Proc. of VI Int. Conf. on Gyro Technology, Navigation, and Motion Control, Kiev, 2007, vol. 2, pp. 104-110

Application of State Space Search Algorithms to Optimal Control

V. Apostolyuk

National Technical University of Ukraine
"Kiev Polytechnical Institute"
Kiev, Ukraine
apostolyuk@astrise.com

Introduction

State space search is usually used in the field of artificial intelligence, in which successive states of an instance are considered, with the goal of finding a path to some desired state. Many different searching algorithms were also developed for the connected graphs or trees traversing [1-3]. Application of such algorithms to continuous state-space is somewhat difficult since it leads to the potentially infinite number of possible states after space partitioning on one hand, or loss of the accuracy on the other. At the same time solving optimal control problems with the existing algorithms often considered to be impractical due to the exponential growth of the states to be analysed. This paper addresses the mentioned above problems as well as other important aspects of solving the optimal control problems.

Optimal control problem

Formulation of general optimal control problem is well known [4] and here we shall give emphasis only to the aspects that are essential to the subject. Let the state of the system at time t be a vector $\vec{x}(t) = \{x_1(t),...,x_n(t)\}$ in an n-dimensional Euclidean space which we shall call the state space X. The steering device or control we model as an m-dimensional vector function of time $\vec{u}(t) = \{u_1(t),...,u_m(t)\}$. The components of $\vec{u}(t)$ are allowed to be piecewise continuous and the values they can take are bounded so that at any time t, $\vec{u}(t)$ lies in some bounded region U of the control space. Without loss of generality we impose the restriction $|u_i| \le 1$, $i = 1,...,m$. Such controls are deemed admissible in terms of the considered algorithms.

We shall study systems whose behaviour can be modelled by the most general state transfer function F that calculates state of the system given its current state and control vectors:

$$\vec{x}(t+\Delta t) = F(\vec{x}(t), \vec{u}(t)). \tag{1}$$

Function F here is assumed to be defined for all $\vec{x} \in X$ and all admissible $\vec{u}$.

This approach to system representation significantly expands number of systems that could be successfully controlled compared to the conventional systems representation via systems of either ordinary or non-linear differential equations.

We now wish to control the system from the initial state $\vec{x}(t_0)$ to the given final state $\vec{x}(t_d)$. It is also could be done in such a way that some cost functional

$$J = \int_{t_0}^{t_1} f_0(\vec{x}(t), \vec{u}(t))dt \tag{2}$$

is minimized. We are, of course, assuming that there are admissible controls that transfer the system from $\vec{x}(t_0)$ to $\vec{x}(t_d)$ and we are looking amongst this subset of admissible controls for a control that minimizes J.

In order to apply inherently discrete state-space traversal algorithms to this problem it is necessary to introduce certain level of discretization to the optimal control problem formulation.

System state transition

Let us assume that every component of the control vector $\vec{u}(t) = \{u_1(t),...,u_m(t)\}$ can take a value only from the given fixed set of possible values: $u_i(t) \in \{u_1^*,...,u_p^*\}$. In this case control vector $\vec{u}(t)$ has $s = m^p$ different possible fixed values $\vec{u}(t) \in \vec{u}_1^*,...,\vec{u}_s^*$. Application of the control $\vec{u}_i = \vec{u}(t_i)$ to the system at the current state $\vec{x}_i = \vec{x}(t_i)$ can result therefore in s new states at the next moment of time $t_{i+1} = t_i + \Delta t$. In terms of the optimal control problem formulated earlier we are searching for the finite sequence of q control inputs $\vec{u}_i(t) \in \{\vec{u}_1^*,...,\vec{u}_s^*\}$ that transfers the system from its initial state $\vec{x}_0$ to the desired state $\vec{x}_d$.

$$\begin{tikzcd} \boxed{\vec{x}_0} \arrow[r, "\vec{u}_1"] & \boxed{\vec{x}_1} \arrow[r, "\vec{u}_2"] & \cdots \arrow[r, "\vec{u}_d"] & \boxed{\vec{x}_d} \end{tikzcd}$$

Figure 1. System state transition

Each of the control $\vec{u}_i$ is assumed to be acting on the system and remained constant within the time interval Δt. Such a limitation causes the found solution to be a sub-optimal rather than optimal, to which it tends when $\Delta t \to 0$.

State-space search algorithm

As a framework and a starting point for the further development the A-Star algorithm is used [2], which could be considered as a modification to the well known Dijkstra's connected graph traversing algorithm [1]. The proposed modification of these algorithms, which is used for control searching, is shown in the Table 1.

Here the variable `x` holds the current state $\vec{x}_i$ of the system as well as the control that leads to this state, successor states `x'` are the system states $\vec{x}_{i+1}$ that were produced by applying different controls from the given set $\{\vec{u}_1^*,...,\vec{u}_s^*\}$ to the system (1).

There are four functions in the algorithm that completely define its operation: "Equal", "Constrained", "Cost", and "Score".

The "Equal" function compares the two given states and returns True if these states appear to be the same based on the chosen criteria. The simplest criteria would be to compare the Euclidean distance between the two states with some threshold function

$$\left|\vec{x}_i - \vec{x}_j\right| \le D(\vec{x}_i, \vec{x}_j). \tag{3}$$

Here the threshold function $D(\vec{x}_i, \vec{x}_j)$ could be simply related to the constant acceptable error of control, or could also depend on either one or both of the compared states. Alternatively, the criteria could be based on the states vectors element comparison. One should note that this function is also heavily used to find the given state in the lists of closed and open states.

The "Constrained" function is used to check any state against known constraints. It returns True if the state falls under any of the constraints in the system. There is no required special form whatsoever, to which constraints should be limited in such an algorithm. The "Cost" function implements calculation of the cost functional (2).

Table 1. Listing of the algorithm framework

```
function FindControl : Boolean;
var
     x      : state;
     x′     : state;
     Open   : priorityqueue;
     Closed : list;
begin
     push x0 on Open
     while Open is not empty do begin
           pop state x from Open
           if Equal( x, goal ) then begin
              produce control to x;
              result := true;
              exit;
           end;
           for each successor x′ of x do begin
               if Constrained( x′ ) then continue;
               NewCost := x.cost + Cost( x, x′ );
               if (x′ is in Open or Closed)
                   and x′.cost <= NewCost then continue;
               x′.cost := NewCost;
               x′.Score := Score( x′ );
               if x′ is in Closed then remove x′ from Closed;
               if x′ is not in Open then push x′ on Open;
           end;
           push x on Closed
     end;
     result := false;
end
```

Finally, the “Score” function must provide the criteria for prioritizing states in the “Open” priority queue. In the classical A-Star algorithm this function returns sum of the cost $C(\vec{x}_0, \vec{x}_i)$ to the given state and some heuristic evaluation of the cost from the given state $\vec{x}_i$ to the goal state $\vec{x}_d$:

$$S^*(\vec{x}_i) = C(\vec{x}_0, \vec{x}_i) + H(\vec{x}_i, \vec{x}_d). \tag{4}$$

Another useful criterion that can be used to prioritize states in the queue is the angle between the following two vectors: vector from the previous state to the current state, and vector from the previous state to the goal state. This criterion is calculated as

$$A(\vec{x}_i) = 1 - \frac{(\vec{x}_i - \vec{x}_{i-1})(\vec{x}_d - \vec{x}_{i-1})}{|\vec{x}_i - \vec{x}_{i-1}||\vec{x}_d - \vec{x}_{i-1}|}. \tag{5}$$

Additional scoring functions $S(\vec{x}_i)$ can be derived combining criteria (4) and (5):

$$S(\vec{x}_i) = S^*(\vec{x}_i) \cdot A(\vec{x}_i), \tag{6}$$

$$S(\vec{x}_i) = S^*(\vec{x}_i) + A(\vec{x}_i). \tag{7}$$

All of these scoring functions (4)-(7) can now be evaluated in terms of solving the optimal control problem.

Algorithm testing

The testing case is the control of a linear system that is defined by simple ordinary differential equations

$$\frac{d}{dt}\begin{bmatrix} X \\ V \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -0.5 \end{bmatrix}\begin{bmatrix} X \\ V \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} U, \tag{8}$$

from the state $\vec{x}_0 = \{1,1\}$ to the state $\vec{x}_d = \{0,0\}$ with respect to either minimal time or minimal distance in the state-space. The system (8) will be controlled with and without presence of constraints. The controls are allowed to have only three admissible values [-1, 0, 1], and the control introduction time step is 0.25s. Algorithm is tested on Athlon64-3000 CPU with 1GB memory. For the given conditions the algorithm testing results are presented in the Table 2.

Table 2. Algorithm testing results

Scoring function	**States**	**Solution time [s]**	**Distance**	**States**	**Solution time [s]**	**Time [s]**
	Distance (figure 2)			*Time (figure 4)*		
(4) – S*	3890	5.951	3.008	*3000*	*1.984*	*3.25*
(5) – A	**251**	**0.031**	**3.008**	304	0.044	11.25
(6) – A·S*	341	0.036	3.008	353	0.051	11.25
(7) – A+S*	1683	0.697	3.279	**1368**	**0.430**	**4**
	Distance with constraints (figure 3)			*Time with constraints (figure 5)*		
(4) – S*	6642	14.672	4.48	*5694*	*8.661*	*4.5*
(5) – A	**366**	**0.054**	**4.45**	370	0.054	10.75
(6) – A·S*	357	0.040	4.45	378	0.047	10.75
(7) – A+S*	2099	1.138	4.48	**3011**	**2.144**	**4.75**

The best found trajectories in state-space along with its control functions are presented in the following figures 2-5.

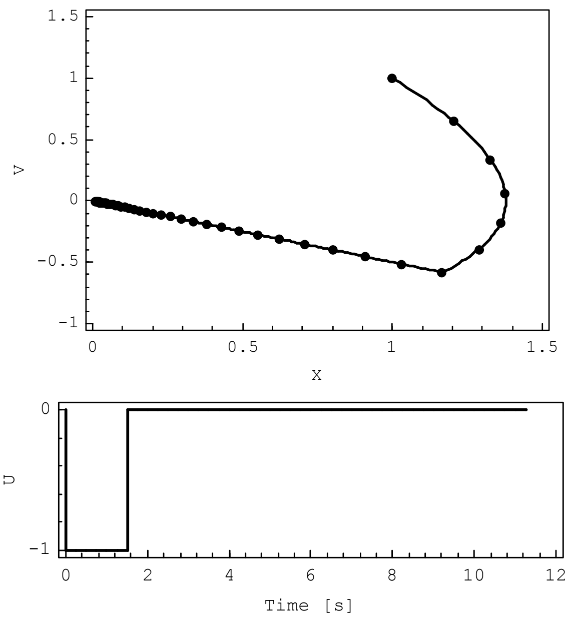

Figure 2. Minimal distance solution

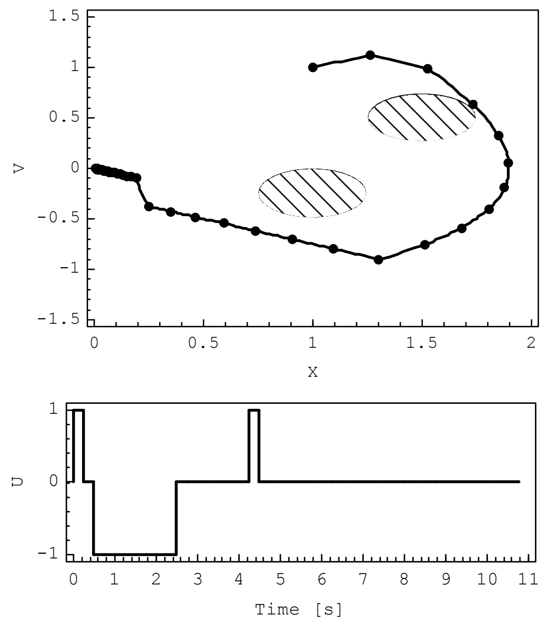

Figure 3. Distance with constraints solution

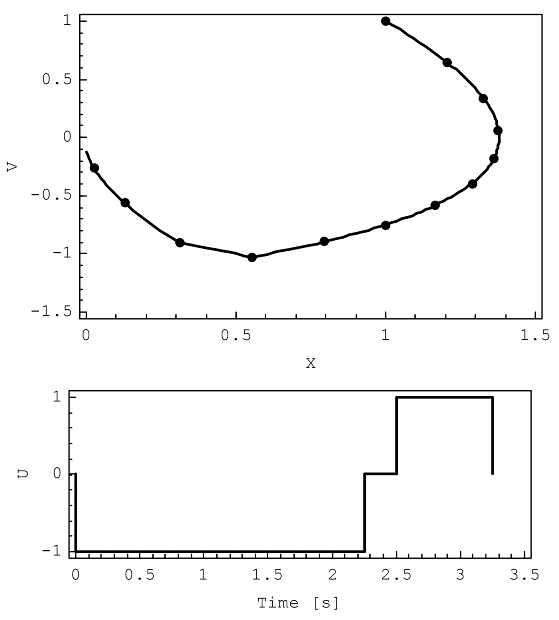

Figure 4. Minimal time solution

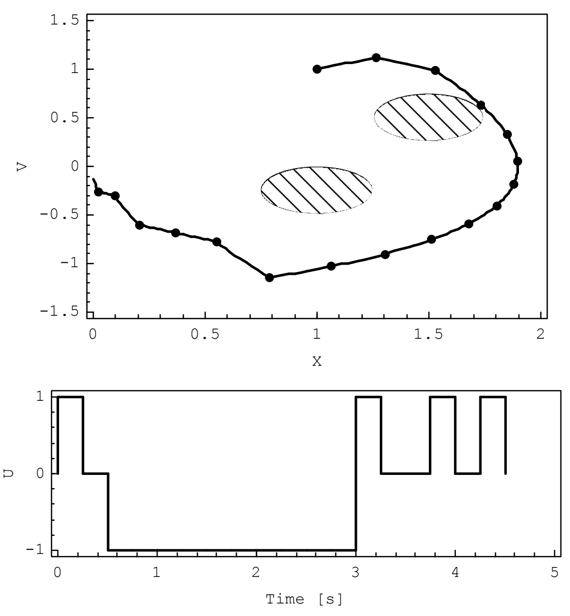

Figure 5. Time with constraints solution

Analysis of the obtained results shows that the best results in solving minimal distance problem are achieved with the directional scoring function (5). Although the function (4) delivers the most optimal control solution, while solving the minimal time problem, its performance is not acceptable for the on-line applications. However, the scoring function (7) delivers near-optimal solutions but in 4-5 times faster. Needless to say that the tested above scoring functions do not cover all of the possibilities and certainly may require further developments with respect to the different optimization criteria.

One should also note that the directional scoring (5) delivers the optimal solution at about 200 times faster than the conventional A-Star scoring (4). Such an improvement allows this modification to be used in on-line control application as well.

Execution time of the algorithm is approximately linearly related to the square of traversed states number. This dependence is shown in the figure 6. If the algorithm is intended for on-line applications, the problem of reducing the states number without loss of accuracy is the most crucial problem to be addressed.

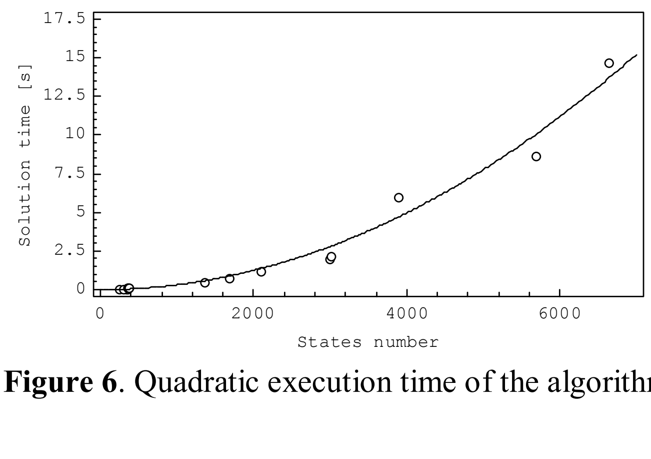

Figure 6. Quadratic execution time of the algorithm

Conclusions

Investigated algorithms allow sophisticated and highly flexible control system to be developed for the wide range of different systems. Solely by its nature the presented here approach delivers the following benefits as compared to the conventional optimal control methods:

- Non-analytical system representation (state transfer function instead of differential equations).
- Straightforward dealing with arbitrary constraints and optimization criteria.
- Both optimal control and trajectory planning problems are solved simultaneously.
- Insensitivity to unpredictable drift of system parameters.
- Simple and easy to implement algorithm structure.

Nevertheless, the vast amount of problems is still there, including the further improvement of scoring functions for the specific optimisation criteria.

References

[1] Dijkstra E. W., "A note on two problems in connexion with graphs", Numerische Mathematik, 1 (1959), S. 269–271.
[2] Nilsson N. J., "Problem solving methods in artificial intelligence", McGraw Hill, 1971.
[3] S. Koenig and M. Likhachev. Real-Time Adaptive A*. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pages 281-288, 2006.
[4] Pinch E.R., "Optimal Control and the Calculus of Variations", Oxford University Press, 1993.