Front. Econ. China 2013, 8(2): 260–271
DOI 10.3868/s060-002-013-0013-4

RESEARCH ARTICLE

Hui He, Hao Zhang

A Rapid Grid Search Method for Solving Dynamic Programming Problems in Economics

Abstract We introduce a rapid grid search method in solving dynamic programming problems in economics. Compared to mainstream grid search methods, by using local information of the Bellman equation, this method can significantly increase the efficiency in solving dynamic programming problems by reducing the grid points searched in the control space.

Keywords dynamic programming, Bellman equation, grid search, concavity, searching efficiency

JEL Classification C02, E13, E27

1 Introduction

High-dimensional dynamic programming (DP) problems have been gaining more and more popularity in economics. Yet solving high-dimensional DP problems numerically is still quite challenging. For instance, those powerful numerical methods for solving one-dimensional optimization problems, such as golden section search and Brent's method, are difficult to implement in a high-dimensional DP context. On the other hand, grid search, as a widely used numerical method in solving optimization problems, can serve as a stable and reliable method to find solutions to high-dimensional DP problems. Compared to other sophisticated methods, such as New-

Received October 18, 2012

Hui He
School of Economics, and Key Laboratory of Mathematical Economics, Shanghai University of Finance and Economics, Shanghai 200433, China
E-mail: he.hui@mail.shufe.edu.cn
Hao Zhang (✉)
School of Labor and Human Resources, Renmin University of China, Beijing 100872, China
E-mail: hao.zhang@ruc.edu.cn

ton's method or Quasi-Newton methods, the basic "brute force" grid search method does not rely on any local or global information of the objective function. In particular, for problems with non-smooth objective functions or multiple local optima, grid search can achieve the global optimum with stable precision and search speed, while methods based on the gradient of objective functions cannot offer. The stability and convergence properties of grid search can be greatly appreciated in the study of high-dimensional DP problems. The drawback of this method, however, is that it can be extremely slow and can impose a huge computational burden in practice. For high precision solutions, the computational cost will increase exponentially, since the precision is determined by the fineness of pre-set grid points. In addition, the overwhelming computational cost caused by the "curse of dimensionality" often arising in high-dimensional DP problems makes the task of solving these problems using grid search intractable.

Efforts have been made to reduce the computational burden imposed by grid search in DP problems. For example, İmrohoroğlu, İmrohoroğlu and Joines (1993) applies a bracketing grid search algorithm to solve a dynamic general equilibrium model with incomplete markets. They first discretize the state and control space using evenly distributed grid points. Starting from coarse grid points to determine an initial optimum, they then make subsequent searches over successively finer grids around the previous optimum. Their method obtains a large improvement in the searching speed by reducing grid points searched in the control space.[1] On the other hand, Grune and Semmler (2004) introduce an adaptive grid scheme for DP problems based on local error estimates. Their method reduces the number of grid points searched in the state space and gains great efficiency especially in computing the dynamic models which exhibit kinks or steep curvature of the value function and complicated dynamics due to the existence of multiple equilibria, thresholds (Skiba sets) separating domains of attraction, and periodic solutions.

In this paper, we propose a rapid grid search (RGS) method that can significantly enhance the efficiency of solving dynamic problems by reducing the grid points searched in the control space. No matter the grid points are pre-set (as in İmrohoroğlu et al. 1993) or adaptively allocated (as in Grune and Semmler 2004), this method can further reduce the total computing time by increasing the "searching speed" over these grid points. The idea is to use some local information (e.g., concavity) of the objective function to speed up the searching process by skipping the evaluation and comparison of unnecessary grid points. Different from the two methods as mentioned above, our

[1] As shown in Table 1, the total number of grid points that need to be searched by the "brute force" grid search method in the code implemented by İmrohoroğlu, İmrohoroğlu and Joines (1995, 1999a) is 1.28073e+10 for 4097 grid points in the state and control space. Using their bracketing method, the actual number of grid points searched is just 1.48898e+8, which accounts for only 1% of the total number of grid points.

method is not about how to efficiently allocate grid points over state or control space, but to restrict the area in which the search is conducted to a relevant range.

This method has the following advantages. First, it inherits all of the advantages of the mainstream grid search method in economics, such as stability and convergence properties. It can be applied to those problems which can only be solved by grid search methods, such as the problems with kinky objective functions.[2] Second, it does not require further information than that required by typical grid search methods. Third, it could in principle be applied to all grid-search-based methods including the two methods mentioned above without compromising their benefits. Therefore, improvement in searching efficiency can be gained without sacrificing the merits of different methods. Last but not least, it can be straightforwardly extended to high-dimensional DP problems with a stable efficiency gain. Although this method is not designed to break the "curse of dimensionality," it helps to reduce considerably the computational cost arising from high-dimensional DP problems.

The remainder of the paper is organized as follows. Section 2 describes the idea of the rapid grid search method and provides the algorithm. Section 3 applies the method to a one-dimensional and a two-dimensional DP example, respectively. Section 4 concludes.

2 Rapid Grid Search Method

In this section we describe a standard Bellman equation arising from a typical DP problem. We then prove a proposition which demonstrates how concavity can be used to reduce the number of effectively searched grid points in solving DP problems. An illustrative algorithm for both one-dimensional and two-dimensional DP problems is also provided.

2.1 Bellman Equation

In a DP model, let $A \subset R^L$ be the space of state variables, and let $C \subset R^M$ be the space of control variables. Under some conditions, it is well known that the solution of DP problems can be obtained by solving the following Bellman equation:

$$V_t(a_t) = \max_{c_t \in C} \{U_t(a_t, c_t) + \beta V_{t+1}(a_{t+1})\}, \tag{1}$$

[2] In a separate appendix which is available upon request, we construct a concave objective function with many kinks which its maximization cannot be solved using Newton method. We employ both brute force and bracketing grid search methods to find the global maximum. And then we apply RGS method on top of each method. In both cases RGS significantly increases search efficiency by skipping a large proportion of unnecessary grid points.

subject to

$$a_{t+1} = G_t(a_t, c_t),$$

$$a_t \in A.$$

We assume that $U_t : A \times C \to R$ is a function of state and control variables and is strictly concave in both A and C, and the function $G_t : A \times C \to R$ is convex in A and C. A and C are compact, and $\beta \in (0, 1)$.[3] Given these assumptions, a unique policy function $c_t(a_t)$ of this maximization problem exists. One can prove that $V_t(a_t)$ is strictly concave and the objective function $U_t(a_t, c_t) + \beta V_{t+1}(a_{t+1})$ is strictly concave in c_t.[4]

In most cases, numerical methods are needed to solve Eq. (1). For example, the standard grid search method first discretizes the state space A and the control space C by grid points. Then the objective function in the right-hand-side of the Bellman Eq. (1) is evaluated at each grid point and the values are compared until we find the optimum.[5]

2.2 Propositions of Strictly Concave Function

For a strictly concave function f, we have the following proposition:

Proposition 1 *Provided the function $f : X \to R$ be strictly concave on a compact set $X \subset R^M$, x_1, $x_2 \in X$ and $f(x_1) > f(x_2)$, $\forall a \in R^+$, we have $f(x_1) > f(x_2 + a(x_2 - x_1))$ and $f(x_2) > f(x_2 + a(x_2 - x_1))$.*

Proof. Since $a \in R^+$, we have constant $b = a/(1 + a) \in (0, 1)$. Since f is strictly concave on a compact set X, we have $bf(x_1)+(1-b)f(x_2+a(x_2-x_1)) < f(bx_1+(1-b)(x_2+a(x_2-x_1))) = f(x_2)$. Hence, $(1-b)f(x_2+a(x_2-x_1)) < f(x_2)-bf(x_1)$. Since $f(x_1) > f(x_2)$, we also have $f(x_1) - bf(x_1) > (1-b)f(x_2+a(x_2-x_1))$. Therefore, $f(x_1) > f(x_2+a(x_2-x_1))$. And since $f(x_2)-bf(x_2) > f(x_2) - bf(x_1) > (1-b)f(x_2 +a(x_2 -x_1))$, we have $f(x_2) > f(x_2+a(x_2-x_1))$. □

This proposition can be interpreted as a binomial relationship between two grid points. Graphically, if the value of one point is higher than another, the value of the

---

[3] These assumptions are usually satisfied in well-defined dynamic economic models. Our method, moreover, can also handle value functions with kinks since we do not require smoothness of the value function. However, the RGS method (and the bracketing approach proposed by İmrohoroğlu et al. 1993) cannot deal with complicated dynamic behavior due to multiple equilibria and thresholds (Skiba sets) separating domains of attraction and periodic solutions as shown in Grune and Semmler (2004).

[4] See Stokey, Lucas and Prescott (1989), Chapter 3, for details.

[5] Interpolation is usually used when evaluating value function for points which are not grid points.

higher one dominates the value of any point in X which is located on the extended line along the descending direction from the one with lower value. Fig. 1 shows this relationship for a one-dimensional function $f(x)$ for $x \in R^+$. Pick three grid points on the x-axis: x_1, x_2, and x_3. Since $f(x_1) > f(x_2)$, for any $x_3 \in [x_2, \infty)$ we have $f(x_1) > f(x_3)$ and the optimum must lie in the range between zero and x_2.

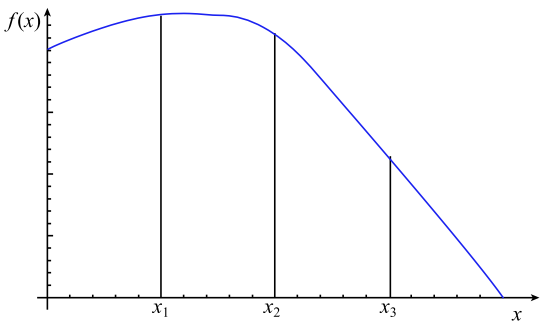

Fig. 1 Domination in a One-Dimensional Case

Fig. 2 shows the intuition of the proposition for a two-dimensional function $f(x, y)$. $\forall a \in R^+$, if $f(x_1, y_1) > f(x_2, y_2)$, then for any point on the extended line along the direction from point (x_1, y_1) to point (x_2, y_2), for example, point $(x_3, y_3) = (x_2 + a(x_2 - x_1), y_2 + a(y_2 - y_1))$,we have $f(x_1, y_1) > f(x_3, y_3)$. Therefore, we do not need to search any (x_3, y_3).

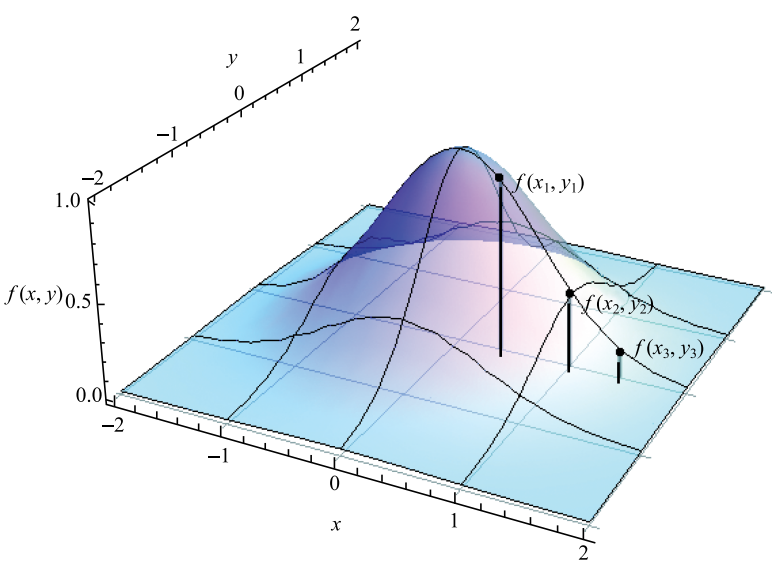

Fig. 2 Domination in a Two-Dimensional Case

In a typical DP problem in economics, the utility function U_t and constraint G_t are usually well-defined based on assumptions about preferences and production set. Given the strict concavity of utility function U_t, when we choose c_t over the control space C to maximize the Bellman equation, Proposition 1 can help to dramatically reduce the searching range over the control space with the ranking information of some grid points. As we showed in the two previous graphs, given the domination relation of any two grid points, all points in the "downward" direction along the extended line beyond the lower ranking point can be skipped. This significantly saves the computational time for solving the Bellman equation.

2.3 Algorithm

Proposition 1 helps to reduce the searching range over the control space in both single and multi-dimensional cases. Here we give the algorithm of the RGS method for a one-dimensional and a two-dimensional case, which applies the proposition above. A similar algorithm can also be applied in higher dimensional DP problems. For the purposes of comparison, this algorithm is based on the bracketing technique as in İmrohoroğlu, İmrohoroğlu and Joines (1993). In their article, the maximum of an objective function is narrowed down round by round and the values of five pre-set grid points are evaluated and compared in each round. We use their technique as our benchmark grid search method and apply RGS on top of it to improve searching efficiency.

2.3.1 Algorithm 1: One-Dimensional Case

For a Bellman equation with a one-dimensional control space C , given the state variable a_t, we have the following algorithm:

Step 1: Set the maximum iteration number k according to the precision required.

Step 2: Discretize the control space C in a closed subset $[c_{\min}, c_{\max}]$. Five grid points are evenly distributed in the control space labeled as $\{x_A, x_B, ..., x_E\}$ (see Fig. 3).

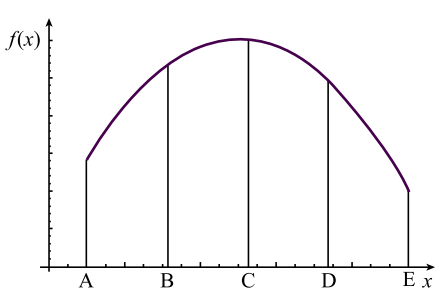

Fig. 3 Rapid Grid Search in a One-Dimensional Case

Step 3: Evaluate the values of point A and B as $f(x_A)$ and $f(x_B)$.

Step 4: If $f(x_A) > f(x_B)$, go back to step 2 and reset the searching space as $[x_A, x_B]$. If not, compute $f(x_C)$. Next, if $f(x_B) > f(x_C)$, go back to step 2 to reset the searching space as $[x_A, x_C]$. If not, compute $f(x_D)$. Next, if $f(x_C) > f(x_D)$, go back to step 2 and reset the searching space as $[x_B, x_D]$. If not, compute $f(x_E)$. Next, if $f(x_D) > f(x_E)$, go back to step 2 to reset the searching space as $[x_C, x_E]$. If not, $f(x_E)$ is the highest among the five grid points, go back to step 2 and reset the searching space as $[x_D, x_E]$. The iteration number increases by one.

Step 5: Keep going until the maximum iteration is reached. The point that dominates in the last iteration is the numerical solution of the Bellman equation.

Notice that in the best case of using the RGS method, we only need to evaluate and compare two grid points in each iteration, which is the case when $f(x_A) > f(x_B)$; in the grid search method employed by İmrohoroğlu, İmrohoroğlu and Joines (1993), one has to go over every point among five grids to find the optimum. By using the local information of the Bellman equation, RGS skips evaluating and comparing unnecessary grid points and hence speeds up the search in each iteration.

2.3.2 Algorithm 2: Two-Dimensional Case

For a Bellman equation with a two-dimensional control space C , given the state variable a_t, we can apply the following algorithm:

Step 1: Set the maximum iteration number k according to the precision required.

Step 2: Discretize the control space C in a closed subset. 25 grid points, five on each dimension, are evenly distributed in the search space.[6] These points are labeled as $\{x_1, x_2, ..., x_{25}\}$ (see Fig. 4). We divide these 25 points into five groups according to their values in control variable 1. For example, $\{x_1, x_2, x_3, x_4, x_5\}$ belongs to the same group since they share the identical value in control variable 1.

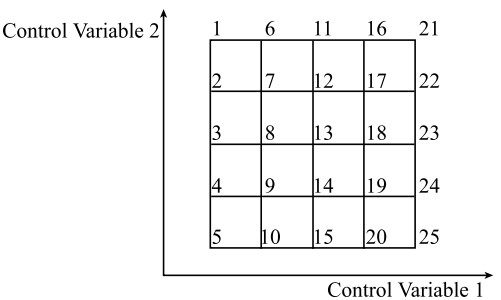

Fig. 4 Rapid Grid Search in a Two-Dimensional Case

[6] For purpose of demonstration, we discretize the grid points in a rectangular space. However, our method is not subject to this specific discretization technique.

Step 3: Use Algorithm 1 to evaluate the five grid points $\{x_1, x_2, x_3, x_4, x_5\}$ in the first group and obtain the maximum along the dimension of control variable 2. Then move to the second group $\{x_6, x_7, x_8, x_9, x_{10}\}$ and find the maximum again among these five points using Algorithm 1. Keep going for the remaining three groups. We end up with five local maximum points, one for each group. Thereafter, we compare these five points to find the (temporary) global maximum.[7] Next, we narrow down the search space to the neighborhood around this (temporary) global maximum point and go back to step 2. The iteration number increases by one.

Step 4: Keep going until the maximum iteration is reached. The point that dominates in the last iteration is the numerical solution of the Bellman equation.

Notice that for the sake of simplicity, Algorithm 2 is a straightforward extension of Algorithm 1 and it does not apply the RGS in its full length.[8] For example, we could further improve the method by taking a radial approach (rather than a line by line approach), and eliminating grid points on all rays originated from the grid point in focus. There are different ways to improve the efficiency here, and we would like to leave that to readers to deal with their specific problems. However, it is worth noting that in the best-case scenario of each iteration, we only need to go over four grid points out of 25 by using the RGS. For example, if we evaluate points 1, 2, 6 and 7 and we have $f(x_1) > f(x_2) > f(x_6) > f(x_7)$ or $f(x_1) > f(x_6) > f(x_2) > f(x_7)$, we do not need to continue searching other points because x_1 is the global maximum out of these 25 points.

3 Application

This section describes the application of our algorithm to two DP problems in macroeconomics. In both problems, it is clearly shown that the RGS method is significantly more efficient than the benchmark grid search method–the bracketing algorithm in terms of computation speed.

3.1 One-Dimensional Model

İmrohoroğlu, İmrohoroğlu and Joines (1995, 1999a) study the optimal social security replacement rate and the welfare benefits associated with it in an overlapping generations general equilibrium framework. Individuals face mortality risk and idiosyncratic

[7] For the five local maximum points we find, if all or some of them are aligned along the same straight line in the domain, we can still apply Algorithm 1 to them to increase searching efficiency.

[8] We are not able to write an optimal algorithm in a generalized n-dimensional case due to its increasing complexity. We leave that for future research.

income shock over the life cycle. However, due to the absence of a private credit annuity market, they have to use savings to self-insure against these shocks. In the model, each individual has to solve a finite-horizon finite-state DP problem, which is summarized in the following Bellman equation

$$V_j(a_j, s_j) = \max\{U(c_j) + \beta\psi_{j+1}E_{s_{j+1}}V_{j+1}(a_{j+1}, s_{j+1})\}, j = 1, 2, ..., J, \quad (2)$$

where j is the age, a_j is the asset holding at the beginning of age j, s is the state associated with employment status, c represents consumption, and ψ_{j+1} stands for the conditional probability of survival from age j to age $j+1$. The period utility function takes the form of CRRA $U(c) = \dfrac{c^{1-\gamma}-1}{1-\gamma}$. For any age j, the Bellman equation is subject to the following budget constraint

$$c_j + a_{j+1} = q_j(s_j) + (1-r)a_j + T,$$

where T is the lump-sum transfer from the government to individuals and $q_j(s_j)$ is the labor income at age j conditional on the state s_j. Notice that we can use the budget constraint to replace c_j in Bellman Eq. (2). Therefore, the DP problem reduces to only choosing the asset holding for next period a_{j+1}.

As in İmrohoroğlu, İmrohoroğlu and Joines (1993), solving the Bellman equation above involves a grid search based on a bracketing technique that we use as a benchmark case.[9] We refer readers to İmrohoroğlu, İmrohoroğlu and Joines (1999b) for the technical details of solving this finite-horizon finite-state DP problem. We then use the RGS method to repeat the exercise. In both cases, to solve the Bellman equation, we discretize the control (and state) space $C = [0, 40]$ by 4097 equally distributed grid points. The same grid points are also used for the control variable a_{j+1}. The total number of theoretical grid points is $4097 \times 4097 \times 2 \times 44$ (working age) + $4097 \times 4097 \times 21$(retirement age) $= 1.82961e + 9$ in each iteration. The model converges to the tolerance of 10^{-3} after 7 iterations for both methods.[10] As shown in Table 1, in the benchmark experiment, it takes 9.26 seconds under the current hardware.[11] Using our Algorithm 1 above, the computing time is reduced to 5.09 seconds. This RGS method saves extra 45.03% of computing time without sacrificing the efficiency gain from the İmrohoroğlu, İmrohoroğlu and Joines (1993) method.[12] The time efficiency comes from the fact that the RGS method can skip lots of unnecessary grid points. Again, as shown in Table 1, the benchmark bracketing method,

[9] The Fortran code to compute the model is downloaded from http://dge.repec.org/codes/marimon-scott/Imrohoroglu/.

[10] Total number of grid points that need to be searched using a brute force grid search is 1.28073e+10.

[11] Environment: AMD Athlon×2 5200, 4G RAM, Intel Fortran compiler for Linux.

[12] Time efficiency is defined as 1 - elapsed time$_{RGS}$/elapsed time$_{benchmark}$.

although already has a huge gain from the brute force grid search, still needs to search 1.48898e+8 grid points totally during 7 iterations; while the RGS method further reduces the number of grid points searched to $6.26218e+7$, which is only 42.06% of the grid points evaluated by the benchmark case. In other words, the RGS method speeds up the computation by skipping 57.94% of grid points evaluated by the standard bracketing method.[13]

As a robustness check, we also double the number of grid points on the state and control spaces to 8193 and solve the model. The model again converges to the tolerance of 10^{-3} after 7 iterations for both methods. Table 1 shows that the searching efficiency and time efficiency are very close to the case with 4097 grid points.

Table 1 Results for RGS Method: One-Dimensional Case

Method	**Benchmark**		**RGS**	
Grids	coarse	fine	coarse	fine
Theoretical grid number	1.28073e+10	5.12166e+10	1.28073e+10	5.12166e+10
Grid number searched	1.48898e+8	3.28655e+8	6.26218e+7	1.36070e+8
Searching efficiency	—	—	57.94%	58.60%
Elapsed time (seconds)	9.26	20.87	5.09	11.98
Time efficiency	—	—	45.03%	42.58%

3.2 Two-Dimensional Model

Braun and Nakajima (2009) investigate an infinite-horizon endogenous growth model with human capital and Epstein-Zin preference. In their model, an individual solves the following DP problem

$$V(a) = \max_{c,a',\omega_k} \{c^{1-1/\psi} + \beta(V(a'^{1-\gamma}))^{\frac{1-1/\psi}{1-\gamma}}\}^{1/(1-1/\psi)}, \tag{3}$$

subject to

$$a' = (a-c)\{R'_k\omega_k + R'_h(1-\omega_k)\},$$

where a is the asset holding at the beginning of the current period, a' is the asset holding for the next period, c is consumption, R'_k and R'_h are the returns to physical and human capital for the next period, respectively, and ω_k is the share of physical capital in total capital.[14] Replacing c in the utility function by the budget constraint,

[13] Searching efficiency is defined as 1–grids number searched$_{RGS}$/grids number searched$_{benchmark}$.

[14] The original model in Braun and Nakajima (2009) allows an idiosyncratic uninsurable shock to the return on human capital. To simplify the computation, we remove this idiosyncratic shock. Our model thus is a deterministic version of Braun and Nakajima's original model.

we can reduce this DP problem to a two-control-variable Bellman equation. The agent chooses a' and ω_k to maximize the Bellman equation. In addition to providing a two-dimensional example for solving DP problems, the specifications of the model allow nearly closed-form solutions, which offers a nice test case to check the accuracy of our numerical algorithm.

To solve this infinite-horizon DP problem using grid search method, we first discretize the state space by 1,000 grid points and the control space by 4,097 grid points in each dimension. The total number of theoretical grid points is therefore $4097 \times 4097 \times 1000$ in each iteration. We then repeatedly solve the Bellman equation for each grid point in the state space until the value function converges to 10^{-8} tolerance and the solution precision (compared to the closed-form solutions) reaches the range of 10^{-4}. We again adopt the bracketing method in İmrohoroğlu et al. (1993) as the benchmark method, and then use the RGS method as described in Algorithm 2 above to repeat the exercise.[15] In both cases, the model converges to the range of tolerances after 11 iterations.[16] As shown in Table 2, a huge efficiency gain shows up both in the computing time and in the number of grids actually searched. The RGS method saves about 66% of computing time and about 59% of grid points searched. As a robustness check, we then double the number of grid points along each dimension in the control space from 4,097 to 8,193 and recompute the model. Not surprisingly, significant efficiency gains appear again with finer grids using the RGS method.

Table 2 Results for RGS Method, Two-Dimensional Case

Method	**Benchmark**		**RGS**	
Grids	coarse	fine	coarse	fine
Theoretical grids number	1.84639e+11	7.38378e+11	1.84639e+11	7.38378e+11
Grid number searched	2.75000e+6	3.02500e+6	1.12868e+6	2.12297e+6
Searching efficiency	—	—	58.96%	29.82%
Elapsed time (seconds)	62.21	68.18	21.24	38.18
Time efficiency	—	—	65.86%	44.00%

4 Conclusion

We introduce a rapid grid search method in solving the dynamic programming problems in economics, which inherits advantages of the standard grid search method. Going one step further, by using the local information of the Bellman equation, this method can significantly increase the efficiency in solving DP problems by reducing

[15] Environment: AMD Athlon×2 5200, 4G RAM, Matlab 2010a in Windows XP.

[16] Total number of theoretical grids thus is $4097 \times 4097 \times 1000 \times 11 = 1.84639\text{e+}11$.

the number of grid points searched in the control space. By applying this method to a one-dimensional and a two-dimensional case, respectively, we obtain a significant gain in efficiency by reducing the computational time compared to the benchmark grid search algorithm. This method can also be easily implemented and applied to higher dimensional DP problems. Therefore, it can offer a possible way to help relieve the "curse of dimensionality" arising from the high-dimensional DP problems in economics.

Acknowledgements We would like to thank Toni Braun, Jim Feigenbaum, Zhigang Feng, Ayse İmrohoroğlu, Selo İmrohoroğlu, Thomas Ramsey, the anonymous referees and the editor for their helpful comments and suggestions.

References

Braun A, Nakajima T (2009). How large is the intertemporal elasticity of substitution? Unpublished mimeo

Grune L, Semmler W (2004): Using dynamic programming with adaptive grid scheme for optimal control problems in economics. Journal of Economic Dynamics and Control, 28: 2427–2456

Ímrohoroğlu A, Ímrohoroğlu S, Joines D (1993). A numerical algorithm for solving models with incomplete markets. International Journal of Supercomputer Applications and High Performance Computing, 7: 211–230

Ímrohoroğlu A, Ímrohoroğlu S, Joines D (1995). A life cycle analysis of social security. Economic Theory, 6: 83–114

Ímrohoroğlu A, Ímrohoroğlu S, Joines D (1999a). Social security in an overlapping generations model with land. Review of Economic Dynamics, 2: 638–665

Ímrohoroğlu A, Ímrohoroğlu S, Joines D (1999b). Computing models of social security. In: R. Marimon and A. Scott (eds.), Computational Methods for the Study of Dynamic Economis. Oxford: Oxford University Press, 221–237

Stokey N, Lucas R, Prescott E C (1989). Recursive Methods in Economic Dynamics. Cambridge, MA: Harvard University Press