Choosing between heuristics and strategies: an enhanced model for decision-making *

Shavit Talman, Rotem Toister, Sarit Kraus
Department of Computer Science, Bar-Ilan University
Ramat-Gan,52900, Israel
{toistet, talmans, sarit}@cs.biu.ac.il

Abstract

Often an agent that has to solve a problem must choose which heuristic or strategy will help it the most in achieving its objectives. Sometimes the agent wishes to obtain additional units of information on the possible heuristics and strategies in order to choose between them, but it may be costly. As a result, the agent's goal is to acquire enough units of information in order to make a decision while incurring minimal cost. We focus on situations where the agent must decide in advance how many units it would like to obtain. We present an algorithm for choosing between two options, and then formulate three methods for the general case where there are $k > 2$ options to choose from. We investigate the 2-option algorithm and the general *k*-option methods effectiveness in two domains: the 3-SAT domain, and the CT computer game. In both domains we present the experimental performance of our models. Results will show that applying the 2-option algorithm is beneficial and provides the agent a substantial gain. In addition, applying the *k*-option method in the domains investigated results in a moderate gain.

1 Introduction

When making decisions, automated agents need to decide which heuristic function or strategy would best assist them in achieving their objectives. In particular, when an agent possesses several alternatives to tackle a problem, and the best alternative is unknown in advance, its success depends on choosing the most beneficial alternative. In [Azoulay-Schwartz and Kraus, 2002] a formal model for solving a problem of choosing between alternatives in e-commerce have been presented. They formulated the problem in terms of units of information and agent's utility, and assumed that the agent should decide in advance on how many information units to obtain about each alternative. Moreover, the agent could not change its decision during the information obtaining process. Then they suggested an algorithm for acquiring an optimal amount of information units, through which the best alternative could be determined.

In this paper, we generalize their model to suit domains that involve choosing between heuristics or strategies. Azoulay-Schwartz and Kraus' assumptions hold in this research as well: the agent chooses the best alternative in advance and is able to hold onto its decision for a long period of time. This approach is valid in many situations, especially when acquiring information is costly. In these cases, the agent would like to execute its decision-making process once and apply the process's output thereafter. Moreover, once the agent realizes the decision made does not produce satisfactory results at any given time, it may re-execute the decision-making process, and proceed with its new output. Nevertheless, we are more concerned with the applicative rather than the theoretical aspect of the model. Indeed, the model suggests many implementation challenges. We present two examples, from the heuristics domain and from the computer games domain. In both domains we compare the utility of the automated agent with and without using our algorithm. Results will show that an agent that applies our model is more likely to choose the best option among the group of possible options. Furthermore, it does not waste too many resources during the decision process, and thus, its overall utility increases.

The paper first briefly presents the model constructed by Azoulay-Schwartz and Kraus. It then introduces the algorithm for choosing between two options, which was formulated by Azoulay-Schwartz and Kraus, and thereafter the three generalized methods for cases where there are more than two options. The first of these three methods is a theoretical one which was formulated by Azoulay-Schwartz and Kraus. We generalized their ideas to form the other two methods, which are much more feasible in real-time environments. We proceed by describing the experimental design and analysis. In section 4 we describe the first domain we considered - the 3-SAT domain, and in the following section we describe the other domain, the CT game domain. In both sections we present the experiments we conducted and the outcomes of applying the 2-option algorithm and the *k*-option generalized models. In section 6 we compare our work with previous researches. The concluding section discusses the experimental results and suggests several avenues for future research.

*This work was supported in part by NSF grant no. IIS-0208608 and by ISF grant no. 1211-04. Sarit Kraus is also affiliated with UMIACS.

2 Model construction

A risk neutral agent has to choose a heuristic from *k* heuristics to solve *M* problems. After choosing alternative *i*, the agent will obtain a value of x_i which is unknown in advance. We assume that x_i is normally distributed, that is, $x_i \sim N(\mu_i, \sigma_i^2)$. In addition, the agent does not know μ_i, but it has some prior beliefs about its distribution. We further assume that μ_i also follows the normal distribution - $\mu_i \sim N(\zeta_i, \tau_i^2)$. We assume that σ_i, ζ_i and τ_i are known and reflect the agent's believes. Otherwise, if σ_i is unknown, the agent can use The student distribution. In addition, if ζ_i and τ_i are unknown, we assume that the agent can estimate their values based on its beliefs as to the possible values of μ_i. Furthermore, in future work we will investigate the effect of eliminating normal distribution assumptions.

The agent has n_i units of information about each alternative *i*, with an average value of $\overline{x}_i$. For instance, in the e-commerce example presented by Azoulay-Schwartz and Kraus, a customer would like to buy an item available from two suppliers. The customer collected n_1 and n_2 customer-impressions from friends or from the web about these suppliers, in order to be able to decide between them. The average impression was calculated to form $\overline{x}_1$ and $\overline{x}_2$.

The agent is able to obtain a combination of $comb = (m_1, \ldots, m_k)$ additional units about the different alternatives. However, this operation is costly, either in time or in direct costs. Given $0 < \delta \leq 1$, the discount time factor, the *cost* model, the list of alternatives and the parameters for each alternative, the agent decides whether to proceed with the information it has so far, or to accumulate additional information units. One could simply use a greedy algorithm in order to find the optimal allocation of additional experiments. In each step the option which proved to be better so far is chosen, executed and its result is added to the data accumulated. The greedy algorithm stops when there is no additional sample that increases the expected utility. However, [Azoulay-Schwartz and Kraus, 2002] showed that the greedy algorithm is not optimal. The reason being that there may be situations where obtaining one unit of information about a particular alternative is not worthwhile since it will not lead to a change in the decision, but obtaining two and more may be worthwhile. In conclusion, first we will present an algorithm for *k*=2 heuristics (or strategies) and then generalize it for $k > 2$.

Algorithm 1 The HSC Algorithm

for m_A from 0 to ∞ **do**
  for m_B from 0 to ∞ **do**
    calculate Fchange.
    calculate Utility.
    if $Utility(m_A - 1, m_B - 1) > Utility(m_A, m_B)$ and $Utility(m_A - 1, m_B - 1) > Utility(m_A - 2, m_B - 2)$ **then**
      return $(m_A - 1, m_B - 1)$.
Add m_i experiments to alternative *i*.
Choose the best alternative according to the new results

2.1 Choosing Between Two Heuristics

Suppose the agent has to decide between alternatives A and B. Currently, $\overline{x}_A > \overline{x}_B$. Since the agent is risk neutral, alternative *A* will be chosen if no additional information is obtained. Firstly, the probability of changing the winning option is $Fchange(\mu_A, \mu_B, \sigma_A, \sigma_B, \overline{x}_A, \overline{x}_B, n_A, n_B, m_A, m_B) = Pr(Z > Z_\alpha)$, where *Z* is a random variable, having the standard normal distribution and Z_α represents the first value where *B* outperforms *A* (see [Azoulay-Schwartz and Kraus, 2002] for more information). Then, the expected benefits from obtaining additional m_A and m_B is a function of *Fchange* and is denoted by $benefits(m_A, m_B)$. The agent's utility is $utility(m_A, m_B) = \int_{\mu_A} \int_{\mu B} benefits(m_A, m_B) \cdot \delta^{m_A + m_B} - cost(m_A, m_B) d\mu_B d\mu_A$. The specifics of the *benefits* function depend on the *cost* model. Later we will show the specifications for three different cost models. The agent's goal is to find the pair m_A and m_B, that yields the highest $utility(m_A, m_B)$ value. By considering all possible combinations of information units about *A* and *B*, the optimal combination is achieved. Algorithm 1 presents the Heuristics-Strategies-Choosing (HSC) Algorithm for the two heuristics (or strategies) case.

Proposition 1. *The HSC algorithm stops and returns m_A and m_B that maximize the utility function.*

Proof. The intuition for the correctness of the algorithm lies in the form of the utility function. Since it consists of the multiplication of similar normal distribution functions, its form is Gaussian as well, with only one maximum point. As soon as the algorithm finds that point it stops and returns m_A and m_B. □

2.2 Choosing Between Multiple Heuristics

There are $k > 2$ alternatives, and the agent can obtain up to *M*-1 units of information about each of them. We suggest three models:

1. The *Statistical model* in which we considered the agent utility function given a combination of information units as suggested by [Azoulay-Schwartz and Kraus, 2002]. Nevertheless, in order to find a combination of additional units for each alternative, one must solve a quadruple integral. As a result, the model proved to be inapplicable for our purposes;
2. The *Binary Tree model* where the alternatives construct the leaves' level of the tree. We apply the HSC algorithm for each pair. The winning alternative goes up a level in the tree. We repeat the procedure until the best alternative reaches the root. This model deviates from our initial assumption that all experiments are conducted prior to decision-making. The binary tree model is an intermediate approach between the greedy algorithm and the HSC algorithm. However, since the comparisons are done in pairs, a large number of experiments may be executed for alternatives that would not have been considered at all when regarding all the alternatives together;
3. The *Fixed number of experiments model* - in which we distribute a fixed number of experiments denoted by (*N*) between the different alternatives using the information we have so far. For example, suppose we have five alternatives to choose from. Alternative 1 produces the best results and alternative 5 produces the worst ones. Thus, it is worthwhile to execute more experiments of alternative 1 than of alternative 5, as less time will be wasted

on fewer alternatives while accumulating information on all possible alternatives. Following preliminary experiments, we set *N* to be four times the number of alternatives. Thus, it is large enough to accommodate experiments of all alternatives and nonetheless, economical in the number of additional experiments. Consequently, for the example of five alternatives *N*=20, and we suggest that the best alternative be assigned one third of *N*. The rest will be assigned two thirds of the remaining experiments: $m_1 = [20 \cdot \frac{1}{3}] = 7$; $m_2 = [13 \cdot \frac{2}{3}] = 5$; $m_3 = [8 \cdot \frac{2}{3}] = 3$; $m_4 = [3 \cdot \frac{2}{3}] = 2$ and $m_5 = [1 \cdot \frac{2}{3}] = 1$. This ad-hoc approach will be refined in future work.

3 Experimental design and analysis

Our investigation using the HSC algorithm was conducted in two domains. The first, a classic NP-complete problem, i.e. the 3-SAT problem. To demonstrate the HSC algorithm's vast usage possibilities, it was employed in two different cost models:

1. Minimal Time (MT) scenario - the agent had to solve *M* formulas as quickly as possible;
2. Maximal Formulas (MF) scenario - the agent had *T* units of time to solve as many formulas as possible.

We chose three best-known search algorithms as the agent's possible heuristics: Greedy-SAT (GSAT), Simulated Annealing (SA) and GSAT with Random Walk. The agent had to perform its task by using one of these search algorithms. The second domain was a computer game, the CT game (for game specifications see [Grosz *et al.*, 2004]). Here, the agent's task was to maximize its game score. To this end, our agent had seven different strategies to employ against its opponent agent in the game, and it had to choose the best strategy against the opponent. The experiments compared the agent's achievements without using the HSC algorithm and its achievements after executing this algorithm. We also compared these results with methods where a large set of additional experiments had been conducted. ([Selman *et al.*, 1993] for instance conducted an unlimited number of experiments in order to choose the best heuristic). These methods found the best heuristic more frequently than our algorithm. Nevertheless, though the decisions made were slightly improved, given our cost model, the approach applied by Selman yielded a huge loss in the number of experiments executed. For space reasons, these results are not detailed here.

We first executed preliminary runs to prepare an offline database. The database comprised vital information, namely, which option is the best, and moreover, the mean quality of each option. Our hypothesis was that the execution of the HSC algorithm will indeed benefit the agent. Moreover, by executing a small number of additional experiments the agent's utility will increase.

4 The 3-SAT domain

We assumed that each Truth-assignment flip takes one unit of time. Thus, the MT scenario in this domain is a minimum problem: the agent must solve *M* formulas within a minimal number of flips. Without loss of generality, when comparing two possible heuristics, we assumed that the better heuristic after n_A experiments is heuristic *A*. As a result, the agent would have chosen heuristic *A* for the *M* formulas. In that case, it would have taken the agent the average number of flips multiplied by the number of formulas, $M \cdot \mu_A$, to solve *M* formulas. After the HSC algorithm is executed, the total number of flips will be assembled from,

1. The number of flips in the $m_A + m_B$ extra experiments yielded by the HSC algorithm;
2. The number of flips to solve the remaining $M - m_A - m_B$ formulas in case heuristic *B* prevails;
3. The number of flips to solve the remaining formulas in case heuristic *A* still leads.

δ in this case is 1, since the agent solves the required formulas in the additional experiments. Accordingly, the utility function in this domain is $utility(m_A, m_B) = \int_{\mu_A}\int_{\mu_B} benefits(m_A, m_B) \cdot \delta^{m_A+m_B} - cost(m_A, m_B)d\mu_B d\mu_A = \int_{\mu_A}\int_{\mu_B} M \cdot \mu_A - Fchange(M - m_A - m_B)\cdot\mu_B - (1-Fchange)(M-m_A-m_B)\cdot\mu_A - (m_A\mu_A + m_B\mu_B)d\mu_B d\mu_A = \int_{\mu_A}\int_{\mu_B}(M-m_A-m_B)\cdot Fchange\cdot(\mu_A-\mu_B) - m_B\cdot(\mu_B-\mu_A)d\mu_B d\mu_A$. The agent searches for m_A and m_B that maximize this equation.

The MF scenario in the domain is a maximum problem: the agent must solve as many formulas as possible within *T* flips. Considering heuristic *A* is the better heuristic after n_A experiments, the agent would have solved T/μ_A 3-SAT formulas. However, if it would have used the HSC algorithm, the total number of 3-SAT formulas would have been the summation of:

1. The number of formulas solved during the extra experiments yielded by the HSC algorithm, $m_A + m_B$;
2. The number of formulas to be solved using heuristic *B*, if it prevails, $Fchange \cdot [T/\mu_B]$;
3. The number of formulas to be solved if heuristic *A* is not changed, $(1 - Fchange) \cdot [T/\mu_A]$.

Here, δ is 1 as well, and $cost(m_A, m_B)$ consists of the number of flips lost in each case, namely $cost(m_A, m_B) = Fchange \cdot [(m_A\mu_A + m_B\mu_B)/\mu_B] + (1 - Fchange) \cdot [(m_A\mu_A + m_B\mu_B)/\mu_A]$. Lastly, the HSC algorithm will yield m_A and m_B that will maximize $utility(m_A, m_B) = \int_{\mu_A}\int_{\mu_B} m_A + m_B + Fchange \cdot \frac{T}{\mu_B} + (1 - Fchange) \cdot \frac{T}{\mu_A} - Fchange \cdot \frac{m_A\mu_A + m_B\mu_B}{\mu_B} - (1 - Fchange) \cdot \frac{m_A\mu_A + m_B\mu_B}{\mu_A} d\mu_B d\mu_A = \int_{\mu_A}\int_{\mu_B} m_B - \frac{\mu_B m_B}{\mu_A} + Fchange \cdot [\frac{T-\mu_A m_A}{\mu_B} - \frac{T-\mu_B m_B}{\mu_A} + m_A - m_B]d\mu_B d\mu_A$.

4.1 Implementation issues

We constructed 305 different 3-SAT formulas, each consisting of 100 different variables and 430 clauses. The formulas were tested in advance for the existence of a valid truth assignment. The GSAT algorithm restarted with a new random truth assignment after 5,000 flips, and the total number of restarts was set at 18. The temperature of the SA algorithm was set at 2%, and the experiment was stopped after 100,000 unsuccessful flips. The Random Walk with GSAT was implemented using three different probabilities of Walk: 50%,

Heuristic	GSAT	Simulated Annealing	Random 50%	Random 60%	Random 80%
Flips #	26.2	18.3	5.1	4.5	16.0

Table 1: Offline results of number of flips (in thousands)

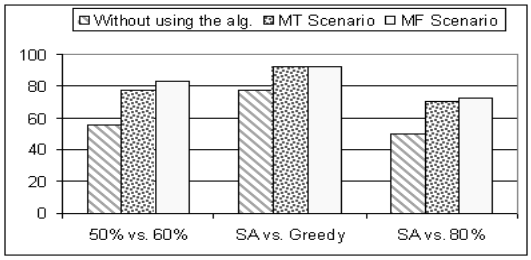

Figure 1: 3-SAT % of choosing the best heuristic without the HSC algorithm, with the algorithm in the MT and with the algorithm in the MF scenarios

60% and 80% (The three different heuristics of this algorithm will be denoted by Random 50%, 60% and 80%). In each experiment the maximal number of flips was set at 15,000 and the number of restarts was set at five. We established all the algorithms' parameters such that, on the one hand, they will have a reasonable potential to solve any formula, but on the other, their execution time will remain low.

The preliminary experiments executed each of the five heuristics on the 305 3-SAT formulas. Table 1 presents the average number of flips each heuristics obtained. The parameters in the equations of section 2 were estimated using the data accumulated during the preliminary experiments stage, since this data comprises our whole population. Thus, the a-priori parameters ζ_A and ζ_B were estimated with the mean of all the offline results of all heuristics (55.2), and τ_A and τ_B were estimated by their standard deviation (22.14). In addition, σ_A and σ_B were estimated by heuristic *A* and heuristic *B* standard deviations, respectively. A sensitivity analysis of these parameters determined that the results were not sensitive to changes in the parameters. In the analyses we varied the values of σ_A and σ_B, of ζ_A and τ_A and of ζ_B and τ_B and re-executed the experiments. Except for several extreme situations we obtained similar results for the specific values described above.

Furthermore, in each experiment n_A and n_B were set to five, *M* to 300, and *T* to 200,000. That is, in each experiment of both scenarios, five formulas chosen randomly from the 305 formulas mentioned above, were solved first, and were used as the agent's preliminary units of information. Then, it had to decide with which heuristic to proceed solving the remaining 300 formulas. We allowed the agent only five units of prior information since this is often the case in the real world - agents need to base their decision on only few observations due to uncertainty in their environment or due to a cost associated with the information.

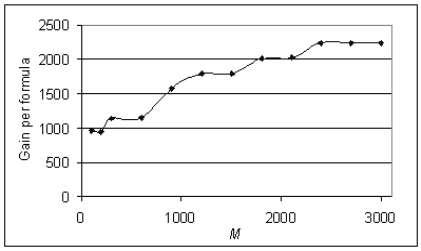

Figure 2: 3-SAT MT scenario influence of *M*

4.2 2-heuristic experimental results

According to the results presented in table 1 we compared (1) Random 60% to Random 50%; (2) SA to GSAT; (3) Random 80% to SA. Other pairs, although feasible, would not effectively demonstrate the HSC algorithm's performance: the difference between their means is too large for the algorithm to improve. From empirical results we know that the algorithm will not advise on further experiments in those cases, and simply yield the better heuristic according to the information the agent has. Thus, it will not endure any loss for these other pairs. The offline results revealed that the best heuristic in each pair was Random 60%, SA and Random 80%, respectively. The total number of experiments in each scenario was 120, 40 experiments per pair.

In the MT scenario, when the agent did not use the HSC algorithm, it always chose the heuristic with the better 5-game average. In contrast, when the agent applied the HSC algorithm, it mostly executed more experiments for both heuristics, and then either changed its mind or continued with the better 5-game heuristic. On average, only 12 additional experiments were executed for each heuristic. Figure 1 summarizes the percentage of experiments in which the agent chose the best heuristic, with and without the HSC algorithm across the three heuristic pairs. As we expected, the algorithm improved the agent's decision-making, and directed it to the best heuristic in 80% of the experiments. Moreover, without the HSC algorithm, the agent would have chosen the best heuristics only in 61% of the experiments. This improvement resulted in a gain in the number of flips: the average number of flips for 300 3-SAT formulas without the algorithm (4.20 million) was significantly higher than with the HSC algorithm (4.06 million)(Wilcoxon pv=0.07).

To show the influence of *M* on the average gain, we executed 120 additional experiments, with 12 different values for *M*. Figure 2 summarizes the average gain per formula solved for *M* varying between 100 and 3000. As expected, the average gain is in linear relation to the size of *M*, since the HSC algorithm poses a greater potential benefit as the number of formulas increases.

The agent's task in the MF scenario was to solve as many formulas as possible within 200,000 flips. Here, on average nine additional experiments were executed for each heuristic. Figure 1 presents the percentage of experiments in which the agent chose the best heuristic, with and without using the HSC algorithm (note that the percent of without the HSC algorithm is the same as in the MT scenario). In this scenario

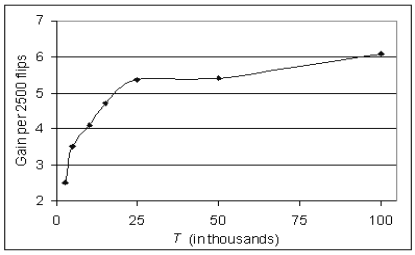

Figure 3: 3-SAT MF scenario influence of *T*

as well, the agent succeeded in choosing the best heuristic more frequently by using the algorithm (83% vs 61%). Consequently, it managed to solve six more formulas on average: 212 formulas with the HSC algorithm in contrast to 206 without it. To demonstrate the influence of *T*, we executed an additional 70 experiments for seven different values of *T* varying from 2,500 to 100,000. As the average gain is dependent on *T* (the larger the *T* the larger the gain), we normalized the gain by calculating the average gain per 2,500 flips. Thus, when *T* was set to 5,000 the average gain was divided by 2, and when is was set to 100,000 the average gain was divided by 40. Figure 3 summarizes the average gain per 2,500 flips. It supports our hypothesis that the gain of using the HSC algorithms increases as *T* increases. It seems that the increase is very steep at the beginning, but when $T > 25,000$ it becomes moderate.

4.3 *k*-heuristic experimental results

We tested the generalized *k*-heuristic model using the binary tree algorithm and the fixed number of experiments algorithm. The binary tree was built according to the pairs we described in section 4.2. That is, the binary tree's leaves were Random 50% and Random 60%, then GSAT and SA and finally Random 80%. The fixed number of experiments procedure was executed according to the description presented in 2.2. We repeated the procedures 40 times for both algorithms.

Table 2 summarizes the number of times (of the forty experiments) the agent chose each heuristic with and without each algorithm for the MT scenario. As expected, both algorithms improved the agent's decision-making, in the binary tree algorithm by 27.5% and in the fixed number of experiments algorithm by 25%. Nevertheless, the gain in the agent's utility was not very impressive in the binary tree case: on average 1,500 flips were lost per experiment due to the execution of heuristics that although seemed promising in their pair, were in fact time consuming in the overall aspect. On the other hand, in the fixed number of experiments case 1,522 flips were gained per experiment (the average total number of flips per experiment was 4,800). The number of additional experiments were 14 for each heuristic on average. Due to the disappointing results of the binary tree method, for the MF scenario we executed only the fixed number of experiments method. Here, on average 1.6 formulas more were solved per experiment when applying the algorithm.

	Greedy	SA	Random 80%	Random 50%	Random 60%
Without	1	2	2	16	19
With	0	0	1	9	30

	Greedy	SA	Random 80%	Random 50%	Random 60%
Without	1	2	2	16	19
With	0	0	0	11	29

Table 2: MT Scenario chosen heuristic distribution, with and without applying the HSC algorithm: (top) The binary tree results; (bottom) The fixed number of experiments results

5 The CT domain

We investigated the HSC algorithm in a two-player negotiation game that uses the CT game [Grosz *et al.*, 2004]. In this game each player has a goal placed on the game board and certain resources to help it reach the goal. The players can exchange resources, and at the end of the game are assigned a score that corresponds to their performance throughout the game. During the game the agents may negotiate on resources, commit to resource exchanges and execute exchanges. However, the commitments made by an agent are not enforceable, and it might decide to back down on a commitment or even deceive an opponent agent by committing to an exchange it does not intend to keep.

In [Talman *et al.*, 2005], an automated agent able to play repeated CT games was developed. The agent characterizes itself and its opponent in terms of cooperation and reliability. The cooperation trait measures the willingness of an agent to share its resources with others, whereas the reliability trait measures the agent's willingness to keep its commitments in the game. Accordingly, the agent is capable of employing seven strategies, differing in the level of cooperation and reliability the strategy dictates. Each strategy is suitable to a different type of an opponent, but the optimal matching scheme is unknown. For example, it may prove beneficial to play a low-reliability strategy against a highly-cooperative opponent by deceiving it. On the other hand, perhaps a more logical strategy against such an opponent will be a high-cooperation strategy which promotes reciprocity in the game, and may benefit the agent in the long run. Thus, in order to maximize the agent's score in the game, it must determine which strategy best-suits each opponent type. Each strategy trait can be low (L), medium (M) or high (H), and a strategy will be referred to by its cooperation-reliability level, such as a low-cooperation medium-reliability strategy (or LM strategy). Therefore, the seven possible strategies in the game are LL, LM, LH, MM, MH, HM and HH. The remaining two strategies, ML and HL are not applicable as an agent applying low reliability strategy has by definition a low cooperation level - when an agent almost never keeps it commitments, it is not willing to share its resources.

We expect that our suggested model will assist the agent in its decision-making, which will result in higher score in the game. Following n_i games of each strategy the agent executes the HSC algorithm and determines which strategy best-

Strategy	LL	LM	LH	MM	MH	HM	HH
Avg. Score	80.9	85.7	68.6	69.4	69.5	69	69.2

Table 3: Average score of each strategy against an LH opponent in the CT domain

suits each opponent type. The model adjusted to the CT game is similar to the 3-SAT model except for the introduction of the time discount factor (δ). Each game the agent plays with a non-optimal strategy results in a lower score. Hence, each experiment added by the HSC algorithm brings about a time cost of δ. In contrast, $cost(m_A, m_B) = 0$ in this domain: the execution of $m_A + m_B$ additional games does not incur an additional cost associated with m_A and m_B. In conclusion, for two strategies *A* and *B* (with *A* as the current winning strategy) having a mean score of μ_A and μ_B, respectively, the agent searches for m_A and m_B that maximize the utility function: $\int_{\mu_A}\int_{\mu_B} Fchange(\mu_B - \mu_A) \cdot \delta^{m_A+m_B} d\mu_B d\mu_A$. In addition, we define the agent's gain from applying the HSC algorithm as follows: let $Score_A$ be the score of the strategy the agent chose before the algorithm and $Score_B$ be the score of the strategy it chose after using the algorithm. Then, its total gain from the algorithm is obtained by $(Score_B - Score_A) \cdot \delta^{m_A+m_B}$.

5.1 Implementation issues

For our study, we constructed 300 game scenarios, which varied the game boards, the resources each player was assigned and the dependency relationship between the two players. For evaluation purposes the HSC algorithm was executed to establish which strategy best suits an LH type opponent (best matches for the other opponent types are carried out in a similar manner). To this end, we first executed all 300 games, in which the agent played all seven strategies against an LH type opponent. The results served as the CT domain offline database as shown in table 3. Interestingly, the LM strategy produced the best results. In addition, we set δ at 0.90, to allow a certain amount of exploration. We later varied δ and investigated its influence on the algorithm's performance.

5.2 2-heuristic experimental results

In light of the results in table 3, we compared strategies: (1) LL and LH; (2) LM and MH; (3) MM and HM. Other pairs can be compared in a similar manner as long as the difference between them is not too diverse for the HSC algorithm to introduce an improvement nor a loss. Each comparison was executed 40 times, totaling 120 experiments. In each experiment, our agent played five 2-player CT games against an LH opponent applying each strategy. Then, the agent executed the HSC algorithm, which yielded the additional number of games of each strategy the agent needs to play. On average only three games of each strategy were additionally executed.

Figure 4 summarizes the percentage of experiments in which the agent chose the best strategy, with and without using the HSC algorithm across the pairs of the three strategies. By using the HSC algorithm, the agent was able to apply the better strategy in 72% of the games, whereas without the HSC algorithm it would have applied the better strategy in only 60% of the games. Nevertheless, as we anticipated, when the

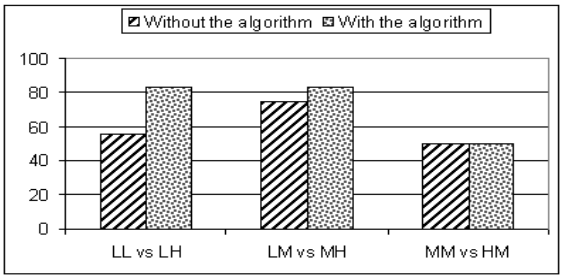

Figure 4: CT % of choosing the best strategy with and without the HSC algorithm

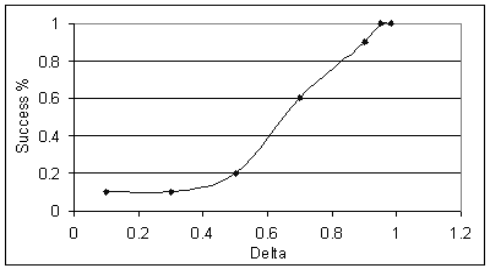

Figure 5: CT % of correct re-evaluations for different δs

difference between the two alternatives was negligible as in the MM vs HM case, the algorithm did not improve the performance. This is because the two strategies are almost identical. Thus, applying the algorithm resulted in no gain since it rarely decided to perform additional experiments. However, no loss was incurred as well (the average number of additional experiments was 3 for each alternative). Accordingly, the agent's gain was on average 0.8. This gain is somewhat low because it is multiplied by $\delta^{m_A+m_B}$.

Naturally, the time discount factor has a great influence on the HSC algorithm's performance. We envisioned that if we increase δ the HSC algorithm will shift more frequently towards the better strategy. To test our hypothesis, we randomly selected ten experiments from the experiments described above, across the three pairs, in which the worse strategy would have been chosen by the agent according to the preliminary experiments. We then re-evaluated the HSC algorithm's performance in those games with δ varying between 0.10 and 0.98. Figure 5 presents the percentage of experiments the HSC algorithm pointed out the better strategy, for each possible δ.

5.3 *k*-heuristic experimental results

We tested the generalized *k*-heuristic model using the binary tree algorithm and the fixed number of experiments algorithm. In the binary tree, the pairs were established according to the settings described in 5.2: LL vs LH, followed by MM vs HM, then LM vs MH, and finally we added the HH strategy. Then, we repeated the HSC algorithm bottom up. The fixed number of experiments procedure was executed according to the description in 2.2. We repeated the procedures 40 times for both algorithms.

The results in this domain were similar to those of the 3-SAT domain: both algorithms improved the agent's decision-making while keeping the number of additional experiments to a minimum (on average, 11 additional games per strategy) - the binary tree algorithm by 30% and the fixed number of experiments algorithm by 27.5%. The agent's gain was 0.015 in both cases. Again, this figure seems low, but any positive figure indicates a gain in performance due to the algorithm.

6 Related work

Our problem is different from the k-armed bandit problem [Tins, 1989] since once the agent decides which alternative to use it cannot change its decision. In the k-armed bandit problem, an item is chosen repeatedly and the decision of which item to select may change over time. Another approach is the statistical one. In [Pizarro *et al.*, 2000] the statistical ANOVA was employed for choosing the best model for a neural network. The procedure enabled them to isolate a subset of models whose mean error was the smallest, and subsequently the simplest model was chosen (Occams razor criteria). The main difference between that work and ours is that they were not interested in minimizing the number of additional experiments since no cost was involved. The same approach can be found in heuristics-related works. [Selman *et al.*, 1993], for instance, conducted a large number of experiments in order to identify the best heuristic. We suggest a method that will minimize the number of experiments to be executed, while providing the best option in a high percentage of the cases. In another work [Tseng and Gmytrasiewicz, 1999] developed an information-gathering system that uses the information value to guide the process. However, they assume that the number of possible answers for a query in the gathering process is finite while we consider a continuous set of possible answers. Moreover, they consider a myopic sequential procedure for the information gathering process. Thus, their solution is not optimal: they only consider the nearest step of information gathering, assuming that in each step the agent can decide about the next information to be obtained. [Grass and Zilberstein, 2000] developed a decision theoretic approach that uses an explicit representation of the user's decision model in order to plan and execute information-gathering actions. However, their system is based on information sources that return perfect information about the asked query.

In the world of strategies, researchers also have been more concerned with finding the most beneficial strategy in a given domain rather than formulating a general model. For example, in [Carlsson and Johansson, 1998] three types of strategies - generous, even-matched and greedy - were investigated as concepts for analyzing games. They granted the participating agents a strategy and examined their performances in two types of games. Their aim was to determine which kind of strategy was preferable and in which environments.

7 Conclusions and future work

In this paper, we generalized a model developed in [Azoulay-Schwartz and Kraus, 2002], which considered the problem of choosing between alternatives in e-commerce. We have presented an adjusted model which was designed to improve automated agents' decision-making. We evaluated this model in two different domains - an NP-complete problem and a computer-game framework. In each domain we executed a large number of experiments for several aspects of the domain. We have shown that agents that apply the HSC algorithm have improved their decision making by at least 20%, while executing a minimal number of additional experiments. Moreover, we have demonstrated how to adjust the HSC algorithm to suit these different domains and to perform well when solving both minimum and maximum problems.

Our future goal is to generalize the fixed number of experiments method. In this work we established its parameters empirically, and although it produced satisfactory results, we are interested in investigating its feasibility in additional domains. We also intend to examine the normal distribution assumption sensitivity. We assume that when the information units' distribution and the a-priori distribution are symmetric with one maximum, our assumptions hold. Nevertheless further analysis is required to ascertain this assumption.

References

[Azoulay-Schwartz and Kraus, 2002] R. Azoulay-Schwartz and S. Kraus. Acquiring an optimal amount of information for choosing from alternatives. In *Proceedings of the 6th International Workshop on Cooperative Information Agents VI*, pages 123–137, 2002.

[Carlsson and Johansson, 1998] B. Carlsson and S. Johansson. Generous and greedy strategies. In *Proceedings of Complex Systems*, 1998.

[Grass and Zilberstein, 2000] J. Grass and S. Zilberstein. A value-driven system for autonomous information gathering. *J. of Intel. Information Systems*, 14:5–27, 2000.

[Grosz *et al.*, 2004] 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 *Proceedings of AAMAS*, pages 782–789, 2004.

[Pizarro *et al.*, 2000] J. Pizarro, E. Guerrero, and P.L. Galindo. A statistical model selection strategy applied to neural networks. In *ESANN '2000*, pages 55–60, 2000.

[Selman *et al.*, 1993] B. Selman, H. A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In *the 2nd DIMACS Challenge on Cliques, Coloring and Satis.*, 1993.

[Talman *et al.*, 2005] S. Talman, K. Gal, M. Hadad, and S. Kraus. Adapting to agents' personalities in negotiations. In *Proceedings of AAMAS*, 2005.

[Tins, 1989] J. C. Tins. *Multi-armed Bandit Allocation Indices*. John Wiley & Sons, 1989.

[Tseng and Gmytrasiewicz, 1999] C. Tseng and P. J. Gmytrasiewicz. Time sensitive sequential myopic information gathering. In *HICSS-32*, 1999.