Active Function Optimization with Parallel Stochastic-Duration Experiments

Javad Azimi AZIMI@EECS.OREGONSTATE.EDU
Alan Fern AFERN@EECS.OREGONSTATE.EDU
Xiaoli Fern XFERN@EECS.OREGONSTATE.EDU
Oregon State University, 1148 Kelley Engineering Center, Corvallis, Oregon 97331, USA

Abstract

Budgeted optimization involves optimizing an unknown function that is costly to evaluate by requesting a limited number of function evaluations at intelligently selected inputs. Typical problem formulations assume that experiments are selected one at a time with a limited total number of experiments, which fails to capture important aspects of many real-world problems. This paper defines a novel problem formulation with the following important extensions: 1) allowing for concurrent experiments; 2) allowing for stochastic experiment durations; and 3) placing constraints on both the total number of experiments and the total experimental time. We develop both offline and online algorithms for selecting concurrent experiments in this new setting and provide experimental results on a number of optimization benchmarks. The results show that our offline algorithm produce highly effective schedules compared to the proposed online algorithms.

1. Introduction

We study the optimization of an unknown function f by requesting n experiments, each specifying an input x and producing a noisy observation of $f(x)$. In practice, the function f might be the performance of a device parameterized by x. We consider the setting where running experiments is costly (e.g. in terms of time), which renders methods that rely on many function evaluations, such as stochastic search or empirical gradient methods, impractical. Bayesian optimization (BO) (Jones, 2001) addresses this issue by leveraging Bayesian modeling to maintain a posterior over the unknown function based on previous experiments. The posterior is then used to intelligently select new experiments so as to trade-off exploring new parts of the experimental space and exploiting promising parts.

Traditional BO follows a sequential approach where only one experiment is selected and run at a time. However, it is often desirable to select more than one experiment at a time so that multiple experiments can be run simultaneously to leverage parallel facilities. Recently, Azimi et al. (2010) proposed a batch BO algorithm that selects a batch of $k \geq 1$ experiments at a time. While this broadens the applicability of BO, it is still limited to selecting a fixed number of experiments at each step. As such, prior work on BO, both batch and sequential, completely ignores the problem of how to schedule experiments under fixed experimental budget and time constraints. Furthermore, existing work assumes that the durations of experiments are identical and deterministic, whereas in practice they are often stochastic.

Consider one of our motivating applications of optimizing the power output of nano-enhanced Microbial Fuel Cells (MFCs). MFCs (Bond & Lovley, 2003) use micro-organisms to generate electricity. Their performance depends strongly on the surface properties of the anode (Park & Zeikus, 2003). Our problem involves optimizing nano-enhanced anodes, where various types of nano-structures, e.g. carbon nano-wire, are grown directly on the anode surface. Because there is little understanding of how different nano-enhancements impact power output, optimizing anode design is largely guess work. Our original goal was to develop BO algorithms for aiding this process. However, many aspects of this domain complicate the application of BO. First, there is a fixed budget on the number of experiments that can run due to limited funds and a fixed time period for the project. Second,

Presented at *the ICML 2011 Workshop on Combining Learning Strategies to Reduce Label Cost, Bellevue, WA, USA.*

we can run multiple concurrent experiments, limited by the number of experimental apparatus. Third, the time required to run each experiment is variable due to the fact that each experiment requires the construction of a nano-structure with specific properties. Nano-fabrication is highly unpredictable and the amount of time to successfully produce a structure is quite variable. Clearly prior BO models fail to capture critical aspects of the experimental process in this domain.

We consider the following extended BO setting. It is possible to run up to l concurrent experiments in l available labs. The labs might correspond to experimental stations at one location or to physically distinct laboratories. Experiments are assumed to have stochastic duration that follow a known distribution p_d. The experimental time period is constrained by a time horizon h and there is a budget of n total experiments. The goal is to maximize the unknown function by selecting experiments and when to start them while satisfying the constraints.

We propose offline (Section 4) and online (Section 5) scheduling approaches for this problem, which aim to balance two competing factors. First, a scheduler should ensure that all n experiments complete within the horizon h, which encourages high concurrency. Second, we wish to select new experiments given as many previously completed experiments as possible, in order to make more intelligent experiment selections, which encourages low concurrency. We introduce a novel measure of the second factor, cumulative prior experiments (CPE) (Section 3), which our approaches aim to optimize. Our experimental results indicate the offline approach significantly outperform the online algorithms across a range of benchmark optimization problems.

2. Problem Setup

Let $\mathcal{X} \subseteq \Re^d$ be a d-dimensional compact input space, where each dimension i is bounded in $[a_i, b_i]$. An element of $\mathcal{X}$ is referred to as an *experiment*. We assume an unknown real-valued function $f : \mathcal{X} \rightarrow \Re$ that represents the expected value of the dependent variable after running an experiment. For example, $f(x)$ might correspond to the result of a wet-lab experiment described by x. Conducting an experiment x produces a noisy outcome $y = f(x) + \epsilon$, where ϵ is a random noise term. Our goal is to find an experiment $x \in \mathcal{X}$ that approximately maximizes f by requesting a limited number of experiments and observing their outcomes.

This paper extends the standard BO setting and enriches the model for requesting experiments as follows. First, we assume there are l labs, allowing for up to l concurrent experiments. Our model considers using a variable number of the l labs over time, unlike traditional BO, which always requests a fixed number of k (nearly always k=1) experiments. Second, we assume that experiment durations are independently and identically distributed according to a known probability density function p_d. For example, in a wet lab setting, experiment durations are often stochastic due to unpredictable setup times. In our experiments, we use the one-sided truncated normal distribution for p_d, though our approaches generalize to other distributions. This distribution denoted by $\mathcal{N}_{\text{tr}}(a, \mu, \sigma 2)$ specifies that the duration is normally distributed with mean μ and variance σ^2 conditioned on the fact that it is at least as large as a (the minimum experimental duration). Finally, our problem specifies a maximum number of total experiments n and a maximum time horizon h over which we can run experiments.

Given p_d, n, l, and h, our problem is to design a policy π for selecting when to start experiments and which ones to start. The inputs to π are the set of completed experiments and their outcomes, the set of currently running experiments with their elapsed running time, the number of free labs, and the remaining time till the horizon. Given this information, π must select a set of experiments (possibly empty) to start that is no larger than the number of free labs. Any run of the policy ends when either n experiments are completed or the time horizon is reached, resulting in a set $\mathcal{X}$ of n or fewer completed experiments. The objective is to obtain a policy with small *regret*, which is the expected difference between the optimal value of f and the best value of f for any experiment in $\mathcal{X}$. In theory, the optimal policy can be found by solving a POMDP with hidden state corresponding to the unknown function f. However, this POMDP is beyond the reach of any existing solvers. Thus, we focus on defining and comparing several principled policies that work well in practice, but lack optimality guarantees.

3. Objective Function

A policy for our problem must make two types of decisions: 1) scheduling when to start new experiments, and 2) selecting the specific experiments to start. In this work, we factor the problem based on these decisions and focus on approaches for scheduling experiments. We consider both offline (Section 4) and online (Section 5) schedulers, which at any time t will ask for $k \geq 0$ new experiments to be started. We assume a black box function *SelectBatch* for intelligently select-

ing the k experiments based on both completed and currently running experiments. The implementation of *SelectBatch* is described in Section 6.

Optimal scheduling to minimize regret appears to be computationally hard for non-trivial instances of *SelectBatch*. Further, we desire scheduling approaches that do not depend on the details of *SelectBatch*, but work well for any reasonable implementation. Thus, rather than directly optimizing regret for a specific *SelectBatch*, our scheduling approaches consider the following surrogate criteria. First, we want to finish all n experiments within the horizon h with high probability. Second, we would like to select each experiment based on as much information as possible, as measured by the number of previously completed experiments. These two goals are at odds, since maximizing the completion probability requires maximizing concurrency of the experiments, which in turn minimizes the second criterion.

To quantify the second criterion, consider a complete execution E of a scheduler. For any experiment e in E, let $\text{prior}(e, E)$ denote the number of experiments in E that *completed* before starting e. We define the *cumulative prior experiments (CPE)* of E to be the sum of the prior over all completed experiments in E, i.e. $\sum_{e \in E} \text{prior}(e)$. Intuitively, a scheduler that achieves a high expected CPE is desirable, since it measures the total amount of information (experimental outcomes) that is used by *SelectBatch* to make its decisions. CPE agrees with intuition when considering extreme schedules. A poor schedule that starts all n experiments at the same time (assuming enough labs) will have a minimum CPE of zero. Further, CPE is maximized by a schedule where all experiments are sequentially executed with no concurrency (assuming enough time).

To empirically examine the utility of CPE, we conducted experiments in a number of BO benchmark domains. For each domain, we used a set of 30 diverse schedulers, some started more experiments early on than later, and vice-versa, while others included random and uniform schedules. We measured the average regret achieved for each schedulers as well as the expected CPE of the executions. Figure 1 shows the results for two of the domains (other results are highly similar), where each point corresponds to the regret and CPE of a particular scheduler. We observe a clear and non-trivial correlation between regret and CPE, which provides empirical evidence that CPE is a useful measure to optimize. Further, as we will see in our experiments, the performance of our methods is also highly correlated with CPE.

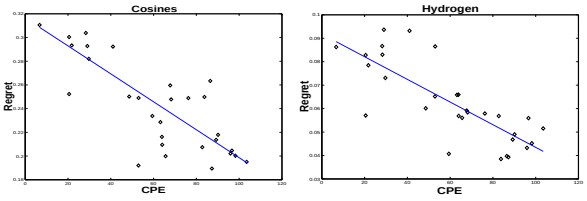

Figure 1. The correlation between CPE and regret for 30 different schedulers on two BO benchmarks.

4. Offline Scheduling

The goal of offline schedulers is assigning start times to all n experiments before the experimental process begins. This approach is often convenient in real experimental domains where it is useful to plan out a static equipment and personnel schedule for the duration of a project. Below we introduce a restricted class of offline scheduling algorithms, called *staged schedules*, for which we are able to characterize a solution in terms of CPE. In the full paper, we have also studies a richer class of offline schedules and associated algorithms. However, staged schedules appears to be highly competitive with those extensions.

4.1. Staged Schedules

A *staged schedule* is a sequence of tuples $\langle (n_1, d_1), \ldots, (n_N, d_N) \rangle$ that satisfies $0 < n_i \leq l$, $\sum_i d_i \leq h$, and $\sum_i n_i \leq n$. It defines a consecutive sequence of N experimental stages. Starting with stage $i = 1$, the i^{th} stage begins by starting up n_i new experiments, as selected by *SelectBatch* using the most recent information, and ends after a duration of d_i, upon which stage $i + 1$ starts. While our approach will try to ensure that most experiments finish within their stage, experiments are never terminated and hence might run longer than their stages duration. If, because of this, at the beginning of stage i there are not n_i free labs, then the experiments that cannot start will start as soon as labs free up.

An execution E of a staged schedule S results in a set of experiments that started and finished at specific times. We say that E is *safe* if the duration of each experiment is no greater than the duration specified for its stage in S. We say that a staged schedule S is *p-safe* if with probability at least p an execution of S is safe. It is straightforward to calculate the CPE of any safe execution of a staged schedule S, which (slightly abusing notation) we denote as $\text{CPE}(S)$ and is given by:

$$\text{CPE}(S) = \sum_{i=2}^{N} n_i \sum_{j=1}^{i-1} n_j \tag{1}$$

Typical applications will use relative high values of p,

since otherwise experimental resources would likely be wasted, and thus with high probability we can expect the CPE of an execution of S to be equal to CPE(S). It turns out that for any fixed number of stages N, it is easy to characterize the schedules that maximize CPE(S).

A staged schedule is defined to be *uniform* if for any i and j, $|n_i - n_j| \leq 1$. That is, a uniform schedule is one where the batch sizes across stages may differ by at most a single experiment. The uniform schedules are exactly those that maximize CPE.

Proposition 1. *For any number of experiments n and labs l, let $\mathcal{S}_N$ be the set of corresponding N stage schedules, where N must be at least $\lceil n/l \rceil$. For any $S \in \mathcal{S}_N$, CPE(S) = $\max_{S' \in \mathcal{S}_N}$ CPE(S') if and only if S is uniform.*

It is easy to verify that for a given n and l, an N stage uniform schedule achieves a strictly higher CPE than any uniform $N-1$ stage schedule (and hence any $N-1$ stage schedule). This implies that, with respect to CPE, we should prefer uniform schedules that use the maximum number of stages allowed by the p-safeness restriction. This motivates us to solve the following problem: *Find a p-safe uniform schedule with maximum number of stages.*

Our approach, outlined in Algorithm 1, considers N stage schedules in order of increasing N, starting at the minimum possible number of stages $N = \lceil n/l \rceil$ for running all experiments. For each value of N, the call to *MaxProbUniform* computes a uniform schedule S that has the highest probability of a safe execution, among all N stage uniform schedules. If the resulting schedule is p-safe then we consider $N+1$ stages. Otherwise, there is no uniform N stage uniform schedule that is p-safe and we return a uniform $N-1$ stage schedule, which was computed in the previous iteration.

Algorithm 1 Algorithm for computing a p-safe uniform schedule with maximum number of stages.

Input: number of experiments (n), number of labs (l), horizon (h), safety probability (p)
Output: A p-safe uniform schedule with maximum number of stages

```
N = ⌈n/l⌉, S ← null
loop
  S' ← MaxProbUniform(N, n, l, h)
  if S' is not p-safe then
    return S
  end if
  S ← S', N ← N + 1
end loop
```

It remains to describe the computation of the *MaxProbUniform* function, which for a given value of N, n, l, and h, must compute a uniform N stage schedule $S = \langle (n_i, d_i) \rangle$ that maximizes the probability of a safe execution. First, it is straightforward to compute the n_i values for S. In particular, any N stage uniform schedule has $N' = (n \mod N)$ stages with $n' = \lfloor n/N \rfloor + 1$ experiments and $N - N'$ stages with $n' - 1$ experiments. Furthermore, the ordering of these stages in the schedule does not matter. In particular, the probability of safe execution is invariant to swapping of stages in a schedule, since we assume i.i.d. distributions on the durations of each experiment.

The *MaxProbUniform* problem is now reduced to computing the durations d_i of S that maximize the probability of safeness for this choice of n_i. For this we will make the assumption that we will only be interested in schedules that have at least 0.5 probability of safe executions. This assumption is quite reasonable in practice and simplifies our analysis.

Lemma 1. *For any $p > 0.5$, if there is a p-safe N stage schedule $S = \langle (n_i, d_i) \rangle$, then there is a p-safe N stage schedule $S' = \langle (n_i, d'_i) \rangle$ such that if $n_i = n_j$ then $d'_i = d'_j$.*

We omit the proof for space reasons, which follows from standard constrained optimization theory and the fact that the $p > 0.5$ assumption implies we will be operating in the concave range of the truncated normal CDF. Since a uniform schedule has only two values of n_i, either n' or $n' - 1$, this lemma implies that we can restrict consideration to schedules with only two durations, d' for stages with n' experiments and d'' for stages with $n' - 1$ experiments. Further since all durations must sum to h, d' and d'' are deterministically related by: $d'' = \frac{h - d' \cdot N'}{N - N'}$. Based on this, for any value of d' it is easy to compute the probability of the uniform schedule using durations d' and d'':

$$\left[P_d(d')\right]^{N' \cdot n'} \left[P_d\left(\frac{h - d' \cdot N'}{N - N'}\right)\right]^{(N-N') \cdot (n'-1)} \quad (2)$$

Based on the $p > 0.5$ assumption we know d' and d'' are greater than μ which implies $\mu < d' < \frac{h-(N-N')\mu}{N'}$. We compute *MaxProbUniform* by maximizing Equation 2 with respect to d' over this range and using the corresponding duration for d''. Putting everything together we get the following result.

Theorem 1. *If we compute* MaxProbUniform *by maximizing Equation 2 over $\mu < d' < \frac{h-(N-N')\mu}{N'}$, then for any $p > 0.5$, if a p-safe uniform schedule exists, then Algorithm 1 returns a p-safe uniform schedule with maximum number of stages.*

In our implementation, we approximately maximize Equation 2 by evaluating values of d' over a dense discretization.

5. Online Scheduling Approaches

We now consider two baseline online scheduling approaches, where the start times of experiments are selected online, rather than via an offline scheduling process as in the previous section. These approaches are primarily intended as baselines for comparison with our offline schedule. In future work, we plan to explore more advanced online scheduling algorithms.

Online Fastest Completion Policy (OnFCP). This policy simply tries to finish all of the n experiments as quickly as possible, serving as a basic baseline. To do this, the policy keeps all l labs busy as long as there are experiments left to run. Specifically whenever a lab (or labs) becomes free the policy immediately uses SelectBatch with the latest information to select new experiments that are immediately started. This policy will achieve a low value of expected CPE since it maximizes concurrency as much as possible, which suggest that the policy will perform relatively poorly.

Online Minimum Eager Lab Policy (OnMEL). The problem with OnFCP is that it does not use the full time horizon, even when doing so would allow for much less concurrency, and hence larger CPE values. The OnMEL policy is simply a version of OnFCP but restricted to use only k labs, where k is the minimum number of labs required in order to guarantee with probability at least p all n experiments complete within the horizon.

To compute the number of labs k, for increasing k values starting with $k = 1$, we decide if the probability of finishing all n experiments with k labs within horizon h is at least p. If so, we output that value of k and otherwise increment k. One approach for estimating the probability of completion is via Monte-Carlo simulation. Rather, in this paper we use a closed form approximation of the CDF of sums of truncated normal random variables (Francis, 1946).

6. Experiments

Implementation of *SelectBatch*. Given the sets of completed experiments $\mathcal{O}$ and currently running experiments $\mathcal{A}$, SelectBatch must select k new experiments. Our implementation of SelectBatch is based on a recent batch BO algorithm (Azimi et al., 2010), which greedily selects a batch of k experiments considering only $\mathcal{O}$. We modify this greedy algorithm to also consider $\mathcal{A}$ by forcing the selected batch to always include the currently running experiments, and greedily selecting k additional experiments. SelectBatch makes its decisions based on a posterior over the unknown function f. We use Gaussian Process with the RBF kernel $K(x, x') = \exp\left(-\frac{1}{2w} \parallel x - x' \parallel^2\right)$, and set the kernel width w to $0.01 \sum_{i=1}^{d} l_i$ where l_i is the input space length in dimension i.

6.1. Benchmark Functions

We evaluate our scheduling policies using 4 well-known synthetic benchmark functions (shown in Table 1).We also consider two real-world benchmark functions *Hydrogen* and *FuelCell* over $[0, 1]^2$, from (Azimi et al., 2010). The Hydrogen data set is the result of a study on biosolar hydrogen production (Burrows et al., 2009), where the goal was to maximize hydrogen production of a particular bacteria by optimizing PH and Nitrogen levels. The FuelCell data set was collected as part of of our motivating application mentioned in Section 1. In both cases, the benchmark function was created by fitting regression models to the available data.

Table 1. Benchmark Functions

Cosines(2)	$1 - (u^2 + v^2 - 0.3cos(3\pi u) - 0.3cos(3\pi v))$ $u = 1.6x - 0.5, v = 1.6y - 0.5$
Rosenbrock(2)	$10 - 100(y - x^2)^2 - (1 - x)^2$
Michalewicz(5)	$-\sum_{i=1}^{5} \sin(x_i).\left(\sin\left(\frac{i.x_i^2}{\pi}\right)\right)^{20}$
Shekel(4)	$\Sigma_{i=1}^{10} \frac{1}{\alpha_i + \Sigma_{j=1} 4(x_j - A_{ji})^2}$ $\alpha_{1\times10}$, $A_{4\times10}$ are constants

6.2. Evaluation

We consider a p-safeness guarantee of $p = 0.95$, the number of available labs l is 10 and let the total number of experiments $n = 20$. For p_d, we set $a = 0$, $\mu = 1$, $\sigma^2 = 0.1$. Additive normal noise is used with zero mean and variance 0.01. Finally, we consider 2 time horizons h of 6 and 4.

Given l, n and h, to evaluate a policy π using a function f and a set of initial observed experiments, we execute the policy, resulting in a set $\mathcal{X}$ of n or fewer completed experiments. We measure the regret of π as the difference between the optimal value of f (known for all of our functions) and the best value of f for any experiment in $\mathcal{X}$.

6.3. Results

Tables 2-3 show the results of our proposed offline and online scheduling approaches. We also include,

as a reference point, the result of the *un-constrained* sequential policy (i.e., selecting one experiment at a time) using SelectBatch. Note that this can be viewed as an effective upper bound on the optimal performance that can be achieved by any constrained scheduler because it ignores the time horizon constraint($h = \infty$). The values in the tables correspond to regrets (smaller values are better) achieved by each policy, averaged across 100 independent runs with the same initial points (5 points for 2-d functions and 20 points for the rest) for all policies in each run.

Comparing the proposed staged offline algorithm with the two online algorithms (OnFCP and OnMEL), it can be seen that the staged algorithm is significantly better. This may seem surprising at first because online policies should offer more flexibility than a fixed offline schedule. However, the offline schedule is more intelligent in terms of waiting for experiments to complete before starting up new experiments, which can result in improved CPE values. To see this, Table 4 gives the average CPE values of each policy. Both OnFCP and OnMEL yield significantly lower CPEs compared to the offline algorithm, which correlates with their significantly larger regrets. Considering the online algorithms, OnMEL outperforms OnFCP consistently. Interestingly, as the difference between the CPE of OnMEL and OnFCP increases via different horizons, the difference between the regret of OnMEL and OnFCP also increases. This simply shows the correlation between the CPE and regret.

Table 2. Long horizon, h=6

Function	$h = \infty$	OfStaged	OnFCP	OnMEL
Cosines	0.140	0.150	0.339	0.217
Fuel Cell	0.155	0.160	0.239	0.215
Hydrogen	0.024	0.027	0.097	0.048
Rosen	0.004	0.005	0.010	0.013
Michal	0.450	0.458	0.541	0.522
Shekel	0.406	0.524	0.661	0.573

Table 3. Short horizon, h=4

Function	$h = \infty$	OfStaged	OnFCP	OnMEL
Cosines	0.140	0.206	0.339	0.304
Fuel Cell	0.155	0.178	0.239	0.242
Hydrogen	0.024	0.057	0.097	0.081
Rosen	0.004	0.005	0.010	0.008
Michal	0.450	0.484	0.541	0.510
Shekel	0.406	0.588	0.661	0.655

Table 4. The CPE values of different policies.

Setting	$h = \infty$	OnFCP	OnMEL	OfStaged
h=4	190	55	66	100
h=6			120	133

7. Summary and Future Work

Motivated by real-world applications we introduced a novel setting for Bayesian optimization that incorporates a budget on the total time and number of experiments and allows for concurrent, stochastic-duration experiments. We considered offline and online approaches for scheduling experiments in this setting, relying on a black box function to intelligently select specific experiments at their scheduled start times. These approaches aimed to optimize a novel objective function, Cumulative Prior Experiments (CPE), which we empirically demonstrate to strongly correlate with performance on the original optimization problem. Our offline scheduling approache significantly outperformed some natural online baselines.

In future work, will extend the class of schedules and consider computing p-safe schedules that maximize CPE. We also plan to consider alternatives to CPE, which, for example, incorporate factors such as diminishing returns. Second, we plan to extend our online scheduling approaches, by attempting to directly maximize the CPE while ensuring p-safeness. In particular, we will consider approached based on replanning with an offline scheduler. We also plan to study further extensions to the experimental model for BO and also for active learning. For example, taking into account varying costs and duration distributions across labs and experiments. In general, we believe that there is much opportunity for more tightly integrating scheduling and planning algorithms into BO and active learning to more accurately model real-world conditions.

Acknowledgment

The authors acknowledge the support of the NSF under the grants IIS-0905678.

References

Azimi, Javad, Fern, Alan, and Fern, Xiaoli. Batch bayesian optimization via simulation matching. In *NIPS*, 2010.

Bond, D. and Lovley, D. Electricity production by geobacter sulfurreducens attached to electrodes. *Applications of Environmental Microbiology*, 69:1548–1555, 2003.

Burrows, Elizabeth H., Wong, Weng-Keen, Fern, Xiaoli, Chaplen, Frank W.R., and Ely, Roger L. Optimization of ph and nitrogen for enhanced hydrogen production by synechocystis sp. pcc 6803 via statistical and machine learning methods. *Biotechnology Progress*, 25: 1009–1017, 2009.

Francis, V. J. On the distribution of the sum of n sample values drawn from a truncated normal population. *Supplement to the Journal of the Royal Statistical Society*, 8(2):223–232, 1946.

Jones, D. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

Park, D. and Zeikus, J. Improved fuel cell and electrode designs for producing electricity from microbial degradation. *Biotechnol.Bioeng.*, 81(3):348–355, 2003.