates all partial bids, composed only of $i_1$, calculates and rounds their utilities with respect to both $u'$ and $u$, and filters out dominated solutions (see Table 3). The same procedure is followed for partial solutions defined over $i_2$. Next, it constructs the Cartesian product of the dominant partial bids, calculates and rounds the utilities of the full bids and filters out dominated solutions (see Table 4). In this

| Partial solutions | Utility values |
|---|---|
| $(1, \cdot)$ | (0.48 , 0.5) |
| ~~$(2, \cdot)$~~ | ~~(0.3 , 0.2)~~ |
| $(3, \cdot)$ | (0.6 , 0.3) |
| ~~$(4, \cdot)$~~ | ~~(0.54 , 0.25)~~ |
| ~~$(\cdot, 1)$~~ | ~~(0.28 , 0.35)~~ |
| ~~$(\cdot, 2)$~~ | ~~(0.2 , 0.25)~~ |
| $(\cdot, 3)$ | (0.24 , 0.5) |
| $(\cdot, 4)$ | (0.4 , 0.35) |

**Table 3** Example of applying IPS: First, calculate the rounded utilities of partial solutions with values assigned only for issue 1 and remove dominated partial solutions. Repeat the process for partial solutions defined over issue 2.

| Pareto set 1 | | Pareto set 2 | | Product | Utility values |
|---|---|---|---|---|---|
| $(1, \cdot)$ | | $(\cdot, 3)$ | | (1 , 3) | (0.72 , 1) |
| | X | | = | (1 , 4) | (0.88 , 0.85) |
| $(3, \cdot)$ | | $(\cdot, 4)$ | | ~~(3 , 3)~~ | ~~(0.84 , 0.8)~~ |
| | | | | (3 , 4) | (1 , 0.65) |

**Table 4** Example of applying IPS: Calculate the Cartesian product of the two Pareto sets and calculate their rounded utilities. After, remove the dominated solutions.

way it constructs the entire Pareto frontier. Lastly, among the left bids, it identifies the bid that maximizes $u'$ given that its evaluation according to $u$ is at least 0.8.

### 4.3.1 Trading Computational Complexity for Approximation Accuracy

As with the analysis for BIDS algorithm, assume we use a regular grid over the utility codomain, with each point being $10^{-p}$ apart from its closest neighbors and where $p \in \mathbb{N}$ is the precision parameter (it can be tuned). Then the approximation error the method introduces in each iteration is at most $10^{-p}$. Given that there are $|\mathcal{I}|$ issues, the algorithm runs $|\mathcal{I}|$ iterations for each solution. Therefore, the absolute error the method can introduce is $|\mathcal{I}| \cdot 10^{-p}$, which means the higher the precision, the smaller the introduced error. On the other hand, having grid points $10^{-p}$ apart implies that the grid is composed of $10^{p}$ points, which means the worst-case space complexity of the algorithm is $O(|\mathcal{I}| \cdot |V| \cdot 10^{p})$ and its time complexity is $O(|\mathcal{I}| \cdot |V| \cdot 10^{p})$. In other words, the more precise the algorithm is, the more space and time is going to require.

## 5 Experiments

Our algorithms permit the implementation of the three most used search queries for outcome spaces. To evaluate the methods, we have implemented them in GeniusWEB [4], and we have designed four experiments: In Experiment 1 we investigate how scalable and rapid our algorithms are by allowing some representative ANAC2021 participants to use them. In Experiment 2 we isolate the search problem and compare BIDS with other scalable methods in

**Algorithm 3** Iterative Pareto Search

```
Signature: IPS($u_t$)
 1: for $i$ in $\mathcal{I} \setminus \{1\}$ do
 2:     $R_i \leftarrow$ removeApproxDominatedElements($V_i$)
 3: end for
 4: $PS \leftarrow$ removeApproxDominatedElements($R_1$)
 5: for $i$ in $\mathcal{I} \setminus \{1\}$ do
 6:     $PS \leftarrow$ removeApproxDominatedElements($PS \times R_i$)
 7: end for
 8: return $\begin{cases} \text{argmax}_{\omega \in PS} & u'(\omega) \\ \text{subject to} & u(\omega) \geq u_t \end{cases}$

Signature: removeApproxDominatedElements($S$)
 1: $D \leftarrow \emptyset$
 2: for $\omega \in S$ do
 3:     for $\omega' \in S$ do
 4:         if $\omega = \omega'$ then
 5:             continue
 6:         end if
 7:         if $d_p[u^P(\omega)] \leq d_p[u^P(\omega')]$ & $d_p[u'^P(\omega)] < d_p[u'^P(\omega')]$ then
 8:             $D \leftarrow D \cup \{\omega\}$
 9:             break
10:         end if
11:         if $d_p[u^P(\omega)] < d_p[u^P(\omega')]$ & $d_p[u'^P(\omega)] \leq d_p[u'^P(\omega')]$ then
12:             $D \leftarrow D \cup \{\omega\}$
13:             break
14:         end if
15:     end for
16: end for
17: return $S \setminus D$
```

terms of accuracy and diversity. In Experiment 3 we compare Sampling-BIDS with other scalable methods in terms of accuracy. Lastly, in Experiment 4, we compare IPS in terms of accuracy with other scalable methods. Note that, ANAC agents are designed to operate with arbitrary utility functions and as a result their search methods are more general. However, when utilities are additive, our algorithms can be used to improve the agents' performance.

The implementation of BIDS and Sampling-BIDS can be found at https://gitlab.com/thimjo.koca/bids, while the impementation of IPS can be found at https://www.iiia.csic.es/~davedejonge/downloads.

## 5.1 Setup

We run simulations for scenarios with arbitrary outcome spaces, containing 5 to 2000 issues, with each issue having 10 possible values. We assign to each

| **Agent** | **Query** | **Used-Method** | **Our algorithm** |
|---|---|---|---|
| AgentFO | Utility-lookup | Enumeration | BIDS |
| AlphaBIU | Utility-sampling | Enumeration | Sampling-BIDS |
| TripleAgent | Trade-off | Enumeration | IPS |

**Table 5** Search queries and algorithms for a selection of ANAC2021 agent.

party an arbitrary utility profile over the generated spaces, i.e. an additive utility with random weights and random issue utilities. For experiments 2, 3, and 4, we generate 1000 queries for each. All our simulations were run on a laptop with an Intel i7 core and 16$GB$ of RAM.

In Experiment 1 we identify for each ANAC2021 participant the used search query (see Table 5) and evaluate the gains in scalability of some representative agents when using our methods with a precision $p = 5$. In particular, we compare the scalability of AgentFO when using BIDS as opposed to its original search method, AlphaBIU (the winner of ANAC2021) when using Sampling-BIDS as opposed to its native enumeration method, and TripleAgent (third place) when using IPS as opposed to using its own search method.

Next, to obtain some accuracy and diversity results for the utility-lookup query, we compare BIDS to other scalable search algorithms in Experiment 2. In particular, we compare it to attribute-planning [9], an adaptation of AgentM's [7] Simulated Annealing that answers the utility-lookup query, and GANGSTER's [8] Genetic Algorithm adapted to the utility-lookup query (for all three search methods we use the parameters proposed by their authors).

In Experiment 3 we investigate the accuracy for methods that tackle the utility-sampling query. In particular, we compare the accuracy of Sampling-BIDS to the accuracy of AgentM's [7] Simulated Annealing, and GANGSTER's [8] Genetic Algorithm.

Lastly, to obtain accuracy results for the trade-off query, we compare in Experiment 3 IPS to Simulated Annealing, and Genetic Algorithm adapted for the trade-off query. A version of NB$^3$(for which discretization is used and with a heuristic adapted for the linear case) is also included in the initial experiments, however, search time for a single query is around 5 minutes, which is too high for our setting. As a result, we excluded NB$^3$ from our final scalability and accuracy experiments.

## 5.2 Metrics to Quantify Scalability, Speed, Accuracy, and Diversity

Each search algorithm is scored on a number of metrics. To measure scalability, we count the highest number of issues for which an agent is able to exchange at least one offer. To also have a sense of how rapid each method is, we run negotiation sessions that last 2 minutes since ANAC2021 agents are designed to participate in negotiations of that length.

Accuracy for the utility-lookup query is estimated by calculating the mean absolute error of the query's response from the defined target utility. More specifically, assume we pose queries for $n$ different utility targets $U_t = \{u_{t_j}\}_{j=1}^{n}$

and get one response per each $\{\omega^1, \ldots, \omega^n\}$. Then the mean error, which we use to measure accuracy is:

$$e = \frac{1}{n} \sum_{j=1}^{n} |u_{t_j} - u(\omega^j)|$$

For the utility-sampling query, we measure accuracy by counting the percentage of the posed queries that are responded correctly. For instance, suppose we look for a bid that brings a utility of at least 0.7 according to the utility function $u$. In case the search algorithm returns a bid $\omega$ with utility $u(\omega) < 0.7$ the response is counted as inaccurate. Otherwise, $u(\omega) \geq 0.7$ and the response counts as accurate. For the trade-off query, we use two metrics for accuracy, as we: (a) count the percentage of the posed queries that satisfy the constraint; and (b) compare the value of the objective function that IPS brings as opposed to the rest of the scalable methods (we cannot identify the optimal bid because of the domain size).

To estimate diversity for the utility-lookup query we quantify the change for two consecutive bids composition, i.e. we calculate the variability (through the standard error) of the concession rates among issues for two consecutive bids. More specifically, we define variability $v(\omega^j)$ of a bid $\omega^j$ with respect to its predecessor $\omega^{j-1}$ in the following way:

$$v(\omega^j) = \frac{1}{|\mathcal{I}|} \sum_{i=1}^{|\mathcal{I}|} [u_i(\omega_i^j) - u_i(\omega_i^{j-1})]$$

Then for a series bids $S = \{\omega^1, \ldots, \omega^n\}$ that answer queries we measure the series variability $v(S)$ as:

$$v(S) = SE(\{v(\omega^2), \ldots, v(\omega^n)\})$$

where $SE$ stands for the standard error.

We do not estimate diversity for the utility-sampling query since the methods used are based on the algorithms used for the utility-lookup, hence their diversity is quantified before. Note also that, for the trade-off query the opponent preferences are known (and codified in the objective function), hence diversity is not relevant.

## 5.3 Results

### 5.3.1 Experiment 1 - Scalability and Rapidness

In the first experiment we investigate the scalability and rapidness of our algorithms and compare them to the search algorithms used by ANAC2021 agents.

Our results show that (see Fig. 3) BIDS and Sampling-BIDS can enable an agent to negotiate over 250 issues — over outcome spaces with $10^{250}$ possible

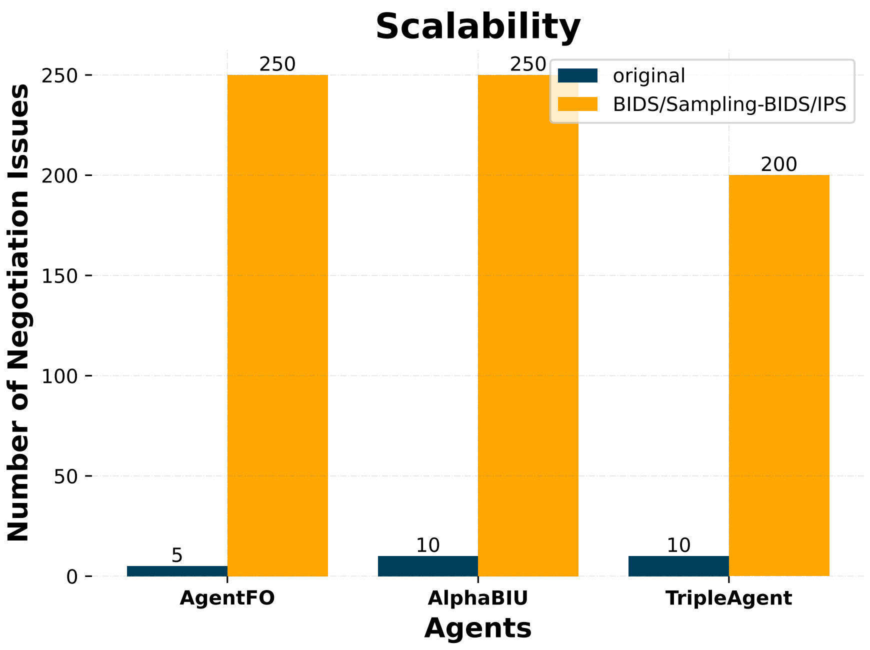

**Fig. 3** Scalability results for three ANAC2021 participants.

outcomes — and IPS can enable an agent to negotiate over 200 issues — over outcome spaces with $10^{200}$ possible outcomes — within a 2-minutes session. Meantime the original search algorithms used by ANAC2021 participants allow them to negotiate upon a maximum of 10 issues — or over outcome spaces with $10^{10}$ possible outcomes. The poor performance of ANAC2021 comes as a result of their search method, with each agent using exhaustive enumeration and cannot negotiate over more than 10 issues. Our BIDS algorithms cannot allow negotiations over more than 250 issues within 2 minutes since the initialization of the dynamic-programming table takes too long. Lastly, note that within GeniusWEB, IPS can generate at least one bid for domains composed of 2000 issues within the 2 minutes interval. That happens since in the beginning the opponent model is unknown, all weights and issue utilities are identical, making the whole domain indifferent with respect to the utility $u'$ used as the objective function. As a result, all possible bids are dominated by the maximum bid (with respect the constraint utility $u$) and search becomes extremely fast. However, once the opponent is updated, IPS gets slow for domains with more than 200 issues, which we report as the scalability limit for IPS.

The results of Experiment 1 support our claim over the scalability of our methods. However, if we focus solely on scalability, similar results can be achieved by using various other methods (Simulated Annealing, Genetic Algorithms, Attribute-Planing). Nonetheless, apart from scalability given some time restrictions, search accuracy and diversity play a crucial role in the quality of a search algorithm.

### 5.3.2 Experiment 2 - Accuracy and Diversity of BIDS

In the second experiment we isolate the search problem to evaluate the search accuracy and diversity of BIDS algorithm and compare it with the other scalable methods (the attribute-planning, simulated annealing, and genetic algorithm) to get an insight of which algorithms can perform better in very large outcome spaces. To do so, we define a series of utility targets from 0 up to 1 with a regular step of 0.1 and use each search method to respond the utility-lookup query.

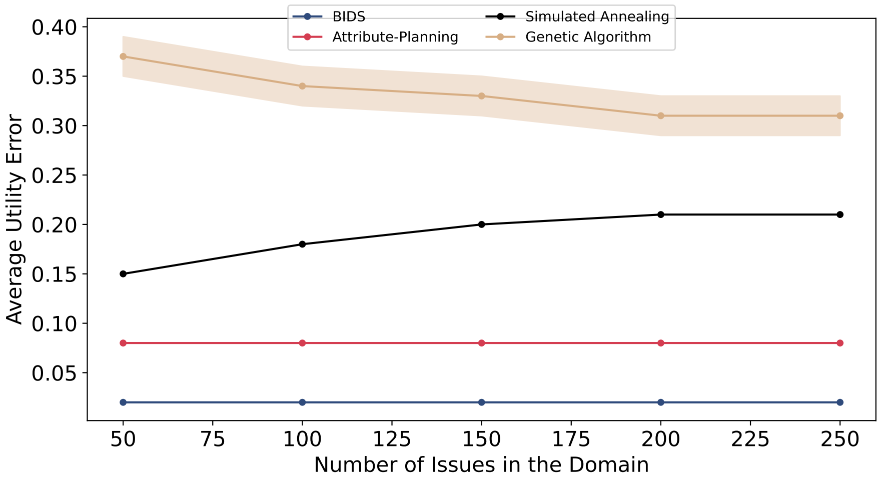

**Fig. 4** Mean standard error for each scalable search method as we vary the number of issues in the outcome space. The smaller the error, the more accurate the search method is.

Our results on accuracy show (see Fig. 4) a clear ranking among the search methods. BIDS is more accurate since the way it explores the outcome space allows it to consider outcomes smartly and guarantee small error bounds (see Error Analysis in Section 4.1.1). Attribute-planning comes second penalized by the fact that it determines individual issue utility targets, which can lead to higher errors in discrete spaces. To illustrate this, suppose that in a given space for a particular issue $\omega_i$ there are only 2 possible values that can bring issue-utilities of 0.1 and 0.2 and that for that issue the weight is $\lambda_i = 0.5$. This means that if attribute-planning assigns a target utility for this issue $u_{t_i} = 0.8$, it will introduce an error of 0.3. Lastly, the meta-heuristics perform poorly with respect to accuracy, penalized by their randomness combined with their trade-off between search-time and accuracy.

The results on diversity show (see Fig. 5) a different ranking. Here the randomness used by Simulated Annealing and Genetic Algorithm gives them

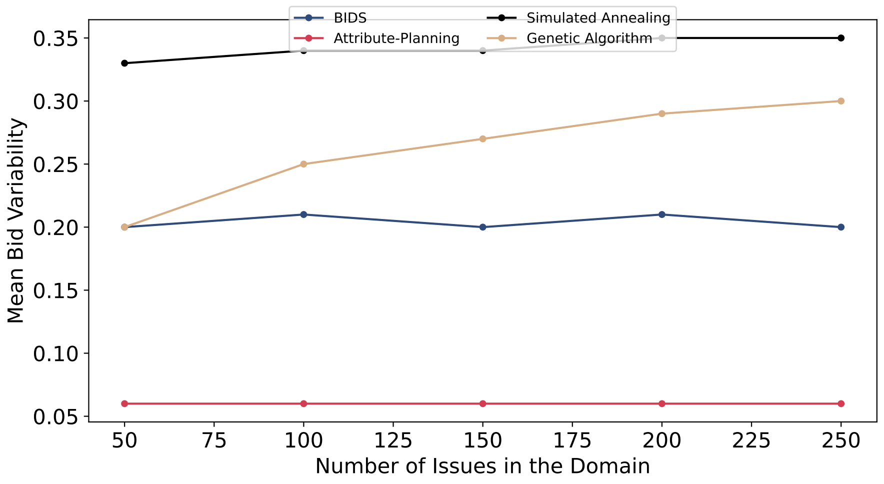

**Fig. 5** Mean variability for each scalable search method as we vary the number of issues in the outcome space. The higher the variability, the more diverse the search method is.

the lead; BIDS comes after with a diversity around the middle of best and worst performing methods; and attribute-planning comes last penalized by the very regular way its heuristic distributes concession among different issues.

To summarize, BIDS can scale to spaces with up to 250 issues. Moreover, even though attribute-planning and the meta-heuristics can scale as high, our method provides higher accuracy and satisfactory diversity, having overall a better balance among the properties.

### 5.3.3 Experiment 3 - Accuracy of Sampling-BIDS

In the third experiment we isolate the search problem associated to the utility-sampling query to evaluate the search accuracy of Sampling-BIDS algorithm and compare it with the other scalable methods (simulated annealing, and genetic algorithm) to get an insight of which algorithms can perform better in very large outcome spaces. To do so, we define a series of utility thresholds from 0 up to 1 with a regular step of 0.1 and use each search method to respond the utility-sampling query.

Our results on accuracy show (see Fig. 6) a clear ranking among the search methods. Sampling-BIDS is more accurate since the deterministic way through which it explores the outcome space allows it to always identify bids the utility of which are higher than the threshold. On the other hand, the probabilistic nature of Simulated Annealing and Genetic Algorithm causes inaccuracies which at very large domain sizes happen frequently (around 40% of the cases for AgentM's Simulated Annealing and 70% of the cases for GANGSTER's Genetic Algorithm).

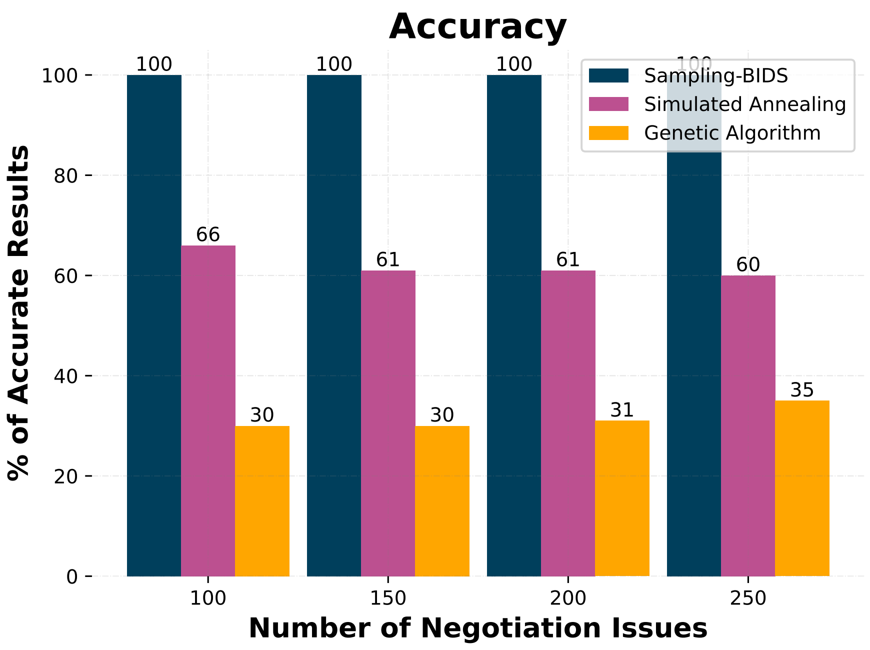

**Fig. 6** Percentage of search results above the set utility threshold.

To summarize, Sampling-BIDS can scale to spaces with up to 250 issues. Moreover, even though the meta-heuristics can scale as high, our method provides higher accuracy and satisfactory diversity (as seen in Section 5.3.2), having overall a better balance among the properties.

### 5.3.4 Experiment 4 - Accuracy of IPS

In the forth experiment we isolate the search problem associated to the trade-off query to evaluate the search accuracy of IPS and compare it with the other scalable methods (simulated annealing, and genetic algorithm) to get an insight of which algorithms can perform better in very large outcome spaces. To do so, we again define a series of utility thresholds from 0 up to 1 with a regular step of 0.1 and use each search method to respond the trade-off query.

Our results on constraint accuracy show (see Fig. 7) a clear ranking among the search methods. IPS is more accurate since the deterministic way through which it explores the outcome space allows it to always identify bids the utility of which are higher than the threshold. Differently, the probabilistic elements of Simulated Annealing and Genetic Algorithm causes inaccuracies which at very large domain sizes happen frequently ($30-40\%$ of the cases for Simulated Annealing and $60-70\%$ of the cases for Genetic Algorithm).

Furthermore, our results on objective accuracy show (see Fig. 8) IPS have a clear advantage with respect to Genetic Algorithm (ranging from 0.33 to 0.36 points of utility) and Simulated Annealing (ranging from 0.4 to 0.42 points

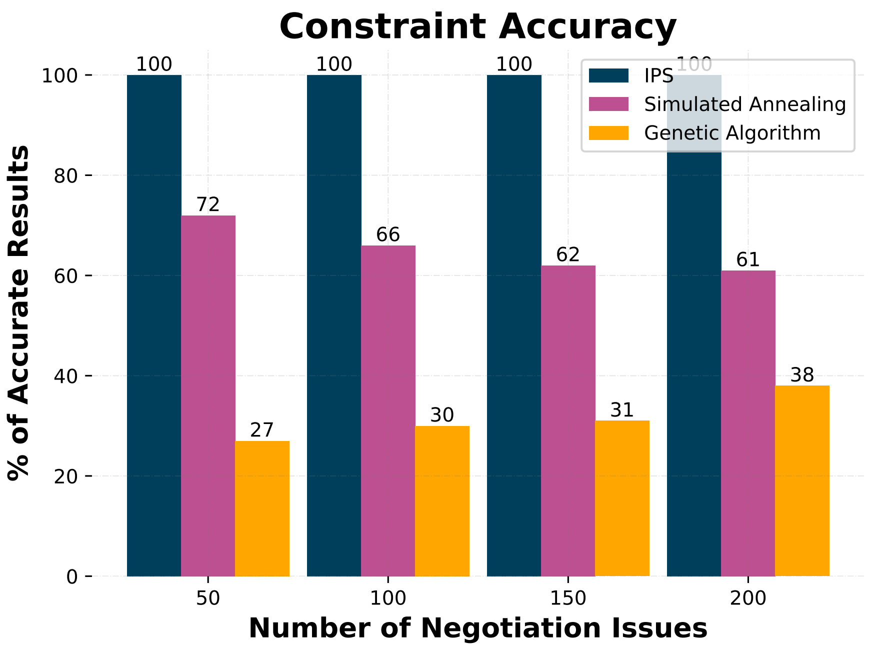

**Fig. 7** Percentage of search results above the set utility threshold.

of utility) since IPS has theoretical accuracy guarantees (in our experiments $|\mathcal{I}| \cdot 10^{-5}$), while the probabilistic nature of the metaheuristics can result in bids arbitrarily far from the optimal one.

To summarize, IPS can scale to spaces with up to 200 issues. In addition, even though the meta-heuristics can scale higher, our method provides higher accuracy (both with respect to the constraint and the objective function), having overall a better balance among the properties.

## 6 Conclusions and Future Work

This work presents several algorithms that exploit the additive structure of the utility function to search very large domains while providing search accuracy and diversity. We find that our methods can increase drastically the domain size in which negotiating agents can still function, while providing very high accuracy and significant outcome diversity. Therefore, our algorithms can enable state-of-the-art (and future) agents to negotiate over very large realistic domains such as the ones found in procurement and supply-chain management.

Future work may build on this to evaluate the robustness of specific negotiation strategies on the accuracy and diversity of the used search method. Moreover, as we discussed, literature typically focuses on proposing negotiation strategies without considering the complexity of the associated search problem. Nonetheless, there are works, as the ones that participated in ANAC 2014 [26],

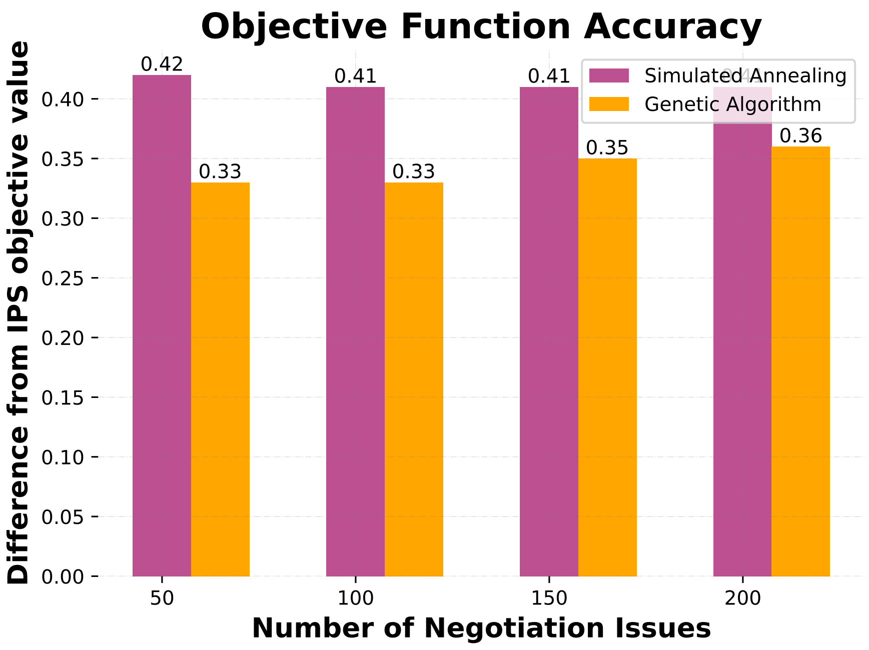

**Fig. 8** Difference in terms of objective function utility from the objective function utility of the IPS result.

that are designed to negotiate over large spaces (albeit not as large as ours) and propose negotiation strategies together with some search algorithms. It would be interesting to investigate whether these negotiation strategies have a competitive advantage when the same underlying search algorithms are used. Lastly, all three methods could be modified to search for partial bids: while for BIDS and Sampling-BIDS the adaptation is straightforward (we need to adapt BIDS recurrent equation to account for partial bids), but it is not yet clear how to adapt IPS such that it preserves its theoretical accuracy guarantees.

## Acknowledgments

Tim Baarslag is funded by the Dutch Research Council (NWO), as part of Vidi research project VI.Vidi.203.044. Thimjo Koça and Dave de Jonge declare that they have no conflict of interest.

## Data Availability Statement

The datasets generated during and analysed during the current study are available in the Gitlab repository, https://gitlab.com/thimjo.koca/data-processed-in-the-experiments-of-paper-on-outcome-space-search-algorithms/-/tree/main.

## References

[1] Byde, A., Yearworth, M., Chen, K.-Y., Bartolini, C.: Autona: A system for automated multiple 1-1 negotiation. In: EEE International Conf. on E-Commerce, 2003. CEC 2003., pp. 59–67 (2003). IEEE

[2] Mohammad, Y., Viqueira, E.A., Ayerza, N.A., Greenwald, A., Nakadai, S., Morinaga, S.: Supply chain management world. In: International Conference on Principles and Practice of Multi-agent Systems, pp. 153–169 (2019). Springer

[3] An, B., Lesser, V.R., Irwin, D.E., Zink, M.: Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In: AAMAS, vol. 10, pp. 981–988 (2010)

[4] Lin, R., Kraus, S., Baarslag, T., Tykhonov, D., Hindriks, K., Jonker, C.M.: Genius: An integrated environment for supporting the design of generic automated negotiators. Computational Intelligence **30**(1), 48–70 (2014)

[5] TU Delft: GeniusWeb platform. [Online; accessed 04.01.2022] (2019)

[6] Mohammad, Y., Nakadai, S., Greenwald, A.: Negmas: a platform for situated negotiations. In: International Workshop on Agent-Based Complex Automated Negotiation, pp. 57–75 (2019). Springer

[7] Niimi, M., Ito, T.: In: Fukuta, N., Ito, T., Zhang, M., Fujita, K., Robu, V. (eds.) AgentM, pp. 235–240. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30307-9_15

[8] de Jonge, D., Sierra, C.: In: Fukuta, N., Ito, T., Zhang, M., Fujita, K., Robu, V. (eds.) GANGSTER: An Automated Negotiator Applying Genetic Algorithms, pp. 225–234. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30307-9_14

[9] Jonker, C.M., Treur, J.: An agent architecture for multi-attribute negotiation. In: International Joint Conference on Artificial Intelligence, vol. 17, pp. 1195–1201 (2001). LAWRENCE ERLBAUM ASSOCIATES LTD

[10] Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., Lin, R.: In: Ito, T., Zhang, M., Robu, V., Fatima, S., Matsuo, T. (eds.) The First Automated Negotiating Agents Competition (ANAC 2010), pp. 113–135. Springer, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-24696-8_7

[11] Fujita, K., Ito, T., Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., Lin, R.: In: Ito, T., Zhang, M., Robu, V., Matsuo, T. (eds.) The Second Automated Negotiating Agents Competition (ANAC2011), pp. 183–197. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/

978-3-642-30737-9_11

[12] Williams, C.R., Robu, V., Gerding, E.H., Jennings, N.R.: An overview of the results and insights from the third automated negotiating agents competition (anac2012). Novel Insights in Agent-based Complex Automated Negotiation, 151–162 (2014)

[13] Fujita, K., Aydoğan, R., Baarslag, T., Hindriks, K., Ito, T., Jonker, C.: In: Fujita, K., Bai, Q., Ito, T., Zhang, M., Ren, F., Aydoğan, R., Hadfi, R. (eds.) The Sixth Automated Negotiating Agents Competition (ANAC 2015), pp. 139–151. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51563-2_9

[14] Aydoğan, R., Fujita, K., Baarslag, T., Jonker, C.M., Ito, T.: Anac 2017: Repeated multilateral negotiation league. In: International Workshop on Agent-Based Complex Automated Negotiation, pp. 101–115 (2018). Springer

[15] Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT press, Cambridge (1994)

[16] Kawaguchi, S., Fujita, K., Ito, T.: Compromising strategy based on estimated maximum utility for automated negotiation agents competition (anac-10). In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 501–510 (2011). Springer

[17] Baarslag, T., Hindriks, K., Jonker, C.: In: Ito, T., Zhang, M., Robu, V., Matsuo, T. (eds.) A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations, pp. 229–233. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30737-9_18

[18] Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems **24**(3-4), 159–182 (1998)

[19] Koeman, V.J., Boon, K., van den Oever, J.Z., Dumitru-Guzu, M., Stanculescu, L.C.: In: Fujita, K., Ito, T., Zhang, M., Robu, V. (eds.) The Fawkes Agent—the ANAC 2013 Negotiation Contest Winner, pp. 143–151. Springer, Tokyo (2015). https://doi.org/10.1007/978-4-431-55525-4_10

[20] Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make negotiation trade-offs. In: Proceedings Fourth International Conference on MultiAgent Systems, pp. 119–126 (2000). IEEE

[21] Baarslag, T., Hendrikx, M.J.C., Hindriks, K.V., Jonker, C.M.: Learning

about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. Auton. Agents Multi Agent Syst. **30**(5), 849–898 (2016). https://doi.org/10.1007/s10458-015-9309-1

[22] de Jonge, D., Bistaffa, F., Levy, J.: A heuristic algorithm for multi-agent vehicle routing with automated negotiation. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. AAMAS '21, pp. 404–412. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021)

[23] Jonge, D.d., Bistaffa, F., Levy, J.: Multi-objective vehicle routing with automated negotiation. Applied Intelligence (2022). https://doi.org/10.1007/s10489-022-03329-2

[24] Buron, C.L., Guessoum, Z., Ductor, S.: Mcts-based automated negotiation agent. In: International Conference on Principles and Practice of Multi-Agent Systems, pp. 186–201 (2019). Springer

[25] De Jonge, D., Sierra, C.: NB$^3$: A multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. Autonomous Agents and Multi-Agent Systems **29**(5), 896–942 (2015)

[26] Aydogan, R., Baarslag, T., Jonker, C.M., Fujita, K., Ito, T., Hadfi, R., Hayakawa, K.: A baseline for non-linear bilateral negotiations: the full results of the agents competing in anac 2014 (2016)

[27] Marinescu, R., Dechter, R.: AND/OR branch-and-bound search for combinatorial optimization in graphical models. Artif. Intell. **173**(16-17), 1457–1491 (2009). https://doi.org/10.1016/j.artint.2009.07.003

[28] Amini, M., Fathian, M.: Optimizing bid search in large outcome spaces for automated multi-issue negotiations using meta-heuristic methods. Decision Science Letters **10**(1), 1–20 (2021)

[29] Ito, T., Hattori, H., Klein, M.: Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In: IJCAI, vol. 7, pp. 1347–1352 (2007)

[30] Hadfi, R., Ito, T.: Modeling complex nonlinear utility spaces using utility hyper-graphs. In: International Conference on Modeling Decisions for Artificial Intelligence, pp. 14–25 (2014). Springer

[31] Marsa-Maestre, I., Klein, M., Jonker, C.M., Aydoğan, R.: From problems to protocols: Towards a negotiation handbook. Decision Support Systems **60**, 39–54 (2014)