# Less is more: Restructuring decisions to improve agent search

# Less Is More: Restructuring Decisions to Improve Agent Search

David Sarne
Department of Computer Science
Bar Ilan University
Ramat Gan 52900, Israel

Avshalom Elmalech
Department of Computer Science
Bar Ilan University
Ramat Gan 52900, Israel

Barbara J. Grosz
School of Engineering and Applied Sciences
Harvard University,
Cambridge MA 02138 USA

Moti Geva
Department of Computer Science
Bar Ilan University
Ramat Gan 52900, Israel

## ABSTRACT

In many settings and for various reasons, people fail to make optimal decisions. These factors also influence the agents people design to act on their behalf in such virtual environments as eCommerce and distributed operating systems, so that the agents also act sub-optimally despite their greater computational capabilities. In some decision-making situations it is theoretically possible to supply the optimal strategy to people or their agents, but this optimal strategy may be non-intuitive, and providing a convincing explanation of optimality may be complex. This paper explores an alternative approach to improving the performance of a decision-maker in such settings: the data on choices is manipulated to guide searchers to a strategy that is closer to optimal. This approach was tested for sequential search, which is a classical sequential decision-making problem with broad areas of applicability (e.g., product search, partnership search). The paper introduces three heuristics for manipulating choices, including one for settings in which repeated interaction or access to a decision-maker's past history is available. The heuristics were evaluated on a large population of computer agents, each of which embodies a search strategy programmed by a different person. Extensive tests on thousands of search settings demonstrate the promise of the problem-restructuring approach: despite a minor degradation in performance for a small portion of the population, the overall and average individual performance improve substantially. The heuristic that adapts based on a decision-maker's history achieved the best results.

## Categories and Subject Descriptors

[**Agent theories, Models and Architectures**]: Bounded rationality

## General Terms

Human Factors, Experimentation

## Keywords

Restructuring Decision Making

## 1. INTRODUCTION

For a variety of decision-making situations, it has been shown that people do not choose optimally or follow an optimal strategy. Research in psychology and behavioral economics has revealed various sources of this suboptimal behavior, rooted in various characteristics of human cognition and decision-making [2]. The phenomena recurs also in agents that are designed by non-specialists in decision-making theory [3]. A number of approaches have been pursued to design computer systems which will improve the decisions made by people [13, 21]. These systems may be characterized as attempting to improve a decision-maker's judgment of a situation and her ability to identify, reason about, and compare the (full set of possible) outcomes of the different choices in a decision setting in ways that yield a better decision-making process.

In this paper we use a different approach, one that parallels recent developments in psychology and behavioral economics [18]. Instead of attempting to change a decision-maker's strategy directly so that it aligns better with the optimal one, we restructure the decision-making problem itself. This strategy mimics approaches to human decision-making that remove options to allow people to focus on an appropriate set of choices and characteristics of those choices. For instance, the buyer of a used car may reach a better decision more quickly if presented with a smaller set of possible cars and only the most important characteristics of those cars. The work described in the paper is an exploration of the hypothesis that a computer agent can make better decisions in certain settings given fewer options, with the characteristics of each option adjusted to compensate for possible reasoning biases of the agent.

Restructuring processes manipulate the choices originally available to the decision-maker. Manipulations include elimination of a subset of the alternatives available and changing the values of their characteristics. The decision-maker then makes the choices it believes to be optimal given the restructured problem. To maintain the reliability of the restructured decision setting, all choices for the manipulated problem must be legitimate choices in the original problem. Similarly, all possible outcomes of each choice in the original problem should be valid in the manipulated problem.

The advantage of the restructuring approach is that it completely avoids the need to persuade the decision-maker (either an agent or a person) of the optimality and correctness of the optimal strategy. When the space of possible strategies is complex to express or the optimal strategy is non-intuitive, substantial effort may be required for such persuasion. Restructuring is also ideal when the strategy of a searcher is pre-set and cannot be changed externally, as in the case of an autonomous agent in eCommerce. This new approach does not, however, guarantee optimality. In some cases, a few decision-makers may perform slightly less well because they lose alternatives or receive inaccurate information about the possible outcomes of the different options. The results given in this paper show that, nonetheless, overall the method substantially improves the expected outcome. While the idea that manipulating information people receive can be beneficial is not new, prior work [9, 7] mainly considered settings in which non-optimal selections derive from people's computational limitations. This paper, in contrast, deals with sequential decision-making in complex settings which require structured decision-making strategies.

The application domain used in this paper for investigating the usefulness of the problem-restructuring approach is economic search. In economic search [19], the searcher chooses one of several opportunities, each associated with a distribution of gains. The actual gain from a particular opportunity can be obtained, but there

**Cite as:** Less Is More: Restructuring Decisions to Improve Agent Search, David Sarne, Avshalom Elmalech, Barbara J. Grosz and Moti Geva, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 431-438.

| Application | Goal | Opportunity | Value | Search Cost | Source of uncertainty |
|---|---|---|---|---|---|
| Marriage market | Maximize lifetime happiness | Date | Lifetime utility | Time spent, being alone while searching | Uncertainty regarding the potential spouse character |
| Job market | Optimize lifetime assets | Job interview | Offered salary and perks | Time spent, unemployment while searching | Uncertainty about potential employers |
| Product purchase | Minimize overall expense | Store | Product price | Time, communication and transportation expenses | Uncertainty regarding asked prices |

**Table 1: Mapping applications to sequential search problem**

is a cost for getting it. The searcher thus needs to take into consideration the trade-off between the cost of further search and the additional benefits of it [1, 5, 11]. The sequential decision setting of economic search models a variety of daily activities. Prior literature shows that neither people nor agents they design for this problem use the optimal search strategy [15, 3]. Furthermore, the optimal solution for an economic search problem is conceptually challenging and has many inherent counter-intuitive characteristics [19] that make it difficult to persuade people to adopt it. Economic search is thus an ideal domain in which to investigate the benefits of problem-restructuring.

The paper presents three problem manipulation heuristics for the sequential search problem, two of them non-adaptive and the third adaptive. The non-adaptive heuristics apply a fixed set of manipulation rules and do not require any prior knowledge about the searcher. The first non-adaptive heuristic, denoted "information hiding", eliminates some of the alternatives available based on the likelihood that these alternatives will not actually be needed by the optimal strategy. The second non-adaptive heuristic, denoted "mean manipulation", attempts to manipulate the distribution of gains associated with each alternative in a way that a searcher who is influenced only by means (rather than the distribution of gains) will actually end up following the optimal strategy. Finally, an adaptive manipulation heuristic, denoted "adaptive learner", is introduced for cases where results of prior interactions with the user are available. This heuristic attempts to model the decision-maker's strategy and classify it according to a set of pre-defined strategies. Based on this classification, it then applies one of the non-adaptive manipulation heuristics.

We evaluate the usefulness of the problem-restructuring method and the effectiveness of the heuristics using computer agents that were programmed by students for a search domain called "job - assignment". The results of the evaluation show that the use of manipulated choices for search problems results in search strategies that more closely resemble the optimal ones for the corresponding non-revised settings. In addition, the evaluation reveals that the heuristics differ in the nature of the improvement that individual agents achieve. With the "mean manipulation" heuristic some searchers get maximum improvement, but others substantially worsen their performance. The "information hiding" heuristic, in contrast, does not achieve the maximum possible individual improvement for any agent, but the average improvement is greater and the maximum degradation in any searcher's expected performance is substantially smaller. As expected, the "adaptive learner" heuristic produces the best results in comparison to the non-adaptive heuristics alone.

In the following section we formally present the economic search problem, its optimal solution and the complexities associated with recognizing its optimality. Section 3 explains and justifies the agent-based methodology for testing. The three heuristics are described in Section 4, and the details of the principles used for evaluating them are given in Section 5. Section 6 summarizes the results. Section 7 surveys literature from several research fields relevant to this research. Finally we conclude in Section 8.

## 2. THE SEARCH MODEL

As the underlying framework for the research, we consider the canonical sequential search problem described by Weitzman [19] to which a broad class of search problems can be mapped. In this problem, a searcher is given a number of possible available opportunities $B = \{B_1, ..., B_n\}$ (e.g., to buy a product) out of which she can choose only one. The value $v_i$ to the searcher of each opportunity $B_i$ (e.g., expense, reward, utility) is unknown. Only its probability distribution function, denoted $f_i(v)$, is known to the searcher. The true value $v_i$ of opportunity $B_i$ can be obtained but only by paying a fee, denoted $c_i$, possibly different for each opportunity. Once the searcher decides to terminate her search (or once she has uncovered the value of all opportunities) she chooses from the opportunities whose values were obtained, the one with the minimum or maximum value (depending on whether values represent costs or benefits). A strategy $s$ is thus a mapping of a world state $W = (q, B' \subset B)$ to an opportunity $B_i \in B'$, the value of which should be obtained next, where $q$ is the best (either maximum or minimum) value obtained by the searcher so far and $B'$ is the set of opportunities with values still unknown. ($B_i = \emptyset$ if the search is to be terminated at this point.) The optimal sequential search strategy $s^*$ is the one that maximizes/minimizes the expected sum of the costs incurred in the search and the value of the opportunity chosen when the process terminates.

| Project | α | ω |
|---|---|---|
| Cost | 15 | 20 |
| Rewards | (0.5,100) , (0.5,55) | (0.2,240) , (0.8,0) |

**Table 2: Information for Simplified Example.**

The search problem as so formulated applies to a variety of real-world search situations. For example, consider the case of looking for a used car. Ads posted by prospective sellers may reveal little and leave the buyer with only a general sense of the true value and qualities of the car. The actual value of the car may be obtained only through a test drive or an inspection, but these incur a cost (possibly varying according to the car make, model, location, and such). The goal of the searcher is not necessarily to end up with the most highly valued car, since finding that one car may incur substantial overall cost (e.g., inspecting all cars). Instead, most car buyers will consider the tradeoff between the costs associated with further search and the marginal benefit of a better-valued opportunity. Table 1 provides mappings of other common search applications to the model. As it suggests, a large portion of our daily routine may be seen as executing costly search processes.

While the problem is common, the nature of its optimal solution is non-intuitive. A simplified version of an example from Weitzman [19] may be used to illustrate. It deals with two possible investments. The benefits of each are uncertain and can only be known if a preliminary analysis is conducted. If funds are limited, then no more than one investment would actually be carried out. Table 2 summarizes the relevant information for decision-making: investment α might yield a total benefit of 100 with probability .5 and of 55 with probability .5 and alternative investment ω with a probability of .2 might deliver a possible benefit of 240 and no benefit with probability .8. Preliminary analysis shows costs of 15 for the α investment and 20 for ω.

The problem is to find a sequential search strategy which maximizes expected value. When reasoning about which alternative to explore first, one may notice that by any of the standard economic criteria, α dominates ω. Investment α has a lower cost, higher expected reward, greater minimum reward and less variance. Consequently, most people would guess that α should be researched first [19]. However, and somewhat paradoxically, it turns out that the

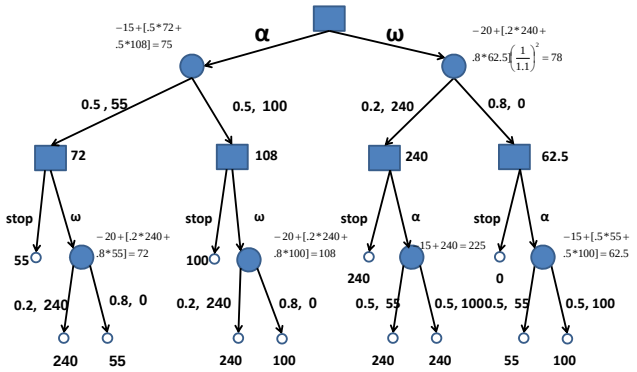


**Figure 1: Solution to simplified example**

optimal sequential strategy is to check ω first and if its payoff turns out to be zero then to develop α. This is shown in the decision tree in Figure 1.

It is quite simple to compute the optimal solution to the sequential search problem [19]. The solution is based on setting a reservation value (a threshold) denoted $r_i$ for each opportunity $B_i$. For the expected cost minimization version of the problem, the reservation value to be used should satisfy Equation 1(a):

$$(a)\ \ c_{B_i} = \int_{x=-\infty}^{r_i} (r_i - x) f_i(x) dx \quad ; \quad (b)\ \ c_{B_i} = \int_{x=r_i}^{\infty} (x - r_i) f_i(x) dx. \tag{1}$$

Intuitively, $r_i$ is the value where the searcher is precisely indifferent: the expected marginal benefit from obtaining the value of the opportunity exactly equals the cost of obtaining that additional value. The searcher should always choose to obtain the value of the opportunity associated with the minimum reservation value and terminate the search once the minimum value obtained so far is less than the minimum reservation value of any of the remaining opportunities. For revenue maximization, the reservation value should satisfy equation 1(b) and values should be obtained according to descending reservation values, until a value greater than any of the reservation values of the remaining opportunities is found.

One important and non-intuitive property of the above solution is that the reservation value calculated for each opportunity does not depend on the number and properties of the other opportunities, but rather on the distribution of the value of the specific opportunity and the cost of evaluating it.

Sequential search problems provide a good, and important, arena for investigating whether restructuring the problem is preferred over supplying the optimal search strategy to the decision-maker. As evidenced in the results section (and in prior literature [3, 15]), both people and the agents they program fail to follow the optimal strategy when engaged in sequential search. Supplying the optimal solution to the searcher may require extensive argumentation and effort because of the counter-intuitive nature of the optimal solution, in particular its myopic nature and the fact that it often favors risky opportunities [19]. A possible way to persuade a person that this is the optimal strategy is by giving her the optimality proof, but that is relatively complex and requires strong mathematical and search theory background. A possible way to persuade an agent that this is the optimal strategy is by calculating the expected value of every other sequence of decisions and compare with the expected outcome of the optimal strategy. However, the number of possible sequences for which the expected outcome needs to be calculated is theoretically infinite in the case of continuous value distributions or exponential (combinatorial) for discrete probability distribution functions. Thus, both these methods for proving optimality have substantial overhead. In contrast, the problem restructuring approach can improve performance without requiring such complex persuasion.

# 3. AGENT-BASED METHODOLOGY

We used computer agents rather than people to test the effectiveness of the general approach of restructuring the problem space and of the heuristics for manipulating the choices presented for several reasons. First, from a methodological perspective, this approach enables the evaluation to be carried out over thousands of different search problems, substantially improving the statistical quality of the result. Even more importantly, it eliminates people's computational and memory limitations as possible causes of inefficiency. The inefficiency of the agents' search is fully attributable to their designs, and result from the agent designers' limited knowledge of how to reason effectively in search-based environments.

Second, from an applications perspective, the ability to improve the performance of agents for search-related applications and tasks, especially in eCommerce, could significantly affect future markets. The importance and role of such agents have been growing rapidly. Many search tasks are delegated to agents that are designed and controlled by their users (e.g., comparison shopping). Many of these agents use non-optimal strategies. Once programmed, their search strategy cannot be changed externally, but it can be influenced by restructuring of the search problem.

Finally, the results from agent-based evaluations may be useful in predicting the way the proposed heuristics would affect people's search. Some prior research has shown close similarities between a computer agent's decisions and the decisions made by people in similar settings [15], in particular in search-based settings [3].

# 4. HEURISTICS

In this section, we define the three problem-reconstruction heuristics used in our investigations: Information Hiding, Mean Manipulation and Adaptive Learner. These heuristics differ primarily in whether they adapt to a searcher's strategy. The first two heuristics do not adapt; they assume no prior information about the searcher is available and apply a fixed set of manipulation rules. The third heuristic uses information from a searcher's prior searches to classify it and decide which of the other two heuristics to use.

For a manipulation heuristic to be considered successful, it needs not only to improve average overall agent performance, but also to avoid significantly harming the performance of any of the agents.

## 4.1 The Information Hiding Heuristic

This heuristic removes from the search problem opportunities for which the probability that their value will need to be obtained according to the optimal strategy $s^*$ is less than a pre-set threshold α. By removing these opportunities, we prevent the searcher from choosing them early in the search, yielding a search strategy that is better aligned with the optimal strategy in the early, and more influential, stages of the search. While the removal of alternatives is likely to worsen the performance of fully rational agents (ones that use the optimal strategy), the expected performance decrease is small; the use of the threshold guarantees that the probability is relatively small that these removed opportunities are actually required in the optimal search.

Formally, for each opportunity $B_i$ we calculate its reservation value, $r_i$, according to Equation 1. The probability of needing to obtain the value of opportunity $B_i$ according to the optimal strategy, denoted $P_i$, is given by $P_i = \prod_{r_j \le r_i} P(v_j \ge r_i)$ for the cost minimization version of the problem, and $P_i = \prod_{r_j \le r_i} P(v_j \le r_i)$ for its revenue maximization version. The heuristic omits from the problem every opportunity $B_i$ $(i \le n)$ for which $P_i \le \alpha$.

## 4.2 The Mean Manipulation Heuristic

This heuristic addresses the problem that people tend to overemphasize mean values, reasoning about this one feature of a distribution rather than the distribution more fully. Their search strategies typically choose to obtain the value of the opportunity for which the difference between its expected net value and the best value obtained so far is maximal. We denote this strategy "naive mean-

based greedy search". Formally, denoting the mean of opportunity $B_i$ by $\mu_i$, searchers using the naive mean-based greedy strategy calculate for each opportunity the value $w_i = \mu_i - c_i$ (in the cost maximization version) or $w_i = \mu_i + c_i$ (in the revenue minimization version) and choose to obtain the value of opportunity $B_i = argmax_{B_j}\{w_j | B_j \in B' \cap w_j \geq v\}$ (or $argmin_{B_j}\{w_j | B_j \in B' \cap w_j \leq v\}$ in the cost minimization version), where $v$ is the best value obtained so far.

The heuristic restructures the problem such that $w_i$ of each opportunity $B_i$ in the restructured problem equals the reservation value $r_i$ calculated for that opportunity in the original problem. This ensures that the choices made by agents that use the naive mean-based greedy search strategy for the restructured problem are fully aligned with those of the optimal strategy for the original problem. The restructuring is based on assigning to each opportunity $B_i$ $(0 < i \leq n)$ a revised probability distribution function $f_i^{'}$ such that $w_i^{'} = r_i$, where $r_i$ is the reservation value calculated according to Equation 1. The manipulation of $f_i$ is simple, as it only requires allocating a large mass of probability around $\mu_i^{'}$ (the desired mean, satisfying $w_i^{'} = r_i$). The remaining probability can be distributed along the interval such that $\mu_i^{'}$ does not change.

## 4.3 The Adaptive Learner Heuristic

The adaptive learner heuristic attempts to classify the strategy of a searcher and uses this classification to determine the best problem restructuring method to apply. For this purpose we need to have a representative strategy for each strategy class that has all the typical characteristics of strategies in the class. The heuristic thus requires the development of agents, denoted "class-representing agents". Each "class-representing agent" employs the representative-strategy of its class. The measure of similarity between any searcher's strategy and a given class is the relative distance between its performance and the performance of the class-representing agent for that class over the same set of problems. The searcher is classified as belonging to the class for which the relative performance distance to its representing-agent is minimal, and below a threshold $\gamma$. Otherwise, it is classified as belonging to a default class. Once the searcher is classified, the restructuring heuristic for its class can be applied. The use of the threshold $\gamma$ assures that for any agent that cannot be accurately classified, a default manipulation heuristic is used, one that guarantees no substantial possible degradation in the performance of the agent. The adaptive learner heuristic is given in Algorithm 1.

**Algorithm 1** Adaptive Learner

**Input:** $O$ - Set of prior problem instances.
$S$ - Set of strategies.
$Threshold$ - classification threshold.
**Output:** $s^*$ - the classification strategy for the searching agent (*null* if not classified or no previous data).

```
Initialization: d_s ← 0 ∀s ∈ S
for every o ∈ O do
    for every s ∈ S do
        d_s ← d_s + ||Performance_agent(o) − Performance_s(o)|| / Performance_s(o)
    end for
end for
if min{d_s} ≤ Threshold then
    return argmin_s(d_s)
else
    return null
end if
```

The algorithm receives as an input the results of prior searches and a set $S$ of strategy classes. The function $Performance_s(o)$ returns the performance of the class-representing agent $s \in S$ given the problem instance $o$ (where $Performance_{agent}(o)$ is the performance of the searcher being classified based on $o$). The algorithm returns the strategy $s^*$ to which the agent is classified (or *null*, if none of the distance measures are below the threshold set). Based on the strategy returned we apply the manipulation heuristic which is most suitable for this strategy type (or the default manipulation if *null* is returned).

The results we report in this paper are based on two strategy classes: optimal strategy and naive mean-based greedy search strategy.[1] For a searcher that cannot be classified as one of these two strategies, we use the information hiding manipulation. To produce the functionality $Performance_s(o)$ required in Algorithm 1, we developed the following two agents:

- *Optimal Agent*. This agent follows Weitzman's optimal solution [19].
- *Mean-based Greedy Agent*. This agent follows the naive mean-based greedy search strategy described in Subsection 4.2.

The more observations of prior searcher behavior that the adaptive heuristic's algorithm is given, the better the classification it can produce, and consequently the better the searcher's performance is likely to be after the appropriate manipulation is applied. Obviously, an even greater improvement in performance could be obtained if the heuristic had access to the searcher (i.e., the agent) rather than just the records describing prior searches. If direct access to the agent is allowed, then a straightforward improvement of the method would be executing the agent over each set of choices obtained, using any of the different methods and classifying it accordingly.

# 5. EVALUATION

To evaluate the three heuristics and our hypothesis that agents' performance in search-based domains can be improved by restructuring the search problem, we used a search domain called "job-assignment". Job-assignment is a classic server-assignment problem in a distributed setting that can be mapped to the general search problem discussed in this paper. The problem considers the assignment of a computational job for execution to a server chosen from among a set of homogeneous ones (servers). The servers differ in the length of their job queue. Only the distribution of each server's queue length is known. To learn the actual queue length of a server, it must be queried, an action that takes some time (server-dependent). The job can eventually be assigned only to one of the servers that were queried. The goal is to find a querying strategy that minimizes the overall time until the job starts executing. The mapping of this problem to the sequential search problem (in its cost minimization variant) is straightforward: each server represents an opportunity where its queue length is its true value and the querying time is the cost of obtaining the value of that opportunity.

## 5.1 Agent Development

The evaluation used agents designed by computer science students in a core Operating Systems course. While this group does not represent human searchers in general, it fairly represents future agent developers who are likely to design the search logic for eCommerce and other computer-aided domains. As part of her regular course assignment, each student created an agent that receives as input a list of servers, their distribution of waiting times and querying costs (times); queries the servers (via a proxy program) to learn its associated waiting time; and then chooses one of them for executing a (dummy) program. The students' grade in the assignment was correlated with their agent's performance, i.e., the time it takes until the program is executed on one of the servers.

[1] We use the optimal strategy class as a means for representing the class of strategies that are better off without applying problem restructuring. The optimal strategy is part of this class, though, as reported in the evaluation section, none of the agents we evaluated actually used the optimal strategy.

As part of their assignment, students provided documentation that described the algorithm used for managing the search for a server.

An external proxy program was used to facilitate communication with the different servers. The main functionality of the proxy was to randomly draw a server's waiting time, based on its distribution, if queried, and to calculate the overall time elapsed from the beginning of the search until the program is assigned to a server and starts executing (i.e., after waiting in the server's queue). To simplify the search problem representation, distributions were formed as multi-rectangular distribution functions. In multi-rectangular distribution functions, the interval is divided into sub intervals $x_0,..,x_n$ and the probability distribution is given by $f(x) = \frac{P_i}{x_i - x_{i-1}}$ for $x_{i-1} < x < x_i$ and $f(x) = 0$ otherwise, ($\sum_{i=1}^{n} P_i = 1$). The benefit of using a multi-rectangular distribution function is its simplicity and modularity, in the sense that any distribution function can be modeled through it with a small number of rectangles.

## 5.2 Analysis Methodology

The agents that the students developed were executed on a set of problems with the full set of search choices and no restructuring. The problems in the set varied in their characteristics (e.g., number of opportunities, characteristic of distribution functions, querying costs). Each agent was then run on each problem restructured according to the different restructuring heuristics. The performance of the agents was logged. In parallel, the class-representing optimal and mean-based greedy agents were executed over the same problem set. The results obtained by the students' agents on the non-manipulated problem set were compared with their results on the restructured problems. The results of the optimal agent were used as a baseline for evaluating the improvement achieved by each of the restructuring heuristics. Results were tested for statistical significance using t-test (with $\alpha = 0.05$), whenever applicable.

The designs of the students' agents were also analyzed to identify a set of common search strategy characteristics. We then looked for common features among agents that performed similarly.

## 5.3 Performance Measures

The evaluation of the different heuristics used two complementary measures: (1) relative decrease in the time until the job is executed; and (2) the relative reduction in search inefficiency. Formally, we denote the expected time until execution for the optimal search strategy by $t_{opt}$ and the time until execution for an agent on the manipulated and non-manipulated problem by $t_{man}$ and $t_{\neg man}$, respectively. The first measure, calculated as $\frac{t_{\neg man} - t_{man}}{t_{\neg man}}$, relates directly to the time saved. It depends on the problem set, because $t_{\neg man}$ can vary widely. The second measure, calculated as $\frac{t_{man} - t_{opt}}{t_{\neg man} - t_{opt}}$, takes into account that the search time using either the manipulated or original data is bounded by the performance of the optimal agent. It thus highlights the efficiency of the heuristic in improving performance.

For each of the two measures, the average over the entire set of problem instances and across all agents was calculated from both social and individual perspectives. For the social perspective, we calculated the relative improvement in both measures over the aggregated times obtained for all agents in all problems. For the individual perspective, we calculated the average of individual improvements for both measures. For each evaluated heuristic the maximum decrease in individual average performance was also identified, because an important requirement for a successful heuristic is that it does not substantially worsen any of the agents' individual performance.

# 6. RESULTS AND ANALYSIS

Seventy six agents, each designed by a different student, were used to evaluate the heuristics. The test set used for evaluation consisted of 5000 problems that were generated with a random number of servers in the range $(2, 20)$, costs of querying the different servers uniformly drawn from the range $(1, 100)$, and a multi-rectangular distribution function to each server generated by randomly setting a width and probability for each rectangle and then normalizing it to the interval $(0, 1000)$.

In this section we present the main analysis carried out over these agents using this problem set. The results using two other problem sets are given in Subsection 6.5.

## 6.1 Agent Strategy

The strategies students used reveal several characteristics along which agent designs vary when programmers who are not search experts do the design. Our analysis of the agents using program documentation (and occasionally the code itself) revealed several problem features commonly used in their search strategies, including expected value (in 41 of the agents), variance (in 6 of the agents), and the median (in 2 of the agents) of each server. Additional factors used in some designs were the time cost of querying servers (in 37 of the agents), randomness in the decision-making process (in 11 of the agents), a preliminary selection of servers for querying (in 57 of the agents), the inclusion of the cost incurred so far (i.e., "sunk cost") in the decision-making process (in 4 of the agents) and the use of the probability of finding a server with a lower waiting time than the minimum found so far (in two of the agents).

Several interesting observations may be made based on these characteristics. First, many of the agents use the mean waiting time of a server as a parameter that directly influences the search strategy, even though the optimal strategy is not affected directly by means (see Section 2). Second, a substantial number of agents (39 of 76) do not take into account the cost of search in their strategy. One possible explanation for this phenomena is that the designers of these strategies considered cost to be of very little importance in comparison to the mean waiting times. Interestingly, several students (11 of 76) use randomization in their search strategy, even though, as explained in Section 2, randomness is not useful for these problems (and plays no role in the optimal strategy).

The average performance of the different agents on the 5000 test problem instances is given in Figure 2. The vertical axis represents the average overall time until execution and the horizontal axis is the agent id. The two horizontal lines in the figure represent the performance of an agent searching according to the optimal strategy and an agent using the random selection rule, "randomly query a random number of servers and assign the job to the server with the lowest waiting time". As can be seen from the figure, none of the students' agent strategies reached the performance of the optimal strategy. The average overall time obtained by the agents is 445.77, while that of the optimal agent is 223.1. Furthermore, many of the strategies (41 out of 76) did even worse than a random agent.

We attempted to identify clusters of agents, based on agent performance and characteristics of agent design, as identified by our analysis of agent designs. The following clusters emerged from this assessment:

- The naive mean-based greedy search strategy and its variants (e.g., agents 3-7).
- Mean-based approaches that involve preliminary filtering of servers according to means and costs (e.g., agents 15-17).
- A variation of the naive mean-based greedy search strategy that also takes the variance of each server as a factor (e.g., agents 22-23).
- Querying the two servers with the lowest expected queue length and assigning the job to the one with the minimum value found (e.g., agents 24-27).
- Assigning the job to the first/last/random server (e.g., agents 42-71).

For many agents, similarities in performance could not be explained by resemblances among the strategies themselves. Although in some cases, agents used different variants of the same basic strategy, apparently the differences among the variants resulted in substantial differences in performance. The most interesting and

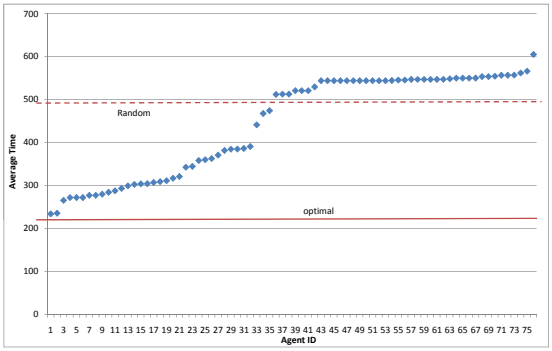


**Figure 2: Agent performance without any manipulation (to make the results easier to follow, IDs are ordered based on performance)**

.

unique strategies deployed include: (a) adding up the costs of the servers already queried, and based on this sum deciding whether to continue to the next server or to assign the job to the server with the lower execution time found so far; (b) taking 10% of the servers with the highest variance and querying them one by one, until the real value of one of them is less than the one of the server with the minimal expected value.

There was one major distinction between agent designs that led us to separate out a group of agents. Although many of the agents used search strategies that took into account affected information obtained in searching, a significant number did not follow any sequential decision-making rule, but rather queried only one server chosen arbitrarily. While any selection rule was considered legitimate for the students' assignment, strategies of the latter type are not true search strategies. Because these strategies are simple for the adaptive learner to identify (they choose a single server according to a simple pattern) and a simple problem reconstruction method could easily improve their behavior (provide only one choice: the server with the minimum sum of expected waiting time and querying time), we removed the results for these agents (agents 32-75) from the main analyses given in this paper. If included in the analysis, the improvement in the average overall performance of the adaptive agent reported in the following subsections would have been substantially better.

## 6.2 Analysis of Information Hiding

The threshold $\alpha$ used for removing alternatives from the problem instance is a key parameter affecting the "Information Hiding" heuristic. Figure 3 depicts the average time until execution (over all agents, for the 5000 problem instances) for different threshold values. For comparison purposes, it also shows the average performance on the non-manipulated set of problems, which corresponds to $\alpha = 0$ (the horizontal line). The shape of the curve has an intuitive explanation. First, for small threshold values, an increase in the threshold increases agent performance as it further reduces the possible deviation from the optimal sequence. However, as the threshold increases, the probability increases that the opportunities this increase allows to be removed are ones that would be examined by the optimal strategy.

As the graph in Figure 3 shows, the optimal threshold is $\alpha = 10\%$, for which an average time of 299.33 is obtained. This graph also shows that for a large interval of threshold values around this point — in particular for $3\% \leq \alpha \leq 30\%$ — the performance level is similar. Thus, the improvements are not extremely sensitive to the exact value; relatively good performance may be achieved even if the $\alpha$ value used is not exactly the one which yields the minimum average time. In fact, any threshold below $\alpha = 55\%$ results in improved performance in comparison to the performance obained

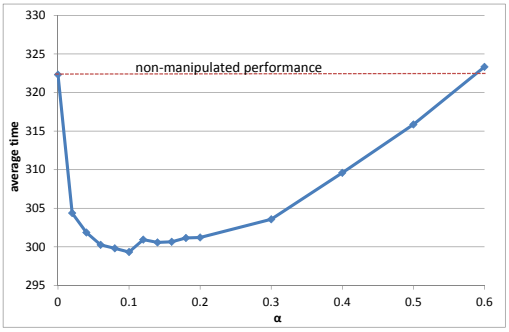


**Figure 3: The effect of $\alpha$ in "information hiding" over average performance (cross-agents)**

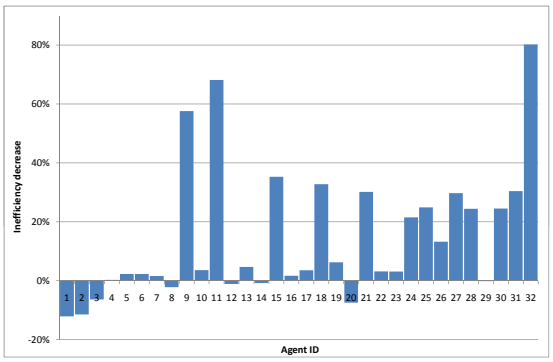


**Figure 4: Average reduction in the search inefficiency of information hiding for $\alpha = 10\%$**

without the use of this manipulation heuristic (i.e., with $\alpha = 0$).

Figure 4 depicts the average reduction in search inefficiency (over the 5000 problem instances) of each agent with $\alpha = 10\%$. As the figure shows, this heuristic decreased the search inefficiency of 24 of the 32 agents. The maximum improvement was obtained by agent 32 (80.18%). The average reduction (individual welfare) is 14.49% and the overall reduction (social welfare) is 5.52%. The downside of this heuristic is that it increases the overhead of some of the agents' searches. The highest increase in the overhead of any agent was, however, minimal and equals 12.1% (for agent 1), corresponding to an increase of 0.57% in its average time until its job starts executing.

The main advantages of this strategy are that it improves the performance of most agents and that even in cases in which an individual agent's performance degrades, the degradation is relatively small. Thus, the heuristic is a good candidate for use as a default problem-restructuring heuristic whenever there is no information about searcher strategy or an agent cannot be classified accurately.

## 6.3 Analysis of Mean Manipulation

Figure 5 depicts the average reduction in search inefficiency, using the same standard problem set, of each agent when using the "Mean Manipulation" heuristic. With this heuristic, seven agents almost fully eliminate their search overhead. These agents use variants of the naive mean-based greedy search strategy. Other agents also benefited from this heuristic and substantially reduced the overhead associated with their inefficient search. These agents (e.g., 4, 7, 25) all also included mean-based considerations, to some extent, in their search strategy.

This heuristic has a significant downside, however. Ten agents did worse with the mean manipulation heuristic, 5 of them substantially worse. The search overhead of these agents, in comparison to optimal search, increased by 50-250%. With this heuristic the overall inefficiency (social welfare) actually increased by 2.5%

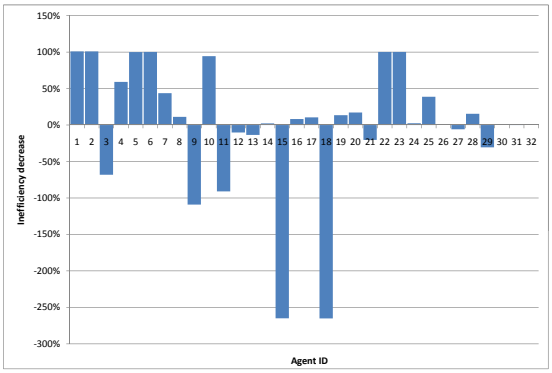


**Figure 5: Average reduction in the search inefficiency of Mean Manipulation**

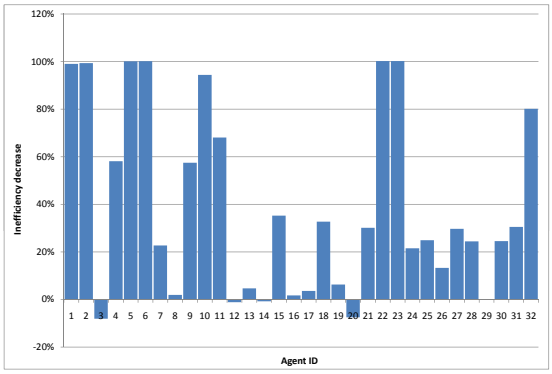


**Figure 6: Average reduction in the search inefficiency of Adaptive Learner**

even though the average overhead decreased by 1.3%, a classical case of Simpson's paradox [17]. The substantial increase in search overhead for some of the agents makes this heuristic inappropriate for general use. It is, however, very useful when incorporated into an adaptive mechanism that attempts to identify those agents that use mean-based strategies and applies this manipulation method on their input.

## 6.4 Analysis of Adaptive Learner

The adapter learner has the best of both worlds. Figure 6 depicts the average reduction in search inefficiency, using the standard problem set, of each agent when using the adaptive learner heuristic with $\gamma = 10\%$. For agents that were badly affected by the mean-manipulation heuristic, the information hiding manipulation was used instead, improving their performance. Overall, the inefficiency (social welfare) decreased by 37.4%. The average reduction in individual inefficiency (individual welfare) is 39%. Out of the 32 agents, 5 agents slightly worsened their performance (maximum of 8% increase in inefficiency, which is equivalent to a 1.3% increase in the expected waiting time of that agent).

## 6.5 Evaluation with Different Problem Sets

To show that the results were not due to a wise selection of problem instance characteristics, we repeated the evaluation with other distributions of queue lengths and different querying costs. Two problem sets were used,

- Increased possible querying time (denoted "Inc Quer"): same as the original set of problems, except that the querying time was taken from an interval that was three times as large (resulting in an increased ratio between querying time and possible waiting times in queue).
- Increased queue time variance (denoted "Inc Var"): same as the original set of problems, except that the possible waiting time interval was increased from 1,000 to 10,000 (resulting in a substantial increased variance in server waiting time in queue).

Each of these problem sets also contains 5000 different problems.

Table 3 presents the results obtained for the new problem sets in comparison to the original set. As can be seen from the table, the improvement obtained from the different heuristics is consistent with the one obtained using the original set.

| | Original | Inc. Var | Inc. Quer. |
|---|---|---|---|
| Non-Manipulated | 322.3 | 2449.4 | 461.9 |
| Adaptive | 223.1 | 2203.0 | 388.1 |
| Optimal | 285.2 | 1349.7 | 332.3 |
| Average individual improvement | 10.2% | 9.7% | 10.5% |
| Overall (social) improvement | 11.5% | 10.1% | 16.0% |
| Maximum individual performance decrease | 2.2% | 2.0% | 1.9% |
| Average individual inefficiency reduction | 39.0% | 29.0% | 46.5% |
| Overall inefficiency reduction | 37.4% | 22.4% | 56.9% |

**Table 3: Performance for different classes of problems**

# 7. RELATED WORK

People are bounded rational [16], unlike computer agents, which may deploy rational strategies and are significantly less bounded computationally. They cannot be trusted to exhibit optimal behavior [14]. Furthermore, people often tend not to use the optimal strategy even when one is provided [10]. Their decision-making may be influenced by selective search, the tendency to gather facts that support certain conclusions while disregarding other facts that support other conclusions [2], and by selective perception — the screening-out of information that one does not think is important [6]. Others [18], have attributed people's difficulty in decision-making to the conflict between a "reflective system" (e.g., involved in decisions about which college to attend, where to go on trips and in most circumstances, whether or not to get married) and an "automatic system" (e.g., that leads to smiling upon seeing a puppy, getting nervous while experiencing air turbulence, and ducking when a ball is thrown at you).

Over the years a variety of work has addressed the challenge of improving people's decision-making, mostly by developing decision support systems to assist users in gathering, merging, analyzing, and using information to assess risks and make recommendations in situations that may require tremendous amounts of the users' time and attention [21]. Recently, several approaches have been proposed that attempt to reconstruct the decision-making problem [18] instead of attempting to change people's decision-making strategies directly. This prior work focused on psychological aspects of human decision-making, and does not involve any learning or adaptation. Furthermore, none of this prior work dealt with a sequential decision-making process.

The search model discussed in this paper, which considers an optimal stopping rule for individuals engaged in costly search (i.e., ones for which there is a search cost) builds on economic search theory,[2] and in particular its sequential search model [12]. While search theory is a rich research field, its focus is on the theoretical aspects of the optimal search strategy and it does not address the non-optimality of search strategies used by people or rationally-bounded agents.

A range of research in multi-agent systems has examined people's use of agents designed to represent them and act on their behalf. For example, Kasba [4] is a virtual marketplace on the Web

[2]A literature review of search theory may be found elsewhere [12].

where people create autonomous agents in order to buy and sell goods on their behalf. Various research has involved programming agents in the decision-theoretic framework of the Colored-Trails game [8]. Here, the agents had to reason about other agents' personalities in environments in which agents are uncertain about each other's resources. In the Trading Agent Competition (TAC) [20], agents are used to collect people's strategies. Work involving people who design agents provides some evidence that people fail to build in the optimal strategy [3], in particular in search-based environments [15]. This work has not, however, provided methods for improving the performance of such agents through problem restructuring of any sort.

# 8. DISCUSSION AND CONCLUSIONS

The results reported in Section 6 are encouraging and a proof of concept for the possibility of substantially improving agent performance in sequential search by restructuring the problem space. The extensive evaluation reveals that even with no prior information regarding an agent's strategy, a heuristic such as information hiding produces substantial improvement in average performance while limiting individual potential performance degradation. With even limited information about the prior search behavior of an agent, heuristics such as the adaptive learner can further improve the overall performance and lower even further the possible decrease in individual agent performance. These results were consistent across three different classes of search environments in extensive evaluations involving a large number of agents, each designed by a different person, and a large number of problems within each class.

Restructuring of the problem space is applicable in settings for which the optimal choice cannot be revealed but rather an optimal sequential exploration should be devised, and the optimal exploration strategy cannot be provided directly to the decision-maker or the decision-maker cannot easily be convinced of its optimality. Instead, we can only control the information the decision-maker obtains in the problem.

The problem restructuring technique has great potential for market designers (who also have the domain-specific information that can lead to more intelligent restructuring heuristics). Consider for example large scale Internet websites like `autotrader.com` or `expedia.com`. These web-sites attempt to attract as many users as possible to increase their revenues from advertisements. Every listing for a flight or a car on these web-sites is an opportunity that needs to be explored further to realize its true value to the user. The welfare of users or the agents they use can thus be substantially improved by manipulating the listings.

The heuristic that provides the best performance is the adaptive learner. As our ability to recognize and differentiate additional strategy clusters and produce appropriate choice manipulations for them improves, we expect the performance improvement obtained by applying the adaptive strategy to increase even further. The adaptive heuristic's architecture is modular, allowing its augmentation using the new manipulation heuristics to be straightforward.

The research reported in this paper is, to the best of our knowledge, the first to attempt to restructure the decision-making problem in order to improve performance in a sequential decision-making setting. The fact that the searcher is facing a sequence of decisions and all manipulations over the choices take place prior to beginning the process, substantially increases the complexity for heuristics. In this case the search strategy used by the searcher becomes more complex as her decisions are also affected by the temporal nature of the problem and the new data that is being obtained sequentially. The challenge faced by the manipulation designer is thus substantially greater than in one-shot decision processes. In the latter case many simple and highly efficient manipulation techniques can be designed. For example, if the searcher is limited to obtaining the value of only one opportunity overall, the simplest and most efficient choice of a manipulation technique would be to remove all opportunities other than the optimal one.

A natural extension of this work involves developing heuristics that will choose the manipulation method to be applied not only based on agent classification but also based on problem instance characteristics. This, of course, requires a more refined analysis in the agent level.

## Acknowledgments

This work was supported by ISF/BSF grants 1401/09 and 2008-404.

# 9. REFERENCES

[1] Y. Bakos. Reducing buyer search costs: Implications for electronic marketplaces. *Mgmt. Science*, 42:1676–92, 1997.

[2] G. Blackhart and J. Kline. Individual differences in anterior EEG asymmetry between high and low defensive individuals during a rumination/distraction task. *Personality and Individual Differences*, 39(2):427–437, 2005.

[3] M. Chalamish, D. Sarne, and S. Kraus. Programming agents as a means of capturing self-strategy. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1161–1168, 2008.

[4] A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *PAAM*.

[5] S. Choi and J. Liu. Optimal time-constrained trading strategies for autonomous agents. In *Proc. of MAMA'2000*, 2000.

[6] A. Drake Roger. Processing persuasive arguments: Discounting of truth and relevance as a function of agreement and manipulated activation asymmetry. *Journal of Research in Personality*, 27(2):184–196, 1993.

[7] Y. Elmaliach and G. Kaminka. Robust multi-robot formations under human supervision and control. *Journal of Physical Agents*, 2(1):31, 2008.

[8] B. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *AAMAS*, pages 780–787, 2004.

[9] S. Iyengar. *The Art of Choosing*. Twelve, 1 edition, March 2010.

[10] D. Kahneman and A. Tversky. *Choices, values, and frames*. Cambridge University Press, New York, 2000.

[11] J. Kephart and A. Greenwald. Shopbot economics. *JAAMAS*, 5(3):255–287, 2002.

[12] J. McMillan and M. Rothschild. Search. In R. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, pages 905–927. 1994.

[13] D. Power and R. Sharda. Decision support systems. *Springer Handbook of Automation*, pages 1539–1548, 2009.

[14] M. Rabin. Psychology and economics. *Journal of Economic Literature*, pages 11–46, March 1998.

[15] A. Rosenfeld and S. Kraus. Modeling agents through bounded rationality theories. In *IJCAI'09*, pages 264–271, 2009.

[16] H. Simon. Theories of bounded rationality. *Decision and organization: A volume in honor of Jacob Marschak*, pages 161–176, 1972.

[17] E. H. Simpson. The interpretation of interaction in contingency tables. 1951.

[18] R. Thaler and C. Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Yale Univ Pr, 2008.

[19] M. L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–54, May 1979.

[20] www.sics.se/tac.

[21] E. S. Yu. Evolving and messaging decision-making agents. In *AGENTS '01*, pages 449–456, 2001.