Enabling Negotiating Agents to Explore Very Large Outcome Spaces

Thimjo Koça[1][0000−0002−8889−8008], Catholijn M. Jonker[2,3][0000−0003−4780−7461], and Tim Baarslag[1,4][0000−0002−1662−3910]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{thimjo.koca,T.Baarslag}@cwi.nl
[2] TU Delft, Delft, The Netherlands
c.m.jonker@tudelft.nl
[3] Leiden University, Leiden, The Netherlands
[4] Utrecht University, Utrecht, The Netherlands

Abstract. This work presents BIDS (**Bi**dding using **D**iversified **S**earch), an algorithm that can be used by negotiating agents to search very large outcome spaces. BIDS provides a balance between being rapid, accurate, diverse, and scalable search, allowing agents to search spaces with as many as 10^{250} possible outcomes on very run-of-the-mill hardware. We show that our algorithm can be used to respond to the three most common search queries employed by ANAC agents. Furthermore, we validate one of our techniques by integrating it into negotiation platform GeniusWeb, to enable existing state-of-the-art agents (and future agents) to scale their use to very large outcome spaces.

Keywords: automated negotiation · very large negotiation domain · search.

1 Introduction

Over the last decades, more and more processes and information are being digitized, allowing the implementation of new technologies that can reduce the duration and complexity of business processes. Automated negotiations is a promising example of such technologies that can bring benefits to various fields, including procurement [8], supply chain management [23], and resource allocation [2].

In such fields, negotiations can take place over high number of finite issues (roughly 100 issues or more). For instance, suppose a buyer (Bob) is trying to negotiate with one of his suppliers (Sally) over the delivery of 100 shipments for the next year, as depicted in Fig. 1. For each shipment there are 365 possible delivery dates, resulting in an outcome space with 365^{100} possibilities. Every time Bob has to propose a new offer to Sally, he needs to define some criteria that his next offer must fulfill (e.g. bring him a certain level of utility, lie within a utility interval, conform some trade-off between his own preferences and Sally's) and then search the enormous outcome space for a bid that best fits his criteria. Moreover, since Bob and Sally keep their preferences private, 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 broadly the outcome space. Hence, Bob needs a *scalable* way to search the enormous outcome space in a manner that is *timely*, *accurate*, and *diverse*.

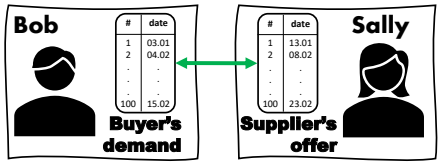

Fig. 1: A negotiation example.

However, searching a large discrete outcome space in the context of automated negotiation is a difficult task, mainly for two reasons. First, because the search process often translate to combinatorial problems that are impossible to solve exactly and challenging to solve approximately. Second, because it is not straightforward to design search algorithms that scale well, have accuracy guarantees, and explore the outcome space rapidly and in a diverse way, since there are trade-offs between the four properties.

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 of their underlying assumptions. In particular, they assume that: (a) either that the outcome space is small enough to be enumerated and explored rapidly [20, 26], and are therefore not scalable; (b) or that the space can be randomly sampled [22, 24, 10], and as a result perform poorly in very large spaces; (c) or that search goals can be defined deterministically for each individual negotiation issue [17], which can lead to poor accuracy in finite domains as well as narrow exploration of the outcome space.

In this work, we propose BIDS (**B**idding using **D**iversified **S**earch) — an algorithm that can search outcome spaces with as many as 10^{250} possible outcomes given that the user's preferences are expressed by the widely-used additive utility function (i.e. with no issue interdependencies). The algorithm employs a dynamic-programming approach to exploit the additive structure of the utility function. We show that our methodology is accurate since it identifies approximate solutions with arbitrary error bounds to the search problem and can provide diversity since it is able to explore the outcome space broadly. Furthermore, we show that our methodology is generic by first surveying the search queries used by the agents that have participated in the Automated Negotiating Agents Competition (ANAC), and then use our algorithm to implement 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. Lastly, we validate BIDS by integrating it into negotiation platform GeniusWeb so that state-of-the-art (and future) agents that need the utility-lookup query can use it.

2 Problem Setting

Our purpose in this work is to 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 $\mathcal{I}$ and each issue $i \in \mathcal{I}$ has an associated finite set of values V_i. For instance, in the task-scheduling scenario of Fig. 1, the issues to negotiate upon are the 100 deliveries and the possible values per issue are the 365 days. All possible combinations of values form the *outcome space*, which is denoted by $\Omega = \prod_{i \in \mathcal{I}} V_i$. Each element $\omega \in \Omega$ is called a negotiation *outcome* and whenever it is convenient we will denote the component of ω corresponding to issue $i \in \mathcal{I}$ by $\omega_i \in V_i$.

The private preferences of each party over Ω 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. There are no dependencies between individual issues since the utility function is a convex sum of individual issue utilities. The reasons why we picked additive utility functions are that they are widely used [6, 14, 27, 13, 4] and because, 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 can code rather complex preference structures, since we do not make any assumptions on u_i, and as consequence we can define over each issue arbitrary utility functions (i.e. not necessarily linear).

A negotiation protocol (e.g. the Alternating Offers Protocol (AOP) [25]) 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) ω 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 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 need to satisfy – examples include an appropriate utility target, a utility interval of interest, a trade-off between the own preferences and the opponent's — and subsequently tries to identify the most appropriate bids that meet one of three important search queries (illustrated in Fig. 2).

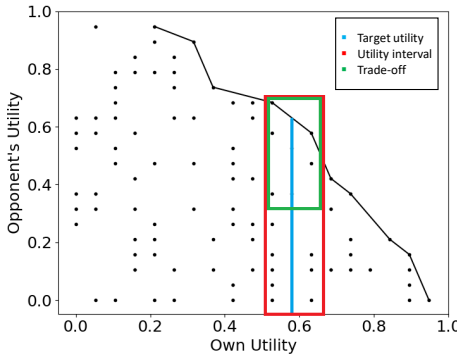

Fig. 2: Illustration of the three most common search queries over a utility diagram. The x-axis shows the agent's own utility and the y-axis the opponent's utility. Dots represent possible bids and the continuous curve represents the Pareto-frontier. While the blue line illustrates the possible picks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

The utility-lookup is the simplest among the three 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). The target can be determined through a time-based strategy (e.g. Agent K [18]), a behavior-based strategy (e.g. Nice-Tic-For-Tac Agent [5]), or through some other criteria (e.g. through a resource-based tactic [12]). Formally, it can be defined as a minimization problem:

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

The second query is the sampling-utility query. 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). Most works in literature determine

the bounds of the interval through a time-based strategy (e.g. Agent M [24]). Formally, the query can be expressed as identifying a set S containing outcomes within a certain utility interval:

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

The third query considers some trade-offs while generating a new bid. Agents search for bids that optimize some objective function $f : \Omega \to \mathbb{R}$, while asking for at least a minimum utility for themselves (illustrated by the green rectangle in Fig. 2). Typical objective functions 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 [11]). Formally, it can be defined as a constrained optimization problem:

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

The three queries are rather generic. In fact, when surveying the search queries used by the participants of Automated Negotiating Agents Competition (ANAC) [6] since its inception (2010-2021), we find 87% of all participating agents use one of the three identified queries (see Table 2). Given their ubiquity, it is important to have a generic, well-founded way to answer these queries, either as part of a well-known negotiation framework (for instance [20], or [22]) 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 Specification of Search Algorithms used by Negotiating Agents

Algorithms that can answer the three discussed queries need to provide a good balance between four specifications. First, the search need to be *scalable*, since we want agents to be able and negotiate in realistic scenarios, where negotiation can take place over even 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 negotiation parties keep their preferences private, 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. Four, the algorithms need to provide *diversity*, so that there are higher chances to achieve an agreement close to the Pareto frontier even though the opponent's preferences are not known.

Example 1 (Importance of diversity). To understand the importance of diversity suppose a buyer negotiates with the seller to obtain a TV considering 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 regularly among the two issues he will offer to pay a low price for a high-quality TV, then an average price for a moderate-quality TV, and finally agree to a high price for a low-quality TV. The regular concession among issues, i.e. exploration of the outcome space with no diversity, made it impossible to agree on a high-quality TV for a high price, which is a better deal for the buyer and still acceptable for the seller.

Designing search algorithms (especially) with such specifications 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 specifications 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 [20], provides high accuracy and some diversity, but it is not scalable. Similarly, a search implemented through random sampling, as in NegMas [22], is scalable, diverse, and rapid, but it becomes really inaccurate as the number of negotiation issues increases.

3 Related Work

Most works in the fields of automated negotiations abstract away the outcome-space search method and assume there is an efficient way to implement it because the main topics of interest in field are the proposal of negotiation strategies and the design of negotiation protocols.

Table 2: Comparison of search algorithms from literature with respect to the four design specifications.

	Scalable	**Rapid**	**Accurate**	**Diverse**
BIDS	✓	✓	✓	✓
Attribute-Planning [17]	✓	✓	✓/×	×
Enumeration [20]	×	×	✓	✓
Random Sampling [22]	✓	✓	×	✓
MCTS [7]	✓	×	✓	✓
NB^3 [7]	✓	×	✓	✓

Jonker & Treur [17] 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 and use it to define a utility target for each issue (additive utility functions are used). The method scales well and is rapid, however, it can have accuracy problem when applied to discrete issues and also the heuristic used to alter the target utility for each issue provides almost no diversity. The well known negotiation platform GENIUS [20] provides a default search method through which all possible outcomes are enumerated during a search process. The method provides high accuracy and some diversity, however, it does not scale well and it gets slow for moderately large outcome spaces. In NegMas [22] the proposed search method is based on random sampling and is therefore scalable and has high diversity, however because it is tuned to be rapid it is not accurate on very large spaces. Participants of ANAC 2014 [3] 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 [24] and genetic algorithm [10]. Similarly to random sampling, the proposed methods are scalable and provide high diversity, however, there is a trade-off in their tuning between being accurate and rapid. Buron et al. [7] propose a bidding strategy that uses MCTS to explore the outcome space. Their method is heavily coupled with their negotiation strategy and while it can be scalable, accurate, and provide diversity, it is designed to operate under no time pressure.

In some other relevant works, Amini & Fathian [1] compare the performance of different stochastic search techniques in certain scenarios, with space sizes ranging from $59,049$ to $1,048,576$ possible outcomes, while de Jonge & Sierra [9] propose NB^3, a search algorithm in a setting where utility functions are publicly known, but computationally expensive (NP-hard) to calculate and there is no time pressure. Lastly, there is body of works which assumes dependencies between issues, represents them by graphs, and proposes negotiating strategies (and as a consequence search techniques) over them [16, 15, 21], however, the methods do not scale to the space sizes we are interested in.

4 Searching through BIDS

We propose BIDS (**Bi**dding using **D**iversified **S**earch) — an algorithm that exploits the additive structure of a utility function to rapidly search very large outcome spaces providing accuracy & diversity. BIDS can answer the utility-lookup query, and it can also serve as a building block to tackle the utility-interval-sampling query and the trade-off query. To provide a tractable solution of the utility-lookup query, BIDS discretizes the codomain of the utility function and applies a dynamic-programming-based search to obtain an approximate solution to the associated optimization problem.

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

A useful property of the utility-lookup query (see Eq. 1) is that its solution can be expressed through a recurrent relationship. Intuitively, if we suppose we know the solutions of Eq. 1 for $n-1$ issues and all possible utility thresholds no larger than u_t, then to solve the problem for n issues we have to simply pick the value of the n^{th} issue that minimizes our objective function.

To formalize the idea we firstly need to consider partial outcomes: A *partial outcome* $\omega|_I$ is an outcome defined over only some issues $I \subset \mathcal{I}$, while Ω^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 Ω^P:

$$u^P(\omega|_I) = \sum_{i \in I} \lambda_i \cdot u_i(\omega_i) \tag{4}$$

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 $\omega|_I = (\omega_1, \ldots, \omega_i)$:

$$\omega|_I + v_j = (\omega_1, \ldots, \omega_i, v_j). \tag{5}$$

Given this and denoting by $\sigma_n(u_t)$ the solution of Eq. 1 for a target utility u_t when the first n issues of Ω are used, the recurrent equation of utility-lookup query is:

$$\sigma_n(u_t) = \begin{cases} \text{argmin}_{\omega_1 \in V_1} |u(\omega_1) - u_t|, & n = 1 \\ \text{argmin}_{\omega_n \in V_n} u[\sigma_{n-1}(u_t - u^P(\omega_n)) + \omega_n], & \text{otherwise} \end{cases} \tag{6}$$

An algorithm that uses (6) to solve Eq. 1 will have exponential time complexity with respect to the number of negotiation issues. This is a result of the fact that 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 (6), 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 delivery dates of the example in Fig. 1. To calculate $\sigma_3(0.7)$, 365 sub-problems of calculating $\sigma_2(\cdot)$ need to be solved, each requiring yet another 365 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 365^3 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:

$$d_p(u_t) = \begin{cases} \lfloor u_t \rceil_p, u \geq 0, \\ 0, otherwise \end{cases} \tag{7}$$

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

Algorithm 1 BIDS Algorithm

Input: u_t, Ω
1: **if** $n = 1$ **then**
2: **return** $\text{argmin}_{\omega_1 \in V_1} \quad |u(\omega_1) - u_t|$
3: **end if**
4: **return** $\text{argmin}_{\omega_n \in V_n} u^P[\text{BIDS}(d_p(u_t - u(\omega_n)), \prod_{i=1}^{n-1} V_i) + \omega_n]$

The table used by dynamic programming has $n_i \cdot n_g$ entries, where n_i is the number of issues and n_g is the number of points in the grid defined over the utility codomain. As a consequence, the space computational complexity of BIDS is $O(n_i \cdot n_g)$. Moreover, given that there are n_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 $O(n_i \cdot n_v \cdot n_g)$ for the table-construction and $O(1)$ to search it during run-time. For what is more, there are trade-offs between the approximation accuracy of BIDS and its computational complexity.

Trading Computational Complexity for Approximation Accuracy For simplicity, assume we use a regular grip 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 n_i issues, the algorithm runs n_i iterations for each solution. Therefore, the absolute error the method can introduce is $n_i \cdot 10^{-p}$, which means the higher the precision, the smaller the introduced error. On the other hand, having grid points 10^{-p} apart implies that the grip is composed of 10^p points, which means the space complexity of the algorithm is $O(n_i \cdot 10^p)$ and the time complexity of the table-construction is $O(n_i \cdot n_v \cdot 10^p)$. 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 & the Trade-off Query

BIDS can be used as a building block for algorithms that address the sampling-utility and the trade-off queries.

Algorithm 2 presents Sampling-BIDS, a method that provide n_s samples within a specified utility interval $\mathbb{I} = [u_{min}, u_{max}]$ in scalable, rapid, accurate, and diverse manner. The algorithm samples n_s utility targets $U_t \subset \mathbb{I}$ 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. $O(n_i \cdot n_g)$, while the time complexity of an "offline" implementation is $O(n_s)$.

Algorithm 2 Sampling-BIDS Algorithm

Input: n_s, Ω, $[u_{min}, u_{max}]$
1: $U_t :=$ determineUtilSamples(n_s, u_{min}, u_{max})
2: $B := \emptyset$
3: **for** $u_t \in U_t$ **do**
4: $\quad B := B \cup \{\text{BIDS}(u_t, \Omega)\}$
5: **end for**
6: **return** B

Similarly, Algorithm 3 presents Optimizing-BIDS, a method that builds upon Sampling-BIDS to approximately solve the trade-off query. Its space and time complexities are identical with Sampling-BIDS, i.e. $O(n_i \cdot n_g)$ space complexity and $O(n_s)$ time complexity. Note that while Optimizing-BIDS improves the state-of-the-art by being scalable, rapid, and diverse, it does not provide accuracy guarantees.

Algorithm 3 Optimizing-BIDS Algorithm

Input: n_s, Ω, u_t
1: $B :=$ SamplingBIDS$(n_s, \Omega, u_t, 1.0)$
2: **return** $\text{argmin}_{\omega \in B} f(\omega)$

5 Experiments

BIDS permits the implementation of the three most used search queries for outcome spaces. To validate the utility-lookup query, we have implemented it in GeniusWEB [20], so that state-of-art agents can use it. Furthermore, to verify that it complies to the four design specifications, we have designed two experiments: In Experiment 1 we investigate how scalable and rapid BIDS is by implementing a simple agent that uses the algorithm to explore the outcome space and compare it to various other agents. In Experiment 2 we isolate the search problem and compare BIDS with the scalable methods of the first experiment in terms of accuracy and diversity.

5.1 Setup

We run simulations for scenarios with arbitrary outcome spaces, containing 50 to 300 issues, with each issue having 10 possible values. We assign to each party an arbitrary utility profile over the generated spaces, i.e. an additive utility with random weights and random issue utilities.

First, we compare a simple agent that uses BIDS (BAgent) to all ANAC2021 participants. Since we are interested in the search process and not the negotiation strategy as a whole, in each round BAgent refuses the opponent's offer, sets an arbitrary utility target u_t, and uses BIDS to identify the bid with utility as close as possible to u_t. In order to obtain an upper bound for scalability for a fixed negotiation time, we allow each agent to use as much time as possible, by letting them negotiate against an opponent that uses minimal time to respond since it always rejects the opponent's offer and proposes a random bid as a counter-offer. Next, to obtain some accuracy and diversity results for very large outcome spaces, we compare BIDS to other scalable search algorithms. In particular, we compare it to attribute-planning [17], an adaptation of AgentM's [24] Simulated Annealing that answers the utility-lookup query, and GANGSTER's [10] Genetic Algorithm adapted to the utility-lookup query.

Lastly, note that we do not need any extensive computational resources for such large spaces. We run them our simulations in a laptop with an i7 core and $16GB$ of RAM.

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 we mean error, which we use to measure accuracy is:

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

Lastly, to estimate diversity 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 ω^j with respect to its predecessor ω^{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.

5.3 Experiment 1 - Scalability & Rapidness of BIDS

In the first experiment we investigate the scalability and rapidness of BIDS algorithm and compare it to ANAC2021 agents.

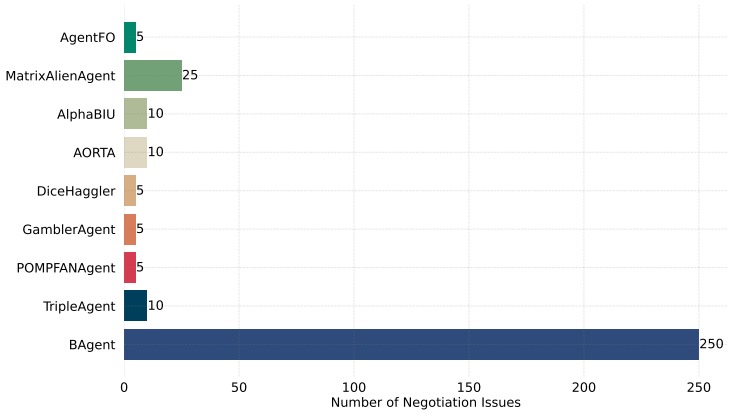

Fig. 3: Scalability results for BAgent and ANAC2021 participants.

Our results show that (see Fig. 3) BIDS can enable an agent to negotiate over 250 issues — in other words over outcome spaces with 10^{250} possible outcomes — within a 2-minutes session, while the best performing ANAC2021 participant can negotiate upon a maximum of 25 issues — or over outcome spaces with 10^{25} possible outcomes. The poor performance of ANAC2021 comes as a result of their search method, with each agent using either: (a) exhaustive enumeration and cannot negotiate over more than 10 issues; or (b) some random sampling with no time constrains and are able to generate at least one offer for domains with up to 25 issues. Our BAgent cannot negotiate over more than 250 issues within 2 minutes since the initialization of the dynamic-programming table takes too long.

The results of Experiment 1 support our claim over the scalability of BIDS. However, if we focus solely on scalability, similar results can be achieved by letting BAgent use other search methods (see Fig. 4). Nonetheless, apart from scalability given some time restrictions, search accuracy and diversity play a crucial role in the quality of a search algorithm.

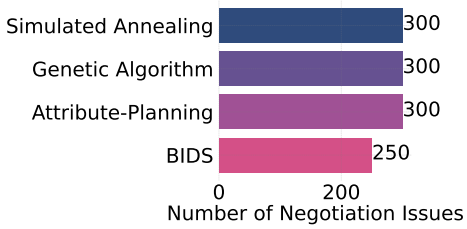

Fig. 4: Scalability results for BIDS, Attribute-Planning, Simulated Annealing, and Genetic Algorithm.

5.4 Experiment 2 - Accuracy & Diversity

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.

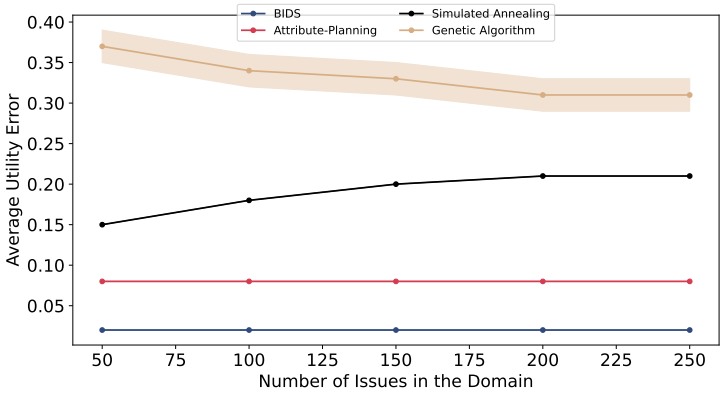

Fig. 5: Mean standard error for each scalable search method as we vary the number of issues in the outcome space. The smaller the error, the more accurate the search method is.

Our results on accuracy show (see Fig. 5) 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). 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 ω_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.

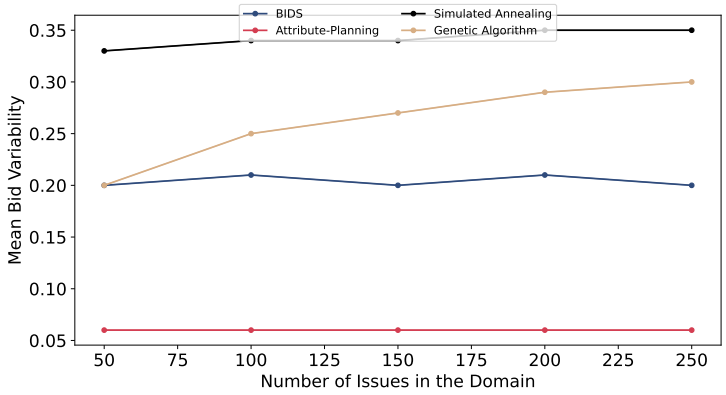

Fig. 6: Mean variability for each scalable search method as we vary the number of issues in the outcome space. The higher the variability, the more diverse the search method is.

The results on diversity show (see Fig.6) a different ranking. Here the randomness used by Simulated Annealing and Genetic Algorithm gives them 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 scale higher, our method provides higher accuracy and satisfactory diversity, having overall a better balance among the properties.

6 Conclusions & Future Work

This work presents BIDS — an algorithm that exploits the additive structure of the utility function to search very large outcome spaces while providing search accuracy and diversity. We find that BIDS can increase drastically the domain

size in which negotiating agents can still function, while providing very high accuracy and significant outcome diversity. Therefore, BIDS algorithm can enable state-of-the-art (and future) agents to negotiate over very large realistic domains such are the ones found in procurement and supply-chain management.

Future work may build on this one to evaluate the robustness of specific negotiation strategies on the accuracy and diversity of the used search method, or investigate how strategies that do not consider the space size performed compared to strategies designed for large spaces. Moreover, the existing experimental setup can be extended to evaluate the performance of BIDS when used for the utility-sampling and the trade-off queries.

Acknowledgements The research reported in this article is part of Vidi research project VI.Vidi.203.044, which is financed by the Dutch Research Council (NWO).

References

1. 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)
2. 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)
3. 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)
4. 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. Springer (2018)
5. Baarslag, T., Hindriks, K., Jonker, C.: A tit for tat negotiation strategy for real-time bilateral negotiations. In: Complex automated negotiations: theories, models, and software competitions, pp. 229–233. Springer (2013)
6. Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., Lin, R.: The first automated negotiating agents competition (anac 2010). In: New Trends in agent-based complex automated negotiations, pp. 113–135. Springer (2012)
7. 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. Springer (2019)
8. 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. IEEE (2003)
9. 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)
10. De Jonge, D., Sierra, C.: Gangster: an automated negotiator applying genetic algorithms. In: Recent advances in agent-based complex automated negotiation, pp. 225–234. Springer (2016)

11. 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. IEEE (2000)
12. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems **24**(3-4), 159–182 (1998)
13. Fujita, K., Aydoğan, R., Baarslag, T., Hindriks, K., Ito, T., Jonker, C.: The sixth automated negotiating agents competition (anac 2015). In: Modern Approaches to Agent-based Complex Automated Negotiation, pp. 139–151. Springer (2017)
14. Fujita, K., Ito, T., Baarslag, T., Hindriks, K., Jonker, C., Kraus, S., Lin, R.: The second automated negotiating agents competition (anac2011). In: Complex automated negotiations: theories, models, and software competitions, pp. 183–197. Springer (2013)
15. Hadfi, R., Ito, T.: Modeling complex nonlinear utility spaces using utility hypergraphs. In: International Conference on Modeling Decisions for Artificial Intelligence. pp. 14–25. Springer (2014)
16. Ito, T., Hattori, H., Klein, M.: Multi-issue negotiation protocol for agents: Exploring nonlinear utility spaces. In: IJCAI. vol. 7, pp. 1347–1352 (2007)
17. Jonker, C.M., Treur, J.: An agent architecture for multi-attribute negotiation. In: International joint conference on artificial intelligence. vol. 17, pp. 1195–1201. LAWRENCE ERLBAUM ASSOCIATES LTD (2001)
18. 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. Springer (2011)
19. Koeman, V.J., Boon, K., Oever, J.Z., Dumitru-Guzu, M., Stanculescu, L.C.: The fawkes agent—the anac 2013 negotiation contest winner. In: Next Frontier in Agent-Based Complex Automated Negotiation, pp. 143–151. Springer (2015)
20. 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)
21. 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)
22. Mohammad, Y., Nakadai, S., Greenwald, A.: Negmas: a platform for situated negotiations. In: International Workshop on Agent-Based Complex Automated Negotiation. pp. 57–75. Springer (2019)
23. 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. Springer (2019)
24. Niimi, M., Ito, T.: Agentm. In: Recent advances in agent-based complex automated negotiation, pp. 235–240. Springer (2016)
25. Osborne, M.J., Rubinstein, A.: A course in game theory. MIT press (1994)
26. TU Delft: GeniusWeb platform. https://ii.tudelft.nl/GeniusWeb/technicians.html (2019), [Online; accessed 04.01.2022]
27. 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 pp. 151–162 (2014)