Proceedings of the 2004 IEEE
Conference on Cybernetics and Intelligent Systems
Singapore, 1-3 December, 2004

An Efficient Turnkey Agent for Repeated Trading with Overall Budget and Preferences

I.B. Vermeulen*, D.J.A. Somefun*, J.A. La Poutré†*
*Center for Mathematics and Computer Science (CWI)
P.O. Box 94079, Amsterdam, The Netherlands
Email:{vermeule,koye,hlp}@cwi.nl
†Eindhoven University of Technology, School of Technology Management
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

***Abstract*— For various e-commerce applications autonomous agents can do the actual trading on behalf of their users. We consider an agent who trades repeatedly on behalf of his user, given an overall budget and preferences per time step, both specified at the start. For many e-commerce settings such an agent has limited computational resources, limited prior information concerning price fluctuations, and little time for online learning. We therefore develop an efficient heuristic that requires little prior information to work well from the start, even for very roughed nonsmooth problem instances. Extensive computer experiments conducted for a wide variety of customer preferences show virtually no difference in performance between a dynamic programming (DP) approach and the developed heuristic carrying out the agent's task. The DP approach has, however, the important drawback of generally being too computationally intensive.**

I. Introduction

E-commerce is rapidly becoming a more and more important economic activity. Among other things, e-commerce provides buyers and sellers with the opportunity to conduct important aspects of trading online: i.e., the actual sales and even the actual terms of trade may be determined online. For such situations, the concept of autonomous agents who, on behalf of their users, do the actual online aspects of trading has received a lot of attention (cf. [1], [2], [3]).

In this paper, we consider the approach of having an agent trade repeatedly on behalf of his user given an *overall* budget and preferences per time step, both specified at the start. There are various goods or services (possibly offered by different sellers) from which, depending on the prices, the agent's user is willing to purchase a subset per time step. The agent's task is to determine the content of this subset per time step. Performance is measured by the overall utility as specified at the start, under the restriction of spending at most the overall budget. The agent's task is related to the problem setting in the literature on online sequential auctions (cf. [4] for related work and [5] for an overview on sequential auctions). The *novelty* of our approach lies in the fact that we address the problem of online and dynamically assigning a budget per time step based on an overall budget. In contrast, the literature on online sequential auctions focuses on bidding strategies.

With online trading, prices can fluctuate more easily in response to changing circumstances than with traditional trade (cf. [6]). Thus potentially the prices of the goods purchased can fluctuate significantly between time steps, making it more difficult for an agent to perform well. In this paper, we therefore design a heuristic, for performing the agent's task, which can deal with significant fluctuations in the prices by delaying or accelerating consumption to the future or present. The heuristic as developed is especially applicable to real world situations characterized by fluctuating prices and substitutability of consumption through time.

The heuristic utilizes the observation that under relatively mild conditions certainty equivalence holds for the agent's task: i.e., with respect to the stochastic process underlying prices the heuristic only uses the (estimated) expected values. Based on the estimated expected values the developed heuristic (re)assigns "soft" budgets to the remaining time steps, at the beginning of every time step. It uses this soft budget as a starting point for approximating, through local search, the utility gains from delaying or accelerating consumption to the remaining time steps or current time step, respectively.

Alternatively, the agent's task can also be performed by a dynamic programming (DP) algorithm (which also utilizes the certainty equivalence property). For many e-commerce applications there will be many agents simultaneously and online performing tasks on behalf of their owners. For these settings a DP approach is often too computationally intensive. The developed heuristic has the important advantage of being computationally much more efficient than a DP algorithm: for relatively small problem instances the DP algorithm already requires millions of extra iterations and this difference grows exponentially in the size of the problem instance. Moreover, extensive computer experiments conducted—in this paper—for a wide variety of customer preferences show virtually no difference in performance between a DP algorithm and the developed heuristic carrying out the agent's task.

Besides combining efficiency with a good performance, the heuristic has the additional advantages of requiring no prior information concerning the stochastic process underlying price fluctuations and working well from the start. Not requiring such prior information is an important advantage in many settings. When the agent, for example, trades on behalf of a customer in a retail market it is often not very realistic to assume that a customer can provide the agent with such (detailed) prior information about the market. The *turnkey* aspect of the heuristic (i.e., working well from the start) is another important advantage in many settings. Often the period, for which the agent trades on behalf of his user, is not long enough to warrant long learning periods.

From an agent design perspective the developed heuristic has also the advantage of allowing users to set a priori the size of the neighborhood in which the heuristic searches for improvements, with respect to the initial soft budget assignments. The smaller this neighborhood the stronger the agent is biased towards evenly spreading the expenses, all else equal. Limiting the search space can be a desirable property whenever users fear that their preferences specified at the start may not fully reflect their bias towards evenly spread expenses. Clearly, limiting the search space will go at the cost of the performance. In a second series of conducted experiments we investigate this trade off.

Application areas of the developed heuristic could be: determining on a daily basis the subscription fee for receiving all news items in a number of news categories (e.g., sport, finance, culture and science) from an online news service; buying (on a daily or hourly basis) banner-space from a number of sellers; choosing today's dinner (or lunch) from the various menus offered by various food delivery services where the source of price fluctuations may be changes in, for instance, the "menu of the day."

The next Section defines the agent's task. Section (III) and Section (IV) discuss the heuristic and DP algorithm. Experimental results are presented in Section (V) and conclusions follow in Section (VI).

II. Defining the Problem

A. Agent's Task

Starting at time $t_s = 1$, we consider a period of T time steps; per time step the agent, on behalf of the user, agrees upon a contract with the seller(s) of M goods (or services). The contract specifies prices and a so called (contract) *configuration*. A configuration specifies for all M goods whether or not a user purchases the good. Given a contract at time step t, the M dimensional vector $\vec{c}_t$ specifies the contract configuration: i.e., $\vec{c}_t(i) = 1$ if the i^{th} good is purchased and 0 otherwise (for $1 \leq i \leq M$). Morever, the M dimensional vector $\vec{p}_t$ denotes the prices (thus $\vec{p}_t(i)$ denotes the price for good i, with $1 \leq i \leq M$).

At time step t, $U(\vec{c}_t)$ denotes the utility of purchasing the goods specified by the contract configuration $\vec{c}_t$; note, the user's utility for a configuration does not depend on the time of purchase. The agent's performance depends on the overall utility obtained in the T time steps given the restriction that the overall expenses do not exceed some initially allocated budget b_1. Randomness of the prices represents a complicating factor (i.e., the sequence of prices, $\{\vec{p}_t\}_{t=1}^T$, gives rise to multivariate random variables). Let b_t denote the remaining overall budget at the beginning of time step t and h_t denote the history, which we define recursively: $h_1 = \emptyset$ and $h_t =< \{b_{t-1}, c_{t-1}, p_{t-1}\}, h_{t-1} >$ for $2 \leq t \leq T$. The agent's task is then to find a policy π, which for each time step determines the contract configuration—contingent on h_t—such that the total expected utility is maximized. More formally, a policy consist of a sequence of functions: i.e., $\pi = \{\mu_1, \ldots, \mu_T\}$, where (for $1 \leq t \leq T$) the function μ_t takes h_t, the history, as an input and outputs c_t, the current contract configuration. (Dependable on the distribution underlying the multivariate random variables $\{\vec{p}_t\}_{t=1}^T$, it may suffices to take $\{b_{t-1}, c_{t-1}, p_{t-1}\}$—or some other truncation of the history—as an input of μ_t.) The agent's task can then be defined as follows:

$$\begin{aligned} \max_{\pi} \quad & E\left[\textstyle\sum_{t=1}^T U(\mu_t(b_t))\right] \\ \text{s.t.} \quad & b_{t+1} = b_t - \vec{p}_t \cdot \mu_t(h_t), \\ & b_1 > 0, \text{ and } b_{t+1} \geq 0. \end{aligned} \tag{1}$$

$E[\cdot]$ denotes the expected value with respect to the (multivariate) random variables $\{\vec{p}_t\}_{t=1}^T$.

B. Difficulties with Agent's Task

Equation (1) gives rise to a non-linear stochastic programming problem (cf. [7]). The following four considerations, however, make standard techniques less applicable:

1)(*discreteness*) Due to the discreteness of $\vec{c}_t$ (i.e., $\vec{c}_t \in \{0,1\}^M$) there will generally not exist a closed solution to the problem and there may be multiple (local) optima. The essential difficulty with the discreteness of c_t lies in the fact that—whenever the prices of the individual goods differ—relative small changes in the expenditures could lead to a completely different optimal choice of c_t; with a small increase in the expenditures a higher priced (and much higher valued) good may be purchased at the expense of lower priced (and much lover valued) goods. Thus generally the problem specified by equation (1) can only be solved numerically.

2) (*No prior information*) For many real world application no prior knowledge is available of the stochastic process underlying the (multivariate) random variables $\{\vec{p}_t\}_{t=1}^T$.

3)(*Small* T) The period T is not long enough to warrant (online) learning of the underlying stochastic process through, for example, standard reinforcement learning techniques, cf. [8].

4)(*Limited Computational Resources*) For many agent applications there will be many agents simultaneously performing tasks on behave of their owners. To restrict the computational resources necessary for running these agents simultaneously it is a very desirable property to perform a task well while making efficient use of the computational resources.
In order to take these considerations into account we develop a so called *efficient turnkey* heuristic: it should immediately work well and be computationally efficient.

III. Efficient Turnkey Heuristic

A. Redefining the Agent's Task

The heuristic uses the fact that the agent's task is essentially a budget allocation problem. To see this, we first define at every time step t (with $1 \leq t \leq T$) the function $U^*_{t+i}(e)$, for all remaining time steps, as follows:

$$\begin{aligned} U^*_{t+i}(e) \equiv \quad & \max_{\vec{c}} \quad U_{t+k}(\vec{c}) \\ & \text{s.t.} \qquad e - E[\vec{p}_{t+k}|h_t] \cdot \vec{c} \geq 0, \end{aligned} \tag{2}$$

with $0 \leq i \leq T - t$, and $e \in \mathbb{R}^{+}$. Thus $U^{*}_{t+k}(e)$ specifies the maximum utility attainable at time step $t + i$ when the expected expenses, predicted at time step t, are at most e. The agent's task as specified in equation (1) can now be redefined as follows:

$$\begin{aligned} \max_{\pi'} \quad & \textstyle\sum_{t=1}^{T} U^{*}_{t}(E[\mu'_t(h_t)]) \qquad (3) \\ \text{s.t.} \quad & b_{t+1} = b_t - \mu'_t(h_t), \\ & b_1 > 0, \ \text{and } b_{t+1} \geq 0, \end{aligned}$$

where the policy π' consists of the sequence of functions $\{\mu'_1, \ldots, \mu'_T\}$ and (for $1 \leq t \leq T$) the function μ'_t takes h_t, the history, as an input and outputs e_t the current expenses. (We assume that the current prices are known at the beginning of the current time step.) $E[\mu'_t(h_t)]$ denotes the expected expenses at time step t. Equation (3) specifies the agent's task as a problem of allocating the overall budget b_1 to the T time steps. More generally we can say that at the beginning of time step t the agent's task is to (re)allocate the remaining budget b_t to the remaining $T - t$ time steps.

One general observation we can make, based on rewriting the agent's task, is that under certain relatively mild conditions the following property holds:

PROPERTY *1:* (*Certainty Equivalence*) It follows from equation (2) and (3) that if prices are independent of the (past) expenses (i.e., $E[\vec{p}_{t+k}|h_t] = E[\vec{p}_{t+k}|p_0, \ldots, p_t]$) then the so called certainty equivalence property holds.

With certainty equivalence the agent's task reduces to a deterministic problem, where the actual prices are replaced by the expected values of the (multivariate) random variables $\{\vec{p}_t\}_{t=1}^{T}$. Especially when there are numerous (potential) buyers, their individual impact on prices will be limited. Therefore certainty equivalence is a good starting point for the developed turnkey heuristic: i.e., with respect to the stochastic process the heuristic only uses knowledge of the expected values of the (multivariate) random variables.

B. Assigning the Soft Budgets

The heuristic performs the task of (re)allocating the overall budget remaining at time step t, b_t, by first roughly allocating b_t to the remaining $T + 1 - t$ time steps. These first budget assignments represent the "soft" budgets of the remaining time steps. The heuristics then searches in the vicinity of these soft budgets for local improvements. At the (beginning of the) current time step t, the soft budgets for the remaining time steps e^{s}_{t+i} (for $0 \leq i \leq T - t$) are assigned according to the rule

$$e^{s}_{t+i} = \frac{b_t}{(T + 1 - t)} \cdot \frac{\vec{1} \cdot E[\vec{p}_{t+i}|h_t]}{\vec{i} \cdot \vec{p}_t}. \qquad (4)$$

where $\vec{1}$ denotes a M dimensional vector of 1's (e.g., $\vec{1} \cdot p_t$ denotes the sum of the prices).

The idea underlying equation (4) is to have the soft budget assignments approximate the optimal budget assignments whenever discreteness is not a real issue. To see this, first recall that the utility for purchasing configuration $\vec{c}$ at time step t and t' results in the same utility $U(\vec{c})$: i.e., user's preferences do not differ between time steps. Furthermore if discreteness is not a real issue the preference relation can be closely approximated by a continuous utility function, which for most goods would be well behaved (cf. [9]). Consequently, the optimal budget allocation would be such that the agent is indifferent between consuming now or later. If there are no expected differences in the price level between two time steps, the overall budget is spread equally over the remaining time steps. Otherwise, the budget per time step will be monotonically increasing in the relative price level: the smallest budget is assigned to the most expensive day, the second smallest budget is assigned to the second most expensive day, and so on. Equation (4) exactly mirrors these optimal budget assignments (for smooth and well behaved utility functions) whenever there are no expected differences in the price levels, otherwise it mirrors the qualitative aspects.

C. k-Local Search

After these initial budget assignments, the idea is to accommodate for the discreteness of the problem by locally searching for improvements. To that end, the heuristic compares the utility of purchasing exactly the *current* soft budget with either spending more or less than the soft budget, in the *current* time step. Deviations from the current soft budget are shared equally among the minimum number of remaining time steps necessary to fully compensate for such a deviation. Following the budgets adjustments in the necessary time steps, the heuristics computes the average utility for all the time steps with adjusted budgets (see also algorithm (1)).

The size of the neighborhood in which the search is conducted depends on the parameter k (for a $0 \leq k \leq 2^{M}$): k determines the actual deviation from the current soft budget, by specifying the maximum deviation from the goods purchased with such a budget. In case $k = 0$, the current soft budget becomes a "hard" budget. Moreover, in case $k = 2^{M}$, the actual expenses in the current time step can fluctuate between buying 0 goods, all M goods, and any combination that lies in between. Thus k specifies how hard or soft the initially assigned budget is. Alternatively, we can interpret k as a parameter with which a user can control the agent's autonomy with respect to distributing the overall budget based on the provide preferences: the smaller k the stronger the agent is biased towards evenly spreading the budget, all else equal.

The k-local search algorithm, applied whenever $t < T$, first makes an ordering of all possible configurations of goods (2^{M}) in ascending expenditure. Next it removes all configurations with a lower utility than a configuration earlier in the order. These operations result in the list of optimal configuration for all possible expenses. Given this list, $\vec{c}'_t$, the best configuration which can be bought with the soft budget (e^{s}_{t}), is selected. The search neighborhood, Ng, is defined from k steps down, to k steps up from $\vec{c}'_t$ in the ordering (with the restriction that the neighborhood is within spending nothing and spending the complete remaining budget). Algorithm (1) gives the pseudocode for selecting the best combination in the neighborhood.

Algorithm 1 Procedure for selecting the best configuration.

```
1. $u_{max} = 0$ and $\vec{c}* = \vec{0}$
2. for all $\vec{c} \in Ng\{$
3.     $l = 1$ //stores no. future steps influenced by buying $\vec{c}$
4.     while $(\vec{p}_t \cdot \vec{c} > e^s_t + \sum_{i=1}^{l} e^s_{t+i})$ $\{l = l + 1\}$
5.     $u = \frac{1}{l+1} \cdot \left[U(\vec{c}) + \sum_{k=1}^{l} U^*_{t+k}(e^s_{t+k} - \frac{\vec{p}_t \cdot \vec{c} - e^s_t}{l})\right]$
6.     if $u > u_{max}\{$
7.         $u_{max} = u$
8.         $\vec{c}* = \vec{c}\}$
9. $\}$return $\vec{c}*$
```

D. Complexity Heuristic

The asymptotic running time of the developed heuristic depends on the procedure for k-local search. At time step t, the procedure performs at most $T+1-t$ calls to a function $U^*_{t+i}(\cdot)$ (for a $0 \leq i \leq T-t$), for every element in the neighborhood in which the search is conducted, Ng with $|Ng| \leq \min(2k, 2^M)$. Given sufficient memory, it is most efficient to compute/update $U^*_{t+i}(\cdot)$ (for all $0 \leq i \leq T-t$) at the beginning of every time step and store it in lookup tables. With lookup tables, the actual function calls can be conducted in O(1) time. The worst-case (asymptotic) upper bound on the running time at time step t, of the developed heuristic, is then $(T+1-t) \cdot \min(2k, 2^M) \cdot O(1)$ plus the cost for creating the lookup tables.

For all remaining $T+1-t$ functions $U^*_{t+i}(\cdot)$, a lookup has to be created and each table contains the utility for (at most) all 2^M possible bundles. Therefore creation of these lookup tables has a (tight) worst-case asymptotic upper bound of $O((T+1-t) \cdot 2^M)$. Thus we have the following:

PROPERTY *2:* (*Computational Complexity Heuristic*) The computational complexity of the heuristic is $O((T+1-t) \cdot (2^M + \min(2k, 2^M)))$.

Because the required storage capacity is exponential in M, the lookup tables can only be generated for relatively small values of M. This can however easily be done, for the values of M we consider.

IV. DYNAMIC PROGRAMMING BENCHMARK

We can also develop a dynamic programming (DP) algorithm for performing the agent's task (as specified by equation (3)), which like the turnkey heuristic uses certainty equivalence as the starting point (see property (1)). Such a DP algorithm only requires knowledge of the expected values of the (multivariate) random variables. An important drawback of such a DP algorithm is its efficiency; it nevertheless can serve as a benchmark for the developed heuristic.

Let B denote the domain of all possible budget values; B is bounded by the overall budget b_0 and is (generally) finite due to the discreteness of prices. Additionally, let U^*_{t+i} for all $0 \leq i \leq T-t$ be given. We can then define the DP algorithm recursively as follows:

$$\begin{aligned} V^*_T(b') &= U^*_T(b') \\ V^*_{i-1}(b) &= \max_e \left\{U^*_{i-1}(e) + V^*_i(b-e)\right\} \\ &\quad i = t, \ldots, T-1, \end{aligned} \tag{5}$$

for all $b', b, e \in B$ and $e \leq b$. The performance of the DP algorithm depends on the accuracy of the predicted price level, incorporated in the functions U^*_i.

Note that given $V^*_i(\cdot)$ the DP algorithm can obtain $V^*_{i-1}(\cdot)$ by having to compare for all $b \in B$—in the worst-case—the utility $U^*_{i-1}(e) + V^*_i(b-e)$ for all 2^M possible values of e (i.e., all possible bundle combinations given M individual goods). Thus in the worst-case there are $|B| \cdot 2^M$ calls to a function U^*_{i-1} necessary at time step i and these calls have to be repeated at all $T+1-t$ remaining time steps.

It is most efficient to use lookup tables for these function calls such that, after creating/updating these lookup tables, the actual function calls can be conducted in O(1) time; creating the tables has a (tight) worst-case asymptotic upper bound of $O((T+1-t) \cdot 2^M)$ (see Section $(III-D)$). Thus we have the following:

PROPERTY *3:* (*Complexity DP*) The worst-case complexity of the DP algorithm is $O\left((T+1-t) \cdot 2^M(1+|B|)\right)$.

In the experiments we compare the DP algorithm with the full search turnkey heuristic (i.e., $k = 2^M$). The difference in the complexity of these two procedures is the following:

PROPERTY *4:* (*Difference Complexity DP vs. Heuristic with full search*) The difference in computational complexity between the DP algorithm and the heuristic with full search is $O\left(2^M(T+1-t)(|B|-1)\right)$.

To fix ideas table (I) computes the difference in complexity between the DP algorithm and the heuristic with *full* local search, for a few examples.

	$M = 7, T = 15$	$M = 10, T = 15$
$b_1 = 100\$$	1279872	10238976
$b_1 = 1000\$$	12799872	102398976

TABLE I

EXTRA STEPS OF DP ALGORITHM;A CENT IS THE SMALLEST UNIT

V. EXPERIMENTS

A. Problem Space

To get a robust evaluation of our heuristic we experimented on a wide range of (relatively) realistic settings. In this Section we discuss settings related to the "discreteness" of the problem. In Section $(V-B)$ we discuss variations in the stochastic process underlying the prices.

a)(*Number of Goods*) The discreteness of the problem is directly related to the number of goods (the larger the number of goods the less discreteness becomes an issue, all else equal). For the applications we have in mind (choosing between various news categories, buying banner spaces, etc.)

the number of goods is relatively limited. We therefore set the number of goods M to an arbitrary but relatively small number of 7. By sufficiently varying the other problem settings, which influence the discreteness of the problem, we can nevertheless get a good insight in the (relative) performance of the heuristic in relation to the discreteness of the problem.

b)(*Curvature of* U^*) In equation (2) we define the function U_i^*, for all $1 \leq i \leq T$. (Henceforth we use U^* when discussing common aspects of these T functions.) With divisible goods U^* is generally assumed to have decreasing marginal utility in the expenses. Decreasing marginal utility arises (at least partly) due to the fact that dollars are first spent on the highest valued goods, then they are spent on the second highest valued good, and so on. Due to nondivisibility the order in which goods are purchased depends also on the available budget. Consequently, the functions U_i^* will generally have intervals with decreasing and increasing marginal utility.

In order to create a variety of functions U_i^* with different curvatures we fixed the relative difference between the prices for all our experiments. (In the conducted experiments we do however vary the (expected) price level over the various time steps.) Prices are set such that the highest valued good is also the most expensive good; the second highest valued good is the second most expensive good; and so on so forth. We then vary the shape of the functions U_i^* through the user's preference, by varying a parameter α.

PROPERTY 5: *(Property of α)* Suppose (without loss of generality) that the utility of good i, $U(i)$, is lower or equal to the utility of good $i+1$, $U(i+1)$, for all $1 \leq i < M$ (consequently $p(i+1) \geq p(i)$ also holds). Then α has the property $\frac{U(i+1)-U(i)}{p(i+1)-p(i)} = \alpha^i$ for all $1 \leq i < M$.

Thus for $\alpha = 1$, $\alpha < 1$, and $\alpha > 1$ the gains in utility of moving from the i^{th} good to the $(i+1)^{th}$ ranked good is exactly proportional, less than proportional, and more than proportional to the associated price increment. In the experiments 2.5 represents the maximum for α; for $\alpha = 2.5$ the utility-price-increment ratio grows exponentially at a rate of 2.5, which is already an extreme case. Values near 1 are probably most common in every day situation. Figure 1 depicts typical U^* functions for different values of α.

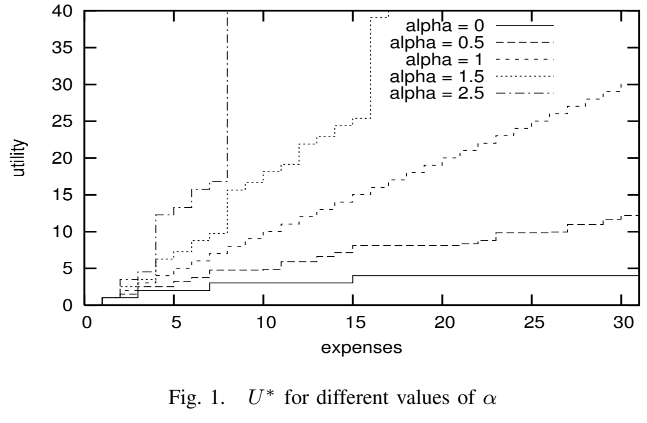

Fig. 1. U^* for different values of α

c)(*The Budget*) If the budget is relatively small, the number of goods that can be purchased is also small, which means a higher level of discreteness. Alternatively, if the budget is relatively large then it is always possible to buy the best goods, and the remaining money can be spend on goods with lesser utility (which make a significantly smaller contribution to total utility). To get a general performance measure we conduct all our experiments over a range of budget values (between 3% and 50% of the budget that buys everything). We do not experiment with a budget above 50% because then it is always possible to buy the best good at every time step (the problem becomes so simple that the extra complexity of the DP algorithm is obviously not necessary).

B. *Experimental results*

In this Section we compare the performance of the heuristic with full search to the DP algorithm benchmark. Both procedures need to compute U_{t+i}^* at the current time step t, for all remaining time step. To compute U_{t+i}^* requires estimating $E[\vec{p}_{t+i}|h_t]$. The two algorithms use the average of the past t prices as an estimator: i.e., $\hat{E}[\vec{p}_{t+i}|h_t] = \frac{1}{t}\sum_{j=1}^{t} p(t)$. We then consider three different situations for the price level:

1) no randomness, the prices are constant
2) the price level fluctuates but the expected price level is constant: i.e., the estimator is consistent
3) the price level follows a random walk: i.e., the estimator is inconsistent.

The objective of these three series of experiments is to see how both procedures perform while we increase the level of uncertainty through the stochastic process underlying the price level. We do not vary both the price level and the relative prices; controlling the discreteness of the problem through α is then no longer possible. By varying the price level we mimic the occurrence of expensive and cheap time steps.

In all the conducted experiments there are 7 individual goods, $M = 7$, and the overall period contains 15 individual time steps, $T = 15$. In order to run the benchmark in reasonable time we had to severely limit the domain ($|B| < 1000$). Such a limitation has no effect on the relative performances, but the necessity for it does show the high complexity of our benchmark. (For all results, the values presented are averaged over 35 different budgets times 30 different random seeds.)

1)(*Constant prices*) Figure (2a) depicts the relative performances for the situation with constant prices. With constant prices the agent is able to make an exact prediction of the prices of future time steps. In this situation the benchmark always gives the *optimal* solution. The performance of the heuristic is presented relative to the performance of our benchmark, for different values of α. We can observe that our heuristic performs near optimal over the whole range.

2)(*Consistent Estimator*) The price level at each time step is drawn from an independent uniform distribution which lies between -40% and $+40\%$ of the average level. Figure (2b) depicts the results of the conducted experiments.

Due to certainty equivalence (see property (1)) the benchmark would be optimal whenever the expected prices would be

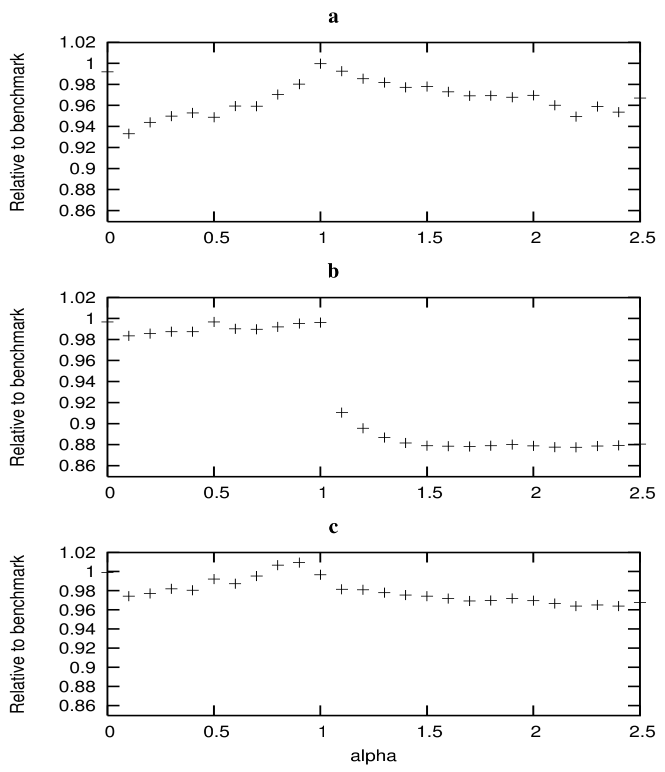

Fig. 2. Relative performances: a) constant prices, b) consistent predictor, c) inconsistent predictor, for different values of α

known. Since both approaches use a consistent estimator, the benchmark still has a relative good performance. For $\alpha < 1$ the difference is minimal, for $\alpha > 1$ we observe a larger difference, but the heuristic still has a good performance.

3)(*Inconsistent Estimator*) The price level follows a random walk. The price level at each time step is the price level of the previous time step plus noise. The noise is randomly drawn from a uniform distribution which lies between $+15\%$ and -15% of the price level at the first time step. (We add the restriction that the price level can never drop below 10% of the price level of the first time step.) For this underlying stochastic process certainty equivalence also still holds. The idea is, however, that the heuristic may suffer less from larger prediction errors.

Figure 2c depicts the relative performance for different values of α. The fact that both procedures use an inconsistent estimator has a larger negative effect on the dynamic programming approach. Our heuristic with a full search performs almost just as well, and in some cases even slightly better.

C. Limiting local search

Another advantage of our heuristic is that the user has control over the autonomy of the agent. By limiting the local search range a user can, ensure more distributed spending over the whole period (rule out extremes) and limit the complexity, at the cost of reduced optimality. Figure 3 depicts the performances for smaller values of k relative to full search (for the random walk case).

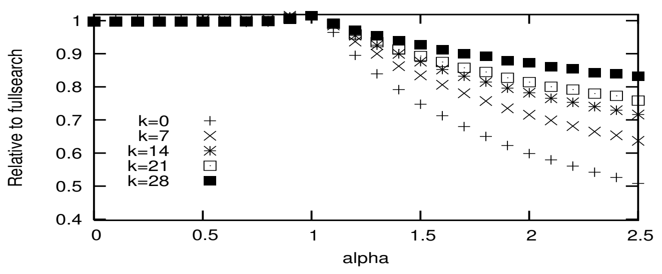

Fig. 3. Performances for different search fields relative to full search

VI. Conclusions

We have presented an efficient turnkey heuristic for repeated trading with a bounded overall budget, and compared its performance to a dynamic programming (DP) benchmark. The heuristic uses a soft budget to specify the expenses at each time step, within the vicinity of the soft budget it then searches for improvements. Extensive computer experiments conducted for a wide variety of customer preferences show virtually no difference in performance between a DP algorithm and the developed heuristic carrying out the agent's task. The DP algorithm has, however, the important drawback of generally being too computationally intensive. Moreover the heuristic's performance improves further when both procedures use an inconsistent estimator; for a certain range of problems it even slightly outperforms the DP benchmark. An additional virtue of the developed heuristic is that it allows the user to control the agent's autonomy. A second series of experiments illustrates the trade-off between limiting the local search and the performance.

References

[1] M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam, "Negotiating complex contracts," *Group Decision and Negotiation*, vol. 12, pp. 111–125, 2003.

[2] P. Faratin, C. Sierra, and N. R. Jennings, "Using similarity criteria to make issue trade-offs," *Journal of Artificial Intelligence*, vol. 142, no. 2, pp. 205–237, 2003.

[3] D. J. A. Somefun, E. Gerding, S. Bohte, and J. A. La Poutré, "Automated negotiation and bundling of information goods," in *Proceedings Agent Mediated Electronic Commerce V*, ser. LNAI, J. A. Rodriguez-Aguilar (et al.), Ed. Berlin: Springer-Verlag, 2004, vol. 3048.

[4] C. Boutilier, M. Goldszmidt, and B. Sabata, "Sequential auctions for the allocation of resources with complementarities," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999, pp. 527–523. [Online]. Available: citeseer.ist.psu.edu/boutilier99sequential.html

[5] V. Krishna, *Auction Theory*. Academic Press, 2002.

[6] R. Amit and C. Zott, "Value creation in ebusiness," *Strategic Management Journal*, vol. 22, pp. 493–520, 2001.

[7] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning. Cambridge Massachusetts: The MIT Press, 1998.

[9] A. Mas-Collel, M. D. Whinston, and J. R. Green, *Mircoeconomic Theory*. Oxford University Press, 1995.