KYBERNETIKA — VOLUME *49* (2013), NUMBER 5, PAGES 720–737

APPROXIMATE DYNAMIC PROGRAMMING BASED ON HIGH DIMENSIONAL MODEL REPRESENTATION

Miroslav Pištěk

This article introduces an algorithm for implicit High Dimensional Model Representation (HDMR) of the Bellman equation. This approximation technique reduces memory demands of the algorithm considerably. Moreover, we show that HDMR enables fast approximate minimization which is essential for evaluation of the Bellman function. In each time step, the problem of parametrized HDMR minimization is relaxed into trust region problems, all sharing the same matrix. Finding its eigenvalue decomposition, we effectively achieve estimates of all minima. Their full-domain representation is avoided by HDMR and then the same approach is used recursively in the next time step. An illustrative example of N-armed bandit problem is included. We assume that the newly established connection between approximate HDMR minimization and the trust region problem can be beneficial also to many other applications.

Keywords: approximate dynamic programming, Bellman equation, approximate HDMR minimization, trust region problem

Classification: 90C39

1. INTRODUCTION

The main focus of this article is to develop an approximate tool suitable for enlarging the class of computationally feasible decision-making problems. It copes with the principal problem within the stochastic dynamic programming, which is known as the *curse of dimensionality*, see [21]. The central notion of stochastic dynamic programming is the Bellman function, see for instance [11]. Once we are able to find and store this function, it is easy to derive the optimal strategy. However, the exact calculation of the Bellman function is computationally infeasible in the majority of practical applications, and also its representation as a lookup-table is intractable.

Next, we present a survey of approximate solutions to these problems. One way to reduce the size of the lookup-table is to aggregate the state space of the original problem into smaller sets. As it is not clear how to pick the best level of aggregation, several methods of multiple-level aggregation are developed [5]. A similar way to lookup-table reduction is approximation of the Bellman function which does not require any simplifications in the state space. A grid-based approximation with value interpolation is a typical example of such method [8]. The Bellman function can also be estimated using regression models which are able to exploit specialized structures (“basis functions”) in the state space [12]. Nonetheless, such methods are suitable for maximally hundreds of regression parameters. Another tool suitable for approximation is the artificial neural networks utilized to learn the shape of the Bellman function, see [17] and references therein. Based on random sampling of the state space, a variety of Monte Carlo methods may be also applied, see for instance [14]. Temporal Difference methods are of quite a different nature. Opposite to the algorithm developed later, they do not operate with system model. They use simulated or experience-based sampling of system trajectories instead, and thus they have no ambition to cover the whole state space. Nonetheless, they definitely do well for many real-world problems [7, 9, 27].

In this article, we develop a new approximate technique which considerably reduces both computational and memory demands of a decision-making problem. To this end, an approximation tool called High Dimensional Model Representations (thereinafter "HDMR") is useful [22]. It was applied to continuous function approximation in calculating reliability of uncertain mechanical systems [23]. It was also utilized for solution of stochastic partial differential equations [15] and compared to Monte Carlo sampling. Another application of HDMR was volatility calibration [3] where it was compared to cubic spline approximation. These successful implementations of HDMR in other fields encourage us to apply it to approximate dynamic programming. In the previous applications, it was used mainly for reducing the amount of the data. The memory space necessary to store all the values of function $g(x_1, \dots, x_d)$ grows exponentially with the dimension d, whereas the size growth of HDMR components is just quadratic in d. This is, of course, important even in our case, but the newly established fact that HDMR permits fast approximate minimization may be even more essential in the context of the decision making theory.

The outline of this work is as follows. Section 2 deals with the approximation technique of HDMR, which is determined by a system of linear equations. Its linearity does not match with the inherently non-linear Bellman equation. On that account, an algorithm for approximate minimization of function having HDMR form is developed in Section 3. Then, the current state of the art in the decision making theory is summarized at the beginning of Section 4. Next, a viable technique for approximate decision making based on HDMR is introduced there, and then the N-armed bandit problem is tackled as an example. Section 5 is devoted to conclusion.

Throughout this work, a few general conventions are followed. The domain of the quantity x is denoted X, $x \in X$, $|X|$ denotes the count of elements of finite set X. Next, x_m denotes mth coordinate for vector valued quantity $x \subset \mathbb{R}^d$, $x = (x_1, \dots, x_d)$. This convention holds with one exception: if we use letter t as a subscript, e. g. x_t, it stands for quantity x at the time instant $t \in T$ with T finite. Next, we reserve letter "f" for conditional probability density functions, arguments in the condition are separated by "$|$" in the argument list. For the domain of function $h(x)$ we use $\operatorname{dom}(h)$, and HDMR of $h(x)$ is marked by $\widetilde{h}(x)$ with several exceptions pointed out later.

2. HIGH DIMENSIONAL MODEL REPRESENTATION

The approximation technique of HDMR has a particularly simple form [22]. For a general function $g(x)$, the second order HDMR $\tilde{g}(x)$ reads

$$\tilde{g}(x) = \tilde{g}_{\emptyset} + \sum_{m=1}^{d} \tilde{g}_m(x_m) + \frac{1}{2} \sum_{m,n=1}^{d} \tilde{g}_{mn}(x_m, x_n). \tag{1}$$

Here, $\tilde{g}_{\emptyset}$ denotes a constant value over dom(g); one-dimensional functions $\tilde{g}_m(x_m)$ describe independent effects of each particular coordinate x_m, and two-dimensional functions $\tilde{g}_{mn}(x_m, x_n)$ represent the joint effect of coordinates x_m and x_n. In the context of HDMR, these functions are called zero-order, first-order, and second-order components of HDMR, respectively. Experience shows that second-order HDMR provides a sufficient approximation of $g(x)$ as only low-order correlations amongst the input variables have a significant impact upon the outputs of a typical model [3, 15, 23].

Remark 2.1. (Zero Mean Property of HDMR Components) We note that for any HDMR $\tilde{g}(x)$ we may assume that identities

$$\sum_{x_m \in X_m} \tilde{g}_m(x_m) = 0 \tag{2}$$
$$\sum_{x_m \in X_m} \sum_{x_n \in X_n} \tilde{g}_{mn}(x_m, x_n) = 0$$

are satisfied for all $m, n \in \{1, \dots, d\}$. If this is not the case, we shift each component in (1) by the respective auxiliary constant $\sigma_{\emptyset}, \sigma_m, \sigma_{mn}$ in such a way that (2) holds non-changing the overall value of $\tilde{g}(x)$. The details are left to an interested reader as an exercise.

There are many ways how to construct HDMR [2, 22]. Typically, it is constructed as an approximation error minimizer in L^2-norm $\|h\|^2 = \int_X h(x)^2\, dx$. In practise, however, a more difficult situation may occur if the approximated function $g(x)$ is defined only on a strict subset of X, dom$(g) = R \subsetneq X$. Under such conditions, it is important not to consider points $X \setminus R$ when constructing HDMR $\tilde{g}(x)$. Thus, instead of L^2-norm, we have to use a weighted norm

$$\|h\|^2_{\chi_R} = \int_X \chi_R(x)\, h(x)^2 \,\mathrm{d}x = \int_R h(x)^2 \,\mathrm{d}x \tag{3}$$

with a weight function equal to the characteristic function

$$\chi_R(x) = 1 \quad \text{for} \quad x \in R, \qquad \chi_R(x) = 0 \quad \text{for} \quad x \notin R. \tag{4}$$

The optimal HDMR of the function $g \in L^2_{\chi_R}(X)$ is now defined as a minimizer of the approximation error $\|g - \tilde{g}\|_{\chi_R}$. Then, the uniqueness of projection on closed subspaces of $L^2_{\chi_R}(X)$ implies the uniqueness of $\tilde{g}(x)$. Nonetheless, there may exist various components $\tilde{g}_{\emptyset}$, $\tilde{g}_m$ and $\tilde{g}_{mn}$ summing up to the same $\tilde{g}$, cf. also Remark 2.1.

For $R = X$ we can directly minimize the approximation error with respect to (3) obtaining component-wise decoupled equations matching "ANOVA-HDMR" in [22]. However, for general $R \subsetneq X$, we obtain one large linear system determining all the optimal HDMR components of $\tilde{g}(x)$, see [20]. For smaller problems this system may be computationally feasible; nonetheless, a more convenient way is to obtain decoupled equations for HDMR components again. Thus, instead of searching for an optimal approximation within the class of all functions having HDMR form (1), we search for it within a smaller class of functions having mutually independent HDMR components.

In this article we are interested in a discrete setting of HDMR. Let X be d-dimensional product of finite sets X_i

$$X = \prod_{i=1}^{d} X_i, \tag{5}$$

and let the integration in (3) be summation over $R \subset X$. Next, for any subset of indices $I \subset \{1, \dots, d\}$ we define

$$X_I^{\perp} = \prod_{i \in \{1,\dots,d\} \setminus I} X_i. \tag{6}$$

Then, the optimal HDMR may be obtained using marginal operators defined for a function h as

$$\begin{aligned} \mathrm{M}_{\emptyset}[h] &= \sum_{y \in X} h(y_1, \dots, y_d) \\ \mathrm{M}_m[h](x_m) &= \sum_{y \in X_m^{\perp}} h(y_1, \dots, y_{m-1}, x_m, y_{m+1}, \dots, y_d) \\ \mathrm{M}_{mn}[h](x_m, x_n) &= \sum_{y \in X_{mn}^{\perp}} h(y_1, \dots, y_{m-1}, x_m, y_{m+1}, \dots, x_n, \dots, y_d). \end{aligned} \tag{7}$$

Proposition 2.2. (Weighted HDMR with Mutually Independent Components) Consider a finite set X in a form of (5) and function g such that $\mathrm{dom}(g) = R \subset X$. Then, HDMR $\tilde{g}(x)$ minimizing approximation error $\|g - \tilde{g}\|_{\chi_R}$ such that $\tilde{g}_{\emptyset}$ does not depend on any other HDMR component, each $\tilde{g}_m(x_m)$ depends only on $\tilde{g}_{\emptyset}$, and finally each $\tilde{g}_{mn}(x_m, x_n)$ depends only on $\tilde{g}_m(x_m)$, $\tilde{g}_n(x_n)$ and $\tilde{g}_{\emptyset}$, is given by

$$\begin{aligned} \tilde{g}_{\emptyset} &= \frac{\mathrm{M}_{\emptyset}[\chi_R \cdot g]}{\mathrm{M}_{\emptyset}[\chi_R]}, \\ \tilde{g}_m(x_m) &= \frac{\mathrm{M}_m[\chi_R \cdot g](x_m)}{\mathrm{M}_m[\chi_R](x_m)} - \tilde{g}_{\emptyset}, \\ \tilde{g}_{mn}(x_m, x_n) &= \frac{\mathrm{M}_{mn}[\chi_R \cdot g](x_m, x_n)}{\mathrm{M}_{mn}[\chi_R](x_m, x_n)} - \tilde{g}_m(x_m) - \tilde{g}_n(x_n) - \tilde{g}_{\emptyset}, \\ \tilde{g}_{mm}(x_m, x_m) &= 0. \end{aligned} \tag{8}$$

Proof. We build the second order HDMR in three steps. First, we compute zero order component $\tilde{g}_{\emptyset}$ in such a way that it minimizes the approximation error $\|g - \tilde{g}_{\emptyset}\|_{\chi_R}$. In the next step we fix this component and find such first order components $\tilde{g}_m(x_m)$ that minimize approximation error $\|g - \tilde{g}_{\emptyset} - \tilde{g}_m\|_{\chi_R}$ with $\tilde{g}_{\emptyset}$ kept fixed. Finally, we find second order components $\tilde{g}_{mn}(x_m, x_n)$ as minimizers of $\|g - \tilde{g}_{\emptyset} - \tilde{g}_m - \tilde{g}_n - \tilde{g}_{mn}\|_{\chi_R}$ with $\tilde{g}_{\emptyset}$, $\tilde{g}_m(x_m)$, and $\tilde{g}_n(x_n)$ kept fixed. The optimality conditions for such HDMR may be derived in three steps where each step is analogous to the original derivation of the full HDMR [22]. $\square$

We note that this simple construction of HDMR is beneficial to our application, as the domain of the Bellman function could be too large to operate with all the function values at once. Still, its HDMR components can be computed by point-wise evaluation of the function values which are immediately added to proper sums in (8).

3. FAST MINIMIZATION OF HDMR

In this section, the main novelty of this article is developed. The key ingredient of the proposed approximate dynamic programming technique is a fast approximate minimization of functions in HDMR form. We consider function $\tilde{g}(x,z)$, $\mathrm{dom}(\tilde{g}) = X \times Z$, having the following structure

$$\tilde{g}(x,z) = \frac{1}{2}\sum_{m,n=1}^{\mu} \tilde{g}_{mn}(z_m,z_n) + \sum_{m=1}^{\mu} \tilde{g}_m(z_m) + \sum_{m=1}^{\mu}\sum_{n=1}^{\kappa} \tilde{g}_{\mu+n,m}(x_n,z_m), \tag{9}$$

where we denoted by κ and μ the dimension of X and Z, respectively. This function corresponds to full HDMR of $\tilde{g}(x,z)$ without all HDMR components independent of z. Since we are interested in a point-wise minima of $\tilde{g}(x,z)$,

$$p(x) = \min_{z\in Z} \tilde{g}(x,z), \tag{10}$$

the previous restriction on components of $\tilde{g}(x,z)$ is without loss of generality and it considerably eases the notation.

Regardless of a specific choice of $x \in X$, the parametrized minimization in (10) is equivalent to the search for the clique of the minimal weight in a complete multi-partite edge-weighted graph [16]. To show it, identify different Z_m as partite sets of the graph, $z_m \in Z_m$ as vertices in particular partite set Z_m and $\tilde{g}_{mn}(z_m,z_n)$ as weight of edge between vertices $z_m \in Z_m$ and $z_n \in Z_n$ taken from distinct partite sets with $\tilde{g}_{mm} = 0$, as we claimed in (2). The additional weights of vertices $\tilde{g}_m(z_m)$ and $\tilde{g}_{\mu+n,m}(x_n,z_m)$, the latter parametrized by $x \in X$, can be simply added to the weights of proper edges. This problem is known to be NP-hard [10] and as it plays a role of repeatedly solved sub-problem here, we search only for an approximate solution of (10).

At the moment, we rewrite function $\tilde{g}(x,z)$ in a more convenient form. For a finite set B and $i \in \{1,\dots,|B|\}$ we denote $B[i]$ the ith element of B. Then, for all $m,n \in \{1,\dots,\mu\}$ we define matrices F^{mn} in this way

$$F^{mn}_{ij} = \tilde{g}_{mn}(Z_m[i], Z_n[j]). \tag{11}$$

In the same manner, we define matrices G^{mn}

$$G^{mn}_{ij} = \tilde{g}_{mn}(Z_m[i], X_n[j]) \tag{12}$$

for all $m \in \{1,\dots,\mu\}$ and $n \in \{1,\dots,\kappa\}$ and vectors h^m

$$h^m_i = \tilde{g}_m(Z_m[i]) \tag{13}$$

for all $m \in \{1,\dots,\mu\}$. Further, we compose all matrices F^{mn} into one matrix F with F^{mn} being the mnth sub-block of F. Similarly, we create matrix G out of matrices G^{mn}

and vector h consisting of sub-vectors h^m. Thus, we obtain a concise reformulation of $\tilde{g}(x,z)$

$$\gamma(u,v) = \frac{1}{2}\, v^T F v + h^T v + u^T G v, \tag{14}$$

where the only question left is to clarify the relation between vectors u, v, and the original variables $x \in X$, $z \in Z$, respectively.

We define

$$\theta = \sum_{m=1}^{\mu} |Z_m| \tag{15}$$

and follow the logic of the previous construction to deduce the structure of the newly introduced vector $v \in \mathbb{R}^\theta$. We see that it consists of μ sub-vectors

$$v^m \in \{0,1\}^{|Z_m|}, \tag{16}$$

which are related to coordinates $z_m \in Z_m$ of the original variable $z \in Z$ as

$$v_i^m = 1 \Longleftrightarrow z_m = Z_m[i], \qquad v_i^m = 0 \quad \text{otherwise.} \tag{17}$$

The relation of the vector u to the original parameter $x \in X$ is analogous. Such constructions of $v(z)$ and $u(x)$ guarantee that

$$\gamma(u(x), v(z)) = \tilde{g}(x,z), \tag{18}$$

for all $(x,z) \in X \times Z$, and thus the evaluation of $p(x)$, see (10), is fully equivalent to the minimization of $\gamma(u(x), v)$ with respect to all vectors v obeying (17). Therefore, the latter problem is also a NP-hard problem. It is, however, more amenable to the relaxation technique developed further.

3.1. Trust region based relaxation

We observe that each $x \in X$ in (10) yields a different value of parameter u in (14) while matrix F remains unchanged. Thus, we can afford some intensive matrix preprocessing in order to exploit the repetitive nature of this minimization. This is why we turn our attention to the trust region problem [26] which permits fast exact solution even for an indefinite matrix F. To match the form of the trust region problem, we have to relax constraints (17) into $\|v\| = r$ with $r > 0$ specified lately. Thus, we obtain problem

$$\min_{\|v\| \le r} \left\{ \frac{1}{2}\, v^T F v + h^T v + u^T G v \right\}. \tag{19}$$

The only question left is to adjust the diameter r properly. We can set $r^2 = \mu$ immediately, as each feasible vector v of the original problem consists of μ sub-vectors v^m of unit norm, see (17). Yet there is a possibility of obtaining a tighter relaxation.

Proposition 3.1. (Optimal Choice of Diameter r) Assume that all HDMR components in (9) have zero mean in the sense of Remark 2.1. Then, the smallest value of r such that (19) is a relaxation of (10) is

$$r = \sqrt{\mu - \sum_{m=1}^{\mu} \frac{1}{|Z_m|}}. \tag{20}$$

P r o o f . Assuming the zero mean property of all HDMR components in (9), we observe that the minimized criteria in (19) does not depend on the average value of any subvector v^m of v, cf. definitions (11), (12) and (13), of matrices F, G and vector h, respectively. Hence, we may shift all elements of each v^m by a constant factor $-\frac{1}{|Z_m|}$ and thus rewrite constraint (17) as

$$v_i^m = 1 - \frac{1}{|Z_m|} \Longleftrightarrow z_m = Z_m[i], \qquad v_i^m = -\frac{1}{|Z_m|} \quad \text{otherwise}, \tag{21}$$

non-affecting the value of the minimized criteria in (19). Thus, we may set r equal to the norm of any feasible solution satisfying constraint (21), and so we have

$$r^2 = \sum_{m=1}^{\mu} \left\{ \left(1 - \frac{1}{|Z_m|}\right)^2 + \sum_{i=2}^{|Z_m|} \frac{1}{|Z_m|^2} \right\} = \mu - \sum_{m=1}^{\mu} \frac{1}{|Z_m|}.$$

□

Thus, we obtained as tight relaxation of the original problem as possible and we are ready to solve the trust region problem (19). From a wide spectra of solution methods of the trust region problem, see [24] and references therein, we choose one which is computationally expensive for a one step minimization, but effective in our repetitive setting.

Theorem 3.2. (Exact Solution of Trust Region Problem) Consider trust region problem (19) parametrized by vector u. Since F is symmetric, we may find its eigenvalue decomposition $F = U^T D U$ with orthogonal matrix U and diagonal matrix D having its diagonal composed of all eigenvalues ordered from the lowest one to the highest one. For simplicity, we assume that F is indefinite matrix, i. e., $D_{11} < 0$, and that for $b = Uh + UG^T u$ it holds $b_1 \neq 0$. Then, the minimizer $\hat{v}$ of problem (19) is given by

$$\hat{v} = -U^T (D - \lambda \mathbb{I})^{-1} b, \tag{22}$$

where $\mathbb{I}$ is a unit matrix and $\lambda \in (-\infty, D_{11})$ is uniquely determined by one-dimensional equation

$$\sum_{i=1}^{\theta} \left(\frac{b_i}{D_{ii} - \lambda} \right)^2 = r^2. \tag{23}$$

P r o o f . An explicit solution to (19) for a general setting (allowing $b_1 = 0$ or $D_{11} \geq 0$) is derived in [26]. For more details see also [1, 24] and references therein. □

We note that λ satisfying (23) may be computed by the Newton's method. Also, the assumptions of Theorem 3.2 are not much restricting when a real world problem is considered. Indeed, it is highly improbable that F would be positive semi-definite or $b_1 = 0$. However, even in these cases a similar result may be obtained as already discussed. In some practical problems matrix F may be zero or may have a very small norm. Then, we may either use a different approach, e. g. linear integer programming [25], or we may solve (19) analytically with the optimal choice of $\hat{v}$ determined by formula

$$\hat{v} = -\frac{\|r\|}{\|h + G^T u\|} \left(h + G^T u \right). \tag{24}$$

3.2. Estimate of the exact minimizer

At the moment, we briefly summarize the previous procedure. We related $v(z) \in \mathbb{R}^{\mu}$ to each $z \in Z$ by (17), and also $u(x) \in \mathbb{R}^{\kappa}$ to $x \in X$ in a similar manner. Next, we found the exact minimizer $\hat{v} \in \mathbb{R}^{\theta}$ of the relaxed problem (19), which is in fact parametrized by $x \in X$ as $\hat{v} = \hat{v}(u(x)) = \hat{v}(x)$. Such $\hat{v}(x)$ generally does not correspond to any feasible solution $z \in Z$ of the original problem (10). Yet, we may still use the knowledge of $\hat{v}(x)$ to estimate the value of $p(x)$. We simply interpret each value $\hat{v}_i^m(x)$ as an indicator of sub-optimality of the related element $Z_m[i] \in Z_m$. In other words, the higher the element $\hat{v}_i^m(x)$ is, the lower cirteria $\tilde{g}(x, z_1, \ldots, z_\mu)$ we may expect when adjusting z_m to $Z_m[i]$. One can came up with many different ways of such "rounding" of $\hat{v}(x)$ to some $z \in Z$, and thus there is not any guarantee that the following heuristic is the best possible.

From now on, we again omit the parameter $x \in X$ in the notation for the sake of simplicity. We start with normalizing vector $\hat{v}$ in two steps. We shift it to be non-negative

$$\hat{v} = \hat{v} - \min_{i \in \{1,\ldots,\theta\}} \hat{v}_i, \tag{25}$$

and then we rescale all its subvectors $\hat{v}^m$, $m \in \{1, \ldots, \mu\}$, as follows

$$\hat{v}^m = \hat{v}^m \Big/ \max_{i \in \{1,\ldots,|Z_m|\}} \hat{v}_i^m. \tag{26}$$

Thus, for all m there is at least one element of $\hat{v}^m$ equal to 1, and for all $i \in \{1, \ldots, |Z_m|\}$ it holds that $\hat{v}_i^m \in [0, 1]$. Further, we define function $q(z)$ indicating the quality of a particular $z \in Z$ (with respect to an implicit parameter $x \in X$)

$$q(z) = \prod_{m=1}^{\mu} \hat{v}_{i_m}^m \quad \text{where} \quad z = (Z_1[i_1], \ldots, Z_\kappa[i_\kappa]). \tag{27}$$

From non-negativity of $\hat{v}$ we observe that $q(z) \in [0, 1]$, and the maximum of $q(z)$ with respect to $z \in Z$ is equal to 1 by (26). Then, we define

$$Z^{\phi} = \{z \in Z : q(z) \geq \phi\} \tag{28}$$

for any $\phi \in [0, 1]$. Thus, $Z^0 = Z$ and Z^1 contains only such $z \in Z$ that all the corresponding $\hat{v}_{i_m}^m$ are maximizers used in the denominator in (26). We note that Z^{ϕ} can be enumerated in a component-wise manner using (27) without passing the whole Z. Then, we substitute $Z^{\phi} \subset Z$ for Z in (10), and we find an upper bound $\overline{p}^{\phi}(x)$ on $p(x)$

$$\overline{p}^{\phi}(x) = \min_{z \in Z^{\phi}} \tilde{g}(x, z), \tag{29}$$

by enumerating $\tilde{g}(x, z)$ for all $z \in Z^{\phi}$. The lower the value of ϕ we choose, the larger the Z^{ϕ} that we obtain and the tighter the upper bound $\overline{p}^{\phi}(x)$ we find; nonetheless, at the price of slower enumeration in (29).

Once the diagonalization of F assumed in Theorem 3.2 is done, it is in fact easy to compute $\overline{p}^{\phi}(x)$ for any $x \in X$. We construct $u(x)$ by the one-to-one correspondence (17), then we find the related value of $\lambda(x)$ following (23), and finally calculate candidate $\hat{v}(x)$ which enters the already introduced procedure that leads to $\overline{p}^{\phi}(x)$ defined by (29). Thus, we found approximate minima of a general function $\tilde{g}(x,z)$ in HDMR form over $z \in Z$ for all parameters $x \in X$. This permits us to apply HDMR to effectively approximate the Bellman equation in Section 4.

3.3. Minimization of a random function

Now, we dedicate a short section to a numerical verification of the previously introduced technique. We solved problem (10) exactly for a random function $\tilde{g}(x,z)$. For the sake of simplicity, we omitted the parameter x and set $G = 0$ in (19). Next, we choose the minimization domain $Z = \{1,\dots,150\}^3$, we generated HDMR components $\tilde{g}_{mn}$ randomly with values chosen from the uniform distribution on interval $[0,1]$ and finally we adjusted them to satisfy (2). Then, we found upper estimates on minima $\overline{p}^{\phi}$ for various choices of parameter ϕ following (29). All results were averaged with respect to 20 random samples of F and h and depicted in Figure 1.

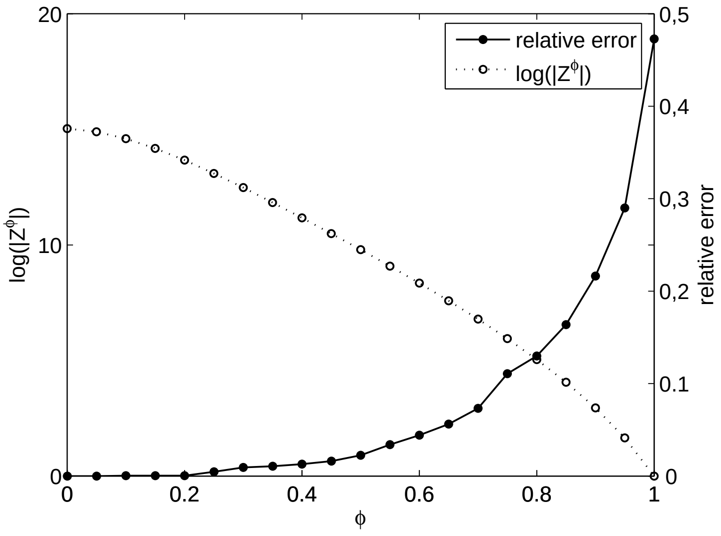

Fig. 1. Value of $\log(|Z^{\phi}|)$ and a relative error of $\overline{p}^{\phi}$ plotted against various values of ϕ. The relative error is the distance of $\overline{p}^{\phi}$ from the minimum of $\tilde{g}(z)$ rescaled and shifted in such a way that exact minimum corresponds to 0 whereas the average value of the minimized criteria corresponds to 1. The depicted results were averaged over 20 different realizations of matrix F and vector h.

The relative error of upper bound $\overline{p}^{\phi}$ is defined as the distance from minimum of $\tilde{g}(z)$ rescaled and shifted in such a way that the exact minimum corresponds to 0 whereas the average value of the minimized criteria corresponds to 1. We observe that the lower the value of ϕ is, the better the approximation we obtain as we expected. On the other hand, there was a linear grow of $\log(|Z^{\phi}|)$ when decreasing ϕ. We suppose that a detailed elaboration of this relation could serve as a basis for an error estimation heuristics.

These experiments were carried out on CPU Intel Core i3, 2.10 GHz with 4GB of RAM in Matlab 7. It took 169 seconds to find the exact minimum, whereas the average time necessary to diagonalize matrix F was 1.3 seconds. We note that this matrix diagonalization is done only once in the full setting of (10), whereas the time necessary for exact minimization of $\tilde{g}(x,z)$ for each $x \in X$ is still the same.

4. APPROXIMATE DP BASED ON HDMR

This is the right time to briefly introduce the decision-making theory. A decision-making task stands for selecting a decision-maker's strategy in order to reach his aim with respect to the part of the world (so-called system). The decision maker observes or influences the system over a finite decision making horizon $\tau < \infty$. Value $y_t \in y_t$, $t \in T = \{1, \ldots, \tau\}$, provides the decision maker with all the knowledge influencing the future behaviour of the system. Thus, y_t includes the current state of the system together with other external data observed up to the time instant t. Nonetheless, we will reference y_t simply as a state of the system. Next, the decisions (actions) of a decision-maker are denoted as $a_t \in A_t$. A strategy is a collection of mappings of the current state $y_{t-1} \in Y_{t-1}$ into the choice of the next decision $a_t \in A_t$; for the optimal strategy we use symbols $\{\hat{a}_t(y_{t-1})\}_{t\in T}$. To formalize the decision-maker's aims, a concept of the additive loss function is used, $l_t(a_t, y_t)$, depending on the current action a_t and the system state y_t. The involved system is described in a probabilistic manner by the following collection of pdfs called the outer Markov model of a system

$$\{f_t(y_t|a_t, y_{t-1})\}_{t\in T}\,. \tag{30}$$

For the expected value of variable x conditioned by y we use

$$\mathcal{E}\,[\,x\,|\,y\,] = \int_X x\, f(x|y)\,\mathrm{d}x. \tag{31}$$

Knowing the collection of loss functions $\{l_t(a_t, y_t)\}_{t=1}^{\tau}$ together with the system model (30), the optimal strategy $\{\hat{a}_t(y_{t-1})\}_{t\in T}$ is fully determined by the Bellman function

$$V_{t-1}(y_{t-1}) = \min_{a_t \in A_t} \mathcal{E}\,[\,l_t(a_t, y_t) + V_t(y_t)\,|\,a_t, y_{t-1}\,]\,, \tag{32}$$

which has to be recursively evaluated at all times $t \in T$ with the boundary condition $V_\tau = 0$. As this standard form of the Bellman equation (32) is not convenient to our purposes, we rewrite it in an equivalent form

$$\begin{aligned} E_t(a_t, y_{t-1}) &= \mathcal{E}\left[\, l_t(y_t, a_t) + \min_{a_{t+1}\in A_{t+1}} E_{t+1}(a_{t+1}, y_t) \,\middle|\, a_t, y_{t-1}\right] \\ E_{\tau+1} &= 0. \end{aligned} \tag{33}$$

Then, $E_{t+1}(a_{t+1}, y_t)$ is the expected loss-to-go provided we choose action a_{t+1} in the system state y_t. In this setting, the optimal strategy $\hat{a}_t(y_{t-1})$ is composed of the actions satisfying

$$\hat{a}_t(y_{t-1}) = \operatorname*{argmin}_{a_t \in A_t} E_t(a_t, y_{t-1}). \tag{34}$$

4.1. Offline part – Approximate evaluation of E_t

Now, we are prepared to apply both HDMR developed in Section 2 and fast approximate minimization of functions in HDMR form, see Section 3, to effectively approximate $E_t(a_t, y_{t-1})$ defined by (33). This part of algorithm is the most demanding concerning the computational complexity. Thus, function $E_t(a_t, y_{t-1})$ is typically computed offline, stored as a look-up table (in our case HDMR), and then used during the online part of a decision-making algorithm to find the approximated optimal action in analogy to (34). The proposed algorithm runs in the backward manner just as the evaluation of the exact Bellman equation (32).

We denote the approximated loss-to-go function by $\tilde{E}_t$ even though for $t < \tau$ it is not HDMR of the (unknown) exact E_t. For the first step, $t = \tau$, we rewrite (33) as

$$E_\tau(a_\tau, y_{\tau-1}) = \mathcal{E}\left[\, l_\tau(y_\tau, a_\tau) \,|\, a_\tau, y_{\tau-1} \,\right]. \tag{35}$$

To obtain all HDMR components $\tilde{E}_{\tau,\emptyset}, \tilde{E}_{\tau,m}, \tilde{E}_{\tau,mn}$ of $E_\tau(a_\tau, y_{\tau-1})$, we evaluate $E_\tau(a_\tau, y_{\tau-1})$ for each pair $(a_\tau, y_{\tau-1}) \in A_\tau \times Y_{\tau-1}$ and add the resulting value to proper sums in (8). Next, suppose we know all $\tilde{E}_{t+1,\emptyset}$, $\tilde{E}_{t+1,m}$, $\tilde{E}_{t+1,mn}$ and we want to find an approximation of E_t in the form of HDMR. Substituting $\tilde{E}_{t+1}$ into (33) we have

$$E_t(a_t, y_{t-1}) \approx \mathcal{E}\left[l_t(y_t, a_t) + \min_{a_{t+1} \in A_{t+1}} \tilde{E}_{t+1}(a_{t+1}, y_t)) \,\middle|\, a_t, y_{t-1} \right]. \tag{36}$$

This suggests defining $\tilde{E}_t$ as HDMR of the expression on the right-hand side, or at least as HDMR of some approximation of this expression.

Algorithm.

```
evaluate $E_\tau(a_\tau, y_{\tau-1})$ according to (35)
find HDMR $\tilde{E}_\tau(a_\tau, y_{\tau-1})$ of $E_\tau(a_\tau, y_{\tau-1})$ using Proposition 2.2
fix $\phi \in [0,1]$ determining the precision of the approximate minimization near (37)
for $t := \tau$ to 2
    shift HDMR components of $\tilde{E}_t(a_t, y_{t-1})$ according to Remark 2.1
    construct matrices $F_t$, $G_t$ and vector $h_t$ regarding (11), (12) and (13)
    find orthogonal matrix $U_t$ and diagonal matrix $D_t$ such that $F_t = U_t^T D_t U_t$
    compute $r_t$ using Proposition 3.1
    for each $y_{t-1} \in Y_{t-1}$
        relax problem $\pi_{t-1}(y_{t-1})$, see (37), into a trust region problem (19)
        compute the exact minimizer $\hat{v}_{t-1}(y_{t-1})$ using Theorem 3.2
```

```
            find an upper bound estimate $\overline{\pi}^{\phi}_{t-1}(y_{t-1})$ using $\hat{v}_{t-1}$, see (29)
            for each $a_{t-1} \in A_{t-1}$
                set $\rho := \mathcal{E}\left[ l_{t-1} + (y_{t-1}, a_{t-1}) + \overline{\pi}^{\phi}_{t-1}(y_{t-1}) \,\middle|\, a_{t-1}, y_{t-2} \right]$, cf. (39)
                add $\rho$ to proper sums in Prop. 2.2 to successively construct $\tilde{E}_{t-1}$
            end
        end
    end
```

On that account we denote

$$\pi_t(y_t) = \min_{a_{t+1} \in A_{t+1}} \tilde{E}_{t+1}(a_{t+1}, y_t) \tag{37}$$

and search for its upper bound $\overline{\pi}^{\phi}_t(y_t)$ following the instructions of Section 3. The choice of an auxiliary parameter $\phi \in [0, 1]$ determining the precision of the upper bound estimate is discussed at the end of this section. Looking at (10), we identify $\tilde{g} = \tilde{E}_{t+1}$, $X = y_t$ and $Z = A_{t+1}$. We note that all the HDMR components of $\tilde{E}_{t+1}$ that depend only on y_t may be directly interchanged with minimization in (37) and thus not considered at this moment. Based on the knowledge of such $\tilde{E}_{t+1,\emptyset}$, $\tilde{E}_{t+1,m}$ and $\tilde{E}_{t+1,mn}$ that depend on a_{t+1}, we construct matrices F_t, G_t and vector h_t according to (11), (12) and (13), and we formulate the relaxed problem (19). Then, we find its exact minimizer $\hat{v}_t(y_t)$ in a direct analogy to (22) with the matrix diagonalization

$$F_t = U_t^T D_t U_t \tag{38}$$

involved. The diagonalized matrix F_t is typically small and does not grow with t as its size (15) corresponds to the space of actions a_t. Knowing $\hat{v}_t(y_t)$, we calculate an upper bound on minimum applying procedure (29), and finally we add (restore) all HDMR components of $\tilde{E}_{t+1}$ depending only on y_t. Thus, we obtained an upper bound on minimum of $\pi_t(y_t)$. We note that diagonalization (38) is carried out just once for each time step t, and so we can effectively evaluate $\overline{\pi}^{\phi}_t(y_t)$ for all $y_t \in Y_t$.

Now, we find $\tilde{E}_t(a_t, y_{t-1})$ by evaluating the right-hand side of the following formula

$$\tilde{E}_t(a_t, y_{t-1}) \approx \mathcal{E}\left[l_t(y_t, a_t) + \overline{\pi}^{\phi}_t(y_t) \,\middle|\, a_t, y_{t-1} \right] \tag{39}$$

for each pair $(a_t, y_{t-1}) \in A_t \times Y_{t-1}$ and by adding the resulting value to proper sums in (8) immediately. Thus, we construct all HDMR components $\tilde{E}_{t,\emptyset}$, $\tilde{E}_{t,m}$ and $\tilde{E}_{t,mn}$, avoiding the full dimensional representation of $\tilde{E}_t$. Finally, we repeat the whole procedure to recursively compute function $\tilde{E}_t(a_t, y_{t-1})$ for all $t \in T$. For the readers convenience, the whole procedure is summarized in Algorithm 1.

4.2. Online part – Approximate minimization of $\tilde{E}_t$

The previously described part of the algorithm has to be implemented in advance, or "off-line" manner because of high computational demands. As functions $\{\tilde{E}_t(a_t, y_{t-1})\}_{t\in T}$ are stored only in the form of HDMR, it is possible to take larger decision horizons τ into consideration. Nonetheless, we still have to choose an approximated (suboptimal) action $\tilde{a}_t$ in the real time, or "on-line" manner. Then, the previously observed system state y_{t-1} is fixed and so we solve just one minimization problem in each time step t in opposite to the recursive evaluation of (36). Substituting $\tilde{E}_t$ into (34), we define

$$\tilde{a}_t(y_{t-1}) = \operatorname*{argmin}_{a_t \in A_t} \tilde{E}_t(a_t, y_{t-1}). \tag{40}$$

We note that $\tilde{a}_t(y_{t-1})$ does not stand for HDMR approximation of $\hat{a}_t(y_{t-1})$ defined by (34).

There are many ways how to find $\tilde{a}_t$, or at least some its approximation. An interesting choice can be a trust region based relaxation as we may exploit our previous calculations. We may represent HDMR components of $\tilde{E}_t(a_t, y_{t-1})$ in the basis obtained in (38). If we store all matrices U_t, D_t, and also matrices G_t and vectors h_t involved in the approximate minimization of $\pi_t(y_t)$ defined by (37), we may find the approximate minimizer of (40) in accordance with Section 3 again. However, even some more accurate technique may be used in one-shot only minimization (40). Any algorithm for binary quadratic programming [18] may be applied to solve (40) via equivalent reformulation (14) constrained by (17). For smaller sets A_t, we can find even exact value of $\tilde{a}_t \in A_t$ by direct enumeration of (40). We decided to use this most accurate approach in Section 4.3 in order to show the extent to which $\tilde{E}_t(a_t, y_{t-1})$ in the form of HDMR may be compared with the exact value of $E_t(a_t, y_{t-1})$.

4.3. N-armed bandit problem

As an illustrative example, we propose here an approximate solution to the N-armed bandit problem, which plays an important role in approximate dynamic programming, see for instance [21, 27]. First, however, is important to note that since the proposed algorithm does not exploit any expert knowledge of the N-armed bandit problem, it can not compete with the whole class of tailor-made solutions to this problem, see [4, 6] and references therein. Thus, the purpose of this section is to show in a detail how to employ the developed algorithm, and also to indicate its position among other general approximate dynamic programming techniques, see the comparison in Figure 2. This question is also discussed in the conclusion.

Conceive a game where the player has to choose between different options, e. g. levers of N-armed bandit, with numerical rewards chosen from various stationary probability distributions. The payoff probabilities of levers are fixed, yet unknown, and thus the player has to estimate them. Then, the problem is to identify the most winning lever. Even though this problem could be formulated easily, it is a real issue for a longer game horizon as it is hard to balance exploration and exploitation. Winning in the first round does not imply that the player should stick to the same lever as it prevents learning of the payoff probability of other levers.

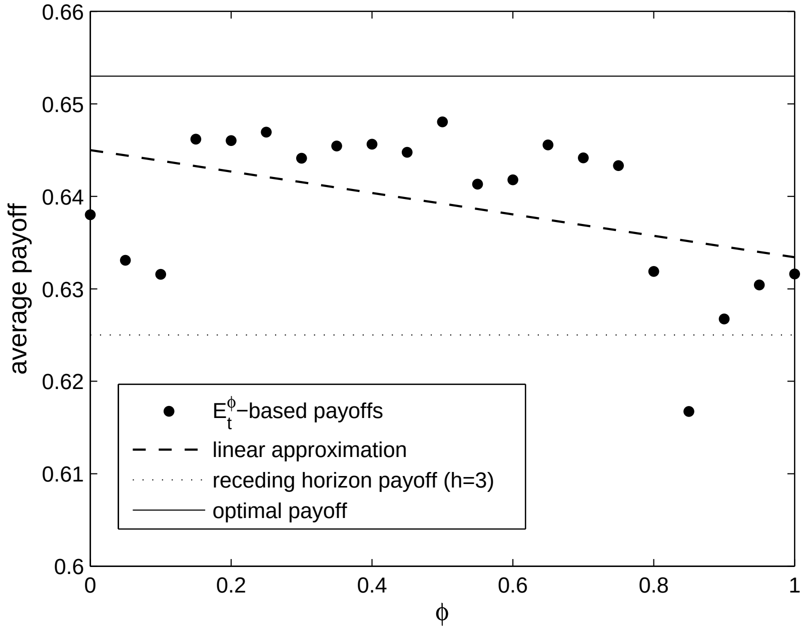

Fig. 2. Average payoffs obtained from approximated strategies derived from $\tilde{E}_t^{\phi}$ for various $\phi \in [0, 1]$. The average payoff of the exact optimal strategy derived from E_t was 0.653; the average payoff of the receding-horizon-based strategy (with horizon of 3 steps) was 0.625. These results are based on 20000 simulated plays with 9-armed bandit, each of them consisting of $\tau = 8$ steps. The payoff probabilities P_{ij} of the bandit were chosen randomly from uniform distribution on interval $[0, 1]$. The only exception was payoff probability $P_{11} = 0.1$, which was kept fixed to avoid complete symmetry of the problem as discussed in Section 4.3.

We considered the game with 9-armed bandit and decision making horizon of $\tau = 8$ steps to be able to compare approximated suboptimal strategies with the exact optimal strategy. Using the previous notation, y_t stands for the observed value (payoff) $y_t \in Y = \{0, 1\}$ and a_t denotes the decision of a player in each time step $t \in T = \{1, \dots, \tau\}$. The arms of the bandit are represented by two-dimensional space of actions, $a_t \in A = \{1, 2, 3\}^2$. The loss function

$$l_t(y_t, a_t) = -y_t \tag{41}$$

represents the aim of maximizing the payoff y_t in each round of the game. Next, we introduce a sufficient statistic s_t, $\mathrm{dom}(s_t) = Y \times A$, which compresses the previous game results in a small vector

$$s_t(y, a) = s_{t-1}(y, a) + \delta_{y_t, y}\, \delta_{a_t, a}, \tag{42}$$

with δ standing for standard Kronecker's symbol. Thus, $s_t(y, a)$ counts how many times we observed a value y after selecting an action a in first t rounds of the game. We set $s_0 = 0$ for the moment. In fact, s_t may be included into the system state y_t, but for the sake of simplicity we treat it separately here. To compute the expected loss in (33), the knowledge of the Markov system model (30) is necessary

$$f_t(y_t|a_t, s_{t-1}) \;=\; \frac{s_{t-1}(y_t, a_t) + 1}{s_{t-1}(y_t, a_t) + s_{t-1}(1 - y_t, a_t) + 2}. \tag{43}$$

This model was obtained using the technique of Bayesian estimation [19]. In the following experiment, the 9-armed bandit was simulated using pseudo-random generator with fixed payoff probability matrix P defined for $a \in A$ as follows

$$P_{ij} = \mathrm{Prob}(y = 1 | a = [i, j]). \tag{44}$$

During the experiment, it turned out that high-symmetry of N-armed bandit is unsuitable for our purposes. If the underlying payoff probability P is completely unknown, and for the prior information it holds $s_0(y, a) = 0$ for all $y \in Y, a \in A$, then all the bandit arms have the same expected loss when averaged over all the possible system trajectories. Thus, F_1 corresponding to differences of the expected loss among various arms is equal to zero. We may still use the previously introduced algorithm, see the note near (24), but we would miss its most interesting part, i. e. the trust region based approximate minimization described in Section 3.1.

Thus, we decided to slightly perturb the experiment to suppress its symmetry. We put a prior information on one arm, $s_0(0, [1, 1]) = 1$, and in this setting we computed the exact values of $\{E_t\}_{t \in T}$ following (33) and also all HDMR functions $\{\tilde{E}_t^{\phi}\}_{t \in T}$ according to Section 4.1. This time we explicitly stated that $\tilde{E}_t^{\phi}$ depends also on the value of ϕ, see (39). The disk space necessary to save $\{E_t\}_{t \in T}$ and each $\{\tilde{E}_t^{\phi}\}_{t \in T}$ in Matlab .mat file was 2.3 MB and 0.1 MB, respectively. The optimal strategy was derived from E_t using (34), and suboptimal strategies parametrized by ϕ were derived according to (40).

All these strategies were used to simulate 20000 plays with 9-armed bandit, each of them consisting of $\tau = 8$ steps. The payoff probabilities P_{ij} of the bandit were chosen randomly from uniform distribution on interval $[0, 1]$ with the only exception of a fixed payoff probability $P_{11} = 0.1$ corresponding to the only non-zero prior $s_0(0, [1, 1])$. The average payoff of the optimal strategy was 0.653, whereas the average payoff of a sub-optimal strategy based on a classical technique of receding horizon (with horizon of 3 steps) was 0.625. We observe that the payoffs of strategies based on HDMR approximation for various $\phi \in [0, 1]$, see Figure 2, lie between these two numbers (with an exception of one outlier). Naturally, the optimal strategy with the high computational demands (for offline calculation) and high memory demands (for both offline and online part) gives the highest payoff, its HDMR based approximation (high offline computational demands, modest memory demands) is slightly worse, and the receding-horizon-based strategy (no offline computation, low online memory demands, high online computational demands) gives the lowest payoff. In our setting, 1710 numbers are necessary to represent the receeding horizon strategy and 342 expected loss minimizations are needed to evaluate it each time step, whereas for HDMR approximation 22745 numbers are needed to represent the approximated strategy, but only 9 expected loss minimizations are necessary each time step.

Next, we discuss the influence of the parameter $\phi \in [0, 1]$. We see that the strategy derived from E_t^1 was rather successful, it gained 0.632 on average. This fact indicates the practical applicability of the less accurate approximation of E_t with $\phi = 1$ and Z^1 containing typically just one element. On contrary, the precision of HDMR approximation itself may be deduced from the average payoff 0.638 obtained for $\phi = 0$, which corresponds to the exact minimization in (37). The closer the ϕ is to 0, the closer the upper bound $\overline{\pi}_t^{\phi}$ is to π_t, cf. the definition near (37). However, this point-wise convergence of approximated Bellman functions does not imply any monotonicity of their respective payoffs, as it is disrupted by an averaging effect of HDMR. Still, on average, we observed higher payoffs when decreasing ϕ to 0, see their linear approximation in Figure 2. The slope of this line is rather small, i. e., the average payoff just slightly increases when decreasing ϕ. However, this observation is likely to be problem-dependent. Indeed, in the case of a randomly generated function, cf. Figure 1, the approximation error was more sensitive with respect to ϕ.

5. CONCLUSION

The aim of this work was to cope with both computational and memory demands necessary to find and represent the optimal decision making strategy. The proposed variant of the approximate dynamic programming based on HDMR is appealing for two reasons. At first, this approximation considerably reduces the memory demands of the algorithm, but, more importantly, it also enables a fast approximate minimization of the approximated Bellman function. The numerical simulation proved that the proposed variant of the approximate dynamic programming is a viable technique. Based on the comparison with the receding-horizon-based approximation, we shown that this technique may be advantageous whenever the online computational power is limited. This could be the case of fast industrial processes where the time step of control strategy is very short.

As for all the approximate methods surveyed at the beginning of Section 1, the one proposed in this article cannot be assigned to any of these classes directly. It is based on the Bellman function approximation, however, looking at its internal structure it may be considered also as an aggregation method where each HDMR component aggregates different coordinates. Next, the point-wise construction of HDMR resembles the learning phase of the artificial neural networks, yet it is more straightforward.

A bottleneck of the proposed approximation technique is the fact that it still needs to pass through the whole decision tree. Nonetheless, it can easily be parallelized, or randomly sampled HDMR may be used [13], or some reinforcement learning algorithm that aims at this problem can be applied. The fact that HDMR enables a fast approximate minimization would still be worthwhile.

ACKNOWLEDGEMENT

The author would like to express his gratitude to RNDr. Ondřej Pangrác, Ph.D., for inspiring discussion about discrete optimization, to Irena Dvořáková, prom. fil., for significant help with the language of the manuscript, and finally to Ing. Václav Šmídl, Ph.D., for constructive criticism and encouragement. This work has been funded by the GAČR project P 102/11/0437.

(Received October 11, 2012)

REFERENCES

[1] S. Busygin: A new trust region technique for the maximum weight clique problem. Discrete Appl. Math. *154* (2002), 2006.

[2] M. Demiralp: High dimensional model representation and its application varieties. In: Proc. Fourth International Conference on Tools for Mathematical Modelling, St. Petersburg 2003, pp. 146–159.

[3] K. S. Feil and N. Shah: Volatility calibration using spline and high dimensional model representation models. Wilmott J. *1* (2009), 179–195.

[4] A. Garivier and O. Cappé: The KL-UCB algorithm for bounded stochastic bandits and beyond. In: Proc. 24th Annual Conference on Learning Theory, Budapest 2011.

[5] A. George, W. B. Powell, and S. R. Kulkarni: Value function approximation using multiple aggregation for multiattribute resource management. J. Machine Learning Res. *9* (2008), 2079–2111.

[6] J. Gittins: Bandit processes and dynamic allocation indices. J. Roy. Statist. Soc. Ser.B *41* (1979), 2, 148-177.

[7] A. Gosavi: Reinforcement learning: A tutorial survey and recent advances. INFORMS J. Comput. *21* (2009), 178–192.

[8] M. Hauskrecht: Value-function approximations for partially observable markov decision processes. J. Artif. Internat. Res. *13* (2000), 33–94.

[9] T. Jaakkola, S. P. Singh, and M. I. Jordan: Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems. MIT Press, 1995.

[10] R. M. Karp: Reducibility among combinatorial problems. In: Complexity of Computer Computations (R. Miller and J. W. Thatcher, eds.), Plenum, New York 1972.

[11] H. Kushner: Introduction to Stochastic Control. Holt, Rinehart and Winston, New York 1970.

[12] M. LeBlanc and R. Tibshirani: Combining estimates in regression and classification. J. Amer. Statist. Assoc. *91* (1996), 1641–1650.

[13] G. Li, J. Hu, S.-W. Wang, P. G. Georgopoulos, J. Schoendorf, and H. Rabitz: Random sampling-high dimensional model representation (rs-hdmr) and orthogonality of its different order component functions. J. Phys. Chem. A *110* (2006), 7, 2474–2485.

[14] R. Luus: Iterative Dynamic Programming. Chapman and Hall/CRC Monographs and Surveys in Pure and Applied Mathematics, 2000.

[15] X. Ma and N. Zabaras: An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. J. Comput. Phys. *229* (2010), 3884–3915.

[16] J. Matoušek and J. Nešetřil: Invitation to Discrete Mathematics. Clarendon Press, 1998.

[17] W. Miller, R. Sutton, and P. Werbos: Neural Networks for Control. Neural Network Modeling and Connectionism. Mit Press, 1995.

[18] C. Olsson, A. Eriksson, and F. Kahl: Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming. In: Computer Vision and Pattern Recognition, 2007.

[19] V. Peterka: Bayesian system identification. In: Trends and Progress in System Identification (P. Eykhoff, ed.), Pergamon Press, Oxford 1981, pp. 239–304.

[20] M. Pištěk: On implicit approximation of the bellman equation. In: 15th IFAC Symposium on System Identification, Saint-Malo 2009.

[21] W. B. Powell: Approximate Dynamic Programming: Solving the Curses of Dimensionality. Wiley-Interscience, 2007.

[22] H. Rabitz and O. Alis: General foundations of high-dimensional model representations. J. Math. Chem. *25* (1999), 197–233.

[23] S. Rahman: A polynomial dimensional decomposition for stochastic computing. Internat. J. Numer. Methods in Engrg. *76* (2008), 2091–2116.

[24] M. Rojas, S. A. Santos, and D. C. Sorensen: A new matrix-free algorithm for the large-scale trust-region subproblem. SIAM J. Optim. *11* (2000), 611–646.

[25] A. Schrijver: Theory of Linear and Integer Programming. Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons, 1998.

[26] D. C. Sorensen: Newton's method with a model trust region modification. SIAM J. Numer. Anal. *19* (1982), 2, 409–426.

[27] R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction. MIT Press, 1998.

Miroslav Pištěk, Institute of Information Theory and Automation – Academy of Sciences of the Czech Republic, Pod Vodárenskou věží 4, 182 08 Praha 8. Czech Republic.
e-mail: pistek@utia.cas.cz