Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem: Linear Quadratic Dynamic Optimization Problem

Yasuhiko TAKAHARA (Chiba Institute of Technology)
Tosio SAITO (Nihon University)

Abstract: We have proposed a systems theory approach to management information system (MIS) development. The approach is called a model theory approach. In general an MIS is required to perform both a problem solving and a transaction processing functions. extSLV is produced as a general solver for the problem solving function when the model theory approach is applied to the function. extSLV has been applied to various problems for an MIS. This paper applies extSLV to a linear quadratic (dynamic optimization) problem (LQP), to examine its generality.

The LQP, which is described by a differential equation, is adopted for four reasons to examine generality of extSLV. First, the LQP is the most typical analytical dynamic optimization problem. Second, because extSLV has been applied only to MIS problems, it is suspected to be applicable only to a combinatorial problem, whose search space is a finite set. The paper will demonstrate that extSLV is not necessarily restricted to a finite search space by dealing with the LQP. Third, as a data processing aspect of an MIS indicates, numerical processing is a base of an MIS. If extSLV can handle a problem described by a differential equation, it can be expected to satisfactorily address numerical aspects of MIS problems. Regarding non-numerical problem solving it has been demonstrated that extSLV is quite effective for it. Fourth, since the target problem is an analytical one, we can delve into deeper properties of extSLV than those MIS development can reveal.

The paper shows that extSLV can address the LQP without any modification and furthermore, because of the backtracking function on local optimization it can present a non-conventional solution to a problem which cannot be solved analytically.

This paper requires some background knowledges which may not be considered within the domain of management information system. They are briefly introduced in Appendices. The essence of the paper, however, should be understood without consulting Appendices.

Keywords: management information systems development, system theory, control problem, extSLV

1. Introduction

We have proposed a system theory approach to management information system (MIS) development, that is called a model theory approach [3,4,5,6,7,8,9,15]. An MIS of this paper indicates an information system which consists of four layers, file system (data base) layer, transaction processing layer, data transformation layer and problem solving (DSS) layer. Figure 1 shows a browser-based MIS in the model theory approach where an adaptive function in the problem solving layer is ignored for the sake of simplicity [3,5].
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

Since the problem solving function is assumed to depend on data handled by the transaction processing function, they are hierarchically modeled. A user input (command) is sent to atomic processes through a user interface (UI) and processed by an atomic process corresponding to the command. The atomic process updates the file system or solves a problem and returns a proper response to the user. Although only two atomic processes are presented in Figure 1, in general, many atomic processes can exist in an MIS. There are two types of atomic processes, transaction type and solver type. A transaction type is a simple process which performs an update action of the file system. A solver type performs problem solving. For example, when a temporary staff recruiting system is developed [16], one atomic process is required to perform an optimum matching task between applicants and job offering clients. Although the recruiting system is conventionally developed as a transaction system, it cannot be realized without implementation of problem solving. A solver type consists of three components, problem specification environment (PSE) constructor, extSLV and interface. The PSE constructor provides information of the target problem. extSLV is the engine for problem solving, which is the main topic of this paper. The family of transaction type atomic...
processes in an MIS will be called a transaction processing system (TPS) whereas the family of solver type processes a problem solving system (PRS).

The model theory approach is based on the assertion that in order to obtain a solid theory for MIS development, it must be investigated from the system theoretic viewpoint rather than the traditional software engineering viewpoint because an MIS is a system.

The model theory approach is characterized by the following features.

- It is not a software engineering approach but a systems theory approach based on a systems model.
- It has the model theory structures (relational structures) of a TPS and a PRS, derived from general systems theory concepts.
- It is a formal approach in the sense that a model of a TPS or a PRS is developed following the relational structure and described in set theory.
- A model of a TPS or a PRS described in set theory can be translated into an executable system automatically as an integral part of the approach.
- System construction of a TPS or a PRS is supplemented by standardized components specified by the formal structures, facilitating rapid systems development.
- The implementation language is extProlog which is an extension of regular Prolog to meet MIS development.

The approach claims the following advantages.

1. It provides a reliable system specification. This is a general assertion claimed for formal approach.
2. It facilitates correct implementation and rapid system development because it provides a user interface (for a TPS) and a goal-seeker (for a PRS) as black box components for MIS development and realizes automatic systems generation. Once a system specification is given in computer-acceptable set theory, an executable TPS or a PRS is generated. The executable system is expected to present a faithful realization of the specification.
3. It can facilitate inexpensive systems development because it realizes rapid systems development and uses open softwares (Linux, apache, postgresql, php).
4. It can realize an intelligent MIS because it covers both problem solving and transaction processing functions on an integrated platform.
5. It can facilitate maintainability by an end user because system construction can be done without using a computer language but elementary set theory.

extSLV is a general solver produced for an MIS when the model theory approach is applied to problem solving. Although it is a general solver, it is not created to replace conventional solvers. It is our standpoint that if a conventional solver like linear programming can be used, it should be used.

extSLV consists of two components, a goal-seeker and a user model. The goal-seeker is standardized and provided as a black box to system development whereas the user model is constructed depending on the target problem. The previous papers have developed extSLV for various MIS problems [3,4,5,6,8]. This paper investigates its true generality by applying it to a problem outside of the conventional MIS domain.

The LQP is the most typical optimization problem of control engineering. A control engineering problem, which is usually described by a differential equation and solved by control engineering techniques, is adopted for three reasons to examine generality of extSLV. (The LQP is used in control engineering approach to economics [12].) First, because extSLV has been applied to MIS problems, it is suspected to be workable only for a combinatorial problem. The basic feature of a combinatorial problem is that its search space is a finite set. On the other hand, a control

Vol.14 No.4, Mar. 2006
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

eering problem provides an infinite search space. The paper will demonstrate that extSLV is not necessarily restricted to a finite search space.

Second, as a data processing aspect of an MIS indicates, numerical processing is a base of an MIS. If extSLV can handle control engineering problems which present the most typical numerical problems (their models are described by differential equations), it can be expected to satisfactorily address numerical aspects of MIS problem solving. It has been already demonstrated that extSLV is quite effective for non-numerical problem solving.

Third, since the control engineering problem is an analytical one, we can delve into deeper properties of extSLV than those MIS development can reveal.

In summary, control engineering can provide a problem whose character is so different from that of an MIS problem that it can be considered suitable for checking generality of extSLV.

Some comment should be made regarding other approaches that have dealt with optimization problems using set theory and logics [17], and logic programming languages for implementation. Although these methods appear quite similar to the model theory approach, there are a number of important differences.

First, the most fundamental difference is that these previous approaches are concerned with the derivation of a solving algorithm for a given specific problem, whereas the model theory approach is first concerned with the structure of a problem solving system in general, which is formalized as a combination of the standardized goal-seeker and a user model, and then development of a solving structure of a specific problem based on this structural analysis. In short, a problem solving system is constructed following the general meta-system model. Consequently, extSLV is uniformly applicable to a non-numeric problem as well as to a numeric problem. Conventional solvers are effective only for numeric problems. For example, the magic square problem could not be solved by the goal-seeker of Excel.

The second difference is therefore in the classification of problems. In algorithm-oriented theories, problems are usually assumed to be defined on numerical spaces (e.g., Euclidean spaces), and are classified using metric space concepts such as linear or nonlinear problems, or convex problems. The classification of the model theory approach is concerned with the structure of a problem and not with its expression (see Appendix V).

Figure 2 extSLV
We briefly summarize extSLV development [3,4,5,6,7,8,9]. Construction of extSLV is based on the goal seeking system model of GST [3,5]. Figure 2 shows its structure.

extSLV is an input output system whose input is a problem specification environment (PSE) and whose output is a solution. The PSE specifies a problem to be solved. It may or may not be a problem itself. In this paper it is an optimization problem (see Figure 5). In extSLV the transformation from a PSE to a solution is realized by the structure of Figure 2. As mentioned above, it consists of two components, user model and goal seeker.

The user model component is realization of the following relational structure [3,4]:

user model=<A,C,delta,genA,constraint,
finalstate,goal,st,c0>,

where

A : set of actions;
C : set of states;
c0∈C : initial state;
Re : set of reals.
delta:C×A→C; //state transition
genA:C→φ(A); //allowable actions for
constraint:C→{true,false}; //permissible state
initialstate():→{c0}; //initial state
finalstate:C→{true,false}; //final state
goal:C→Re; //evaluation of a state
st:C→{true,false}; //stopping condition.

The meaning of functions will be presented in Chap. 4. The user model is prepared in set theory by a system developer according to the PSE. In fact it is the only component a system developer must construct in extSLV.

A user model is an abstract model of the well known concept of problem solving. That is, a problem is a gap between a current state and a desired state and a problem solving process is an activity to transfer the current state to the desired one. The transformation process is formalized as an automaton in the relational structure (see Figure 2). The PSE is used to define the structure of the automaton. The user model has an input a, which is a control variable or a decision action for the automaton. The output of the user model is a state c. The user model is a main topic of the subsequent chapters.

The goal seeker component determines a value for the control variable a to achieve a given goal (using the state information c). It yields a solution y. It is supplied by the approach as an independent standardized component and hence is treated completely as a black box for extSLV development.

The goal seeker is characterized by an adopted strategy to determine the action. Hill climbing method with a push down stack (PD method) is used for the goal seeker of this paper. The PD method determines an action by local optimization where the push down stack is used for back tracking. Appendix I presents an outline of the PD method. Details of it is presented in Ref. [3].

The separation of a user model and the goal seeker component facilitates rapid systems development.

Figure 3 shows the proposed development procedure of extSLV [6].

According to the procedure first a user model is constructed according to a PSE and then it is compiled into extSLV in extProlog at Stage 5. The solver yields a solution. extProlog is a Prolog extended for MIS development [14]. extSLV has been applied to various problems for an MIS. (Table 2 in Appendix V illustrates examples of attacked problems).
2. Linear Quadratic Problem

This paper will investigate a problem described in the following form:

\[ u'' + 2\alpha u' + \beta u = a \]

A typical image of the problem is a dynamic problem shown in Figure 4.

![Figure 4 A control engineering problem](image)

The dynamic problem consists of a body, whose mass is \( m \), and a spring, whose spring constant is \( k \). The body is fixed by the spring to the wall. The input of the problem is a force \( a \), which is a time function, applied to the body and the output is displacement of the body. Suppose that an initial condition \( (u_0, v_0) \) is given arbitrarily, where \( u_0 \) is an initial displacement and \( v_0 \) is an initial velocity. Then the control problem is to determine a force \( a \), which is a time function, so that the body is taken to a state \((s, 0)\), where \( s \) is also an arbitrarily given set point.

The above equation can be transformed into a state space representation in the usual way. That is, the state is \((\text{displacement}, \text{velocity})\). The representation is:

\[
\begin{align*}
\dot{u} &= v, \\
\dot{v} &= -\beta u - 2\alpha v + a.
\end{align*}
\]

Then, a standard linear quadratic optimization problem can be formalized for the equation (1) as below [12].

### Linear quadratic optimization problem with constraint

Let an initial state be \((u_0, v_0)\). Let a desirable state
be \((s,0)\). Find a time function \(a\) which minimizes the following performance function:

\[
g((u,v),a) = \int_{0}^{t_f} ((u,v)-(s,0))^T Q((u,v)-(s,0)) - (s,0)) + ra^2 dt
\]

subject to \(|u(t)-s|\leq|v(t)|\) for \(t\in[0,t_f]\)

where \(Q>0\) (positive symmetric matrix), \(r>0\) and \([0,t_f]\) is a time span for the optimization. \(u, v, a\) are time functions over \([0,t_f]\). The constraint \(|u-s|\leq|v|\) is introduced arbitrarily to produce a situation where an analytical solution could not be found. (It can be interpreted as a "soft landing" condition.)

There are two cases for the optimization problem \([12]\). The first case requires the boundary condition \((u(t_f),v(t_f))=(s,0)\) should hold. The second case allows \((u(t_f),v(t_f))\) to be free. This paper deals with the second case because it is simpler than the first. In the second case the boundary condition \((u(t_f),v(t_f))=(s,0)\) is indirectly achieved by the term \(((u,v)-(s,0))^T Q((u,v)-(s,0))\) (see the case of \(r=0.001\) in Chap. 4).

The LQP (linear quadratic problem) is the most basic dynamic optimization problem \([1,12]\). Although the target problem is a simple second order system, it can display every qualitative behavior of dynamics. (Appendix II shows the standard qualitative classification of dynamic behaviors). Furthermore, if a constraint is posed on the behavior, it becomes quite difficult to get an analytical solution. (Appendix III briefly discusses a standard analytical technique, maximum principle). It should be also noted that the dynamic programming method cannot work for the current problem because the state space is an infinite set. This paper will shows that extSLV derives a semi-optimum solution for the problem. The concept of a semi-optimum solution is discussed in Ref. \([3]\). (Appendix IV compares a result of extSLV with a well known optimum solution for a special case).

3. Analytical Properties of Target Problem

a. Controllability Property

The problem has the controllability property \([2,10]\). That is, there is an action \(a\) which takes \((u_0,v_0)\) to \((s,0)\) or the problem is solvable if no constraint is posed on \(a\). This can be easily checked by the rank condition. Let

\[
F=\begin{pmatrix}
0 & 1 \\
-\beta & -2\alpha
\end{pmatrix}, \quad g=\begin{pmatrix}
0 \\
1
\end{pmatrix}
\]

Then,

\[
\text{rank}[g \ F g]=2.
\]

This result indicates that any initial state can be steered to the target state if the performance function and a constraint are ignored.

b. Stability Property

Although the problem satisfies the controllability property, it does not say anything about difficulty of control. The difficulty is related to stability property of the target dynamics. If the dynamics specified by the matrix \(F\) is unstable, the action \(a\) must fulfill two tasks, suppression of the instability in addition to steering the state to the target point. If the dynamics is stable, \(a\) has only to take care of the steering action.

The dynamic mode of \(F\) is classified according to the roots of the quadratic equation \(\lambda^2+2\alpha\lambda+\beta=0\) \([10]\). (Appendix II shows the classification). The paper will investigate the worst case of stability.
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

4. Development of User Model

We will develop extSLV for the optimization problem following the general procedure of Figure 3. extSLV has a problem classification scheme which can help us with transformation of a PSE to a user model. (Appendix V shows the classification scheme [3,4,5,6]). Regarding the optimization problem, since the output is a sequence of decisions \(a\), the problem has the explicit solving action property. Furthermore, since it has a given goal, the problem has the closed goal property. As for the target state, there are two cases that the target state is fixed as \((s,0)\) or not. Then, in the former case the problem belongs to the E-C-C (Explicit solving action - Closed goal - Closed target) class while the latter belongs to the E-C-O (Explicit solving action - Closed goal - Open target) class according to the problem classification. As mentioned in Chap. 2, this paper will investigate the problem as an E-C-O problem.

4.1 User Model Specification

a. Input-output Block Diagram

Figure 5 shows the stage 1 for extSLV development.

![Input-output diagram for extSLV](image)

This stage confirms a basic character of the target system.

b. User Model Construction

A user model: \(<A,C,delta,genA,constraint,\>

finalstate,goal,st,c> is to be realized for extSLV of

Figure 5.

(i) \(delta\)

It should be noted that although the delta of the user model is conceptually completely different from the state transition function (1), it can be shown for the explicit solving action case that Equ. (1) can be used for the delta [3,6]. Equ. (1) is, therefore, used for the delta.

Since extSLV is implemented on a computer, its input, the optimization problem of the differential equation model, must be transformed into a problem of difference equation model. The difference equation model will be generated by the mean value theorem in the following way:

\[
\frac{du}{dt} = v \rightarrow (u(t+h)-u(t))/h = v(t+\theta h) \quad (0 \leq \theta \leq 1),
\]

\[
\frac{dv}{dt} = -\beta u - 2\alpha v + a \rightarrow (v(t+h)-v(t))/h = -\beta u(t+\theta h) - 2\alpha v(t+\theta h) + a(t+\theta h) \quad (0 \leq \theta \leq 1).
\]

Then, the dynamic system can be described by the following difference equation:

\[
v(k+1) = -h\beta u(k) + (1 - 2\alpha h)v(k) + ha(k),
\]

\[
u(k+1) = u(k) + hv(k+1).
\]

Notice that \(h=1\) is selected for \(du/dt=v\) in order to compensate the delay of the target problem.

If the state set is identified by the conventional way as \(UxV\), the relation (2) implies that the goal is not a function on the set but its trajectory set. This is incompatible with the user model formulation. We must extend the state set so that the goal can be a function of a state variable.

The extention is done by introducing the following two relations:

\[
\frac{dg}{dt} = ((u,v)-(s,0))^TQ((u,v)-(s,0)) + ra^2,
\]

\[
\frac{dt}{dt} = 1.
\]

The last relation is introduced to represent the fact that the solving activity is terminated by the upper time limit of the integral \(t\). This will be shown by the definition of \(st()\) given below.
Let a desirable state be specified as below.
\[ c=(u, v, g, t) \in U \times V \times G \times T = (\mathbb{C}), \]
where \( U = V = G = T = \mathbb{R} \) (set of reals).

Let the set of actions be
\[ A = \{ a_1, \ldots, a_n \} \]
where \( a \in \mathbb{R} \). Since extSLV is implemented on a computer, a value of an action should be discretized.

Then, the state transition function \( \delta: C \times A \rightarrow C \)\( \delta((u, v, g, t), a) = (u_2, v_2, g_2, t_2) \) is given by the following equations:
\[
\begin{align*}
 v_2 &= -h \beta u + (1-2 \alpha h) v + h a \\
 u_2 &= u + h v_2 \\
 g_2 &= g + h(q_1(u_2^2) + q_2 v_2^2 + r u^2) \\
 t_2 &= t + h
\end{align*}
\]
where
\[
 Q = \begin{pmatrix}
 q_1 & 0 \\
 0 & q_2
\end{pmatrix}
\]
is assumed \( q_1 > 0, q_2 > 0 \).

In general, the state transition function of extSLV is a partial function. Usually two functions stated below, \( \text{genA}() \) and \( \text{constraint}() \), are introduced to secure its proper behavior, by which \( \delta \) is constrained as follows:
\[
\delta(c, a) = c' \rightarrow a \in \text{genA}(c) \text{ and } \text{constraint}(c') = \text{true}.
\]

(ii) Output
Let the output set \( Y \) be
\[ Y \subseteq A^* \]
where \( A^* \) is the free monoid of \( A \). It should be noted that a solution is an element of \( Y \).

(iii) \text{genA}
For the current case
\[ \text{genA}(c) = A. \]
\text{genA}(c) is the set of allowable actions for the state \( c \).

(iv) \text{constraint}
\[ \text{constraint}([u, v, g, t]) = \text{true} \leftrightarrow |u-s| \geq |v|. \]

(v) \text{initial state}
The initial state \( c_0 \in C \) is naturally defined as below:
\[ c_0 = (u_0, v_0, 0, 0) : \text{initial state} \]

(vi) \text{final state}
\[ \text{finalstate}: C \rightarrow \{ \text{true, false} \} \text{ is undefined.} \]
This specification comes from the problem character, \( E \rightarrow C \rightarrow O \).

(vii) \text{goal}
The performance function \( \text{goal}: C \rightarrow \mathbb{R} \) is naturally defined as below.
\[ \text{goal}((u, v, g, t)) = g. \]

(viii) \text{stopping condition}
The stopping condition decides when the solving activity must stop. Then, \( st: C \rightarrow \{ \text{true, false} \} \) is given by the last component of the state.
\[ st((u, v, g, t)) = \text{true} \leftrightarrow t = t_f \]

4.2 Implementation and Execution
When the elements of the user model \( <A, C, \delta, \text{genA}, \text{constraint}, \text{finalstate}, \text{goal}, st, c_0> \) are described in computer-acceptable set theory, the entire model can be translated by setcompiler into extSLV as an executable system in extProlog. (It should be noted that, for example, "∪" cannot be read by a computer. It should be replaced by, for example, "union"). The action sequence \( a \in A^* \) in the behavior of extSLV is a solution for the LQP.

Regarding stability, since the unstable focus presents one of the most difficult cases, that case is used for the subsequent arguments, where \((\alpha, \beta) = (-1, 2)\). (Table 1 of Appendix II shows specification of unstability). The difficulty is measured by the required range of \( A \) to control the target. \( h = 0.1, t_f = 5 \) and \( s = 5 \) are fixed.

The following “lqp.set” is the entire model
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

description in computer-acceptable set theory.

/*linear quadratic optimization problem*/
/*mechanical system of Fig.4*/
/*E-C-O problem*/
/*mu"=a-ku*/
/*v"=-ku/m+a/m*/
/*lqp.set*/
.func([goalElement,setProperty]);

delta([U,V,GoalV,T],A)=[U2,V2,GoalV2,T2] <->
  V2=0.1*A -0.1*U+V,
  U2=U+0.1*V2,
  R:=goalElement(U2,V2,A),
  GoalV2:=GoalV+0.1*R,
  T2:=T+0.1,
  constraint([U2,V2,GoalV2,T2]),!;

goalElement(U,V,A)=R <->
  S:=setProperty(),
  R:=(U-S)*(U-S)*30+V*V+1+A*A*0.01;

genA(C)=As <->
  As=[-80,-40,-20,-10,-5,-3,-1,0,1,3,5,10,20,40];

constraint([U,V,GoalV,T]) <->
  abs(U-5) >= abs(V);

initialstate()=[10,4,0,0];

setPoint()=5;
goal([U,V,GoalV,T])=Re <->
  Re:=GoalV;

st([U,V,GoalV,T]) <->
  abs(T-5)<0.01;

The lqp.set is an exact description of the user model in elementary set theory. If we understand that <->, -> and := mean iff, imply and assignment, respectively, the meaning of the model should be clear. ".func" is a meta-command to the extProlog interpreter to indicate that the predicates listed in the arguments are used as functions.

In the above implementation (u0,v0)=(10,4), and Q and r of the state evaluation ((u,v)-(s,0))7Q((u,v)-(s,0))ra2=q1(u-s)2+q3v2+ra2 are specified as:
q1=30, q2=1 and r=0.01.

Figure 6 Trajectory of the unstable focus
"lqp.set" is translated into an executable system by setcompiler. Upon translation the standardized goal seeker is attached to the system so that it can work as extSLV. If the system is executed by the extProlog interpreter, we have a solution. Since discussion of setcompiler is too technical, its presentation is omitted in this paper [13].

Figure 6 shows the free trajectory of \((u,v)\) of the unstable focus when the goal seeker is inactivated \((\alpha=0)\). \((u_0,v_0)=(10,10)\) is assumed.

The figure shows that the trajectory diverges drawing a circle around \((0,0)\). The goal seeker is expected to suppress the unstable behavior and steers the state to the set point \((s,0)\).

---

Figure 7 Managed behaviors of Fig. 6 with respect to initial states

Figure 8 Behavior of \((u,v,a)\) of case \((u_0,v_0)=(10,4)\) in Fig. 7
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

Figure 7 shows stabilized and controlled (u,v)-behaviors of extSLV, where a=solution is hidden.

The trajectories derived from initial states (10,4), (0,4), (0,-4) and (10,-4) display non-linear but stable behaviors. The unstable process of Figure 6 is well controlled by the goal seeker. As such, the desired set point=(5,0) is reached. It should be noted that the constraint |u-s|\geq|v| is naturally satisfied for every trajectory due to the back tracking function. This function most clearly distinguishes extSLV from conventional solvers. If the goal seeker were a PID (proportional-integral-derivative) linear controller, the constraint could not be satisfied.

Figure 8 shows the dynamic behavior of (u,v,a) where (u_0,v_0)=(10,4). The curve whose value shows abrupt change corresponds to a, control input. This is a solution produced by extSLV for the LQP. The curve which approach to 0 corresponds to v, velocity, while the smooth curve corresponds to u, displacement which shows that the behavior converges c=(5,0). The highly non-linear movement of a comes from the constraint |u-s|\geq|v|. If the constraint is ignored, a displays a regular (optimum) bang-bang control behavior.

So far behaviors of extSLV have been examined for one specific goal. In order to check that extSLV can work for an arbitrary goal, the behavior of extSLV is examined changing the value of r with respect to a fixed Q. Figure 9(a) shows behaviors of (u,v) of extSLV for r=0.001, 0.01 and 0.1 where A=[-80,-40,-20,-10,-5,-3,-1,0,1,3,5,10,20,40,80], t_f=5, h=0.1 and (u_0,v_0)=(10,4) are fixed.

Figure 9(a) shows the behaviors are completely controlled by the back tracking function of the goal seeker. The result shows that if r=0.001, the final state is (u,v)\approx(5.5,0.3), where the goal value of (2) is g_f=547;
if r=0.01, the final state is (u,v)\approx(5.5,0.3), where the goal value of (2) is g_f=560;
if r=0.1, the final state is (u,v)\approx(5.5,-0.5), where the goal value of (2) is g_f=835.

In order to check effect of the back tracking function, the case of no constraint is examined. The

![Figure 9(a) Relationship between trajectory of (u,v) and change of goal function with constraint](image-url)
user model is controlled by only local optimization. Figure 9(b) shows the result.

The result shows that if \( r=0.001 \), the final state is \((u,v) = (5.0,0.1)\), where the goal value of (2) is \( g_f=176 \) and the set point is reached; if \( r=0.01 \), which is the case of Figures 7 and 8, the final state is \((u,v) = (4.7,0.0)\), where the goal value of (2) is \( g_f=210 \); if \( r=0.1 \), the user model cannot be controlled by the goal seeker.

The result is compatible with the expectation that if the evaluation is market oriented \((r\text{ is small})\), \((u,v)\) behaves as a stabilized system and reaches a desired state quickly because the activity of the goal seeker is not restricted if \( r \) is small. On the other hand, if the evaluation is production oriented \((r=0.1 \text{ in Figure 9(b)})\), \((u,v)\) display an unstable behavior. extSLV can avoid this failure by using back tracking in Figure 9(a).

The first two cases with the constraint are worse than the corresponding cases without the constraint. This indicates that better paths of \((u,v)\) exists outside the constraint region. This result suggests that we can design an adaptive constraint, no constraint for a small \( r \) and constraint for a large \( r \).

5. Conclusion

This paper investigates generality of extSLV using a second order LQP which is characteristic of an infinite state set, a numerical and analytical model and a continuous action set. These characters are completely different from those of an MIS problem which is the principal target of extSLV. The result shows generality of extSLV is confirmed. That is, extSLV produces a solution which steers a given initial state to a set point even in an infinite state set. Furthermore, extSLV shows interesting behaviors.

(1) extSLV is able to derive a solution to a problem, which is difficult to be attacked by conventional optimization techniques. (See Figures 7 and 8)

(2) A problem, which cannot be solved without imposing a constraint on behavior (see Figure 9(b)), can be made a feasible problem introducing a proper constraint (see Figure 9(a)). This fact relies on the back tracking function.
Application of extSLV (extended Solver) to A Continuous Non–combinatorial Problem

The second fact is a little bit surprising because a constraint condition is usually considered as a negative factor in the conventional optimization theory. In some cases a constraint must be positively introduced for extSLV.

Appendix IV discusses comparison between an optimum solution and a solution by extSLV for the case that the target problem has a well known analytical solution. The solution by extSLV qualitatively imitates the expected optimum solution. It is compatible with the observation that extSLV yields a reasonable solution for a non-numerical problem such as a traveling salesman problem.

References
Appendix I Goal Seeker with Hill-climbing Method

Figure 10 shows the structure of extSLV which uses the goal seeker with the hill-climbing method.

extSLV consists of a user model of extSLV and a goal-seeker. The user model is an automaton model of a problem solving activity. That is, a problem is a gap between a current state and a desired state and a problem solving process is an activity to transfer the current state to the desired one. The user model has an input $a \in A$ which is a decision action variable for the user model. The output of the user model is a state $c \in C$ indicating the status of the problem solving activity. Let the automaton model be $\delta : C \times A \rightarrow C$.

The goal seeker specifies a value for the control parameter $a \in A$ of the user model to achieve a given goal using the state information $c \in C$. Let the goal be $\text{goal} : C \rightarrow \mathbb{R}$ where $\mathbb{R}$ is the set of reals. Although the goal may or may not be explicitly given by the PSE, the goal-seeker always requires the existence of a goal in extSLV. If the goal is not explicitly given, it can be used as a design parameter. The goal-seeker with the hill-climbing method is designed to yield an optimum $a \in A$ with respect to $\text{goal}(\delta(c,a))$ for a given $c \in C$. The optimality means, in particular, minimization in extSLV.

The push-down stack is used for the back tracking operation. The output of the goal-seeker is the (primary) solution $y \in Y$.

Figure 11 shows the total process of the solving activity of the goal-seeker.

All necessary information to yield an action is stored in the push-down stack. The goal-seeker, first, tries to pop up the top information. If the trial fails, in general, it indicates failure of the goal-seeker and the operation of the goal-seeker must finish. If the trial is successful, the goal-seeker gets $(c, A, S, H, a)$, current extSLV
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

state (c), set of available actions (As) = genA(c), the history of the visited states (H), and the history of selected actions (α) as the top information of the stack. genA:C→ϕ(A) is a function which specifies the set of available actions for a given state. It then selects (a*,RA) where a* is assumed to satisfy the relation

$$\min\{\text{goal}(\delta(c,a))|a\in As\} = \text{goal}(\delta(c,a*))$$

and

$$RA = As\{-a^*\}.$$

\(\delta(c,a^*)\) must not be equal to an element of H. If c is a deadlocked state, \(a^* = []\) is assumed. In this case the goal-seeker must do the backtrack operation, that is, it pops up the next information from the stack. If \(a^*\) is a regular action (\(a^* \neq []\)), the goal-seeker applies \(a^*\) to the process \(\delta\) to get the next state \(cc = \delta(c,a^*)\). It generates the set of available actions \(As2 = \text{genA}(cc)\) for the next state cc. The goal-seeker saves (push down) two pieces of information, (c,RA,H,α) and (cc,As2,H2,α2)

---

**Figure 11** Process of goal seeking activity
where $H_2 = H_c$ and $\alpha_2 = \alpha \cdot a^*$, i.e., the current state $c$ and the action $a^*$ are appended to $H$ and $\alpha$, respectively. The former information $(c, RA, H, \alpha)$ will be used when the goal-seeker backtracks to the state $c$. The latter information is used at the next state $c'$. Then, the stopping condition $st(t)$ is checked. If $st(cc) = true$, the total process must finish. If $st(cc) \neq true$, it goes to the pop up stage.

**Appendix II Classification of Stability**

Table 1 shows the classification of stability. Each slot illustrates a typical case of $(\alpha, \beta)$ pair and its stability mode name [10].

The dynamic system of Figure 4 belongs to the case of center. In general dynamic modes of a dynamical system are classified by the mode names of Table 1.

**Appendix III Analytical Solution for Dynamic Optimization Problem**

Let us apply the maximum principle to the optimization problem [12]. According to it, the Hamiltonian $H$ is given as below.

$$H = (((u,v)-(s,0))^\top Q((u,v)-(s,0)) + \alpha^2) + p_1 v + p_2(\beta u - 2\alpha v + a)$$

where $\mathbf{p} = (p_1, p_2)$ is the co-state vector. $H$ must be minimized with respect to $\alpha$. If every constraint is ignored, $\alpha = -\frac{p_2}{2r}$ produces the minimum. Then, if all constraints are ignored again, $(u, v)$ and $(p_1, p_2)$ are given as a solution of the following two-point boundary value differential equation.

$$u' = v, \
v' = -\beta u - 2\alpha v - \frac{p_2}{2r}, \
p_1' = -2q_1(u-s) + \beta p_2, \
p_2' = -2q_2v - p_1 + 2\alpha p_2$$

where

$$u(0) = u_0, v(0) = v_0, p_1(t_f) = p_2(t_f) = 0.$$

The following are the obvious difficulties about the analytical procedure.

<table>
<thead>
<tr>
<th>$\alpha &gt; 0$</th>
<th>$\alpha = 0$</th>
<th>$\alpha &lt; 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha^2 \cdot \beta &lt; 0$</td>
<td>I $(\alpha, \beta) = (1, 2)$ stable focus</td>
<td>II $(\alpha, \beta) = (0, 2)$ center</td>
</tr>
<tr>
<td>$\alpha^2 \cdot \beta = 0$</td>
<td>IV $(\alpha, \beta) = (1, 1)$ stable node</td>
<td>V $(\alpha, \beta) = (0, 0)$</td>
</tr>
<tr>
<td>$\alpha^2 \cdot \beta &gt; 0, \beta &gt; 0$</td>
<td>VII $(\alpha, \beta) = (1, 0.5)$ stable node</td>
<td>none</td>
</tr>
<tr>
<td>$\alpha^2 \cdot \beta &gt; 0, \beta = 0$</td>
<td>IX $(\alpha, \beta) = (1, 0)$ none</td>
<td>X $(\alpha, \beta) = (-1, 0)$</td>
</tr>
<tr>
<td>$\alpha^2 \cdot \beta &gt; 0, \beta &lt; 0$</td>
<td>XI $(\alpha, \beta) = (1, -0.5)$ saddle point</td>
<td>XII $(\alpha, \beta) = (0, -0.5)$ saddle point</td>
</tr>
</tbody>
</table>

Table 1 Stability modes
Application of extSLV (extended Solver) to A Continuous Non-combinatorial Problem

1. If there are constraints, it is not easy to optimize $H$ with respect to $a$.
2. In general, it is not easy to solve a two-point boundary value problem. If there is a constraint for the state, the situation becomes worse. Regarding the above we can solve the two-point boundary value problem using linearity. This paper skips solving it because it requires technical arguments.

Appendix IV Comparison between an optimum solution and a solution by extSLV

A solution derived from extSLV is a local optimum and its relationship to the global optimum depends on the target problem. Let us consider the following optimization problem which has a well known theoretical solution.

Let $u'=v$ and $v'=a$, $|a|\leq 50$, find the control law to bring the system from $(u_0,v_0)$ to $(s,0)$ in the shortest time where $s=4.5$.

This optimization problem is solved with respect to the following performance function.

$$\text{goal} = \int_0^T dt$$

where $(u(T),v(T))=(s,0)$. The solution is the so-called bang-bang control \[12\]. Unfortunately, the PD method cannot use the above performance function as a goal because the function is independent of the local control $a$. However, if $\alpha=0$ and $\beta=0$, and $\text{goal}=(u,v)-(s,0)^TQ(u,v)-(s,0)+ra^2$, where $Q$ and $r$ are design variables, the model theory approach can be considered to present an approximate solution of the above optimization problem. Figure 12 shows both trajectories $(u,v)$ of the optimum solution and extSLV solution, where $\text{goal}=(30*(u-s)^2+v^2)+0.01a^2$.

The dotted curve A-G-E-D-B shows the switching function for the bang-bang control \[12\]. If an initial state lies in the left side of the curve, the optimum action is $a(t)=50$ otherwise $a(t)=-50$. Consequently, if an initial state is $C$, the optimum trajectory is the curve C-D'-E. According to extSLV,
the corresponding trajectory is the curve C-D'-E where the switching function is A'-E-B'. The switching curve is determined by the minimization of \( \text{goal} = (30^*(u-s)^2+v^2)+0.01a^2 \). Although the trajectory C-D'-E seems to yield better performance than the trajectory C-D'-D-E, the optimum trajectory is given by the latter. (The optimum is 6 steps while extSLV requires about 10 steps). Figure 12 shows, however, that extSLV qualitatively imitates the optimum behavior.

### Appendix V Classification of Problems

extSLV uses the following three dimensions for the categorization of problems:

1. Explicit solving action — implicit solving action (E/I)
2. Closed goal — open goal (C/O)
3. Closed target — open target (C/O)

The following table Table 2 shows the classification of problems with typical examples. The classification is used to derive a user model[3,6,8].

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-C-C problem</td>
<td>traveling salesman problem</td>
</tr>
<tr>
<td>E-C-O problem</td>
<td>(linear quadratic dynamic optimization problem)</td>
</tr>
<tr>
<td>E-O-C problem</td>
<td>machine scheduling problem (regulation problem)</td>
</tr>
<tr>
<td>E-O-O problem</td>
<td>(linear quadratic dynamic optimization problem)</td>
</tr>
<tr>
<td>I-C-C problem</td>
<td>cubic root problem</td>
</tr>
<tr>
<td>I-C-O problem</td>
<td>knapsack problem</td>
</tr>
<tr>
<td>I-O-C problem</td>
<td>class scheduling problem</td>
</tr>
<tr>
<td>I-O-O problem</td>
<td>data mining problem</td>
</tr>
</tbody>
</table>

Table 2 Classification of problems and examples