

**ALTERNATIVE ROLLOUT OPTIMIZATION TO NONLINEAR MODEL PREDICTIVE
CONTROL IN L-LYSINE PRODUCTION FOR CHOLESTEROL REDUCTION**

Tatiana S. Ilkova and Mitko M. Petrov

Institute of Biophysics and Biomedical Engineering, Bulgarian Academy of Sciences,
105 Acad. George Bonchev Str., Sofia 1113, Bulgaria
E-mail: tanja@clbme.bas.bg, mpetrov@clbme.bas.bg

Abstract

L-lysine is an essential amino acid, which means that it is essential to human health but cannot be manufactured by the body. For this reason, L-lysine must be obtained from food. Amino acids are the building blocks of protein. Lysine is important for proper growth and it plays an essential role in the production of carnitine, a nutrient responsible for converting fatty acids into energy and helping to lower cholesterol.

NMPC is developed for guarantee robustness to process disturbances for a fed-batch fermentation process for the L-lysine. The method is carried out with an aim control of disturbance of the optimal control variable (feeding rate). For local optimization of choice optimization hour, an algorithm is applied in order to find an optimal profile of the control variable.

For local optimization consider the approximate solution of discrete optimization problems using procedures that are capable of magnifying the effectiveness of any given heuristic algorithm through sequential application a alternative rollout algorithms, which are related to notions of policy iteration.

The developed control algorithm ensures maximal L-lysine at the end of the process and guarantees a feedback on disturbance as well as robustness to process disturbances.

Key words: *alternative rollout optimization, cholesterol reduction, control algorithm, l-lysine, nonlinear model predictive control*

1. INTRODUCTION

In humans, carnitine deficiencies result in the buildup of acid metabolites in tissues and increased levels of triglycerides, a type of fat in the blood. Carnitine deficiencies also lead to poor energy production in the heart and poor metabolism of fats. Dr. Eman Cameron has recently suggested that carnitine deficiency may also account for the muscle wasting, weakness, and poor sense of well-being in a cancer patient. Supplementation with *lysine* and vitamin C may help correct a low carnitine level. The role of carnitine in fat metabolism is to help transport free fatty acids into the mitochondria of cells where the fats can be used as a source of energy. Carnitine has also been found to be important in the synthesis and function of sperm in the testes. In male animals, carnitine deficiency can result in infertility (Hulse et al., 1978, <http://www.diagnose-me.com/treat/T110125.html>).

As an amino acid, it is found in a host of foods and is not normally deficient in the diet. Many foods supply *lysine*, but the richest sources by far include red meats, fish, and dairy products (milk, eggs, cheese). Vegetables, on the other hand, are generally a poor source of *lysine*, with the exception of legumes (beans, peas, lentils).

Of all the amino acids, *lysine* is the most sensitive to the effects of food processing, such as dry heat. The amount of protein available in legumes and other sources of *lysine* can be significantly reduced if they have been toasted or roasted.

While many people satisfy their need for *lysine* through dietary sources, supplements are now popular for treating and preventing specific ailments as well. Supplements come in various forms: tablet, powder, liquid, and capsule (<http://www.diagnose-me.com/treat/T110125.html>).

Like all amino acids, *lysine* functions as a building block for proteins. It is also a key player in the production of various enzymes, hormones, and disease-fighting antibodies. It is necessary for proper growth and helps form collagen which comprises bone cartilage and connective tissues.

Researchers are exploring the value of *lysine* supplementation and the consumption of *lysine*-rich foods for lowering cholesterol, improving athletic performance, and enhancing recovery after surgery. *Lysine* aids in the manufacture of Carnitine, which is essential in the breakdown of fats into energy. *L-Lysine* was recommended to us as a good alternative to drugs, *L-Lysine* breaks down fatty deposits and seems to clean your blood of the bad fats which in our experience drastically reduces Cholesterol (<http://www.gustrength.com/health:the-role-of-carnitine-in-disease>).

Because *Lysine* helps repair tissue, it is a good supplement for anyone recovering from surgery and injuries (http://www.springboard4health.com/notebook/proteins_carnitine.html).

Lysine is helpful in lowering triglycerides and as an anti-aging factor. It is involved in the structural repair of damaged blood vessels.

Recently it has been shown that *L-Lysine* can inhibit the growth of the herpes virus. Herpes virus requires many proteins with the amino acid arginine, and *lysine* competes directly with arginine in many of these processes. This competition is thought to slow down the growth of the herpes virus. While high doses (500-1500 mg/day) are beneficial during the suppression of viral growth, lesser amounts should be taken, if taking on an on going basis to prevent an amino acid imbalance problem (<http://www.diagnose-me.com/treat/T110125.html>).

Model predictive control (*MPC*) is a general methodology for solving control problems in the time domain. More than 25 years after *MPC* appeared, a theoretical basis for this technique has started to emerge in the industry as an effective means to deal with multivariable constrained control problems. The outcomes of the feasibility of the *on-line* optimization, stability and performance are largely understood as systems described by linear model. In fact, that method for optimal control gives the necessary optimal profile, but it does not give the robustness of the optimization systems. Therefore the *MPC*, especially Nonlinear model predictive control (*NMPC*) can be used for ensuring maximal quality concentration at the end of the process and it guarantees a feedback on disturbance and thus – the robustness to process disturbances (Findeisen et al., 2003).

The *rollout algorithm* is a suboptimal control method for deterministic and stochastic problems that can be solved by a dynamic programming. In this short note when an extension is derived from an extension of the *rollout algorithm* it applies to constrained deterministic dynamic programming problems and relies on a suboptimal policy called base heuristic. Under suitable assumptions it is not shown if the base heuristic produces a feasible solution. In this paper we applied *alternative rollout algorithm* that will also produce a feasible solution, whose cost is not less than the cost corresponding to the base heuristic (Wu et al., 2003).

In this paper we apply the ideas of *Alternative rollout optimization to NMPC* to *L-lysine* production for Cholesterol Reduction in Human Body with aim ensuring maximal productivity and guaranteeing robustness to disturbances.

2. MATERIALS AND METHODS

The conversion of *lysine* to carnitine in all cells of the body is dependent upon adequate vitamin C levels. It has been estimated that about 0.1 percent of the dietary *lysine* is converted to carnitine in the body. Carnitine is not found in a vegetable-based diet; it is only found in animal products. Therefore, vegetarians who may consume a *lysine*-deprived diet may have insufficient amounts of carnitine. Animals fed a wheat gluten diet low in *lysine* and carnitine have significantly lower levels of carnitine in their hearts and skeletal muscles. The pathway in mammals is unique using protein-bound *lysine* that is *enzymatically methylated* to form *trimethyllysine* as a post-translational modification of protein synthesis. *Trimethyllysine* undergoes four enzymatic reactions in the course of endogenous *L-carnitine* biosynthesis (**Fig. 1**) (<http://www.gustrength.com/health:the-role-of-carnitine-in-disease>).

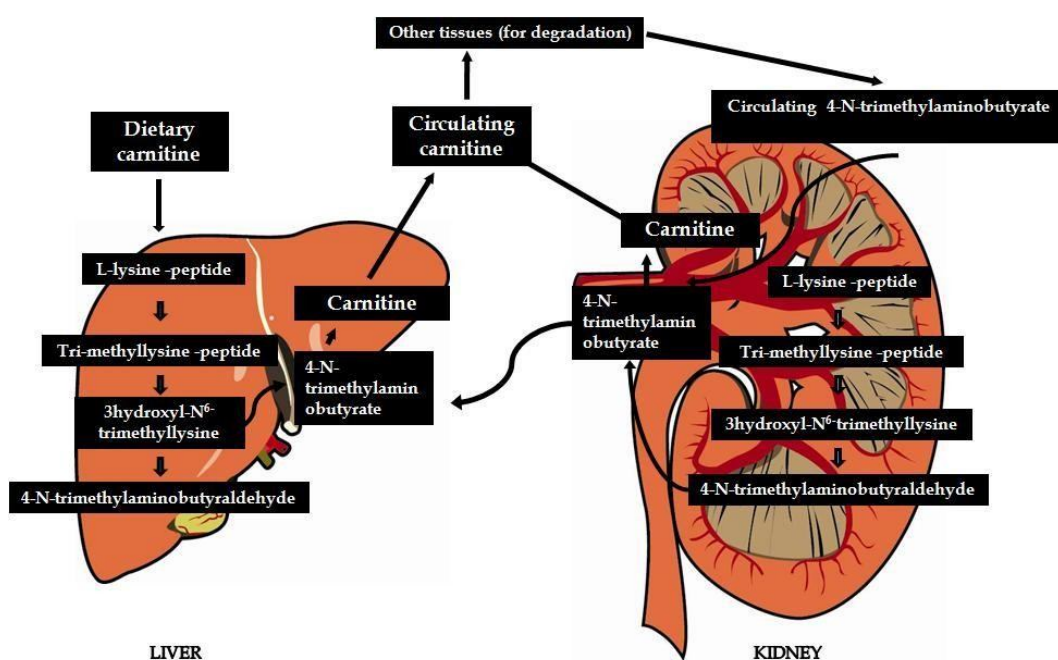


Fig. 1. Carnitine biosynthesis and metabolism

Defined media acquiring pure growth requiring nutrients and essential additives or alternatively undefined media containing natural organic substances such as soybeanhydrolyzate, corn steep liquor, yeast extract or peptone are used for *L-lysine* fermentation. Common fermentation media for *L-lysine* production contain various carbon and nitrogen sources, inorganic ions and trace elements (Fe^{++} , Mn^{++}), amino acids, vitamins (biotin, thiamine-HCl, Nicothin amide) and numerous complex organic compounds. An over expression of genes is also achieved by optimizing the composition of the media and the culture technique in addition to physiological and genetic parameters (Anastassiadis, 2007).

The mathematical model of the process is based of the mass balance equations, as perfect mixing in the bioreactor is accepted and the process model has the type (Anastassiadis, 2007):

$$\frac{dX}{dt} = \mu X - \frac{F}{V} X \quad (1)$$

$$\frac{dS}{dt} = \frac{F}{V} (S_{in} - S) - k_5 \mu X - k_6 X - k_7 \eta X \quad (2)$$

$$\frac{dTr}{dt} = \frac{F}{V} (Tr_{in} - Tr) - k_{13} \mu X - \frac{F}{V} Tr \quad (3)$$

$$\frac{dC_L}{dt} = k_1 a (C^* - C_L) - k_{14} \mu X - k_{15} X - k_{16} \eta X - \frac{F}{V} C_L \quad (4)$$

$$\frac{dL}{dt} = \eta X \quad (5)$$

$$\frac{dV}{dt} = F \quad (6)$$

The specific rate of *L-lysine* synthesis and specific consumption rate have the following form:

$$\mu = \frac{k_1 Tr C_L}{[(k_2 + Tr)(k_3 + S_0 - S)(k_4 + C_L)]}, \quad \eta = \frac{k_8 S C_L}{[(k_9 + S)(k_{10} + S)(k_{11} + C_L)(k_{12} + C_L)]} \quad (7)$$

where: μ – specific rate of *L-lysine* synthesis, h^{-1} ; η – specific consumption rate of *L-lysine*, h^{-1} ; X – biomass concentration, $g\ l^{-1}$; L – *L-lysine* concentration, $g\ l^{-1}$; S – glucose concentration, $g\ l^{-1}$; V – working liquid volume, l ; F – feed flow rate, $l\ h^{-1}$; Tr – Threonine concentration, $mg\ l^{-1}$; t – process time, h ; C_L – dissolved oxygen concentration, $g\ l^{-1}$; C^* – equilibrium dissolved oxygen concentration, $g\ l^{-1}$; S_{in} – input feed substrate concentration, $g\ l^{-1}$; Tr_{in} – input feed Threonine concentration, $g\ l^{-1}$; $k_1 a$ – volumetric liquid mass transfer coefficient, h^{-1} .

The initial conditions in the model (1) – (8) have the follows values:

$$X(0) = X_0 = 3.00\ g\ l^{-1}, S(0) = S_0 = S_i = 100.00\ g\ l^{-1}, Tr(0) = Tr_0 = Tr_{in} = 100.00\ mg\ l^{-1}, \\ L(0) = 0.00\ g\ l^{-1}, C_L(0) = C^* = C_0 = 6.1 \times 10^{-3}\ g\ l^{-1}, V(0) = V_0 = 10.00\ l$$

The model coefficients in (1) have the following values (Anastassiadis, 2007):

$$k_1 = 20.8 \quad k_2 = 42.0 \quad k_3 = 28.0 \quad k_4 = 1.1 \quad k_5 = 1.01 \quad k_6 = 0.07 \quad k_7 = 0.51 \quad k_8 = 62.0 \quad k_9 = 28.0 \\ k_{10} = 37.0 \quad k_{11} = 4.0 \quad k_{12} = 0.12 \quad k_{13} = 6.10 \quad k_{14} = 448.0, \quad k_{15} = 22.0 \quad k_{16} = 209.0 \quad k_1 a = 120$$

3. ALTERNATIVE ROLLOUT OPTIMIZATION METHOD

In revolution papers we discuss the approximate solution of broad classes of combinatorial optimization problems by embedding them within a Dynamic programming framework (*DP* for short). The key idea is to employ a given heuristic in the construction of an optimal cost-to-go function approximation, which is then used in the spirit of the *Neuro-Dynamic Programming/Reinforcement Learning* methodology (*NDP* for short) (Bertsekas at al., 1999) for broad discussions of this methodology). In the next section, we will introduce a general graph search problem that will serve as the context of our methodology. To illustrate the ideas involved, however, let us consider the following type of problem. which includes as special cases problems such as shortest path: assignment, scheduling, matching, etc.

When deterministic optimal control problem is considered the following system is included (Bertsekas at al., 1999):

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, \dots, N-1$$

where: x_k and u_k are the state and control at time k , respectively taking values in some sets which may depend on k and f_k is a function. The initial state is given and it is denoted by x_0 . Each control u_k must be chosen from a finite constraint set $U_k(x_k)$ that depends on the current state x_k . A sequence of the form:

$$T = (x_0, u_0, x_1, u_1, \dots, u_{N-1}, x_N)$$

where $x_{k+1} = f_k(x_k, u_k)$, $u_k \in U_k(x_k)$, $k = 0, \dots, N-1$

is referred to as a *trajectory*. A trajectory is complete in the sense that it starts at the given initial state x_0 , and ends at a state x_N after N stages. Partial trajectories are also referred to which are subsets of complete trajectories, involving fewer than N stages and consisting of stage-contiguous states and controls.

The cost of the trajectory:

$$T = (x_0, u_0, x_1, u_1, \dots, u_{N-1}, x_N)$$

$$\text{is: } V(t) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \quad (8)$$

where g_k , $k = 0, 1, \dots, N$ are given functions. The problem is to find a trajectory T that minimizes $V(T)$ subject to the constraint:

$$T \in C \quad (9)$$

where C is a given set of trajectories.

An optimal solution of this problem can be found by an extension of the dynamic programming algorithm, but the associated computation can be overwhelming. It is much greater than the computation for the corresponding unconstrained problem where the constraint $T \in C$ is absent. This is true even in the special case when C is specified in terms of a define number of constraint functions that is time-additive, i.e., $T \in C$ if:

$$g_N^m(x_N) + \sum_{k=0}^{N-1} g_k^m(x_k, u_k) \leq b^m, m = 1, \dots, M \quad (10)$$

where: g_k , $k = 0, 1, \dots, N$, and b^m , $m = 1, \dots, M$ are given functions and respectively scalars. The literature contains several proposals for suboptimal solution of the problem for the case where the constraints are in form (9). Most of these proposals are cast in the context of the constrained and shortest multi objective path problems (Bertsekas et al., 1999).

In this chapter a suboptimal approach is proposed which may be viewed as an extension of the *rollout algorithm*, a method that has been used with considerable success for general unconstrained DP problems (including stochastic and minimax). Note that it is possible to reformulate the constrained problem of this chapter into a format that is suitable for application of the *rollout algorithm* for the discrete deterministic problems given in Bertsekas et al., 1999. In particular the state at stage k may be redefined to be the partial trajectory:

$$y_k = (x_0, u_0, x_1, \dots, u_{k-1}, x_k),$$

which evolves according to:

$$y_{k+1} = y_k, u_k, f_k(x_k, u_k).$$

The problem then is how to find a control sequence that minimizes a suitable terminal cost function

$F(y_N)$ subject to the constraint $y_N \in C$. This is a format where the *rollout approach* in Wu et al., 2003 is applied and yields algorithms and results that are quite similar to the ones obtained in this chapter. However, this line of development which is conceptually valuable is complicated and unintuitive. It also does not lead to the discussion and development of variants of the main algorithm, therefore in this chapter a more direct extension is chosen for the *rollout approach* that is specially adapted to constrained problems.

The *rollout algorithm* has some additional special properties when it is applied to discrete unconstrained deterministic problems. The *rollout algorithm* makes use of a suboptimal algorithm called *base heuristic*. If the base heuristic is a *DP* policy, i.e. it generates state/control sequences via feedback functions μ_k , $k = 0, 1, \dots, N-1$ so that at any state x_k , it applies to the control $\mu_k(x_k)$, then the *rollout algorithm* can be viewed as a policy improvement step of the policy iteration method. Based on generic results for policy iteration this shows that the *rollout algorithm*'s cost is no worse than the one of the base heuristic. This cost improvement property is extended in Wu et al., 2003 to give base heuristics that are not *DP* policies, but rather satisfy an assumption called *sequential improvement* which will be discussed shortly within the constrained context of this chapter.

The extension of the *rollout algorithm* to discrete deterministic problems with the constraints (10) also uses a base heuristic which has the property to produce a partial trajectory at a given state x_k and at stage k :

$$H(x_k) = (x_k, u_k, x_{k+1}, u_{k+1}, \dots, u_{N-1}, x_N)$$

that starts at x_k and satisfies:

$$x_{i+1} = f_i(x_i, u_i), \quad u_i \in U_i(x_i), \quad i = k, \dots, N-1$$

The cost corresponding to the partial trajectory $H(x_k)$ is denoted by $\tilde{J}(x_k)$:

$$\tilde{J}(x_k) = g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, u_i)$$

Thus at a given partial trajectory that starts at the initial state x_0 and ends at a state x_k , the base heuristic can be used to complete this trajectory by concatenating it with the partial trajectory $H(x_k)$. The trajectory thus obtained is not guaranteed to be feasible (i.e. it may not belong to C), but it is assumed when the state/control trajectory generated by the base heuristic starting from the given initial state x_0 is feasible, i.e.:

$$H(x_0) \in C$$

The *rollout algorithm* is not much more complicated or computationally demanding than the one for the unconstrained *DP* problems (as long as checking feasibility of a trajectory T , i.e., $T \in C$ is not computationally demanding). *rollout algorithm* produces a feasible solution whose cost is no worse than the cost corresponding to the base heuristic. It does not appear possible to relate this *rollout algorithm* to concepts of policy iteration and in fact it is not awarded of an analogy of the policy iteration method for constrained *DP* problem even when the constraint set C is specified by unequal constraints with constraint functions that are additive over time. This *rollout algorithm* also makes essential use of the deterministic character of the problem and does not admit a straightforward extension to stochastic or minimize the problems.

The function $\tilde{J}$ will be called a *scoring function* or *approximate cost-to-go function*, and may contain some adjustable parameter vector that can be tuned using special training methods. In this paper, however, we restrict attention to scoring functions that are based on heuristic algorithms. In particular, we will assume that have a heuristic algorithm, which starting from an n -solution $(u_1, \dots, u_n)$, can

produce a complete N -solution $(u_1, \dots, u_N)$ whose cost is denoted by $H(u_1, \dots, u_n)$. One possibility, studied in this paper, is to approximate the optimal *cost-to-go* function with the scoring function:

$$\tilde{J}(u_1, \dots, u_n) = H(u_1, \dots, u_n)$$

A more general possibility is to use multiple heuristic algorithms, which are weighted with some scalar weights to provide the approximation $J(u_1, \dots, u_n)$. In this paper, we assume that the weights are fixed (although they could be adjusted through a separate trial-and-error process). In a more general *NDP* approach, the weights could be tunable parameters and could depend on some features of the given problem.

The *rollout algorithm* proposed in this chapter starts at stage 0 and sequentially proceeds to the last stage. At stage k it maintains a partial trajectory:

$$T_k = (x_0, \bar{u}_0, \bar{x}_1, \dots, \bar{u}_{k-1}, \bar{x}_k)$$

that starts at the given initial state x_0 , and it is such that

$$\bar{x}_{i+1} = f_i(\bar{x}_i, \bar{u}_i), \quad \bar{u}_i \in U_i(\bar{x}_i), \quad i = 0, 1, \dots, k-1$$

where $\bar{x}_0 = x_0$.

The algorithm starts with the partial trajectory T_0 that only consists of the initial state x_0 .

For each $k = 0, 1, \dots, N-1$ and given the current partial trajectory T_k , it forms a (complete) trajectory $T_k^c(u_k)$ for each control $u_k \in U_k(x_k)$, by concatenating:

1. The current partial trajectory T_k ;
2. The control u_k and the next state $x_{k+1} = f_k(x_k, u_k)$;
3. The partial trajectory $H(x_{k+1})$ generated by the base heuristic starting from x_{k+1} .

Then it forms the subset $\bar{U}_k(\bar{x}_k)$ of controls u_k for which $T_k^c(u_k)$ is feasible, i.e. $T_k^c(u_k) \in C$.

The algorithm then selects from $\bar{U}_k(\bar{x}_k)$ a control u_k that minimizes over $u_k \in \bar{U}_k(\bar{x}_k)$:

$$g(\bar{x}_k, u_k) + \tilde{J}(f_k(\bar{x}_k, u_k))$$

It then forms the partial trajectory T_{k+1} by adding $(\bar{u}_k, \bar{x}_{k+1})$ to T_k , where: $\bar{x}_{k+1} = f_k(\bar{x}_k, \bar{u}_k)$.

Let us use the union symbol $\cup$ for a more compact definition of the *rollout algorithm* to denote the concatenation of two or more partial trajectories. Having in mind this notation, we have the following:

$$T_k^c(u_k) = T_k \cup (u_k) \cup Rf_k(x_k, u_k) \quad (11)$$

$$\bar{U}_k(x_k) = \left\{ u_k \mid u_k \in U_k(\bar{x}_k), T_k^c(u_k) \in C \right\} \quad (12)$$

The *rollout algorithm* is selected at stage k

$$u_k \in \arg \min_{u_k \in U_k(x_k)} [g(\bar{x}_k, u_k) + \tilde{J}(f_k(\bar{x}_k, u_k))]$$

and it extends the current partial trajectory by setting

$$T_{k+1} = T_k \cup (\bar{u}_k, \bar{x}_{k+1}) \quad (13)$$

where:

$$\bar{x}_{k+1} = f_k(\bar{x}_k, \bar{u}_{k+1}) \quad (14)$$

It is worth noting that there is similarity between this *rollout algorithm* and the unconstrained version given in (Bertsekas et al., 1999): the constraint set C enters only in (14) and if $\bar{U}_k(\bar{x}_k)$ is replaced by $\bar{U}_k(\bar{x}_k)$ in (14) one obtains the *rollout algorithm* for unconstrained DP problems.

That it is possible when there is no control $u_k \in U_k(\bar{x}_k)$ that satisfies the constraint $T_k^c(u_k) \in C$ (i.e. the set $\bar{U}_k(\bar{x}_k)$ is empty) and in this case the algorithm breaks down.

In this paper we consider the approximate solution of discrete optimization problems using procedures that are capable of magnifying the effectiveness of any given heuristic algorithm through sequential application. In particular, we embed the problem within a dynamic programming framework, and we introduce several types of rollout algorithms, which are related to notions of policy iteration.

Alternative rollout algorithms: We now consider some generalizations of the results and algorithms discussed up.

Let us introduce the following definition:

Definition 1. Suppose that up algorithm Ψ generates, starting at each node, a path $(i, i_1, \dots, i_m, \bar{i})$ with the property:

$$H(i) \geq H(i_1) \quad (15)$$

Then the algorithm Ψ is said to be *sequentially improving*.

It can be seen that a sequentially consistent Ψ is also *sequentially improving*, we see that this proof carries through verbatim. We thus have the following generalization of **Proposition 1**:

Proposition 1. Let the algorithm Ψ be sequentially improving; and suppose that $R\Psi$ is terminating. Then; if $(i_1, \dots, i_m)$ is the path generated by $R\Psi$ starting from a nondestination node i_1 and ending at a destination node i_m ; we have:

$$H(i_m) = \min \{ H(i_1), \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j) \} \quad (16)$$

Example: Consider the one-dimensional walk problem of, and let Ψ be defined as the algorithm that, starting at a node (k, m) , compares the cost $g(m+N-k)$ (corresponding to taking all of the remaining $N-k$ steps to the right) and the cost $g(m-N+k)$ (corresponding to taking all of the remaining $N-k$ steps to the left), and accordingly moves to node:

$(N, m-N-k)$ if $(m+N-k) \leq g(m+N-k)$, or to node: $(N, m-N-k)$ if $(m+N-k) < g(m+N-k)$

It can be seen that Ψ is not sequentially consistent, but is instead sequentially improving. It is a fact that starting from the origin $(0, 0)$, $R\Psi$ obtains the global minimum of g in the interval $[N, N]$, while H obtains the better of the two points N and N .

We can always modify the problem and the algorithm Ψ so that **Proposition 1** applies. In particular, let us consider the *extended version of the problem*, whereby the graph (Ψ, Ψ) is enlarged by adding for each non-destination node i an arc $(i, d(i))$, where $d(i)$ is the destination at which the path generated by Ψ terminates, starting from i . (This arc is not added if it already exists.) Then Ψ is modified so that starting from each non-destination node i for which:

$$\min_{j \in N(i)} H(j) > H(i) = g(d(i))$$

it generates instead the path $(i; d(i))$. It is seen that the modified version of Ψ so obtained, referred to as the *extended* Ψ and denoted by Ψ_E , is sequentially improving. Thus, **Proposition 1** applies to the *rollout algorithm* based on the extended Ψ , which is referred to as the *alternative rollout algorithm* and is denoted by $R\Psi_E$. This algorithm proceeds exactly like $R\Psi$ up to the first node i for which (16) holds, and then terminates with the destination node $d(i)$.

We note that one may use multiple path construction algorithms in the preceding framework. In particular, let us assume that we have K algorithms $\Psi_1, \dots, \Psi_K$. The k^{th} of these algorithms, given a node $i \notin \Psi$, produces a path $(i, i_1, \dots, i_m, \bar{i})$ that ends at a destination node $\bar{i}$, and the corresponding cost is denoted by $H_k(i) = g(\bar{i})$. Generalizing our earlier approach, we can use the K algorithms in an approximation architecture of the form:

$$\tilde{J}(i) = \min_{k=1, \dots, K} H_k(i), \text{ or of the form: } \tilde{J}(i, r_1, \dots, r_K) = \sum_{k=1}^K r_k H_k(i)$$

where r_k are some fixed scalar weights obtained by trial and error. The *rollout algorithms* easily generalize for the case, by replacing $H(i)$ with $\tilde{J}(i)$, and by defining the path generated starting from a node i as the path generated by the path construction algorithm which attains the minimum. In the case, the *alternative rollout algorithm* $R\Psi$ also generalizes easily by replacing $H(i)$ with $\tilde{J}(i, r_1, \dots, r_K)$, but in order to generalize the algorithms $R\Psi$ and $R\Psi_E$, the path generated from a node i must also be defined. There are several possibilities along this line. A different type of possibility for the case, is to use tunable weights, which are obtained by training using *NDP* methodology.

Finally, let us consider a problem where in addition to the terminal cost $g(i)$, there is a cost $c(i, j)$ for a path to traverse an $(i_1, i_2, \dots, i_n)$. Within this context, the cost of a path $(i, i_1, \dots, i_m, \bar{i})$ that starts at i_1 and ends at a destination node i_n is redefined to be:

$$g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1})$$

One way to transform this problem into one involving a terminal cost only is to redefine the graph of the problem so that nodes correspond to sequences of nodes in the original problem graph. Thus if we have arrived at node i_k using path $(i_1, \dots, i_k)$, the choice of i_{k+1} as the next node is viewed as a transition from state $(i_1, \dots, i_k)$ to state $(i, i_1, \dots, i_m, \bar{i})$.

Both states $(i_1, \dots, i_k)$ and $(i, i_1, \dots, i_m, \bar{i})$ are viewed as nodes of a redefined graph.

Furthermore, in this redefined graph, a destination node has the form $(i_1, i_2, \dots, i_n)$, where i_n is a destination node of the original graph.

After the details are worked out, we see that to recover our earlier algorithms and analysis, we need to modify the cost of the heuristic algorithm Ψ as follows: If the path $(i_1, i_2, \dots, i_n)$ is generated by Ψ starting at i_1 , then:

$$H(i_1) = g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1})$$

Furthermore, the *rollout algorithm* $R\Psi$ at node i_m selects as next node i_{m+1} the node:

$$i_{m+1} = \arg \min_{j \in N(i_m)} [c(i_m, j) + H(j)]$$

The definition of a sequentially consistent algorithm remains unchanged.

Furthermore, **Proposition 1** after modification we can read:

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1}) = \min_{j \in N(i_k)} [c(i_k, j) + H(j)] \quad k = 1, \dots, m-1$$

A sequentially improving algorithm should now be characterized by the property:

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1})$$

if i_{k+1} is the next node on the path generated by Ψ starting from i_k . Furthermore, **Proposition 1** remains unchanged and is modified to read:

$$H(i_k) \geq \min_{j \in N(i_k)} [c(i_k, j) + H(j)] \quad k = 1, \dots, m-1$$

Finally, the criterion $\min_{j \in N(i_k)} H(j) < g(i'_k)$ used in the fortified *rollout algorithm*, given the sequence

$(s, i_1, \dots, i_m)$, where $i_m \notin \Psi$, and the path $P(i_m) = (i_m, i'_m, i'_k)$ should be replaced by:

$$\min_{j \in N(i_m)} [c(i_m, j) + H(j)] < g(i'_k) + c(i_m, i'_{m+1}) + \sum_{l=m+1}^{k-1} c(i'_l, i'_{l+1})$$

Similar to the construction used in the preceding section, we can transform the graph search problem into a *DP* problem. In particular, the nodes correspond to the states of the *DP* problem, the controls available at a given state/node and the corresponding successor states/nodes are the outgoing arcs from the node and the associated end nodes of the arcs, respectively. The destination nodes i are terminal states of the *DP* problem, where the terminal cost $g(i)$ is incurred.

Optimized rollout algorithm. If Ψ is not sequentially improving, it is possible in general that $\mathcal{R}\Psi$ generates a worse solution than the solutions generated by Ψ from the same starting node. However, it is always possible to correct this deficiency by a minor modification $\mathcal{R}\Psi$. In particular, in the process of running $\mathcal{R}\Psi$, one generates several solutions, and upon termination of $\mathcal{R}\Psi$, one can choose out of all these solutions, one that has minimal cost. This version of the *rollout algorithm*, is referred to as the *optimized rollout algorithm* and is denoted by $\mathcal{R}^*\Psi$. Note that if Ψ is terminating, then $\mathcal{R}^*\Psi$ is guaranteed to generate a no worse solution than all of the algorithms Ψ , $\mathcal{R}\Psi$ and $\mathcal{R}\Psi_e$.

Note that while the algorithm $\mathcal{R}\Psi$ will generally yield a suboptimal solution, the path that it constructs may involve a fairly sophisticated suboptimization. For example, Ψ may construct several paths ending at destination nodes according to some heuristics, and then select the path that yields minimal cost.

One possibility for suboptimal solution of the problem is to start at the origin s and use the algorithm Ψ to obtain a solution of cost $H(s)$. We instead propose to use Ψ to construct a path to a destination node sequentially. At the typical step of the sequence, we consider all downstream neighbors j of a node i , we run Ψ starting from each of these neighbors, and we then move to the neighbor from which Ψ gives the best result. The idea of starting with some algorithm, and using it to construct another, hopefully improved, algorithm is implicit in the policy iteration method of *DP* Ψ the use of a *rollout policy*, which is a form of policy iteration.

We now consider some generalizations of the results and algorithms discussed so far.

Let us introduce the following definition:

Definition 2: Suppose that algorithm H generates, starting at each node $i \in \bar{N}$, a path $(i, i_1, \dots, i_m, \bar{i})$ with the property:

$$H(i) \geq H(i_1) \tag{17}$$

Then the algorithm Ψ is said to be sequentially improving.

It can be seen that a sequentially consistent H also sequentially improving, with the above mentioned equation. If it will use, it can be seen have the following generalization:

Proposition 2: Let the algorithm Ψ be sequentially improving, and suppose that $\mathcal{R}\Psi$ is terminating. Then, if $(i, i_1, \dots, i_m)$ is the path generated by $\mathcal{R}\Psi$ starting from a nondestination node i and ending at a destination node i_m , then:

$$H(i_m) = \min \left\{ H(i_1, \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j)) \right\} \quad (18)$$

Example: Consider the onedimensional walk problem, and let H be defined as the algorithm that, starting at a node (k, m) , compares the cost $g(m + N - k)$ (corresponding to taking all of the remaining $N - k$ steps to the right) and the cost $g(m - N + k)$ (corresponding to taking all of the remaining $N - k$ steps to the left), and accordingly moves to node:

$$(N, m + N - k) \quad \text{if} \quad g(m + N - k) \leq g(m - N + k), \text{ or to node:} \\ (N, m - N + k) \quad \text{if} \quad g(m - N + k) < g(m + N - k)$$

It can be seen that H is not sequentially consistent, but is instead sequentially improving. It can be followed that starting from the origin $(0, 0)$, $\mathcal{R}\Psi$ obtains the global minimum of g in the interval $[-N, N]$, while H obtains the better of the two points $-N$ and N .

We can modify the problem and the algorithm H . In particular, let us consider the extended version of the problem, whereby the graph (N, A) is enlarged by adding for each nondestination node i an arc $(i, d(i))$, where $d(i)$ is the destination at which the path generated by Ψ terminates, starting from i . (This arc is not added if it already exists.). Then Ψ is modified so that starting from each nondestination node i for which:

$$\min_{j \in N(i)} N(j) > H(i) = g(d(i))$$

it generates instead the path $(i, d(i))$. It is seen that the modified version of Ψ so obtained, referred to as the extended Ψ and denoted by Ψ_e , is sequentially improving. Thus, the rollout algorithm based on the extended Ψ , which is referred to as the extended rollout algorithm and is denoted by $\mathcal{R}\Psi_e$. This algorithm proceeds exactly like $\mathcal{R}\Psi_e$ up to the first node i holds, and then terminates with the destination node $d(i)$.

If Ψ is not sequentially improving, it is possible in general that $\mathcal{R}\Psi$ generates a worse solution than the solutions generated by H from the same starting node. However, it is always possible to correct this deficiency by a minor modification $\mathcal{R}\Psi$. In particular, in the process of running $\mathcal{R}\Psi$, one generates several solutions, and upon termination of $\mathcal{R}\Psi$, one can choose out of all these solutions, one that has minimal cost. This version of the rollout algorithm, is referred to as the optimized rollout algorithm and is denoted by $\mathcal{R}^*\Psi$. Note that if Ψ is terminating, then $\mathcal{R}^*\Psi$ is guaranteed to generate a no worse solution than all of the algorithms Ψ , $\mathcal{R}\Psi$ and $\mathcal{R}\Psi_e$.

Let us introduce an alternative sequential version of Ψ . This version is referred to as the fortified rollout algorithm and is denoted by $\mathcal{R}\Psi$. As the notation suggests, $\mathcal{R}\Psi$ turns out to be the rollout algorithm based on a path construction algorithm $\bar{H}$, which is derived from H and will be defined shortly. The fortified rollout algorithm starts at s , and maintains, in addition to the current sequence of nodes $(s, i, i_1, \dots, i_m)$, a path:

$$P(i_m) = (i_m, i'_{m+1}, \dots, i'_k) \quad (19)$$

ending at a destination i'_k . Initially, $P(s)$ is the path generated by Ψ starting from s . At the typical step of $\mathcal{R}\bar{\Psi}$, we have a node sequence $(s, i, i_1, \dots, i_m)$, where, and the path $P(i_m) = (i_m, i'_{m+1}, \dots, i'_k)$. Then, if:

$$\min_{j \in N(i_m)} H(j) < g(i'_k) \quad (20)$$

$\mathcal{R}\bar{\Psi}$ adds to the node sequence $(s, i, i_1, \dots, i_m)$ the node:

$$i_{m+1} < \arg \min_{j \in N(i)} H(j)$$

and sets $P(i_{m+1})$ to the path generated by Ψ , starting from i_{m+1} . On the other hand, if:

$$\min_{j \in N(i_m)} H(j) < g(i'_k) \quad (21)$$

$\mathcal{R}\bar{\Psi}$ adds to the node sequence $P(i_m) = (i_m, i'_{m+1}, \dots, i'_k)$ the node:

$$i_{m+1} < i'_{m+1}$$

and sets $P(i_{m+1})$ to the path $(i_m, i'_{m+1}, \dots, i'_k)$. If i_{m+1} is a destination, $\mathcal{R}\bar{\Psi}$ terminates, and otherwise $\mathcal{R}\bar{\Psi}$ starts the next step with $m+1$ replacing m .

The main idea behind the construction of $\mathcal{R}\bar{\Psi}$ is to follow the path $P(i_m)$ unless a path of lower cost is discovered through the following equation. It can be seen that $\mathcal{R}\bar{\Psi}$ may be viewed as the *rollout algorithm* $\mathcal{R}\Psi$ corresponding to a modified version of Ψ , called *fortified Ψ* , and denoted $\bar{\Psi}$. This algorithm is applied to a slightly modified version of the original problem, which involves an additional downstream neighbor for each node i_m that is generated in the course of the algorithm $\mathcal{R}\bar{\Psi}$ and for which the following condition holds. For every such node i_m , the additional neighbor is a copy of i'_{m+1} , and the path generated by $\bar{\Psi}$ starting from this copy is $(i'_{m+1}, \dots, i'_k)$. From every other node, the path generated by Ψ is the same as the path generated by $\bar{\Psi}$. It can be seen that $\bar{\Psi}$ is sequentially improving, so that $\mathcal{R}\bar{\Psi}$ is terminating and has the automatic cost sorting property of **Propositions 1 and 2**; that is:

$$H(i_m) = \min \left\{ H(i_1), \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j) \right\}$$

The above property can also be easily verified directly, using the definition of $\mathcal{R}\bar{\Psi}$. It can also be seen that the *fortified rollout algorithm* $\mathcal{R}\bar{\Psi}$ will always perform at least as well as $\mathcal{R}\Psi_e$. The potential improvement in performance is obtained at the expense of the modest additional overhead involved in maintaining the path $P(i_m)$. Note that when Ψ is sequentially consistent, all three rollout algorithms $\mathcal{R}\Psi$, $\mathcal{R}\Psi_e$, and $\mathcal{R}\bar{\Psi}$ coincide.

We note that one may use multiple path construction algorithms in the preceding framework. In particular, let us assume that we have K algorithms $\Psi_1, \dots, \Psi_K$. The k^{th} of these algorithms, given a node $i \notin N$, produces a path $(i, i_1, \dots, i_m, \bar{i})$ that ends at a destination node $\bar{i}$, and the corresponding cost is denoted by $H_K = g(\bar{i})$. Generalizing our earlier approach, we can use the K algorithms in an approximation architecture of the form:

$$\tilde{J}(i) = \min_{k=1, \dots, K} H_k(i) \quad (22)$$

or of the form:

$$\tilde{J}(i, r_1, \dots, r_K) = \sum_{k=1}^K r_K H_k(i) \quad (23)$$

where r_k are some fixed scalar weights obtained by trial and error. The rollout algorithms $\mathcal{RP}$, $\mathcal{RP}_e$, and $\mathcal{RP}^{\bar{P}}$ easily generalize for the case of Eq. (22), by replacing $H(i)$ with $\tilde{J}(i)$, and by defining the path generated starting from a node i as the path generated by the path construction algorithm which attains the minimum in Eq. (22). In the case of Eq. (23), the rollout algorithm $\mathcal{RP}$ also generalizes easily by replacing $H(i)$ with $\tilde{J}(i, r_1, \dots, r_K)$, but in order to generalize the algorithms $\mathcal{RP}$, and $\mathcal{RP}_e$, the path generated from a node i must also be defined. There are several possibilities along this line. A different type of possibility for the case of Eq. (23), is to use tunable weights, which are obtained by training using NDP methodology.

Finally, let us consider a problem where in addition to the terminal cost $g(i)$, there is a cost $c(i, j)$ for a path to traverse an arc (i, j) . Within this context, the cost of a path $(i_1, i_2, \dots, i_n)$ that starts at i_1 and ends at a destination node i_n is redefined to be:

$$g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1})$$

One way to transform this problem into one involving a terminal cost only is to redefine the graph of the problem so that nodes correspond to sequences of nodes in the original problem graph. Thus if we have arrived at node i_k using path $(i_1, \dots, i_k)$, the choice of i_{k+1} as the next node is viewed as a transition from state $(i_1, \dots, i_k)$ to state $(i_1, \dots, i_k, i_{k+1})$. Both states $(i_1, \dots, i_k)$ and $(i_1, \dots, i_k, i_{k+1})$ are viewed as nodes of a redefined graph. Furthermore, in this redefined graph, a destination node has the form $(i_1, i_2, \dots, i_n)$, where i_n is a destination node of the original graph, and has a cost given by Eq. (24).

After the details are worked out, we see that to recover our earlier algorithms and analysis, we need to modify the cost of the heuristic algorithm Ψ as follows: If the path $(i_1, i_2, \dots, i_n)$ is generated by Ψ starting at i_1 , then:

$$H(i_1) = g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1})$$

Furthermore, the rollout algorithm $\mathcal{RP}$ at node i_m selects as next node i_{m+1} the node:

$$i_{m+1} = \arg \min_{j \in N(i)} [c(i_m, j) + H(j)]$$

The definition of a sequentially consistent algorithm remains unchanged:

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1}) = \min_{j \in N(K)} [c(i_k, j) + H(j)], \quad k = 1, \dots, m-1$$

A sequentially improving algorithm should now be characterized by the property:

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1})$$

if i_{k+1} is the next node on the path generated by Ψ starting from i_k . Furthermore, **Proposition 2** remains unchanged except it is modified to read:

$$H(i_k) \geq \min_{j \in N(K)} [c(i_k, j) + H(j)], \quad k = 1, \dots, m-1$$

Finally, the criterion $\min_{j \in N(K)} H(j) < g(i_k)$ used in the fortified rollout algorithm, given the sequence

$(s, i, i_1, \dots, i_m)$, where $i_m \notin \bar{N}$, and the path $P(i_m) = (i_m, i_{m+1}, \dots, i_K)$, should be replaced by:

$$\min_{j \in N(K)} [c(i_k, j) + H(j)] < g(i'_K) + c(i'_m, i'_{m+1}) + \sum_{l=m+1}^{k-1} c(i'_l, i'_{l+1})$$

Let s is the number of initially available spare parts, and let m and y be the vectors:

$$m = (m_1, \dots, m_T), \quad y = (y_1, \dots, y_T),$$

where $m_t, t=1, \dots, T$, is the number of machines of type t (all assumed to be initially working), and $y_t, t=1, \dots, T$, is the number of breakdowns of machines of type t during the first stage. The decision to be made is:

$$u = (u_1, \dots, u_T)$$

where u_t is the number of spare parts used to repair breakdowns of machines of type t at the end of the first stage. We note that at the second stage, it is optimal to use the remaining spare parts to repair the machines that break in the order of their cost (that is, repair the most expensive broken machines, then if spare parts are left over, consider the next most expensive broken machines, etc). Thus, if we start the second stage with $\bar{s}$ spare parts, and $\bar{m}_t$ working machines of type $t=1, \dots, T$, and during the second stage, $\bar{y}_t$ machines of type t break, $t=1, \dots, T$, the optimal cost of the second stage, which is denoted by $G(\bar{m}, \bar{y}, \bar{s})$, where:

$$m = (m_1, \dots, m_T), \quad y = (y_1, \dots, y_T)$$

can be calculated analytically. We will not give the formula for the function G , because it is quite complicated, although it can be easily programmed for computation.

Let us denote by R the expected value, over the second stage breakdowns, of the second stage cost:

$$R(\bar{m}, \bar{s}) = E_{\bar{y}}[G(\bar{m}, \bar{y}, \bar{s})]$$

Then in the first stage, and once the first stage breakdowns are known, the problem is to find $u = (u_1, \dots, u_T)$ that solves the problem:

$$\text{minimize } \sum_{t=1}^T C_t(y_t - u_t) + R\left(m - y + u, \sum_{t=1}^T u_t\right)$$

$$\text{subject to } \sum_{t=1}^T u_t \leq s, \quad 0 \leq u_t \leq y_t, \quad t = 1, \dots, T$$

This is the problem we wish to solve approximately by using a *rollout algorithm*.

Let reformulate this first stage problem as a path construction problem. In the reformulated problem, the nodes of the graph are triplets (m, y, s) . Destination nodes are the ones for which $y = 0$ and the repair/no repair decision has been made for all the first stage breakdowns. At a non-destination node, the control choices are to select a particular breakdown type, say t , with $y_t > 0$, and then select between two options:

- (1) Leave the breakdown unrepaired, in which case the triplet (m, y, s) evolves to:

$$(m_1, \dots, m_{t-1}, m_t - 1, \dots, m_{t+1}, \dots, m_T, y_1, \dots, y_{t-1}, y_t - 1, \dots, y_{t+1}, \dots, y_T, s)$$

and the cost C_t of permanently losing the corresponding machine is incurred.

- (2) Repair the breakdown, in which case the triplet (m, y, s) evolves to:

$$(m_1, \dots, m_T, y_1, \dots, y_{t-1}, y_t - 1, \dots, y_{t+1}, \dots, y_T, s - 1)$$

and no cost is incurred.

Once we have $y_1 = \dots = y_T = 0$, there is no decision to make, and we simply pay the optimal *cost-to-go* of the second stage, $R(\bar{m}_1, \bar{m}_T, \bar{s})$, and terminate.

We consider *rollout policies* based on heuristic algorithms. We used the following two heuristics, which given the triplet (m, y, s) , produce a first stage solution u :

- (1) *Proportional Heuristic*: In this heuristic, we compute an estimate N of the total number of second stage breakdowns based on the probabilities p_t of breakdown of individual machines of type t . In particular, we have $\bar{N} = \sum_{t=1}^T p_t N_t$, where N_t is a heuristic estimate of the number of working machines of type t at the start of the second stage, based on the already known vectors m and y . We form the estimated ratio of first stage to second stage breakdowns,

$$f = \sum_{t=1}^T y_t$$

We then find the number of spare parts to be used in the first stage to:

$$s_1 = a f s$$

where a is a positive parameter. The first stage problem is then solved by allocating the s_1 spare parts to machines of type t in the order of the costs $C_t(1-p_t)$. (The factor of $1-p_t$ is used to account for the undesirability of repairing machines that are likely to break again.) The constant a provides a parameterization of this heuristic. In particular, when $a < 1$, the heuristic is conservative, allocating more spare parts to the second stage than the projected ratio of breakdowns suggests, while if $a > 1$, the heuristic is more myopic, giving higher priority to the breakdowns that have already occurred in the first stage.

- (2) *Value-Based heuristic*: In this heuristic, given the state, we assign values of C_t and $C_t(1-p_t)$ to each spare part used to repair a machine of type t in the second stage and the first stage respectively. Note that a repair of a machine in the first stage is valued less than a repair of the same machine in the second stage, since a machine that is repaired in the first stage may break down in the second stage and require the use of an extra spare part. We rank-order the values C_t and $C_t(1-p_t)$, $t=1, \dots, T$, and we repair broken down machines in decreasing order of value, using the estimate $p_t(m_T - y_t)$ for the number of machines to break down in the second stage.

It is a fact that *the rollout algorithms* can improve significantly the performance of the original heuristic algorithm Ψ . In particular, the relative improvement is significant. Furthermore, it can be denoted that:

- (a) All of the algorithms $\mathcal{R}^*\Psi$, $\mathcal{R}\Psi_e$, and $\mathcal{R}\bar{\Psi}$, and $\mathcal{R}\bar{\Psi}$ consistently outperform the original heuristic algorithm Ψ . On the other hand, because Ψ is not guaranteed to be sequentially improving, *the standard rollout algorithm* $\mathcal{R}\Psi$ may perform worse than the original heuristic Ψ .
- (b) *The optimized rollout algorithm* $\mathcal{R}^*\Psi$ consistently outperforms the standard and the extended rollout algorithms $\mathcal{R}\Psi$ and $\mathcal{R}\Psi_e$.
- (c) *The fortified rollout algorithm* $\mathcal{R}\bar{\Psi}$ consistently outperforms the standard and the *extended rollout algorithms* $\mathcal{R}\Psi$ and $\mathcal{R}\Psi_e$. On the other hand there is no clear superiority relation between the optimized and the fortified algorithms.

L-lysine production, before and after *alternative rollout optimization* is shown in **Fig. 2**.

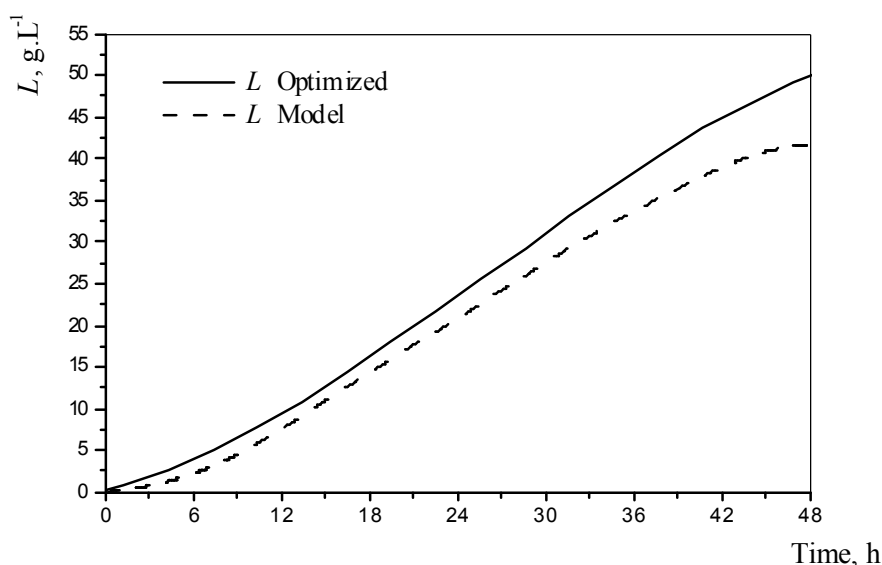


Fig. 2. *L-lysine* production, before and after *alternative rollout optimization*

2. NONLINEAR MODEL PREDICTIVE CONTROL OF THE EXAMINED PROCES

The control of fermentation processes focuses on an open-loop operation owing to their highly nonlinear and inherently difficult dynamic behavior. The optimization is carried out *off-line* and the reactor is fed by the determined optimal feed profile. Once the batch proceeds there is no provision to account for disturbances occurring during the batch (Camacho et al., 2004, Soni, 2002).

Finally the performance of the controller is evaluated and strategies for disturbance compensation are presented. The results of this approach are presented for carnitine production by *L-lysine*.

An added advantage of using a predictive controller is the fact that the decisions made for fed-batch fermentors during the course of the process operation have a significant impact on the final product quality. In such a scenario the model predictive control algorithm proves beneficial since it can predict the effect of the past input moves on the future system behavior. Since process feedback is incorporated into the algorithm at every time step the predicted future constraint violations can be accounted for and subsequently input moves can be chosen so that they can prevent these violations from taking place. On the other hand, constraint handling is not easily incorporated with classical feedback techniques; furthermore classical feedback controllers respond only after the disturbance manifests itself in the process output thereby limiting their applicability for constrained fed-batch control problems (Soni, 2002).

This problem of implementing the optimal policy and ensuring that the system follows the optimal trajectory in the presence of disturbances has received a limited previous consideration. Attention is thus focused on designing a controller to track the optimal policy and on approaches for the disturbance compensation for the closed-loop control problem (Garcia, 1989).

The control scheme for the fed-batch bioreactor is implemented in two stages and the substrate feed rate serves as the manipulated variable. Initially a constrained optimal control problem is solved off-line in order to determine the manipulated variable profile that maximizes the end of the batch product

concentration for the product of interest while maintaining a pre-specified fixed final volume. The next step involves tracking of the optimal control trajectory in a closed-loop operation (Soni, 2002).

Even if the system is linear, the closed-loop dynamics are in general nonlinear due to the presence of constraints. *NMPC* refers to *MPC* schemes that are based on nonlinear models and/or consider non-quadratic cost-functionals and general nonlinear constraints on the states and inputs.

Since its first invention in the 70s of the last century, linear *MPC* has crystallized as one of the key advanced control strategies. By now linear model predictive control, is widely used in industrial applications especially in the process industry (Morari et al., 1997). The practical success is mainly based on the possibility to take constraints on the states and inputs systematically into account while operating the process optimally.

The purpose of this paper is twofold. In the first part a review is provided on the current state of *NMPC*. After a presentation of the basic principle of predictive control some of the key theoretical, computational and implementational aspects of this control strategy is presented. Furthermore, the inherent advantages and disadvantages of *NMPC* are discussed. If it is mainly focus on *NMPC* for continuous time systems using sampled measurement information and do not go into details on discrete time *NMPC* strategies. For more self contained reviews on *NMPC* is referred to Allgower et al., 1999, Chen et al., 1998, De Nicolao et al., 2000, Mayne et al., 2000, Morari et al., 1997.

One important precondition for the application of *NMPC*, is the availability of reliable and efficient numerical dynamic optimization algorithms, since at every sampling time a nonlinear dynamic optimization problem must be solved in real-time. Solving such an optimization problem efficiently and fast is, however, not a trivial task and has attracted strong research interest in recent years).

Model predictive control is formulated as the repeated solution of a (finite) horizon open-loop optimal control problem subject to system dynamics and input and state constraints. Figure 1 depicts the basic idea behind model predictive control. Based on measurements obtained at time t , the controller predicts the dynamic behavior of the system over a prediction horizon T_p in the future and determines (over a control horizon $T_c \leq T_p$) the input such that a predetermined open-loop performance objective is minimized. If there were no disturbances and no model-plant mismatch, and if the optimization problem could be solved over an infinite horizon, then the input signal found at $t=0$ could be open-loop applied to the system for all $t \geq 0$. However, due to disturbances, model-plant mismatch, and the finite prediction horizon the actual system behavior is different from the predicted one. To incorporate feedback, the optimal open-loop input is implemented only until the next sampling instant. In principle the time between each new optimization, the sampling time, can vary. For simplicity of presentation, that it is fixed, i.e the optimal control problem is re-evaluated after the constant sampling time δ . Using the new system state at time $t+\delta$, the whole procedure – prediction and optimization – is repeated, moving the control and prediction horizon forward.

In the following, $\|\cdot\|$ denotes the Euclidean vector norm in $\mathbf{R}^n$ (where the dimension n follows from context) or the associated induced matrix norm. Vectors are denoted by boldface symbols. Whenever a semicolon “;” occurs in a function argument, the following symbols should be viewed as additional parameters, i.e. $f(x;\gamma)$ means the value of the function f at x with the parameter γ .

Models in *MPC* are used for predicting the process output over a prediction horizon. Control actions are calculated over a control horizon in such a way that the predicted process output is as close as possible to the desired reference signal and the first control action in sequence is applied in each step (**Fig. 1**) (Morari 1999, Chen 1998).

In **Fig. 1** the open-loop optimal input is depicted as a continuous function of time. To allow a numerical solution of the open-loop optimal control problem the input is often parametrized by a finite number of “basis” functions, leading to a finite dimensional optimization problem. In practice often a

piecewise constant input is used, leading to T_c/δ decisions for the input over the control horizon.

Summarizing, a standard *NMPC* scheme works as follows:

1. Obtain estimates of the states of the system;
2. Calculate a constraint-conforming optimal input minimizing the desired cost function over the prediction horizon using the system model and the current state estimate for prediction;
3. Implement the first part of the optimal input until the next sampling instant;
4. Continue with 1.

While it can be desirable for computational and performance reasons to choose unequal lengths of the prediction and control horizon, assuming that in the following that $T_c=T_p$ for our presentation.

2.1. Mathematical formulation of state feedback NMPC

Consider the stabilization of time-invariant nonlinear systems of the form:

$$\dot{x}(t) = f(x(t), u(t)), x(0)=x_0 \quad (24)$$

subject to the input and state constraints:

$$u(t) \leq U, \forall t \geq 0 \quad (25)$$

$$x(t) \leq X, \forall t \geq 0 \quad (26)$$

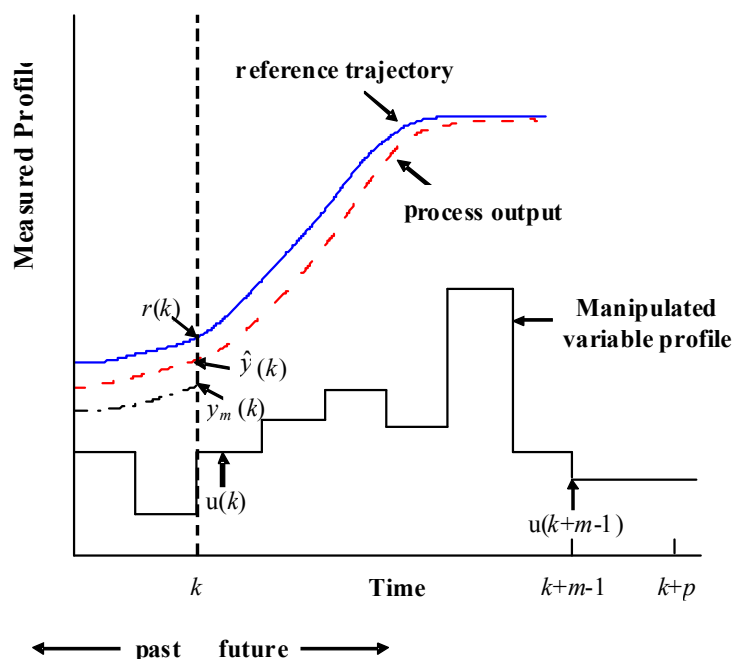


Fig. 3: Principle of model predictive control

With respect to the vector field $f: \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^n$, assuming that it is locally Lipschitz continuous in the region of interest (typically the region of attraction) and satisfies $f(0, 0)=0$. Furthermore, the set $U \subseteq$

$\mathbf{R}^m$ is compact, $\mathbf{X} \rightarrow \mathbf{R}^n$ is connected, and $(0, 0) \leq \mathbf{X} \times \mathbf{U}$. Typically $\mathbf{U}$ and $\mathbf{X}$ are (convex) box constraints of the form:

$$\mathbf{U} = \{u \in \mathbf{R}^m \mid u_{\min} \leq u \leq u_{\max}\} \quad (27)$$

$$\mathbf{X} = \{x \in \mathbf{R}^n \mid x_{\min} \leq x \leq x_{\max}\} \quad (28)$$

with the constant vectors $u_{\min}$, $u_{\max}$ and $x_{\min}$, $x_{\max}$.

In sampled-data NMPC an open-loop optimal control problem is solved at discrete sampling instants t_i based on the current state information $x(t_i)$. Since considering that a constant sampling time δ , the sampling instants t_i are given by $t_i = I, \delta, i=0, 1, 2, \dots$. When the time t and t_i occurs in the same setting, t_i should be taken as the closest previous sampling instant $t_i < t$.

The open-loop input signal applied in between the sampling instants is given by the solution of the following optimal control problem:

$$\textbf{Problem 1: } \min_{\bar{u}(\cdot)} J(x(t_i), \bar{u}(\cdot)), \text{ subject to: } \bar{x}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}_i = x(t_i) \quad (29)$$

$$\bar{u}(\tau) \in \mathbf{U}, \quad \bar{x}(\tau) \in \mathbf{X}, \quad \tau \in [t_i + T_p] \quad (30)$$

$$\bar{x}(t_i + T_p) \in \mathfrak{S} \quad (31)$$

The bar denotes predicted variables (internal to the controller), i.e. $\bar{x}(\cdot)$ is the solution of (6a) driven by the input $\bar{u}(\cdot) : [t_i + T_p] \rightarrow \mathbf{U}$ with the initial condition $x(t_i)$. The distinction between the real system variables and the variables in the controller is necessary, since even in the nominal case the predicted values will not be the same as the actual *closed-loop values*.

The cost functional J minimized over the control horizon $T_p \leq \delta > 0$ is typically given by:

$$J(x(t_i), \bar{u}(\cdot)) = \int_{t_i}^{t_i + T_p} F(\bar{x}(\tau), \bar{u}(\tau)) d\tau + E \bar{x}(t_i + T_p) \quad (32)$$

where the stage cost $F : \mathbf{X} \times \mathbf{U} \rightarrow \mathbf{R}^+$ is often assumed to be continuous, satisfies $F(0, 0) = 0$, and lower bounded by a class $\mathbf{K}$ function $\alpha_F : \alpha_F(\|x\|) \leq F(x, u) \forall (x, u) \in \mathbf{X} \times \mathbf{U}$.

The terminal penalty term E and the so called terminal region constraint $\bar{x}(t_i + T_p) \in \mathfrak{S}$ might or might not be present. These are often used to enforce nominal stability.

The stage cost can for example arise from economical and ecological considerations. Often, a quadratic form for F is used:

$$F(x, u) = x^T Q x + u^T R u \quad (33)$$

with $Q \geq 0$ and $R > 0$.

The state measurement enters the system via the initial condition in (6a) at the sampling instant, i.e. the system model used to predict the future system behavior is initialized by the actual system state. Since all state information is necessary for the prediction, the full state must be either measured or estimated.

The solution of the optimal control problem (30) is denoted by $\bar{u}(\cdot; x(t_i))$. It defines the open-loop input that is applied to the system until the next sampling instant t_{i+1} :

$$u(t; x(t_i)) = \bar{u}(t; x(t_i)), \quad t \in [t_i, t_{i+1}) \quad (34)$$

The control $u(t; x(t_i))$ is a feedback, since it is recalculated at each sampling instant using the new state measurement. In comparison to sampled-data *NMPC* for continuous time systems, in instantaneous *NMPC* (also often referred to as receding horizon control) the input is defined by the solution of

Problem 1 at all times: $u(x(t)) = \bar{u}(t; x(t))$, i.e. open-loop input is applied (De Nicolao 2000, Mayne 2000).

The solution of (1) starting at time t_1 from an initial state $x(t_1)$, applying an input $u: [t_1, t_2] \rightarrow \mathbf{R}^m$ is denoted by $x(\tau; u(\cdot), x(t_1))$, $\tau \in [t_1, t_2]$. If it will refer to an admissible input as:

Definition 3: (Admissible input) An input $u: [0, T_p] \rightarrow \mathbf{R}^m$ for a state x_0 is called admissible, if it is:

a) piecewise continuous

b) $u(\tau) \in \mathbf{U} \forall \tau \in [0, T_p]$

c) $x(\tau; u(\cdot), x_0) \in \mathbf{X} \forall \tau \in [0, T_p]$ d) $x(T_p; u(\cdot), x_0) \in \mathfrak{T}$

Definition 4: (Value function) The value function $V(x)$ of the open-loop optimal control **Problem 1** is defined as the minimal value of the cost for the state x : $V(x) = J\left(x, \bar{u}(\cdot; x)\right)$

The value function plays a central role in the stability analysis of *NMPC*, since it often serves as a Lyapunov function candidate (Allgower 1999, Mayne 2005).

Typically, no explicit controllability assumption on the system is considered in *NMPC*. Instead, the stability results are based on the assumption of initial feasibility of the optimal control problem, i.e. the existence of a input trajectory $u(\cdot)$ s. t. all constraints are satisfied.

The unequality of the predicted and the closed-loop trajectories has two immediate consequences. Firstly, the actual goal to compute a feedback such that the performance objective over the *infinite horizon* of the closed-loop is minimized is not achieved. In general it is by no means true that the repeated minimization over a moving *finite horizon* leads to an optimal solution for the infinite horizon problem. The solutions will often differ significantly if a short finite horizon is chosen. Secondly there is in general no guarantee that the closed-loop system will be stable. It is indeed easy to construct examples for which the closed-loop becomes unstable if a short finite horizon is chosen. Hence, when using finite prediction horizons special attention is required to guarantee stability.

The summarize the key characteristics and properties of *NMPC* are:

- *NMPC* allows the direct use of nonlinear models for prediction;
- *NMPC* allows the explicit consideration of state and input constraints;
- In *NMPC* a specified time domain performance criteria is minimized *on-line*;
- In *NMPC* the predicted behavior is in general different from the closed loop behavior;
- For the application of *NMPC* typically a real-time solution of an open-loop optimal control problem is necessary.

To perform the prediction the system states must be measured or estimated.

Basing the applied input on the solution of an optimal control problem that must be solved on-line is advantageous and disadvantageous at the same time. First, and most important, this allows to directly consider constraints on states and inputs which are often difficult to handle otherwise. Furthermore, the desired cost objective, the constraints and even the system model can in principle be adjusted on-line without making a complete redesign of the controller necessary. However, solving the open-loop

optimal control problem, if attacked blindly, can be difficult or even impossible for large systems.

4.1 State feedback NMPC and nominal stability

One of the most important questions in *NMPC* is whether a finite horizon *NMPC* strategy does guarantee stability of the closed-loop or not. The key problem with a finite prediction and control horizon is due to the difference between the predicted open-loop and the resulting closed-loop behaviour. Ideally, one would seek for an *NMPC* strategy which achieves closed-loop stability independently of the choice of the parameters and, if possible, approximates the infinite horizon *NMPC* scheme as well as possible. An *NMPC* strategy that achieves closed-loop stability independently of the choice of the performance parameters is often referred to as an *NMPC* approach with *guaranteed stability*. Different approaches to achieve closed-loop stability using finite horizon lengths exist. Here only some central ideas are reviewed and no detailed proofs are given. Moreover, no attempt is made to cover all existing methods. Most of the technical details are left out for reasons of a simple presentation.

Finite Horizon NMPC Schemes with Guaranteed Stability: Different possibilities to achieve closed-loop stability using a finite horizon length exist. Most of these approaches modify the standard *NMPC* setup such that stability of the closed-loop can be guaranteed independently of the plant and performance specification. This is usually achieved by adding suitable equality or inequality constraints and suitable additional penalty terms to the standard setup. The additional terms are generally not motivated by physical restrictions or performance requirements but have the sole purpose to enforce stability. Therefore, they are usually called *stability constraints*. One possibility to enforce stability with a finite prediction horizon is to add the so called *zero terminal equality constraint* at the end of the prediction horizon, i.e.

$$\bar{x}(t + T_p) = 0 \quad (35)$$

is added to **Problem 1**. This leads to stability of the closed-loop, if the optimal control problem has a solution at $t=0$. Similar to the infinite horizon case the feasibility at one sampling instant does imply feasibility at the following sampling instants and a decrease in the value function.

Many schemes exist that try to overcome the use of a zero terminal constraint of the form (10). Most of them use the *terminal region constraint* $\bar{x}(t + T_p) \in \mathfrak{S}$ and/or a *terminal penalty term* $E(\bar{x}(t + T_p))$ to enforce stability and feasibility. Typically the terminal penalty E and the terminal region $\mathfrak{S}$ are determined off-line such that the cost function:

$$J(x(t), \bar{u}(\cdot)) = \int_{t_i}^{t_i + T_p} F(\bar{x}(\tau), \bar{u}(\tau)) d\tau + E(\bar{x}(t_i + T_p)) \quad (36)$$

gives an upper bound on the infinite horizon cost and guarantees a *decrease in the value function* as the horizon recedes in time.

Theorem 1 (Stability of sampled-data NMPC) Suppose that:

- a) the terminal region $\mathfrak{S} \subseteq \mathbf{X}$ is closed with $0 \in \mathfrak{S}$ and that the terminal penalty $E(x) \leq C$ is positive semi-definite
- b) the terminal region and terminal penalty term are chosen such that $\forall x \in \mathfrak{S}$ there exists an (admissible) input $u_{\mathfrak{S}} : [0, \delta] \rightarrow \mathbf{U}$ such that $x(\tau) \in \mathfrak{S} \forall \tau \in [0, \delta]$ and:

$$\frac{\partial E}{\partial x} f(x(\tau), u_{\mathfrak{S}}(\tau)) + F(x(\tau), u_{\mathfrak{S}}(\tau)) \leq 0, \quad \forall \tau \in [0, \delta] \quad (37)$$

- c) the *NMPC* open-loop optimal control problem is feasible for $t=0$

Then in the closed-loop system (1) with (9) $x(t)$ converges to the origin for $t \rightarrow \infty$, and the region of attraction $\mathbf{R}$ consists of the states for which an admissible input exists.

Proof. The proof is given here for sake of completeness. It bases on using the value function as a decreasing Lyapunov-like function. As usual in predictive control the proof consists of two parts: in the first part it is established that initial feasibility implies feasibility afterwards. Based on this result it is then shown that the state converges to the origin.

Feasibility: Consider any sampling instant t_i for which a solution exists (e.g. t_0). In between t_i and t_{i+1} the optimal input $\bar{u}(\cdot; x(t_i))$ is implemented. Since no model plant mismatch nor disturbances are present, $x(t_{i+1}) = \bar{x}(t_{i+1}; \bar{u}(\cdot; x(t_i)), x(t_i))$. The remaining piece of the optimal input $\bar{u}(\tau; x(t_i))$, $\tau \in [t_{i+1}, t_i + T_p]$ satisfied the state and input constraints. Furthermore, $\bar{x}(t_i + T_p; x(t_i), \bar{u}(\cdot; x(t_i))) \in \mathfrak{S}$ and that is known from Assumption (b) of the theorem that for all $x \in \mathfrak{S}$ there exists at least one input $u_{\mathfrak{S}}(\cdot)$ that renders $\mathfrak{S}$ invariant over δ . Picking any such input we obtain as admissible input for any time $t_i + \sigma$, $\sigma \in (0, t_{i+1} - t_i]$:

$$\tilde{u}(\tau; x(t_i + \sigma)) = \begin{cases} \bar{u}(\tau; x(t_i)), & \tau \in [t_{i+1} + \sigma, t_i + T_p] \\ u_{\mathfrak{S}}(\tau - t_i + T_p), & \tau \in (t_i + T_p, t_i + T_p + \sigma] \end{cases} \quad (38)$$

Specifically, it is given for the next sampling time ($\sigma = t_{i+1} - t_i$) that $\tilde{u}(\cdot; x(t_{i+1}))$ is a feasible input, hence feasibility at time t_i implies feasibility at t_{i+1} . Thus, if (38) is feasible for $t=0$, it is feasible for all $t \geq 0$.

Furthermore, if the states for which an admissible input exists converge to the origin, it is clear that the region $\mathbf{R}$ that consists of those points belongs to the region of attraction.

Convergence: it is first show that the value function is decreasing starting from a sampling instant. Also, the value of V at $x(t_i)$ is given by:

$$V(x(t_i)) = \int_{t_i}^{t_i + T_p} F(\bar{x}(\tau; \bar{u}(\cdot; x(t_i)), x(t_i)), \bar{u}(\tau; x(t_i))) d\tau + E(\bar{x}(t_i + T_p, \bar{u}(\cdot; x(t_i))), x(t_i)) \quad (39)$$

and the cost resulting from (39) starting from any $x(t_i + \sigma, \bar{u}(\cdot; x(t_i)), x(t_i))$, $\sigma \in (0, t_{i+1} - t_i]$ using the input $\tilde{u}(\tau; x(t_i + \sigma))$, given by:

$$J(x(t_i + \sigma), \tilde{u}(\cdot; x(t_i + \sigma))) = \int_{t_i + \sigma}^{t_i + \sigma + T_p} F(\bar{x}(\tau; \tilde{u}(\cdot; x(t_i + \sigma)), x(t_i + \sigma)), \tilde{u}(\tau; x(t_i + \sigma))) d\tau + E(\bar{x}(t_i + \sigma + T_p; \tilde{u}(\cdot; x(t_i + \sigma)), x(t_i + \sigma))) \quad (40)$$

Reformulation yields:

$$J(x(t_i + \sigma), \tilde{u}(\cdot; x(t_i + \sigma))) = V(x(t_i)) - \int_{t_i}^{t_i + \sigma} F(\bar{x}(\tau; \bar{u}(\cdot; x(t_i)), x(t_i)), \bar{u}(\tau; x(t_i))) d\tau - E(\bar{x}(t_i + T_p; \bar{u}(\cdot; x(t_i)), x(t_i))) + \int_{t_i + \sigma}^{t_i + \sigma + T_p} F(\bar{x}(\tau; \tilde{u}(\cdot; x(t_i + \sigma)), x(t_i + \sigma)), \tilde{u}(\tau; x(t_i + \sigma))) d\tau + E(\bar{x}(t_i + \sigma + T_p; \tilde{u}(\cdot; x(t_i + \sigma)), x(t_i + \sigma))) \quad (41)$$

Integrating inequality (40) from $t_i + \sigma$ to $t_i + \sigma + T_p$ starting from $x(t_i + \sigma)$ is obtained zero as an upper bound for the last three terms on the right side. Thus:

$$J(x(t_i + \sigma), \tilde{u}(\cdot; x(t_i + \sigma))) - V(x(t_i)) \leq - \int_{t_i}^{t_i + \sigma} F\left(\bar{x}(\tau; \bar{u}(\cdot; x(t_i)), x(t_i)), \bar{u}(\tau; x(t_i))\right) d\tau \quad (42)$$

Since $\tilde{u}$ is only a feasible but not necessarily the optimal input for $x(t_i + \sigma)$, it follows that:

$$V(x(t_i + \sigma)) - V(x(t_i)) \leq - \int_{t_i}^{t_i + \sigma} F\left(\bar{x}(\tau; \bar{u}(\cdot; x(t_i)), x(t_i)), \bar{u}(\tau; x(t_i))\right) d\tau \quad (43)$$

i.e. the value function is decreasing along solution trajectories starting at a sampling instant t_i . Especially we have that:

$$V(x(t_i + 1)) - V(x(t_i)) \leq - \int_{t_i}^{t_{i+1}} F\left(\bar{x}(\tau; \bar{u}(\cdot; x(t_i)), x(t_i)), \bar{u}(\tau; x(t_i))\right) d\tau \quad (44)$$

By assumption, this decrease in the value function is strictly positive for $x(t_i) \neq 0$. Since this holds for all sampling instants, convergence can be established as in (Chen 1998) by an induction argument and the application of Barbalat's lemma.

Loosely speaking, E is an F -conform local control Lyapunov function in $\mathfrak{S}$. The terminal region constraint enforces feasibility at the next sampling instant and allows, similarly to the infinite horizon case, to show that the value function is strictly decreasing. Thus stability can be established. Note that this result is nonlocal in nature, i.e. there exists a region of attraction $\mathbf{R}$ which is of at least the size of $\mathfrak{S}$. Typically the region of attraction resulting from this scheme is much larger than $\mathfrak{S}$ and contains the set of all states for which the open-loop optimal control problem has a feasible solution.

Various ways to determine a suitable terminal penalty term and terminal region exist. Examples are the use of a control Lyapunov function as terminal penalty E_i or the use of a local nonlinear or linear control law to determine a suitable terminal penalty E and a terminal region $\mathfrak{S}$.

Quasi-infinite horizon NMPC: We exemplify the choice of a suitable terminal region constraint and a terminal penalty term considering the so called quasi-infinite horizon NMPC (QIH-NMPC) approach (Findeisen 2001). In QIHNMPC E and $\mathfrak{S}$ are obtained on the basis of a locally stabilizing linear control law $u_{\mathfrak{S}} = Kx$ considering a quadratic stage cost of the form (36). The terminal penalty term E in this case is quadratic and of the form $E(x) = x^T Px$. The following procedure gives a systematic approach to compute the terminal region and a terminal penalty matrix *off-line*, assuming that the Jacobian linearization (A, B) of (1) is stabilizable, where $A := \frac{\partial f}{\partial x}(0, 0)$ and $B := \frac{\partial f}{\partial u}(0, 0)$.

Step 1 : Solve the linear control problem based on the Jacobian linearization (A, B) of (1) to obtain a locally stabilizing linear state feedback $u_{\mathfrak{S}} = Kx$.

Step 2 : Define $A_K := A + BK$, and choose a constant $\kappa \in [0, \infty)$ satisfying $\kappa < -\lambda_{\max}(A_K)$ and solve the Lyapunov equation:

$$(AK + \kappa I)^T P + P(AK + \kappa I) = -(Q + K^T RK) \quad (45)$$

to get a positive definite and symmetric P .

Step 3 : Find the largest possible α_1 defining a region

$$\mathfrak{S}_1 := \{x \in \mathbf{R}^n \mid x^T Px \leq \alpha_1\} \quad (46)$$

such that $Kx \leq U$, for all $x \in \mathfrak{I}_1$.

Step 4 : Find the largest possible $\alpha \in (0, \alpha_1]$ specifying a terminal region

$$\mathfrak{I} := \{x \in \mathbf{R}^n \mid x^T P x \leq \alpha\} \quad (47)$$

such that the optimal value of the following optimization problem is non-positive:

$$\max_x \{x^T P \varphi(x) - \kappa \cdot x^T P x \leq \alpha\} \quad (48)$$

where $\varphi(x) := \mathbf{f}(x, Kx) - A_K x$ $\varphi(x) := \mathbf{f}(x, Kx) - AKx$.

This procedure allows to calculate E and $\mathfrak{I}$ if the linearization of the system at the origin is stabilizable. If the terminal penalty term and the terminal region are determined accordingly, the open-loop optimal trajectories found at each time instant approximate the optimal solution for the infinite horizon problem.

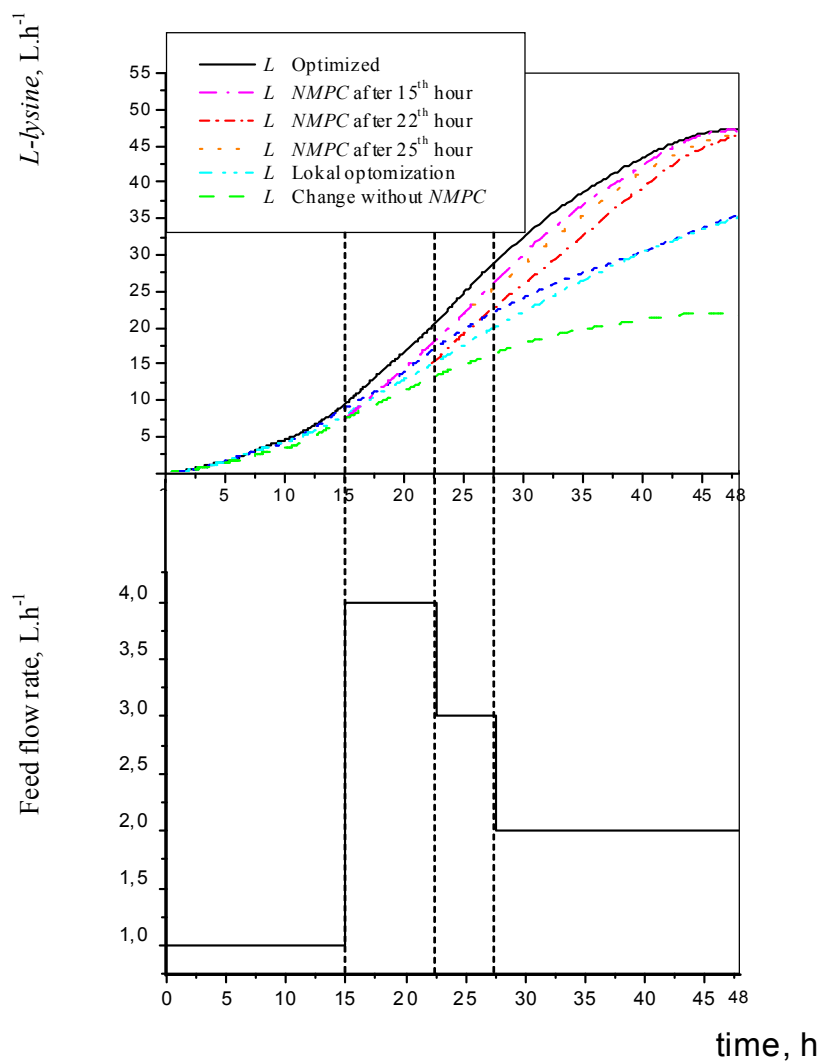


Fig. 4. NMPC to *L*-lysine production

Remark 1 Theorem 2.1 allows to consider the stabilization of systems that can only be stabilized by feedback that is discontinuous in the state (Fontes 2000), e.g. nonholonomic mechanical systems. This is possible since the optimal input signal is applied over the sampling time open-loop to the system. For such systems, however, it is in general rather difficult to determine a suitable terminal region and a terminal penalty term - for instance, the QIH-NMPC method cannot be used.

Remark 2 The use of a terminal inequality constraint leads to computational and feasibility advantages compared to the infinite horizon and zero terminal constraint approach. No zero terminal constraint must be met in finite time. The solution time necessary for solving the open-loop optimal control problem is decreased, since no “boundaryvalue” problem stemming from the zero terminal constraint must be solved. Furthermore, note that in NMPC it is in general not necessary to find always an optimal solutions of **Problem 1** in order to guarantee stability. Only a feasible solution leading to a decrease in the value function is necessary. This can be utilized to decrease the necessary on-line solution time and makes the practical application more robust (Ilkova et al., 2010).

Summarizing, the nominal stability question of NMPC in the state feedback case is well understood. Various NMPC approaches that guarantee stability exist.

NMPC involves the solution of an optimization problem on-line in order to determine the optimal inputs over a future time horizon.

The objective of the optimization is generally a weighted measure of the future tracking error and the difference between the predicted model outputs and the desired set points. In each time step an estimate of current disturbances is updated, the optimization is solved based on this new estimate (and an assumption regarding its future effect) and the first of the resulting optimal inputs is implemented.

The 15th hour is chosen as a first control point. As it may be noted that there is a diversion from the reference profile marked on the figure by “—”, accordingly the optimal profile is changed – NDP profile. The second point is at 22,5th hour. The third point is at 25,7th hour. The obtained control guarantees the robustness and stability of the biomass production. The local optimization at the 5th, 8th and 10th hour is made by the alternative rollout algorithms that has been written in *Section “Alternative rollout optimization method”*. The optimization criterion is maximal biomass concentration at the end of the process (criterion J).

NMPC to *L*-lysine production is shown in Fig. 4.

5. CONCLUSIONS

NMPC is developed for guarantee robustness to process disturbances for a fed-batch fermentation process for the *L*-lysine. The method is carried out with an aim control of disturbance of the optimal control variable (feeding rate). For local optimization of choice optimization hour, an algorithm is applied in order to find an optimal profile of the control variable.

For local optimization consider the approximate solution of discrete optimization problems using procedures that are capable of magnifying the effectiveness of any given heuristic algorithm through sequential application a *alternative rollout algorithms*, which are related to notions of policy iteration.

The developed control algorithm ensures maximal *L*-lysine at the end of the process and guarantees a feedback on disturbance as well as robustness to process disturbances.

REFERENCES

1. Hulse, J.D., Ellis, S.R. and Henderson, L.M. (1978) Carnitine biosynthesis-beta hydroxylation of trimethyllysine by an α -keto glutarate dependent mitochondrial dioxxygenase. *Journal of Biological Chemistry*, 253 (5), pp. 1654 - 1659.
2. <http://www.diagnose-me.com/treat/T110125.html>
3. http://www.springboard4health.com/notebook/proteins_carnitine.html
4. <http://www.gustrength.com/health/the-role-of-carnitine-in-disease>
5. Findeisen, R., Imsland, L., Allgower, F. and Foss, B.A. (2003) State and output feedback nonlinear model predictive control: An overview. *European Journal of Control*, 9, pp. 190-206.
6. Wu, G., Chong, E.P. and Givan, R. (2003) Congestion Control using policy rollout. 2nd IEEE CDC, Maui, Hawaii, pp. 4825-4830.
7. Anastassiadis, S. (2007) L-Lysine fermentation. *Recent pat. Biotechnology*, 1, pp. 11-24
8. Bertsekas, D.P. and Castanon, D.A. (1999) Rollout Algorithms for stochastic scheduling problems. *Heuristics*, 5, pp. 89-108.
9. Camacho, E. and Bordons F.C. (2004) *Model predictive control*. Springer-Verlag, London.
10. Soni, A. (2002) *A multi-scale approach to fed-batch bioreactor control*. University of Pittsburgh Press, PA.
11. Garcia, C., Prett, D. and Morari, M. (1989) Model predictive control, theory and practice – A survey, *Automatica*, 25, 335-348.
12. Allgower, F., Badgwell, T. A., Qin, J. S., Rawlings, J. B. and Wright, S. J. (1999) Nonlinear predictive control and moving horizon estimation - An introductory overview, *Advances in Control, Highlights of ECC'99*, pp. 391-449.
13. Chen, H. and Allgower, F. (1998) Nonlinear model predictive control schemes with guaranteed stability, In: Berber, R. and Kravaris, C. (Eds.), *Nonlinear Model Based Process Control*, Kluwer Academic Publishers, Dordrecht, pp. 465-494.
14. De Nicolao, G., Magni, L. and Scattolini, R. (2000) Stability and robustness of nonlinear receding horizon control, In: Allgower, F. and Zheng, A. (Eds.) *Nonlinear Predictive Control*, pp. 3-23.
15. Mayne, D.Q. and Rawlings, J. B., Rao, C.V. and Scokaert, P. O. M. (2000) Constrained model predictive control: stability and optimality. *Automatica*, 26(6), pp. 789-814.
16. Morari M. and Lee, J.H. (1997) Model predictive control: Past, present and future. *Computers & Chemical Engineering*, 23, pp. 667-682.
17. Bertsekas D.P., Tsitsiklis, J.N., and Wu C. (1997) Rollout Algorithms for Combinatorial Optimization. *Journal of Heuristics*, 3: 245–262
18. Ilkova, T. and Petrov M. (2010) Nonlinear model predictive control of biotechnological process for ethanol production. *Journal of International Scientific Publications: Materials, Methods & Technology*, 4(2), pp. 284-297.