Parallelization of the MARS Value Function Approximation in a Decision-Making Framework for Wastewater Treatment

Julia C. C. Tsai
Krannert School of Management
Purdue University
1310 Krannert Building
West Lafayette, IN 47907-1310

Victoria C. P. Chen (correspondence author)
Department of Industrial and Manufacturing Systems Engineering
Campus Box 19017
University of Texas at Arlington
Arlington, TX 76019
Phone: (817) 272-2342
Fax: (817) 272-3406
Email: vchen@uta.edu

Eva K. Lee and Ellis L. Johnson
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332

COSMOS Technical Report 03-02

Abstract

In this paper, a parallelized version of multivariate adaptive regression splines (**MARS**, Friedman 1991) is developed and utilized within a decision-making framework (DMF) based on an OA/MARS continuous-state stochastic dynamic programming (SDP) method (Chen et al. 1999). The DMF is used to evaluate current and emerging technologies for the multi-level liquid line of a wastewater treatment system, involving up to eleven levels of treatment and monitoring ten pollutants moving through the levels of treatment. At each level, one technology unit is selected out of set of options which includes the empty unit. The parallel-MARS algorithm enables the computational efficiency to solve this ten-dimensional SDP problem using a new solution method which employs orthogonal array-based Latin hypercube designs and a much higher number of eligible knots.

Key Words: dynamic programming, orthogonal arrays, Latin hypercubes, regression splines, parallel computing.

1 Introduction

The goal of a city's wastewater infrastructure is to control the disposal of urban effluents, so as to achieve clean water. Our objective in this application to wastewater treatment is to evaluate

current and emerging technologies via a decision-making framework (**DMF**) based on continuous-state stochastic dynamic programming (**SDP**, see Bellman 1957, Puterman 1994, Bertsekas 2000, for general background). In general, our DMF can provide a forum in which future options may be explored without subjecting humankind to experimental attempts at environmental remediation.

Solution methods for SDP can be extremely computationally intensive and have only become practical due to recent advances in computational power. The most promising high-dimensional continuous-state SDP solution method is OA/MARS (Chen et al. 1999), which utilizes a *statistical modeling* approach by employing orthogonal array (**OA**, Chen 2001) experimental designs and multivariate adaptive regression splines (**MARS**, Friedman 1991).

Chen et al. (1999) and Chen (1999) solved continuous-state inventory forecasting SDP problems with up to nine dimensions in the state space. Application to a ten-dimensional water reservoir SDP problem appears in Baglietto et al. (2001), and preliminary results for a ten-dimensional wastewater treatment system appear in Chen et al. (2000) and Chen et al. (2001). To-date, these are the largest continuous-state SDP application problems in the literature; however, there is great potential to solve much higher-dimensional problems. In this paper, we explore two new variants of the OA/MARS statistical modeling approach:

1. Application of a parallelized version of MARS to reduce computational time and comparisons to the serial version of MARS.

2. Utilization of OA-based Latin hypercube designs (**OA-LHD**, Tang 1993), permitting a larger eligible set of knots for the MARS approximation.

It is expected that the wastewater treatment SDP problem will require more accurate univariate modeling and fewer interactions than the inventory forecasting SDP problems, which involved a significant portion of three-way interactions. Thus, we anticipated the need for more flexibility in MARS knot selection than was permitted with pure OAs. However, the serial version of MARS was already requiring excessive computational effort with the OA eligible knot set, hence, a parallelization of the MARS forward stepwise search for basis functions was developed.

Parallelization of a B-splines variant of the MARS algorithm (BMARS) was presented by Bakin et al. (2000) in the context of mining very large data sets (e.g., one million data points). Their approach utilized *data parallelism* to speed up the calculation of a summation over the data points, which takes place when orthonormalizing the basis functions. Parallelization is necessary because their orthonormalization is conducted for *every eligible* basis function in the forward stepwise search. By contrast, the original MARS conducts the orthonormalization *only* on the basis functions resulting from the search. The parallel-MARS algorithm presented in this paper is based on the original MARS algorithm and is more generally applicable than BMARS.

The wastewater treatment system in this paper was developed by Chen and Beck (1997), and the database of technologies was classified by Chen (1993). Appropriate parameter settings for the purposes of optimization within the DMF were developed by the authors. Details on the wastewater treatment system are given in the next section, and the SDP statistical modeling process is described in Section 3. Parallelization of MARS and background on the message-passing interface (**MPI**) is described in Section 4. Finally, the DMF results are presented in Section 5 and concluding remarks appear in Section 6.

2 Wastewater Treatment System

The wastewater infrastructure and database of technologies utilized in this research is based on the system presented by Chen and Beck (1997), which focuses on the treatment of domestic wastewater

involving both liquid and solid treatment processes with several levels. In this paper, we address only the liquid line, which is shown in Figure 1. The levels represent the different stages of treatment and are numbered in the order in which they are encountered by the wastewater. At each level, a choice must be made as to the type of technology (unit process) to employ. Included among these choices is an "empty unit" which, if selected, indicates that no treatment is performed at that level. Although the diagram in Figure 1 does not show dependencies, there are a few cases in which a unit at a later stage can only be utilized if a certain unit was employed at an earlier stage for a required pretreatment.

2.1 State and Decision Variables

The *state variables* represent the state of the system as it moves through the levels of treatment. For the liquid line of our wastewater treatment system, we monitored ten state variables: (1) chemical oxygen demand substances (**CODS**), (2) suspended solids (**SS**), (3) organic-nitrogen (**orgN**), (4) ammonia-nitrogen (**ammN**), (5) nitrite-nitrogen (**nitN**), (6) total phosphorus (**totP**), (7) heavy metals (**HM**), (8) synthetic organic chemicals (**SOCs**), (9) pathogens, and (10) viruses. The ten-dimensional state vector, denoted by $\boldsymbol{x}_t$, contains the quantities of the above ten state variables remaining in the liquid upon entering the t-th level of the system in Figure 1. These state variables monitor the cleanliness of the liquid at the different levels of the liquid line. Note that all the state variables are continuous.

The *decision variables* are the variables we control. In the case of wastewater treatment, the decision is which technology unit to select in each level of the liquid line. Mathematically, we denote this decision variable by u_t, where the value of u_t essentially indexes the selected technology unit. For a set of U_t possible technologies in level t, the decision variable u_t takes on a value in $\Gamma_t = \{0, 1, 2, \ldots, U_t\}$, where $u_t = 0$ selects the "empty unit."

2.2 Transition Function

The *transition function* for level t determines how the state variables change upon exiting level t. The performance of a particular technology unit is based on the removal of pollutants represented by the ten state variables. Each technology unit in level t determines a different transition from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$ based on performance parameters specified in the database by Chen (1993). We denote the transition function by $f_t(\boldsymbol{x}_t, u_t, \boldsymbol{\epsilon}_{t,u_t})$, where $\boldsymbol{\epsilon}_{t,u_t}$ is the stochasticity placed on the performance parameters. The largest number of parameters for a single technology unit was 25. Since nothing is known about the appropriate probability distributions to represent the stochasticity, only the ranges of the performance parameters are specified, with narrower ranges assigned to well-known technologies and wider ranges assigned to newer, emerging technologies.

2.3 Objectives and Constraints

The basic *constraints* are stringent clean water targets that are specified for the effluent exiting the final level of the liquid line. In addition, to avoid the extreme situation of liquid too polluted to be processed by any technologies available in a specific level (caused by selecting the "empty unit" too often in earlier levels), constraints are added on the cleanliness of the influent entering each level. These range limits on the state variables, shown in Table 1, define the state space for each level of the liquid line. Subsequent lower limits are calculated based on the highest possible pollutant removal in each level. The upper limits are based on the smallest possible pollutant removal, assuming the "empty unit" was not selected.

The primary *objective function* is economic cost, capital and operating, which is minimized. The costs are calculated for each selected technology unit and are subject to stochasticity modeled in the same manner as the performance parameters of each unit. Attainment of the constraints is achieved via a penalty function added to the primary cost objective. The same penalty function was utilized to achieve cleanliness targets and to maintain state space limits. Due to the subsequently extreme penalty on state space limits, these were exceeded very rarely.

Mathematically, the target penalty functions are formulated as quintic functions of the same form as those used in Chen et al. (1999), which involve three "knots," k_-, k, and k_+. The lowermost knot k_- is the target value at which the function hits zero (i.e., zero penalty for being below target). The other two knots are defined as $k = k_- + \Delta$ and $k_+ = k + \Delta$, where Δ is determined such that the upper knot, k_+, coincides with the maximum of the effluent. The exact form of the penalty function is not important, as long as it serves the purpose of penalizing for violating targets, and this form was chosen to facilitate modeling by MARS. Table 2 provides the state variable ranges of the effluent exiting the liquid line, target values, Δ values, and penalty coefficients.

3 Statistical Modeling Within The DMF

As stated earlier, our DMF is based on a SDP formulation. In this section, we present the SDP formulation and the statistical modeling process necessary to acquire a numerical SDP solution for the continuous-state case. The objective of the wastewater treatment SDP is to minimize "costs" over the $T = 11$ levels of the liquid line, i.e., to solve

$$\begin{array}{lll} \min\limits_{u_1,\ldots u_T} & E\left\{\sum\limits_{t=1}^{T} c_t(\boldsymbol{x}_t, u_t, \boldsymbol{\epsilon}_{t,u_t})\right\} & \\ \text{s.t.} & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, u_t, \boldsymbol{\epsilon}_{t,u_t}), \text{ for } t = 1,\ldots,T-1 \text{ and} & \\ & \boldsymbol{x}_t \in S_t, \text{ for } t = 1,\ldots,T & \\ & u_t \in \Gamma_t, \text{ for } t = 1,\ldots,T & \end{array} \tag{1}$$

where $\boldsymbol{x}_t$ is the state vector (cleanliness of the liquid), u_t is the index of the selected technology unit, $\boldsymbol{\epsilon}_{t,u_t}$ is the stochastic component on the performance parameters of unit u_t, $\boldsymbol{x}_{t+1}$ is determined by the transition function $f_t(\cdot)$, S_t contains the range limits on the state variables, Γ_t contains the indices of the available technology units, and the cost function $c_t(\cdot)$ contains both the economic and penalty costs.

The *future value function* provides the optimal cost to operate the system from level t through T as a function of the state vector $\boldsymbol{x}_t$. This is written in summation form as

$$\begin{array}{rlll} F_t(\boldsymbol{x}_t) = & \min\limits_{u_t,\ldots u_T} & E\left\{\sum\limits_{\tau=t}^{T} c_\tau(\boldsymbol{x}_\tau, u_\tau, \boldsymbol{\epsilon}_{\tau,u_\tau})\right\} & \\ & \text{s.t.} & \boldsymbol{x}_{\tau+1} = f(\boldsymbol{x}_\tau, u_\tau, \boldsymbol{\epsilon}_{\tau,u_\tau}), \text{ for } \tau = t,\ldots,T-1 \text{ and} & \\ & & \boldsymbol{x}_\tau \in S_\tau, \text{ for } \tau = t,\ldots,T & \\ & & u_\tau \in \Gamma_\tau, \text{ for } \tau = t,\ldots,T & \end{array}$$

and written recursively as

$$\begin{array}{rlll} F_t(\boldsymbol{x}_t) = & \min\limits_{u_t} & E\{c(\boldsymbol{x}_t, u_t, \boldsymbol{\epsilon}_{t,u_t}) + F_{t+1}(\boldsymbol{x}_{t+1})\} & \\ & \text{s.t.} & \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, u_t, \boldsymbol{\epsilon}_{t,u_t}) & \\ & & \boldsymbol{x}_t \in S_t & \\ & & u_t \in \Gamma_t\ . & \end{array} \tag{2}$$

Given a finite number of levels T, the basic SDP problem is to find $F_1, \ldots, F_T$. Then u_t can be found for any given $\boldsymbol{x}_t$ by standard numerical minimization of (2). The recursive formulation permits the SDP backward solution algorithm, first by assuming $F_{T+1} \equiv 0$, then solving in order for F_T through F_1 (see description in Chen 1999).

The exact solution for the future value functions F_T through F_1 becomes impossible in the presence of continuous state variables (except in simple problems where an analytical solution might be derived). This is where the statistical modeling process introduced by Chen et al. (1999) is employed to estimate these functions. The key is *discretization* of the state space via a finite set of points, which are then used to construct a representation of the continuous future value functions. In the statistical perspective, the state space discretization is equivalent to an experimental design, such as an OA, and the approximation of the future value function is achieved by fitting a statistical model, such as MARS.

Since $\boldsymbol{\epsilon}_{t,u_t}$ in the minimization of (2) is based on a continuous distribution, we must also use discrete approximations to estimate the expected value. For our implementation, we sampled 51 points from a Uniform$(0, 1)$ distribution, then combined that with a Plackett-Burman screening design with 48 points for the edge values $\{0, 1\}$ to ensure that the SDP accounted for the extreme cases (Plackett and Burman 1946).

4 Parallel Multivariate Adaptive Regression Splines

4.1 Background on Parallel Computing

Parallel computing generally refers to solving intensive computational problems via parallel computers. A parallel computer is a set of processors working cooperatively to solve a computational problem. Parallelism has drawn great interest in large scale operations research application problems. Lee et al. (1995) and Linderoth et al. (1999) presented a parallel implementation of branch-and-bound mixed 0/1-integer programming and a parallel linear programming based heuristic in solving large scale set-partitioning problems. Graph partitioning problems were shown to be computationally efficient via a parallelization approach (Bhargava et al. 1993) and the visualization software developed by West et al. (1994) for scientific analysis of high resolution, high volume datasets.

Message-passing is probably the most widely-used paradigm for expressing parallel algorithms (Gropp and Lusk 1994, Foster 1995). The Message-Passing Interface (**MPI**) is a standard message-passing library interface developed by the MPI Forum, a broadly-based group of parallel computer vendors, parallel programmers, and application scientists. Zhuang and Zhu (1994) demonstrated parallelization in a reservoir simulator using MPI. Beyond the basic send and receive capabilities, some MPI features, such as communicators, topologies, communication modes and single-call collective operations (Snir et al. 1998, Gropp et al. 1999) were entertained to parallelize the serial-MARS algorithm.

4.2 The MARS Algorithm

Multivariate adaptive regression splines (MARS), introduced by Friedman (1991), provide a flexible modeling method for high-dimensional data. The model is built by taking the form of an expansion in product spline basis functions, where the number of basis functions as well as the parameters associated with each one are automatically determined by the data. MARS uses the multiple regression model

$$E[y_i] = g(x_{j1}, x_{j2}, ..., x_{jn}), \;\; j = 1, ..., N \tag{3}$$

where n is the number of covariates $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, N is the number of data points, and the "regression function" $g(\cdot)$ is smooth but otherwise arbitrary. In the SDP statistical modeling process of Section 3, the set of covariates corresponds to the set of continuous state variables, and the data points correspond to state space discretization points. There is assumed to be very little noise associated with the modeling of the future value functions in SDP; although, further studies on this subject are warranted. The MARS procedure for estimating the function $g(\cdot)$ consists of (i) a forward stepwise algorithm to select certain spline basis functions, (ii) a backward stepwise algorithm to delete basis functions until the "best" set is found, and (iii) a smoothing method which gives the final MARS approximation a certain degree of continuity.

To give MARS continuity and a continuous first and second derivative at the side knots, a piecewise-linear MARS approximation with quintic functions was derived (Chen et al. 1999). MARS is an *adaptive* procedure because the selection of basis functions is data-based and specific to the problem at hand. We do not employ the backward stepwise algorithm with SDP due to preliminary results indicating minimal benefit for a large additional computational burden.

The forward stepwise algorithm is the most computationally expensive component of MARS, and in the next section we describe its parallelization. It is reprinted as Algorithm 1 and described below to facilitate this discussion. The relevant notation is as follows: $M_{\max}$ is the maximum number of basis functions, which is used to terminate MARS; $B_m(\cdot)$ is the m-th basis function; L_m is the number of splits that gave rise to $B_m(\cdot)$; $v(l, m)$ indexes the covariate in the l-th split of the m-th basis function; and k indexes the eligible knot locations (at data values).

In line S1, Algorithm 1 begins with the constant basis function, $B_1(\boldsymbol{x}) = 1$, and initializes the counter M. Within the M-loop beginning on line S2, basis functions $B_M(\cdot)$ and $B_{M+1}(\cdot)$ are added. Beginning on line S3, the m-loop searches through the $(M-1)$ basis functions that have already been added for the best one to "split." Univariate basis functions "split" the constant basis function at a knot k for covariate x_v in the form of truncated linear functions, $b^+(x_v - k) = [+(x_v - k)]_+$ and $b^-(x_v - k) = [-(x_v - k)]_+$, where $[q]_+ = \max\{0, q\}$. Interaction basis functions are created by "splitting" (multiplying) an existing basis function $B_m(\cdot)$ with a truncated linear function involving a new covariate. Both the existing basis function and the newly created interaction basis function are used in the MARS approximation. Lines S3–S5 loop through the possible choices for basis function (m), covariate (v), and knot (k) in selecting the next two basis functions (M and $M+1$) to add. The potential "splits" are calculated in line S6.

The lack-of-fit (**lof**) criterion in line S7 is based on squared error loss, and is used to compare the possible basis functions. In line S8, the indices m, v, and k are stored for the "split" that currently yields the smallest lof. The algorithm stops when $M_{\max}$ basis functions have been accumulated, where $M_{\max}$ is a user-specified constant. The MARS approximation approaches interpolation as the number of basis functions increases, but there is a tradeoff between $M_{\max}$ and computational time.

In our MARS, we include an option to limit the number of "splits" (L_m) for each basis function. This permits additive modeling or exploration of a limited order of interactions. In addition, our MARS program utilized the orthogonalized version of the forward stepwise algorithm, suggested in the Rejoinder of citetmars. For simplicity, the parallel-MARS forward stepwise algorithm will be presented based on the version reprinted as Algorithm 1.

4.3 Parallelization of MARS

Inside Algorithm 1, the m-loop, beginning on line S3, searches through all the $(M-1)$ previously-added basis functions, and for each basis function m, searches over covariates v and knots k for the best "split." Each of the $(M-1)$ search procedures within the m-th loop is independent of

the others, and hence, the $(M-1)$ search procedures indexed by m can be conducted in parallel. To achieve the greatest computational improvement, the outermost parallelizable loop is selected. For Algorithm 1, it is the m-loop. The number of processors specified by a user, P, was used to assign parallel jobs efficiently to the processors. Algorithm 2 shows the structure of the resulting parallel-MARS algorithm. Where relevant, reference is made to line numbers in Algorithm 1.

MPI function MPI_Type_struct was used to send a collection of data items of various elementary and derived types as a single data type. Thus, data were grouped as blocks with a data type associated with each block and a count to represent the number of elements in each block. With this block structure, message-passing, using the basic message-passing functions, such as MPI_Send, MPI_Receive and MPI_Bcast, is simplified. In Algorithm 2, preprocessing of the parallelization is on line P2, where the master processor sends the necessary input for MARS modeling to all the slave processors. In lines P5–P11, the main body of parallel-MARS distributes the $(M-1)$ search procedures among the processors. In line P7, the loops over covariate v and knot k are conducted in parallel for different basis functions B_{m_i}. The best lof value and corresponding indices are stored separately on each processor in line P8, and then later in line P15, the overall best is identified.

The number of jobs assigned to each processor is set as equal as possible to minimize computational time. The master processor C_0 runs jobs $m_0 = 1,\ 1+P,\ 1+2P,\ \ldots$ and stores the local best in $\text{lof}_0^*, {m_0}^*, {v_0}^*, {k_0}^*$; the slave processor C_1 runs $m_1 = 2,\ 2+P,\ 2+2P,\ \ldots$ and stores its local best, $\text{lof}_1^*, {m_1}^*, {v_1}^*, {k_1}^*$; C_2 runs $m_2 = 3,\ 3+P,\ 3+2P,\ \ldots$; etc. The calculation of the new basis functions and the update of M in lines P16–P17 is the same as serial-MARS, except that the updated information must be broadcasted to all the slave processors in preparation for the next parallel loop.

In this study, all computational experiments were performed on a cluster of machines consisting of seventeen 550MHz eight-processor Pentium III Xeon computers, resulting in up to 136 processors available for parallel computing. All machines were linked via Gigabit Ethernet and used a RedHat Linux 6.2 operating system. Parallelism was implemented with Version MPICH 1.2, developed at Argonne National Laboratory to facilitate parallel programs running across cluster machines and exchanging information between different processors using message-passing procedures.

5 Results for the Wastewater Treatment Application

For discretization of the state space, as described in Section 3, OA-LHDs (Tang 1993) were constructed from OAs of strength two (Owen 1992). These OA-LHDs combine the LHD-property of having all N levels in each dimension represented (once) in the experimental design and the OA-property of balance in bivariate projections (for strength two). In this section, we explore the reduction in computation due to the parallel-MARS algorithm, as applied to the wastewater treatment system, and we present the results of the DMF.

5.1 Computational Benefits of Parallel-MARS

For our computational experiments, we limited MARS to permit at most three "splits" per basis function ($L_m = 3$), and we varied:

- N = number of discretization points,
- $M_{\max}$ = maximum number of basis functions,
- K = number of eligible knots,

- P = number of processors in the parallel-MARS algorithm.

Speedup, *true speedup*, *efficiency* and *true efficiency* (Dongarra et al. 1989) were calculated to evaluate the performance of parallel-MARS. The *speedup* of P processors (S_P) is defined as its run time divided into the time to execute the parallel code on one processor. The *true speedup* of P processors (True S_P) is the run time divided into the run time of the serial code. The *efficiency* (Eff), defined as S_P/P, is used to compare the overhead due to memory contention and waiting in message-passing. Similarly, the *true efficiency* (True Eff) is True S_P/P.

The main findings are presented via a subset of the runs illustrated in Figures 2 through 8. (Note: The run times for the exact same parameter settings can differ slightly for different runs due to the fact that the parallel cluster is a shared facility.) The reduction in run time as P increases is clear, but the impact of parallelization is most pronounced for larger $M_{\max}$. In Figure 2, the run time grows dramatically as $M_{\max}$ increases, and the greater reduction in run time with more processors for larger $M_{\max}$ is apparent. The calculations for S_p, True S_p, Eff, and True Eff when $N = 289$ and $K = 35$ are shown in Table 3. Using S_p and True S_p in Figures 3 and 4, this effect is seen in the steeper slopes of the lines for larger $M_{\max}$. In Figure 5, a generally linear increase in efficiency is shown as a function of $M_{\max}$. For $N = 289$, the parallel overhead for $P = 8$ processors due to memory contention and waiting ($1-$Eff) dropped from 82% for $M_{\max} = 50$ to 56% for $M_{\max} = 200$. However, efficiency degrades as P increases due to a higher probability of simultaneous memory contention and waiting with more processors. For example, in Table 3, the parallel overhead when $M_{\max} = 200$ increases from 18% for $P = 2$ to 56% for $P = 8$. Mathematically, this relationship is seen in the complexity of parallel-MARS. The computational effort for serial-MARS is (Friedman 1991, Chen et al. 1999)

$$\mathrm{O}(nN[M^*_{\max}]^3)$$

while our new parallel-MARS algorithm reduces this to

$$\mathrm{O}(nN[M^*_{\max}]^2[P + (M^*_{\max}/P)]). \tag{4}$$

For parallel-MARS, the overhead is represented by the term $nN[M^*_{\max}]^2P$, which clearly grows with P.

A linear relationship discovered between run time and K for a fixed $M_{\max}$ is illustrated in Figure 6. This figure also indicates that the run times are very close for the serial code and the parallel code with one processor. A slight upward trend between S_p and K in Figure 7 shows that a much larger K leads to tremendous increase in run time, but the increased work load is smoothed by being distributed to multi-processors. However, S_p and N do not demonstrate a significant relationship in Figure 8. The dominant impact of parallelization with larger $M_{\max}$ is logical given that the parallelized loop in Algorithm 2 is the one that depends on $M_{\max}$. In addition, it should be noted that the growth in the computational effort in (4) is most significantly impacted by $M_{\max}$.

5.2 DMF Solution

A strength three OA-LHD with $N = 2197$ discretization points was generated to compare the various SDP solutions. As in Chen (1999), mean absolute deviation (**MAD**) for the last period future value function of the SDP was calculated for the comparisons of model accuracy. The smallest N that yielded reasonable results for the wastewater treatment application was 961. Smaller N resulted in contradictory technology selections due to poor model accuracy. With $N = 961$, the best model accuracy was obtained with $M_{\max} = 200$ and $K = 35$. The serial-MARS run time for this solution was 1.66 hours, while the parallel-MARS run time with $P = 8$ was 31.51 minutes. Figure 9 illustrates how MAD changes with K, and confirms our initial suspicion that

the wastewater treatment SDP problem would require a larger K than the inventory forecasting problems, which utilized a maximum of 11 eligible knots.

The wastewater treatment DMF results provide a quantitative evaluation of the potential process technologies. In this paper, our application only considered economic cost, but other objectives, such as minimizing odor emissions (Chen et al. 2001), are available for exploration. Four measures are used to quantify performance of the technologies in each period. **Count** is the number of times a unit process was selected over the 2197 entering states. **MOD** (mean overall deviation), **MLD** (mean local deviation), and **MLRD** (mean local relative deviation) calculate deviations between the actual cost of selecting a unit process for a particular entering state and a "minimum cost," then averages over the 2197 entering states. For MOD, the "minimum cost" is the smallest cost achieved by any unit for any of the 2197 entering states (overall minimum). For MLD, the "minimum cost" for each entering state is the smallest cost achieved for that state (local minimum). Finally, for MLRD, the local deviations are scaled by the local minimum cost. Table 4 lists these four measures for the SDP solution using $N = 961, M_{\max} = 200, K = 35$. The technologies that provided adequately clean water with lower economic cost are those with higher Count and lower MOD, MLD, and MLRD. In Level 2, Chemical Precipitation appears to be the clear winner; however, looking at MOD, MLD, and MLRD, Vortex SSO is a strong second choice. Our goal here is not simply optimization, but to identify promising technologies by eliminating clearly inferior technologies. For example, in Level 5, Lagoons/Ponds is a clear loser under the economic objective.

6 Concluding Remarks

It is clear that increasing the number of processors P will improve computational performance although Speedup will not improve infinitely since the message-passing process required by parallel computing adds computational effort. On the other hand, while larger N, larger K and larger $M_{\max}$ take longer to execute, Speedup does not change significantly with N and K, and, more importantly, better Speedup can be attained with larger $M_{\max}$, especially when P is sufficient. Further research will explore flexible selection of the MARS $M_{\max}$ parameter.

The most accurate SDP solution (among those tested, based on MAD) utilized $N = 961$, $K = 35$, and $M_{\max} = 200$. Parallel-MARS with 8 processors provided a 68.4% reduction in run time for this SDP solution. Four measures are calculated by the DMF as quantitative criteria for selecting among the technologies, and promising technologies are easily identified. Other objectives that may be studied include odor emissions, land area, robustness (against extreme conditions), and global desirability.

Acknowledgements

The authors' work is supported by a Technology for Sustainable Environment (TSE) grant under the U. S. Environmental Protection Agency's Science to Achieve Results (STAR) program (Contract #R-82820701-0). Dr. Chen's work is also supported by NSF Grant #DMI 0100123.

Disclaimer

Although the research described in this article has been funded in part by the U. S. Environmental Protection Agency, it has not been subject to any EPA review and therefore does not necessarily reflect the views of the Agency, and no official endorsement should be inferred.

References

Baglietto, M., C. Cervellera, T. Parisini, M. Sanguineti, and R. Zoppoli (2001). "Approximating Networks for the Solution of T-stage Stochastic Optimal Control Problems." *Proceedings of the IFAC Workshop on Adaptation and Learning in Control and Signal Processing.*

Bakin, S., M. Hegland, and M. R. Osborne (2000). "Parallel MARS algorithm based on B-splines." *Computational Statistics*, **15**, pp. 463–484.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton: Princeton University Press.

Bertsekas, D. P. and J. N. Tsitsiklis (2000). *Dynamic Programming and Optimal Control.* Belmont, Massachusetts: Athena Scientific.

Bhargava, R., G. Fox, C. Ou, S. Ranka, and V. Singh (1993). "Scalable Libraries for Graph Partitioning." In *Proceedings of the Scalable Parallel Libraries Conference*, Mississippi, October 1993.

Chen, Jining (1993). "Environmentally Efficient Urban Drainage for the 21st Century: A Literature Review." Technical Report, Department of Civil Engineering, Imperial College of Science, Technology and Medicine, London, UK.

Chen, J. and M. B. Beck (1997). "Towards Designing Sustainable Urban Wastewater Infrastructures: a Screening Analysis." *Water Sciences Technology*, **35**, pp. 99–112.

Chen, V. C. P. (1999). "Application of Orthogonal Arrays and MARS to Inventory Forecasting Stochastic Dynamic Programs." *Computational Statistics and Data Analysis*, **30**, pp. 317–341.

Chen, V. C. P. (2001). "Measuring the Goodness of Orthogonal Array Discretizations for Stochastic Programming and Stochastic Dynamic Programming." *SIAM Journal of Optimization*, **12**, pp. 322–344.

Chen, V. C. P., J. Chen, and M. B. Beck (2000). "Statistical Learning within a Decision-Making Framework for More Sustainable Urban Environments." In *Proceedings of the Joint Research Conference on Statistics in Quality, Industry, and Technology*, Seattle, Washington, June 2000.

Chen, V. C. P., D. Ruppert, and C. A. Shoemaker (1999). "Applying Experimental Design and Regression Splines to High-Dimensional Continuous-State Stochastic Dynamic Programming." *Operations Research*, **47**, pp. 38–53.

Chen, V. C. P., J. C. C. Tsai, E. K. Lee, and E. L. Johnson (2001). "A Decision-Making Framework for Evaluating Wastewater Treatment Technologies." In *Proceedings of the 5th Annual Green Chemistry and Engineering Conference*, Washington, D. C. June 2001.

Dongarra, J., I. Duff, P. Gaffney, and S. McKee (1989). *Vector and Parallel Computing: Issues in Applied Research and Development*. New York, NY: John Wiley & Sons, Inc.

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Reading, MA: Addison-Wesley Publishing Company.

Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines (with discussion)." *Annals of Statistics*, **19**, pp. 1–141.

Gropp, W. and E. Lusk (1994). "Implementing MPI: the 1994 MPI Implementors' Workshop." In *Proceedings of the 1994 Scalable Parallel Libraries Conference*, Mississippi, October 1994.

Gropp, W., E. Lusk, and A. Skjellum (1999). *Using MPI—Portable Parallel Programming with the Message-Passing Interface* (2nd edition). Cambridge, MA: The MIT Press.

Lee, E., R. Bixby, W. Cook, and A. Cox (1995). *Parallel Mixed Integer Programming*. Center for Research on Parallel Computation Research Monograph CRPC-TR95554.

Linderoth J., E. Lee, and M. Savelsbergh (1999). *A Parallel, Linear Programming Based Heuristic for Large Scale Set Partitioning Problems*. Industrial & Systems Engineering Technical Report. To appear, INFORMS Journal on Computing.

Owen, A. B. (1992). "Orthogonal Arrays for Computer Experiments, Integration, and Visualization." *Statistica Sinica*, **2**, pp. 439–452.

Plackett, R. L. and J. P. Burman (1946). "The Design of Multifactorial Experiments." *Biometrika*, **33**, pp. 305–325.

Puterman, M. L. (1994). *Markov Decision Processes*. New York, NY: John Wiley & Sons, Inc.

Snir, M., S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra (1998). *MPI—The Complete Reference, The MPI Core* (Vol. 1, 2nd Edition). Cambridge, MA: The MIT Press.

Tang, B. (1993). "Orthogonal Array-Based Latin Hypercubes." *Journal of the American Statistical Association*, **88**, pp. 1392–1397.

West, J. E., M. M. Stehens, and L. H. Turcotte (1994). "Adaptation of Volume Visualization Techniques to MIMD architectures using MPI." *Proceedings of the 1994 Scalable Parallel Libraries Conference*, Mississippi, October 1994.

Zhuang, X. and J. Zhu (1994). "Parallelizing a Reservoir Simulator using MPI." In *Proceedings of the 1994 Scalable Parallel Libraries Conference*, Mississippi, October 1994.

Table 1: Lower and upper range limits (in mg/l) on the ten state variables of the wastewater treatment system for the levels of the liquid line. Levels 1 and 4 are excluded because the state variables are not affected by the technologies in those levels.

Entering ↓	**CODS**	**SS**	**orgN**	**ammN**	**nitN**
Level 2	200.0	220.0	15.00	25.00	0
	210.0	231.0	15.75	26.25	0.01
Level 3	130.0	22.0	4.5	22.500	0
	210.0	115.5	15.0	28.875	0.01
Level 5	104.0	19.8	3.60	13.5	0
	199.5	115.5	14.25	23.1	42.1675
Level 6	2.080	0.396	0.54	0	0
	69.825	7000.000	100.00	21.945	122.27
Level 7	0.208	$3.96(10^{-3})$	0.05	0	0
	69.825	350.0	100.00	21.945	122.27
Level 8	0.0200	$3.96(10^{-5})$	$5.4(10^{-3})$	0	0
	62.8425	52.5	70.0	21.945	122.27
Level 9	$16.64(10^{-3})$	$3.5(10^{-5})$	$4.32(10^{-3})$	0	0
	59.71	52.5	66.5	17.6	170.4
Level 10	$3.328(10^{-3})$	$3.5(10^{-5})$	$4.32(10^{-3})$	0	0
	47.77	52.5	66.5	15.0	170.4
Level 11	$3.328(10^{-3})$	$3.5(10^{-5})$	$4.32(10^{-3})$	0	0
	47.77	52.5	66.5	15.0	170.4
Entering ↓	**totP**	**HM**	**SOCs**	**pathogens**	**viruses**
Level 2	8.0	0.0100	15.00	$5.00(10^7)$	100.0
	8.4	0.0105	15.75	$5.25(10^7)$	105.0
Level 3	0.8	0.00050	1.500	$5.00(10^6)$	10.0
	8.0	0.00945	14.175	$3.15(10^7)$	63.0
Level 5	0.8	0.00050	0.2250	0	0.2
	8.0	0.00945	7.0875	$3.15(10^6)$	6.3
Level 6	0.08	$5(10^{-6})$	$2.25(10^{-5})$	0	$2(10^{-5})$
	10.00	0.00567	4.2525	$1.89(10^6)$	3.78
Level 7	0	$1(10^{-7})$	$1.125(10^{-6})$	0	$1(10^{-7})$
	10.0	0.00567	4.2525	$1.89(10^6)$	3.78
Level 8	0	0	0	0	0
	8.0	0.0018	1.063125	$2.835(10^5)$	0.567
Level 9	0	0	0	0	0
	8.0	0.0018	0.532	$2.835(10^4)$	0.0567
Level 10	0	0	0	0	0
	8.0	0.0018	0.266	4252.5	0.0086
Level 11	0	0	0	0	0
	8.0	0.0018	0.32	212.625	0.0026

Table 2: Minimum and maximum values (in mg/l) of the ten state variables for the effluent exiting the liquid line. Targets are approximately 10% of the maximums. For the quintic penalty function, differences Δ between the knots are calculated as $0.5\times$(maximum $-$ target). Penalty coefficients are calculated as 2000/(maximum $-$ target).

Effluent $\downarrow$	**CODS**	**SS**	**orgN**	**ammN**	**nitN**
Minimum	$3.328(10^{-5})$	$3.5(10^{-7})$	$4.3(10^{-4})$	0	0
Maximum	47.77	52.5	66.5	15.0	170.4
Target	5.00	5.5	7.00	1.5	16.0
Δ	21.25	23.5	29.75	6.7	76.0
Penalty	47.00	43.0	34.00	150.0	13.0
Effluent $\downarrow$	**totP**	**HM**	**SOCs**	**pathogens**	**viruses**
Minimum	0	0	0	0	0
Maximum	8.0	0.0018	0.32	212.625	0.0026
Target	0.80	0.00015	0.04	20.0	0.00030
Δ	3.55	0.00079	0.14	95.5	0.00113
Penalty	275.00	$1.29(10^6)$	$7.2(10^3)$	10.5	$8.88(10^5)$

Table 3: Parallel performance on $N = 289$, $K = 35$.

$\boldsymbol{P}$	**2**	**4**	**6**	**8**	**2**	**4**	**6**	**8**
$\boldsymbol{M}_{\max}$	**50**	**50**	**50**	**50**	**100**	**100**	**100**	**100**
$\boldsymbol{S_P}$	1.192	1.357	1.413	1.444	1.418	1.974	2.126	2.233
Eff	0.596	0.339	0.236	0.181	0.709	0.493	0.354	0.279
True $\boldsymbol{S_P}$	1.199	1.365	1.422	1.453	1.436	1.999	2.153	2.261
True Eff	0.599	0.341	0.237	0.182	0.718	0.500	0.359	0.283
$\boldsymbol{M}_{\max}$	**150**	**150**	**150**	**150**	**200**	**200**	**200**	**200**
$\boldsymbol{S_P}$	1.544	2.313	2.528	2.756	1.641	2.628	2.968	3.527
Eff	0.772	0.578	0.421	0.344	0.821	0.657	0.495	0.441
True $\boldsymbol{S_P}$	1.546	2.317	2.531	2.760	1.660	2.658	3.001	3.567
True Eff	0.773	0.579	0.422	0.345	0.830	0.665	0.500	0.446

Table 4: DMF selected technologies for economic cost optimal. New technologies (perhaps only existing as prototypes) are in **boldface**, and somewhat new technologies (perhaps not yet well understood) are shown in *italics*.

Level	Unit Process	Count	MOD	MLD	MLRD
1	Flow Equalisation Tank				
2	Empty Unit	0	6.3×10^8	6.3×10^8	6.8×10^6
	Vortex SSO	0	166	106.62	1.1530
	Sedimentation Tank	0	898	838.57	9.0661
	Chemical Precipitation	2197	59	0	0
3	Empty Unit	0	2.2×10^8	2.2×10^8	2.7×10^6
	Physical Irradiation	13	1912	59.43	0.7159
	Ozonation	2184	1853	0.07	0.0008
4	Empty Unit				
5	Empty Unit	0	1.9×10^7	1.9×10^7	2.9×10^5
	Activated Sludge(C)	0	5201	2632.05	39.1654
	Activated Sludge(C,N)	0	9901	7332.28	107.9695
	Activated Sludge(C,P)	182	3208	638.62	9.8947
	Activated Sludge(C,P,N)	0	8831	6262.39	92.2593
	High Biomass Act. Sludge	0	4811	2241.72	33.4615
	Activated Sludge(N)	0	9023	6453.76	95.1164
	Multi-reactor/Deep Shaft	0	5269	2700.47	40.3886
	A-B System	151	3209	640.02	10.0114
	Trickling Filter	0	16997	14428.48	211.8184
	Rotating Biological Cont.	3	3443	873.64	13.4603
	UASB System	1861	2636	67.39	0.9050
	Reed Bed System	0	62331	59762.29	875.3710
	Lagoons and Ponds	0	77665	75095.97	1099.9700
6	Empty Unit	107	1.1×10^7	1.1×10^7	2.4×10^5
	Secondary Settler	68	3025	712.77	16.4238
	Microfiltration	509	2506	193.24	4.7106
	Reverse Osmosis	901	2457	144.53	3.5391
	Chemical Precipitation	612	2492	179.72	4.2878
7	Empty Unit	1	1.2×10^7	1.2×10^7	3.2×10^5
	Physical Filtration	96	2014	517.81	14.5462
	Microfiltration	1573	1575	79.04	1.9752
	Reverse Osmosis	12	2210	713.99	20.0764
	Chemical Precipitation	515	1840	344.24	10.3827
8	Empty Unit	8	1.7×10^6	1.7×10^6	4.6×10^4
	Physical Irradiation	642	3111	31.10	0.9486
	Ozonation	1547	3145	64.89	1.5537
9	Empty Unit	11	1.7×10^5	1.7×10^5	5951.2388
	Air Stripping	79	3880	1019.37	40.5287
	Ammonia Stripping	2107	2872	11.93	0.6961
10	Empty Unit	33	25896	22979.52	819.8374
	Chlorine Disinfection	1940	2936	20.04	0.8977
	Chlorating Disinfection	224	3162	245.71	9.1489
11	Empty Unit	877	2772	604.09	31.2061
	GAC Adsorption	1320	2620	452.44	328.8093
	Infiltration Basin	0	42805	40637.77	9803.2567

Algorithm 1 Serial-MARS: Forward Stepwise Algorithm (Friedman 1991, p. 17)

S1: $B_1(\boldsymbol{x}) \leftarrow 1; M \leftarrow 2$
S2: Loop until $M > M_{\max}$: lof$^* \leftarrow \infty$
S3: **for** $m = 1$ to $M - 1$ **do**
S4: **for** $v \notin \{v(l,m) \mid 1 \le l \le L_m\}$ **do**
S5: **for** $k \in \{x_{vj} \mid B_m(\boldsymbol{x}_j) > 0\}$ **do**
S6: $g \leftarrow \sum_{i=1}^{M-1} a_i B_i(\boldsymbol{x}_i) + a_M B_m(\boldsymbol{x})[+(x_v - k)]_+ + a_{M+1} B_m(\boldsymbol{x})[-(x_v - k)]_+$
S7: lof $\leftarrow \min_{a_1,\ldots,a_{M+1}} LOF(g)$
S8: if lof $<$ lof*, then lof$^* \leftarrow$ lof; $m^* \leftarrow m; v^* \leftarrow v; k^* \leftarrow k$ end if
S9: **end for**
S10: **end for**
S11: **end for**
S12: $B_M(\boldsymbol{x}) \leftarrow B_{m^*}(\boldsymbol{x})[+(x_{v^*} - k^*)]_+$
S13: $B_{M+1}(\boldsymbol{x}) \leftarrow B_{m^*}(\boldsymbol{x})[-(x_{v^*} - k^*)]_+$
S14: $M \leftarrow M + 2$
S15: end loop
S16: end algorithm

Algorithm 2 Parallel-MARS: Forward Stepwise Algorithm

P1: Read in data and initialize (line S1)
P2: Master processor C_0 distributes data to slave processors C_i, $i = 1, ..., P-1$
P3: **while** $M > M_{\max}$ **do**
P4: **if** $M > P$ **then**
P5: **for** each processor C_i $(i = 0, 1, \ldots, P-1)$ do in **parallel**: lof$^* \leftarrow \infty$ (line S2); $j \leftarrow 0$; $m_i \leftarrow 1 + i + jP$ **do**
P6: **while** $m_i < M$ **do**
P7: for basis function m_i, loop through covariate v and knot k (lines S4–S7)
P8: store lof$_i^*, {m_i}^*, {v_i}^*, {k_i}^*$ (line S8)
P9: $j \leftarrow j + 1$; $m_i \leftarrow 1 + i + jP$
P10: **end while**
P11: **end for parallel**
P12: **else if** $M <= P$ **then**
P13: run forward stepwise algorithm serially
P14: **end if**
P15: lof$^* \leftarrow \min\{\text{lof}_i^*, i = 0, ..., P-1\}$, then $m^* \leftarrow {m_i}^*; v^* \leftarrow {v_i}^*; k^* \leftarrow {k_i}^*$
P16: Calculate $B_M(\boldsymbol{x})$ and $B_{M+1}(\boldsymbol{x})$ (lines S12–S13)
P17: $M \leftarrow M + 2$ (line S14)
P18: Master Processor C_0 broadcasts the updated M and basis functions to all slave processors
P19: **end while**
P20: end algorithm

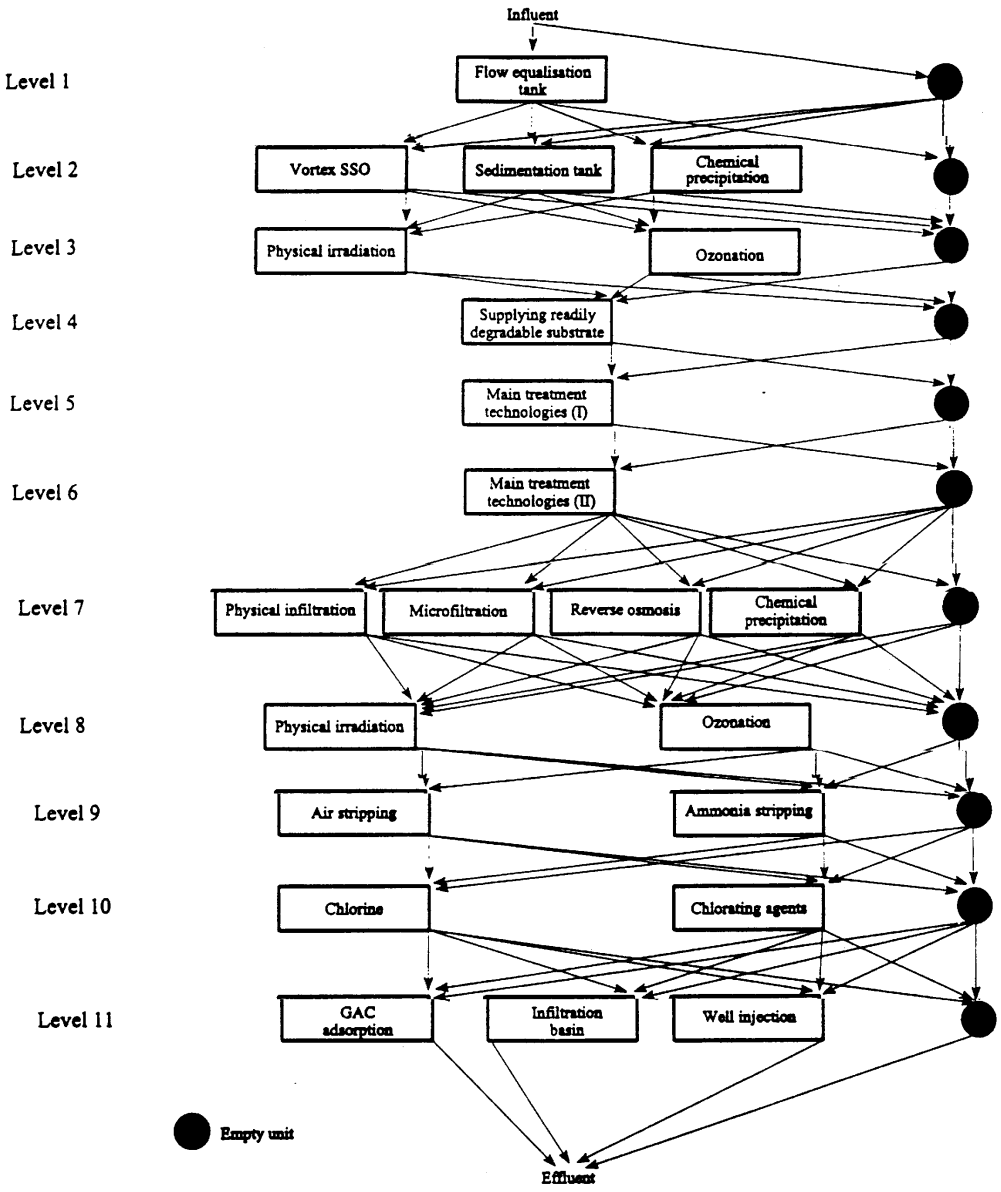

Figure 1: Levels and unit processes for the liquid line of the wastewater treatment system.

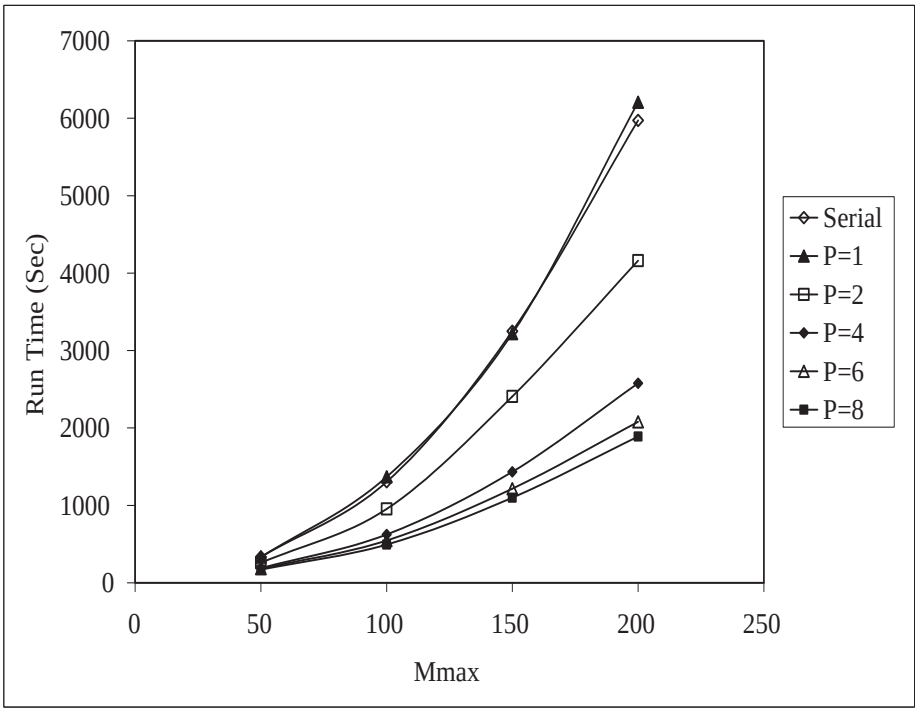

Figure 2: Run time (seconds) vs. $M_{\max}$ for several choices of P (number of processors). The number of discretization points is $N = 961$ and the number of eligible knots is $K = 35$.

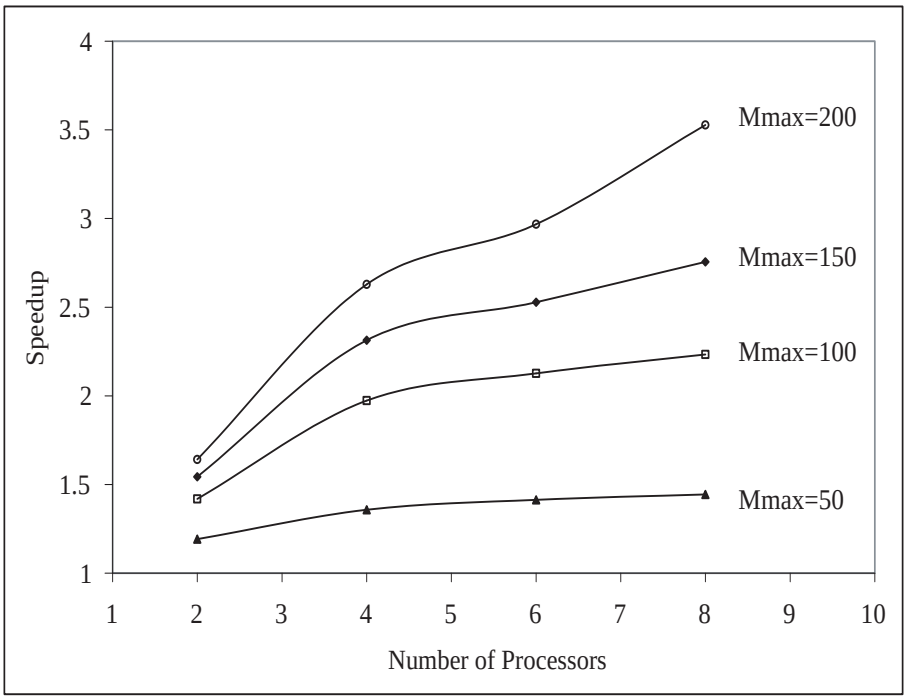

Figure 3: Speedup (S_P) vs. the number of processors (P) for several choices of $M_{\max}$. The number of discretization points is $N = 289$ and the number of eligible knots is $K = 35$.

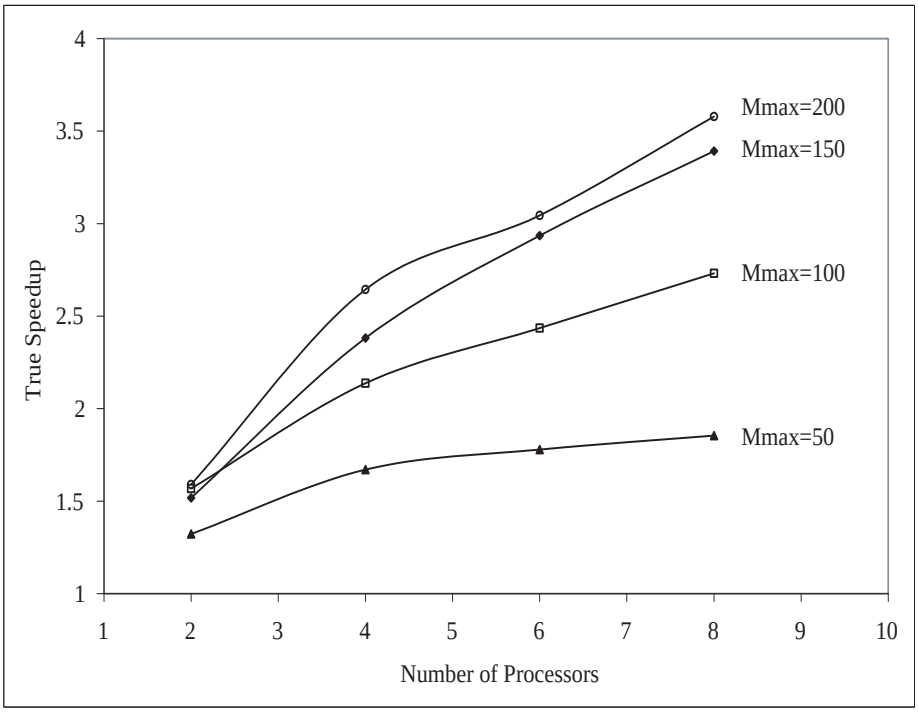

Figure 4: True Speedup (True S_P) vs. the number of processors (P) for several choices of $M_{\max}$. The number of discretization points is $N = 841$ and the number of eligible knots is $K = 35$.

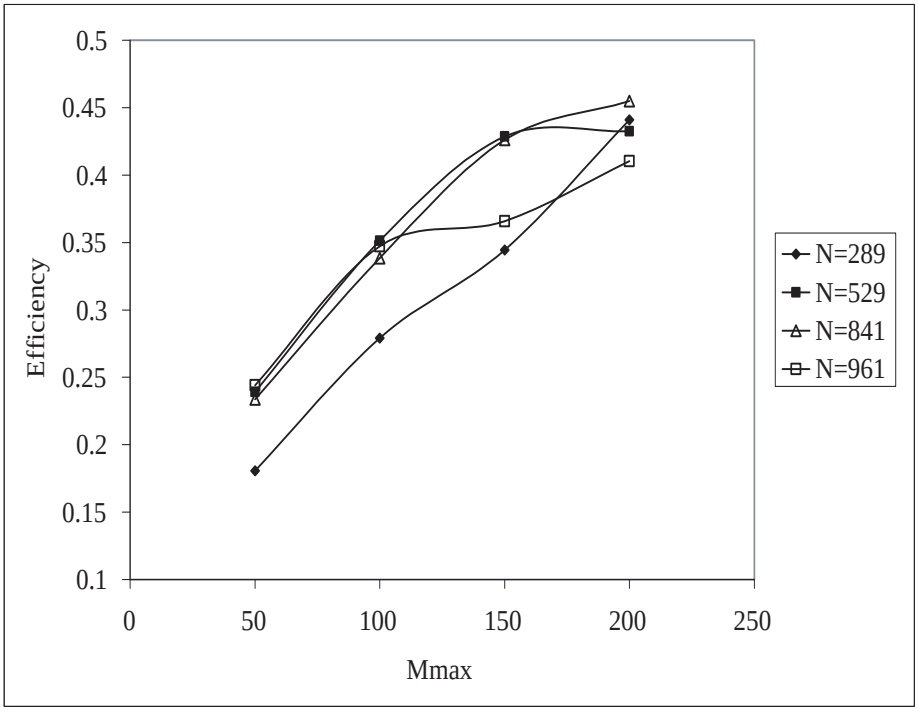

Figure 5: Efficiency (Eff) vs. $M_{\max}$ for several choices of N (number of discretization points). The number of eligible knots is $K = 35$ and the number of processors is $P = 8$.

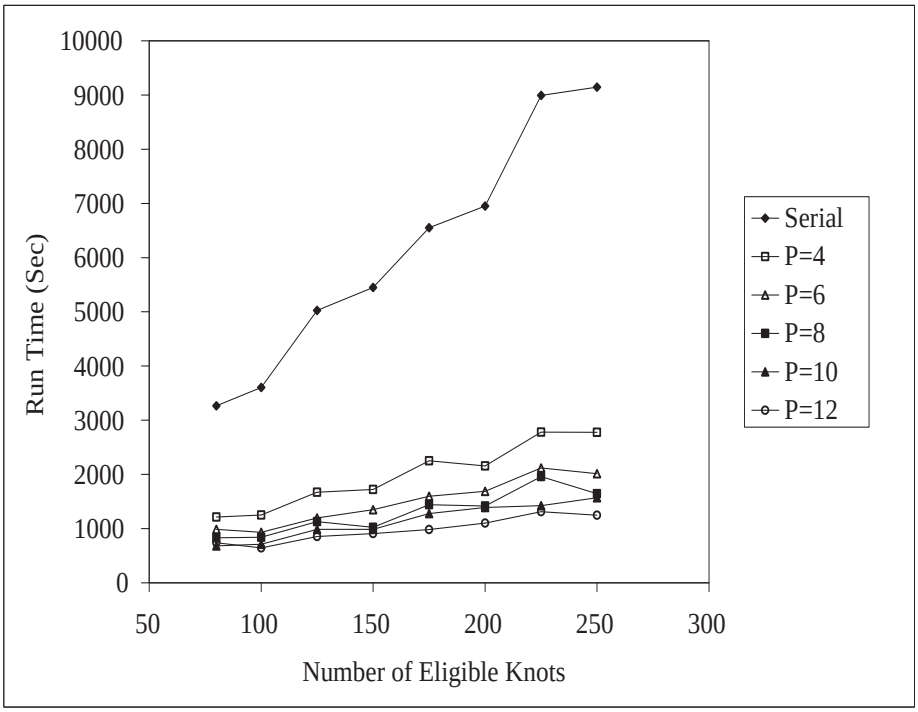

Figure 6: Run time (seconds) vs. the number of eligible knots (K) for several choices of P (number of processors). The number of discretization points is $N = 289$ and $M_{\max} = 200$.

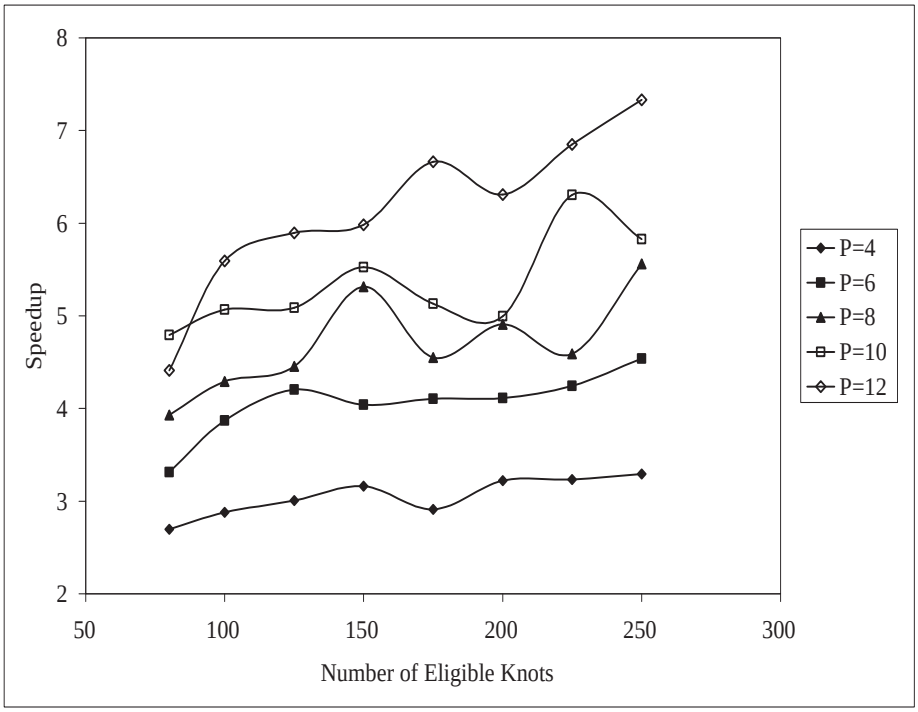

Figure 7: Speedup (S_P) vs. the number of eligible knots (K) for several choices of P (number of processors). The number of discretization points is $N = 289$ and $M_{\max} = 200$.

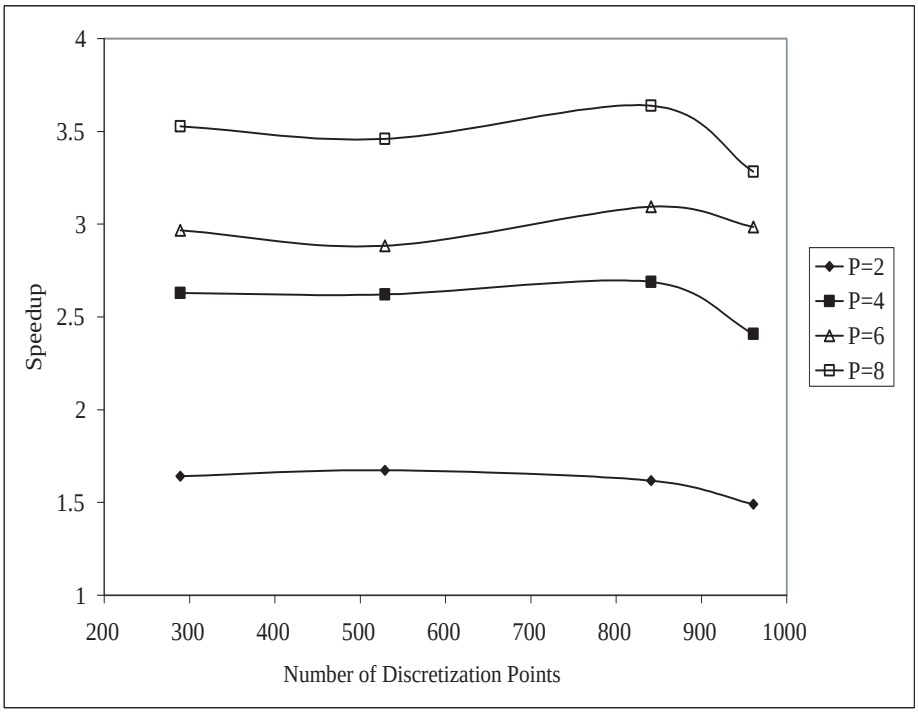

Figure 8: Speedup (S_P) vs. the number of discretization points (N) for several choices of P (number of processors). The number of eligible knots is $K = 35$ and $M_{\max} = 200$.

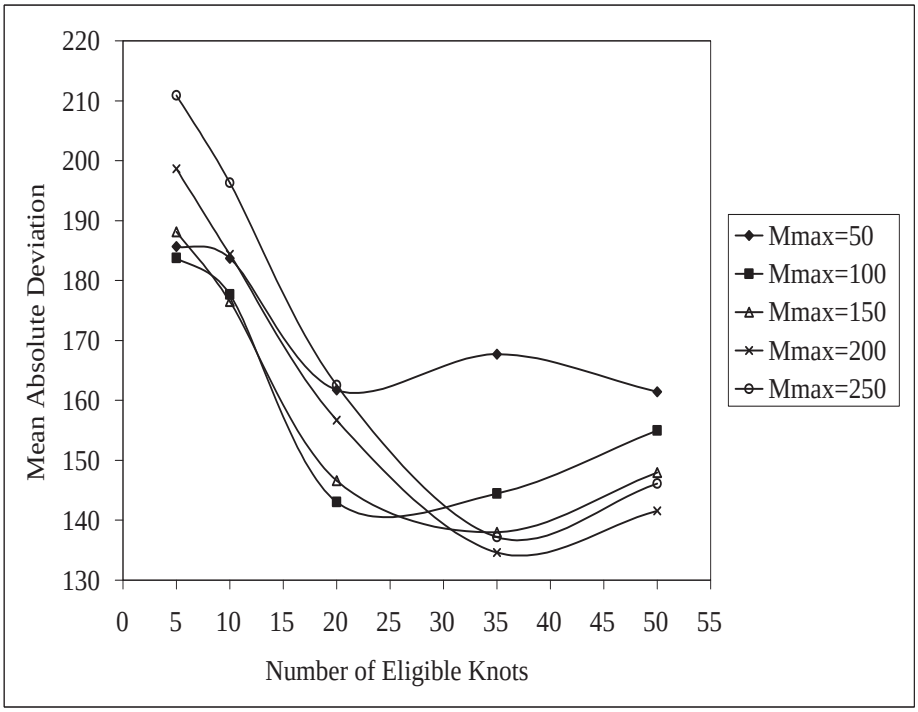

Figure 9: Mean absolute deviation (MAD) on the validation data set of 2197 points vs. the number of eligible knots (K) for several choices of $M_{\max}$. The number of discretization points is $N = 981$.