INFORMATICA, 1994, Vol. 5, No. 1-2, 123-166

# BAYESIAN APPROACH ADAPTING STOCHASTIC AND HEURISTIC METHODS OF GLOBAL AND DISCRETE OPTIMIZATION

Audrius MOCKUS
Department of Statistics
Carnegie-Mellon University, Pittsburgh, PA 15217
Jonas MOCKUS
Department of Optimization
Institute of Mathematics and Informatics
2600 Vilnius, Akademijos St.4, Lithuania
Linas MOCKUS
School of Chemical Engineering
Purdue University, W.Lafayette, IN 47907-1283

**Abstract.** We consider here the average deviation as the most important objective when designing numerical techniques and algorithms. We call that a Bayesian approach.

We start by describing the Bayesian approach to the continuous global optimization. Then we show how to apply the results to the adaptation of parameters of randomized techniques of optimization. We assume that there exists a simple function which roughly predicts the consequences of decisions. We call it heuristics. We define the probability of a decision by a randomized decision function depending on heuristics. We fix this decision function, except for some parameters that we call the decision parameters.

We repeat the randomized decision procedure several times given the decision parameters and regard the best outcome as a result. We optimize the decision parameters to make the search more efficient. Thus we replace the original optimization problem by an auxiliary problem of continuous stochastic optimization. We solve the auxiliary problem by the Bayesian methods of global optimization. Therefore we call the approach as the Bayesian one.

We discuss the advantages and disadvantages of the Bayesian approach. We describe the applications to some of discrete pro-

gramming problems, such as optimization of mixed Boolean bilinear functions including the scheduling of batch operations and the optimization of neural networks.

**Key words:** optimization, discrete, combinatorial, global, Bayesian, stochastic, Boolean, bilinear, scheduling, neural network.

**1. Introduction.** We usually need algorithms of exponential complexity if we wish to get the exact solution of many global and discrete optimization problems. The screening techniques such as branch-and-bound help a lot, but a high complexity remains there. The exponential complexity often remains too, in the case when we are ready to accept an aproximate solution, if we insist on keeping a strict error limit. It means that an important factor of exponential complexity is our desire to garantee the satisfactory results for all the cases, including the worst case. The reason is that the worst case can be very bad, if the family of functions to be optimized is large, see Mockus (1989).

Therefore many applied global and discrete optimization problemas are considered using heuristics. We define by heuristics simple functions such that roughly predict the consequences of decisions in the optimization process. We use randomized decision procedures, if we wish to provide the convergence, in the probabilistic sense, to the global minimum by muliple repetition of randomized decisions. If we wish to increase the efficiency of search we define the decision probabilities as a function of heuristics. We do not know in advance what the best decision function is. We can get the answer only after extensive experimantation. We may adapt the decision function to the given family of optimization problems by optimizing decision parameters as a result of repeated optimization procedures.

Most authors of randomized heuristics techniques adapt the decision parameters "by hand" as a part or research. This

adaptation is not a creative process and may be formalized as an optimization problem. We regard the decision parameters as real numbers. The outcome of optimization is stochastic. The function to be optimized is multimodal in many cases. Thus the optimization of decision parameters in general is the problem of continuous stochastic global optimization. The optimization techniques just for such type of problem may be designed using the Bayesian approach, see Mockus (1989). Therefore we start this paper from a brief description of the Bayesian approach to the continuous global optimization. Later on we show how to apply those techniques to the optimization of the decision parameters using the randomized heuristics.

## 2. Bayesian approach: definitions and extensions.

**2.1. Optimality criteria.** We consider the continuous global optimization problem

$$\min_{x \in A} f(x), \tag{1}$$

where

$$f(x) = f(x, \omega), \quad A \subset R^m, \quad \omega \in \Omega.$$

Here the index set $\Omega$ defines a family of functions $f(x)$.

We assume the linear loss function

$$L(d, \omega) = f(x(d), \omega) - f^*(\omega). \tag{2}$$

Here $x(d)$ is the final decision of the optimization method $d \in D$ and $f^*(\omega)$ is the global minimum of $f(x, \omega)$. Thus the loss function is a deviation from the global minimum $f^*(\omega)$ taking the final decision $x(d)$.

We define the average deviation $R(d)$ as an expectation of the loss function $L(d, \omega)$. We can do that by defining some measure $P$ on the set $\Omega$. Now we can define the Lebesgue integral

$$R(d) = \int_{\Omega} (f(x(d),\omega) - f^*(\omega))dP(\omega). \tag{3}$$

The expectation of $f^*(\omega)$ does not depend on $d$, therefore we can simplify the risk function omitting the constant. Then

$$R(d) = \int_{\Omega} f(x(d,\omega))dP(\omega). \tag{4}$$

We call the decision $d^*$, that minimizes the risk function $R(d)$, the Bayesian decision.

**2.2. Distribution on the set of continuous functions.** We consider the objective $f(x)$ as a stochastic function $f(x) = f(x,\omega), x \in A \subset R^m, \omega \in \Omega$. Here $\omega$ is the index of the function $f(x)$ and $\Omega$ defines the set of continuous functions, see Mockus (1989). We observe the vector $z_n = (y_i, x_i, i = 1,\ldots,n)$ and make decisions depending on $y_x$. Here $y_i = f(x_i), i = 1,\ldots,n$ and $y_x = f(x) = f(x,\omega)$, for the fixed $x \in A$, $\omega \in \Omega$. We define prior on $\Omega$ by the family of finite dimensional Gaussian distribution functions:

$$F_{x_1,\ldots,x_n}(y_1,\ldots,y_n) = P(f(x_1) < y_1,\ldots,f(x_n) < y_n), \quad n = 1,2,\ldots.$$

Then the posterior of $f(x) = y_x$ is

$$p_x(y_x) = \frac{1}{\sqrt{2\pi}\sigma(x)} e^{-\frac{1}{2}(\frac{y_x - \mu_n(x)}{\sigma_n(x)})^2}. \tag{5}$$

Here

$$\mu_n(x) = \mu + \Sigma_x \Sigma^{-1}(y_x - \mu), \tag{6}$$

and

$$\sigma_n^2(x) = \sigma^2 - \Sigma_x \Sigma^{-1} \Sigma_x^T, \tag{7}$$

where $\Sigma_x = (\sigma_{xj})$, $\Sigma = (\sigma_{ij})$. Here $\sigma_{xj}$ is the covariance of the pair $f(x)$, $f(x_j)$, $\sigma_{ij}$ is the covariance of the pair $f(x_i)$, $f(x_j)$, $\mu$ is the expectation of $f(x)$, and $\sigma^2$ is the variance of $f(x)$.

This Gaussian stochastic function is homogeneous, because the parameters (6), and (7) don't depend on $x$. The matrixes $\Sigma$ and $\Sigma_x$ depend on the results of $n$ observations. It means that the conditional mean and the standard are functions of $z_n$ and $x$. Thus, $\mu_n(x) = \mu(z_n, x)$ and $\sigma_n(x) = \sigma(z_n, x)$.

Several authors use similar prior distributions on the set of continuous functions. Most popular is the Wiener process, see Kushner (1964), Diaconis (1988), Žilinskas (1986), Šaltenis (1971), Betro (1991). A homogeneous multi-dimensional extension of the Wiener process is given by Mockus (1989). It is a Gaussian stochastic function defined on $A = [-1, 1]^n$ with constant expectation $\mu$ and constant variance $\sigma^2$ of $f(x)$. We denote the covariance matrix by $\Sigma$.
We define the covariance of the pair $f(x_j)$, $f(x_k)$ as:

$$\sigma_{jk} = \sigma^2 \prod_{i=1}^{m} \left(1 - \frac{|x_j^i - x_k^i|}{2}\right). \tag{8}$$

We call a stochastic function with the covariance function defined by expression (8) the Wiener model.

We may assume that the stochastic function $f(x)$ is simpler if its values at different points $x$ are "less" statistically dependent. Thus a Gaussian stochastic function is simpler if the absolute value of the correlation $|\sigma_{ij}|/\sigma^2$ is smaller.

Therefore the Wiener model is the simplest stochastic function providing a continuity of almost all sample functions. The Wiener model also shows that the Bayesian approach can

be regarded as some instrument of extrapolation under uncertainty. We understand the uncertainty as the conditional variance. The uncertainty is an increasing function of the distance from the nearest observation. The uncertainty is zero, if the distance is zero and there is no noise.

It means that the Bayesian approach should be efficient, if the statistical model reflects the relation between the uncertainty and the distance well enough. Otherwise, we can't expect a significant improvement as compared to the uniform search.[1] Therefore the definition of a distance is an important part of the Bayesian approach to global optimization. It is almost obvious how to define the distance in the continuous problems of global optimization. It is not so clear in the problems of discrete optimization. In the problems of combinatorial optimization the distances could be defined almost arbitrarily. Therefore we would prefer to reduce the discrete optimization problems to the problems of the continuous global optimization.

**2.3. One-step approximation of a sequential decision problem.** In general the minimization of the risk function $R(d)$ is a multi-stage sequential decision problem. At each stage $n < N$ we define the optimal point of the next observation. We choose the final decision $x(d)$ completing all $N$ observations. The sequential decision problem is too difficult. So we replace the multi-stage decision problem by the one-stage approximation. We assume that the next observation $x_{n+1}$ is the last one. The result is the minimum of conditional expectation of $f(x)$ given $y_1, \ldots, y_{n+1}$. We denote it by

$$\mu_{n+1} = \min_{x \in A} \mu_{n+1}(x).$$

Here

[1] When the probabilities of each feasible point are equal.

$$\mu_{n+1}(x) = \mu(z_{n+1}, x) = \mu(z_n, y_{n+1}, x_{n+1}, x)$$

or approximately

$$\mu_{n+1} = \min(c_n, y_{n+1}),$$

where

$$c_n = \min_{1 \leqslant i \leqslant n} y_i - \epsilon_n. \tag{9}$$

Here $\epsilon_n > 0$ is a correction. There are two reasons for the correction. The first one is to compensate the error of the one-stage approximation. An approximate compensation is $\sigma\sqrt{2\ln(N-n)}$, where $N$ and $n$ are a total and a current observation number. The second reason is to estimate the difference between the minimal observed value and the minimal expected value. This difference is zero for the Wiener process without noise.

If there is a noise $\xi_i$, then we observe the sum $\phi(x_i) = f(x_i) + \xi_i$. Here we define $c_n$ in a different way, to provide the convergence of the Bayesian algorithm

$$c_n = y_{0n} - \epsilon_n, \tag{10}$$

where

$$y_{0n} = \begin{cases} \min_{1 \leqslant i \leqslant n} y_i, & \text{if } \min_{1 \leqslant i \leqslant n} y_i \geqslant y_0, \\ y_0, & \text{otherwise.} \end{cases} \tag{11}$$

Here $y_0$ is the value of the objective $f(x)$ which satisfies our needs.

Using the one-stage approximation we can express the loss as

$$L(d, \omega) = \min(c_n, y_{n+1}). \tag{12}$$

Here the decision $d = x_{n+1}$ defines the $n+1$ observation. Thus the risk $R(x)$ depends on $x = x_{n+1}$. In the Gaussian case

$$R(x) = \frac{1}{\sqrt{2\pi}\sigma_n(x)} \int_{-\infty}^{+\infty} \min(c_n, y_x) e^{-\frac{1}{2}\left(\frac{y_x - \mu_n(x)}{\sigma_n(x)}\right)^2} dy_x. \quad (13)$$

Here $x = x_{n+1}$ defines the next observation. We can see that $R(x)$ is an increasing function of

$$\rho(x) = \frac{\mu_n(x) - c_n}{\sigma_n(x)}. \quad (14)$$

Thus we can replace the minimization of the integral $R(x)$ by the minimization of the function $\rho(x)$. In this sense the Gaussian model is the simplest one.

**2.4. Approximation to the posterior distribution (consistency conditions).** The main disadvantage of proper posterior (5), (6) and (7) is that we need an inversion of the covariance matrix $\Sigma$. It is the price we pay for keeping the consistency conditions. The way to avoid the inversion is to drop the consistency conditions.

Mockus (1989) replaced the consistency condition by three conditions:

i. The continuity of risk function (13).

ii. The convergence of the method to the global minimum of any continuous function.

iii. The simplicity of expressions of the "posterior" mean $\mu_n(x)$ and the "posterior" standard $\sigma_n(x)$.

The result is the Gaussian stochastic function with the step-wise mean

$$\mu_n(x) = y_i, \ x \in A_i, \quad (15)$$

and the piece-wise quadratic standard

$$\sigma_n(x) = (\|x - x_i\|)^2, \; x \in A_i. \tag{16}$$

We define the set $A_i$ by the condition of continuity of the risk function $R(x)$ (or the function $\rho(x)$)

$$x \in A_i, \quad \text{iff } \frac{y_i - y_{0n} + \epsilon_n}{\|x - x_i\|^2} = \min_{1 \leqslant k \leqslant n} \frac{y_k - y_{0n} + \epsilon_n}{\|x - x_k\|^2}. \tag{17}$$

Here both the mean and the standard are discontinuous. It means that sample functions are discontinuous, too. To keep the continuity of samples we need a more complicated model. Therefore we keep only the continuity of risk $R(x)$.

Now we may define the Bayesian point of the next observation as

$$x_{n+1} = \arg\min_{x \in A} \min_{1 \leqslant i \leqslant n} \frac{y_i - c_n}{\|x - x_i\|^2}. \tag{18}$$

The "posteriors" (15), (16) and (17) do not satisfy the consistency conditions (Mockus, 1989); consequently they do not define the proper conditional expectation and variance. If we wish to put everything into the conventional framework, we may regard improper "posteriors" $\mu_n(x), \sigma_n(x)$ as a sequence of proper priors. It means "updating" the priors after each observation. The "updating" of priors follows from the one-step approximation, which "updates" the sequential decision problems after each observation, too.

Such "updating" is not proper, if we regard it in the framework of the classical statistical decision theory. Thus we see no "properk way to avoid the inversion of an $n$-dimensional covariance matrix, where $n$ is the number of observations.

Therefore we extend the notion of Bayesian approach to improper posteriors, too, to overcome the computational difficulties. We think that improper posteriors are better than

pure heuristics, because even improper posteriors can be used as some reasonable measure defining the average deviation. Thus we may consider the consequences of the assumptions theoretically and better define the areas of applications. We can also define the asymptotics, we can show that an asymptotic density of observations is much greater around the global minimum:

$$\delta_0/\delta_a = \left(\frac{f_a - f^* + \epsilon}{\epsilon}\right)^{1/2}. \tag{19}$$

Here $\delta_0$ is the asymptotic density of observations around the global minimum, $\delta_a$ is the average density. The limit $\epsilon = \lim_{n\to\infty} \epsilon_n$, where $\epsilon_n$ is correction (9), $f_a$ is the average value of $f(x)$,

$$f_0 = \begin{cases} f^*, & \text{if there is no noise or} \quad f^* > y_0, \\ y_0, & \text{otherwise.} \end{cases} \tag{20}$$

**2.5. Comparison to other approaches.** Now we can compare the Bayesian approach with the well known simulated annealing algorithm of global optimization. In those algorithms we make random perturbations and pass to a new point $n+1$ with probability

$$p_n = \begin{cases} e^{-\frac{f(x_{n+1})-f(x_n)}{T_n}}, & \text{if } f(x_{n+1}) > f(x_n), \\ 1, & \text{otherwise.} \end{cases} \tag{21}$$

Here $T_n = T_0/\log_2(1+n)$.

Using simulated annealing (21) we get the same asymptotic result (19) as in the Bayesian case, if $T_0$ is large enough. However the simulated annealing algorithm is much simpler, as compared with Bayesian algorithm (18). The reason is that we designed the Bayesian approach for a finite, not large number of observations. It extrapolates the expected gain over the entire feasible area $A$ after each observation.

The simulated annealing is an asymptotic method. It is similar to local stochastic methods. The main difference is that the simulated annealing algorithm may go "up" with some probability defined by expression (21). In this way the algorithm gets out of the local minimum. The field of application of the simple asymptotic simulated annealing is "cheap" functions, in the case of a great number of observations $n$. The field of application of the complicated Bayesian approach is "expensive" functions, when $n$ is small.

We use the term the Bayesian approach as a synonym of the average case analysis. It is a very broad extension of the term. However, this extension does not contradict the basic Bayesian idea "to give a composite picture of final beliefs about the unknown parameters using the posterior which combines the prior beliefs about the unknown parameters with the information about the parameters contained in the sample", see Berger (1985). We shall take care to clearly define the "proper" and "improper" assumptions. Thus, the reader may see which results fit into the proper Bayesian framework and which do not, and to what extent.

It is not so easy to draw the exact line dividing the proper and improper Bayesian ideas. For example, the classical uniform prior on the line of real numbers is improper. Therefore to narrow the definition of Bayesian approach to the set of only "proper" assumptions is not convenient. A narrow definition not only restricts the possible applications but also limits new ideas for a future theoretical development. Some assumptions that seem improper now, may be useful when extending the mathematical base of the Bayesian approach and the average case analysis in the future.

**2.6. Application to discrete problems.** Many applied optimization problems are discrete. We can extend the Bayesian approach directly, defining the feasible set $A$ as the discrete set and minimizing the risk function $R(x)$. It may

work if the objective $f(x)$ is a continuous function but the feasible set $A \in R^m$ is discrete. This is an "easy" case, because the neighborhood is uniquely defined and we may use the statistical model which we did use in the continuous case. Here we minimize the risk $R(x)$ on the discrete set $A$.

More difficult are combinatorial problems when we define the objective only for discrete, usually boolean variables $x$. The trouble is that here both the neighborhood and the distance could be defined in many ways. Thus the result depends mainly on the definition of neighborhood and distance. Sometimes we can guess how to do that, see Stoyan and Sokolovskij (1980). Usually we can't. Consequently we should look for other ways. First we mention two important well known cases of using the asymptotic Bayesian approach in the discrete optimization.

The first case is the improvement of the conventional simulated annealing techniques of discrete optimization using the asymptotic Bayesian approach, see Van Laarhoven *et al.* (1989). Here an observation is the result of a Markov chain generated by the simulated annealing algorithm. To make the Bayesian analysis less complicated the authors use assumptions which are only asymptotically satisfied. The results improve the cooling schedule $T_n$. The ratio between the CPU time for the Bayesian and for the non-Bayesian schedule is between 1.2 and 1.5. The difference is due to additional computational efforts involved in the Bayesian approach. In general, we think that asymptotics is not a strong point of Bayesian techniques. It is well known that by increasing the number of observations we usually make the prior less important.

For other case of the asymptotic Bayesian approach to the discrete optimization, see Karp (1976). The author considers the Euclidean travelling–salesman problem assuming the uniform distribution of $n$ points on the unit square. The technique is:

1. Subdivide the unit square into a regular grid of $n/t(n)$ subsquares, where $t \sim \log_2 \log_2 n$.
2. Using the dynamic programming, construct the shortest route through the set of points in each subsquare.
3. Construct a minimum–length spanning tree joining the subroutes.
4. Construct a closed walk that traverses each subroute once, and each tree edge twice.

There are two algorithms. The first one, the exact dynamic programming algorithm constructs the minimal walk between the $t(n) \sim \log_2 \log_2 n$ points inside each subsquare. Then a simple algorithm of the spanning tree connects the $n/t(n)$ subroutes.

The author proves that the technique solves the Euclidean travelling–salesman problem within $1+\epsilon$ and runs within time $O(n \log n)$ with probability one. Here $\epsilon$ is a deviation from the minimal walk. It means that the Bayesian approach provides asymptotically exact results for almost all the sets of points within the low–degree polynomial time. The problem is NP-complete, thus, if we wish to get the exact solution for all points, then we need algorithms running within the exponential time.

The result is very important theoretically. It shows a great difference between the notions of conventional and average complexity. The technique can be useful in solving very large problems, when the asymptotics works. When solving not large travelling–salesman problems we can get a significant deviation $\epsilon$ for many sets of points. For example, if $n = 1024$, then one subsquare contains just $\log_2 \log_2 1024 \sim 5$ points. It means that when solving this problem we connect 205 subroutes by the spanning–tree algorithm, which is approximate to the travelling–salesman problem. We use the exact dynamic programming techniques 205 times wile solving very small problems and getting the minimal walk for 5

points within the subsquare. The size of subsquares grows very slowly, as a double logarithm.

We see now that the Bayesian techniques can be directly used in some discrete optimization problems. That is, in very large problems, when the asymptotics works, and also in the problems, when the "distances" between the decisions are well defined. Now we consider the indirect Bayesian approach by reducing the discrete optimization problem to the problem of the continuous global optimization. We can do that using randomized heuristics algorithms.

We regard the discrete optimization as a multi-stage decision problem. At each stage we define the heuristics as a simple function which roughly predicts the consequences of decisions. The probability of a decision depends on the heuristics. We fix the randomized decision function, with an exception of some parameters. We repeat the sequence of randomized decisions several times at fixed values of those parameters and regard the best decision as a result. We optimize the parameters by the Bayesian methods of continuous global optimization. Here we use the Bayesian methods to adapt the continuous unknown parameters to the given family of discrete optimization problems, see Mockus (1993).

The randomized heuristics algorithms might be useful for the continuous global optimization, too. If the objective is "cheap"[2] , then the auxiliary problem of risk $R(x)$ minimization may be more complicated as compared with the original problem (1). In such cases we fix some decision, make perturbations and define heuristics as the difference between the results after and before the perturbation. A simple example is the simulated annealing algorithm (21). Here the parameter of randomization is $T_0$. We optimize this parameter by the Bayesian techniques.

[2] "Cheap" means that the calculation of $f(x)$ given $x$ is not expensive in terms of CPU time.

**3. Sequential decision problem of discrete optimization.** We consider a multi-stage decision problem. Describing discrete optimization problems we denote by $v(d)$ the objective[3] corresponding to the decision $d$.

$$v^* = \min_{d \in D} v(d). \tag{22}$$

At each stage $i = 1, \ldots, I$ we chose one of $M_i$ feasible decisions $d_i = d_i(m), m \in \{1, \ldots, M_i\}$. The set of feasible decisions at the stage $i$ we denote by $D_i = \{d_i(1), \ldots, d_i(M_i)\}$.

The set $D_i$ usually depends on all the previous decisions, which we shall represent as the vector $d^i = (d_1, \ldots, d_{i-1})$. Let us denote by $d = d^{I+1} = (d_i, i = 1, \ldots, I)$ a sequence of decisions. The set $D$ defines all the sequences of feasible decisions $d$. Thus at each stage $i$ we select a branch $d_i$ of some decision tree. We consider two algorithms: sequential and iterative.

The sequential algorithm builds the system from the scratch, adding one element at each stage. Thus the number of decision stages $I$ is equal to the number of systems elements.

The iterative algorithm starts from a feasible initial decision. At each stage we make some permutation. The iterative algorithm updates the current feasible decision by accepting one of those permutations. In the iterative case the set of feasible decisions $D_i$ at the stage $i$ corresponds to the set $M_i$ of feasible permutations $m \in \{1, \ldots, M_i\}$. Here the decision $d_1$ denotes the initial decision. The decision $d_{i+1}, i = 1, \ldots, I$ denotes the decision taken at the stage $i$. At the stage $i$ we update the previous decision $d_i$ by choosing one of feasible permutations $m \in M_i$.

---

[3]In the continuous optimization we used different notation: we denoted by $f(x)$ the objective , given decision $x$.

The permutation $m$ is feasible $m \in \{1, \ldots, M_i\}$ if the corresponding decision $d_i(m)$ is feasible $d_i(m) \in D_i$. The set of feasible decisions $D_i$ usually depends on the previous decision $d_{i-1}$. We stop, if we can't improve the results by using the given set of permutations. In such a case the number of stages may be considerably larger than the number of system elements.

**4. Some methods of the discrete optimization.** Denote by $v_i$ the conditional minimal value of the objective function for fixed $d^i$

$$v_i = \min_{d \in D^i} v(d), \quad \text{for fixed } d^i. \tag{23}$$

Here $D^i \subset D$ is the set of feasible decisions $(d_i, \ldots, d_I)$ when the decisions $d^i = (d_1, \ldots, d_{i-1})$ are fixed. Therefore $v = v_1$.

Denote by $v_i(m)$ the conditional minimal value of the objective function for both fixed $d^i$ and $d_i = d_i(m)$

$$v_{i+1} = \min_{1 \leqslant m \leqslant M_i} v_i(m). \tag{24}$$

The high complexity of exact techniques (both the dynamic programming algorithm and the branch-and-bound algorithm) is the reason to seek simpler approaches. For example, we can apply the randomized technique where the decision $d_i(m)$ is made with some probability $r_i(m)$. We may repeat the randomized decisions many times and regard the best decision as the optimization result.

If $r_i$ is positive for all feasible decisions, $r_i \geqslant \epsilon > 0$ and repetitions are unlimited, then the randomized technique converges to the best value $v$ with probability one. In the sequel referring to the convergence we omit the words "with probability one". Usually the convergence is very slow, therefore

we stop far from the optimal value $v$, since the repetitions are limited.

Suppose that there exists a simple function $h_i(m)$ that roughly predicts the consequences of the decision $m$. We may scale the heuristics in the unit interval, then

$$\min_m h_i(m) = 0, \quad \max_m h_i(m) = 1, \; i = 1, \ldots, I. \tag{25}$$

We can leave the function $h_i(m)$ unscaled, too.

Applying the heuristics in iterative algorithms we may calculate the objective function (or its estimate) for all feasible permutations, or some part of them. We also may stop at the first improvement. Sometimes we can prefer to do just one permutation[4] . We need here fewer calculations, but the choice is minimal. Just two alternatives: to go or not to go to the next point.

We may optimize twice. The first time we use a sequential algorithm that starts from the empty set. The second time we improve the solution by feasible permutations. We can eliminate the first part of optimization, if we get a good initial solution.

Usually we apply the heuristics for choosing good decisions, see the next section. However, we can use them for cutting off bad decisions too, see Mockus (1993).

## 5. Algorithm of randomized heuristics.

**5.1. General description.** If we relate probabilities $r_i(m)$ to the heuristics $h_i(m)$, then we can expect a faster convergence. As a first approximation assume that the probabilities $r_i(m)$ are proportional to the heuristics $h_i(m)$. This assumption means the linearity of $r_i$ as function of $h_i$. In general, we have to consider nonlinear functions $r_i = r(h_i)$, too. We define a family of functions $r_i = r(h_i)$ by a fixed number

[4] Or just one perturbation, as in the continuous simulated annealing.

of parameters $x = (x_n, n = 1, \ldots, N)$. Then we can write $r_i(m) = r(m, h_i, x)$. Here

$$\sum_{m \in M_i} r_i(m) = 1. \tag{26}$$

For example, we can execute the decision algorithm including condition (26) by the following three steps:

i. Partition the unit interval $U = [0, 1]$ into parts $U_i(m)$, $m = 1, \ldots, M_i$ equal to the probabilities $r_i(m)$,
ii. Generate a random number $\xi_i$ uniformly in the unit interval.
iii. Choose the decision $d_i(m)$, if $\xi_i \in U_i(m)$.

**5.2. Polynomial randomization.** Now the problem is to improve the expression of probabilities $r_i$ as a function of heuristics $h_i$. We wish to represent this function using the parameter $x$ with a minimal number of components. In such a case the orthogonality of components is desirable. We think that in the scalar heuristics case the Legendre polynomial $L_n$ could be suitable, see Mockus *et al.* (1994).

The disadvantage of Legendre representation is that the probability $r_i$ is not a monotonous function of the heuristics. We can understand the results easier if the components of the expression $r_i$ are monotonous.

An ordinary polynomial can better correspond to our task if we prefer the probabilities $r_i$ to be expressed as some monotonous functions of the heuristics $h_i$.

$$r_i^0 = r^0(h_i, x) = \sum_{n=0}^{N-1} a_{in} x_n h_i^n(m), \tag{27}$$

where

$$\sum_{n=0}^{N-1} x_n = 1, \quad x_n \geqslant 0, \; n = 0, \ldots, N-1, \tag{28}$$

and from condition (26)

$$a_{in} = \frac{1}{\sum_{m=1}^{M_i} h_i^n(m)}.$$

The number $n$ in expression (27) may be regarded as the "degree of greed". The number $n = 0$ means no greed, because all feasible decisions are equally desirable. If the number of greed $n$ is large, then we prefer the decisions with the best heuristics. By optimizing $x$ we define degrees of greed such that provide the most efficient randomized decision procedure.

**5.3. Discontinuous randomization (simulated annealing).** So far we considered only continuous functions $r^0$ of heuristics $h_i$. Let us to consider some discontinuous functions, too. A well known discontinuous function is defined by the simulated annealing. We define the heuristics $h_i(m)$ as the difference between the values of the objective before and after a random perturbation at the stage $i$.[5] Here we do not scale the heuristics $h_(m)$, therefore

$$h_i(m) = v(d_i(m)) - v(d_{i-1}).$$

Here $v(d_i(m))$ is the objective at he stage $i$,if we choose the permutation $m$, and $v(d_{i-1})$ is the objective before the permutation. From (21) we can write

$$r_i^0 = \begin{cases} e^{-\frac{h_i(m)}{x/\ln(1+i)}}, & \text{if } h_i(m) > 0 \\ 1, & \text{otherwise.} \end{cases} \qquad (29)$$

where the initial "temperature" $x$ is the only parameter to be optimized.

There could be several possible explanations why the randomization (29) is used so wide. The most obvious one is the

[5]Corresponding to the usual conditions of simulated annealing.

convenience to prove asymptotic results. How well randomization (29) works for a finite number of iterations is not obvious. We usually need some additional experimentation.

**5.4. Transformation to "unit box".** Suppose that we prefer to transform the set of feasible values of $x$ defined by condition (28) to the "unit box" $A$.

$$A = \{x^0 : 0 \leqslant x_n^0 \leqslant 1, \quad n = 0, \ldots, N-1\}. \tag{30}$$

We can do that by a simple transformation

$$x_n = \frac{x_n^0}{\sum_{n=0}^{N-1} x_n^0}, \quad n = 1, \ldots, N-1. \tag{31}$$

Expression (31) transforms the original function $f(x)$, defined on the set (28), into some other function $f^0(x^0) = f(x^0(x))$ defined on the $N$-dimensional unit cube (30). The notation $x^0(x)$ means that $x^0$ depends on $x$ by expression (31).

Let us fix some $x$ on the set (28). Then the values of the function $f_0(x^0)$ remain equal to the value $f(x^0(x))$ for all $x^0$ which satisfy condition (31) at the fixed $x$.

It follows from condition (31) that for the fixed $x = (x_0, \ldots, x_{N-1})$

$$x_n^0 = \frac{x_n}{x_0} x_0^0, \quad n = 1, \ldots, N-1.$$

Fixing $x_0^0 = 0$ we get the zero point $x_n^0 = 0$, $n = 1, \ldots, N-1$. Fixing $x_0^0 = x_0$ we get the point $x_n^0 = x_n, n = 1, \ldots, N-1$, i.e., $x^0 = x$. Expression (31) defines the line in the set $A$ which goes from zero through the point $x^0 = x$.

This property of the function $f^0(x^0)$ defined by transformation (31) usually is not very important applying global optimization methods. It does not contradict the basic assumptions of the Bayesian methods designed for the global

optimization on the unit box, see Mockus (1989). The peculiarity of the function $f^0(x^0)$ related to transformation (31) seems less important as compared to computational advantages of the unit box.

**5.5. Convergence of algorithms.** We regard $r_i(m)$, $i = 1, \ldots, I$ as a probability distribution on the set $1, \ldots, M_i$ of feasible decisions. Expressions (27)–(29) define the multistage randomized procedure $r = (r_i,\ i = 1, \ldots, I)$ as a function of some parameters $x$, so we can write $r = r(x)$.

Denote by $v(K, x)$ the expectation of the best value of the objective function if we repeat the randomized decision $r(x)$ with a fixed parameter $x$ $K$ times. We assume $r(x)$ to be such that the probability of any feasible decision were positive. It means that the best value of the objective function converges to the exact optimal value if the number of repetitions $K$ is large.

$$\lim_{K \to \infty} v(K, x) = v, \quad \text{for any} \quad x. \tag{32}$$

The exact computation of $v(K, x)$ is no less complicated as the solution of the original discrete optimization problem (22). A natural and convenient way to estimate $v(K, x)$ is by Monte Carlo simulation. Let $f_K(x)$ be the best value of the objective function obtained after $K$ repetitions of the randomized decision procedure $r(x)$ at the fixed $x$ The expectation of $f_K(x)$ is equal to $v(K, x)$ by definition.

**6. Reduction of the discrete optimization to the continuous stochastic one.** We denoted by $f_K(x)$ the best value of the objective function obtained after $K$ repetitions of the randomized decision procedure $r(x)$ for a fixed $x$. We defined the randomized decision procedure so, that

$$\lim_{K \to \infty} f_K(x) = v, \quad \text{for any fixed} \quad x. \tag{33}$$

It means that all the values of parameters $x$ are "good" asymptotically. However, the right choice of $x$ could be very important, if the number of repetitions $K$ is not large.

Thus we get the following continuous problem of the stochastic programming

$$\min_x f_K(x),$$

where

$$x = (x_0, \ldots, x_{N-1}), \quad \sum_{n=0}^{N-1} x_n = 1, \quad x_i \geqslant 0, \quad n = 0, \ldots, N-1. \tag{34}$$

Thus we reduced, in the described sense, the discrete optimization problem (22) to the problem of continuous stochastic optimization (6). Let $R$ be the number of iterations of the stochastic optimization. So the total number of observations $K_T$ (calculations of the objective function $v(d)$ for the fixed decision $d$) is the product of iterations $R$ and repetitions $K$, therefore $K_T = RK$. If $K = 1$, then the iteration and the observation have the same meaning.

Now we can replace condition (33) by a weaker condition

$$\lim_{K_T \to \infty} f_K(x) = v, \quad \text{for any fixed } x \text{ and } K \geqslant 1. \tag{35}$$

This expression follows directly from the convergence condition $r_i^0(m) \geqslant \epsilon > 0$, $n = 1, \ldots M_i$, $i = 1, \ldots, I$.

Condition (35) means, that the method will achieve the global optimum for any fixed number of repetitions $K \geqslant 1$. So the results of optimization may depend a reat deal on the number of repetitions $K$. Therefore for any specific problem some preliminary investigation is desirable to define the right number of repetitions $K$.

We can carry out the stochastic optimization in three steps.

We expect the algorithm of randomized heuristics to be efficient only if $v_i(m)$ statistically depends on some simple heuristics $h_i$. Thus, as a first step we investigate the "quality" of heuristics by a simplified representation of probabilities $r_i(m)$:

$$\begin{aligned} r_i^* = r_i^*(m) &= r^*(m, h_i, x) \\ &= a_{i0}x_0 + a_{i1}x_1 h_i(m) + a_{i2}x_2 \Delta_i(h_i(m)), \end{aligned} \tag{36}$$

where

$$\Delta_i(h_i(m)) = \begin{cases} 1, & \text{if} \quad h_i(m) = \max_{k \in 1,\ldots,M_i} h_i(k), \\ 0, & \text{otherwise.} \end{cases} \tag{37}$$

and

$$a_{i0} = \frac{1}{M_i}, \quad a_{i1} = \frac{1}{\sum_{m=1}^{M_i} h_i(m)}, \quad a_{i2} = 1. \tag{38}$$

We consider the heuristics $h_i$ as useless if the optimal value of the zero component $x_0$ obtained using simplified expressions (36)–(38) is close to one. The reason is that we have got the best result using the simplest uniform Monte Carlo method, without any heuristics.

We regard the heuristics $h_i$ very good if the optimal value of the second component $x_2$ is close to one. Here we have got the best result by choosing a decision that corresponds to the best heuristics, without any randomization and parametrization.

The heuristics $h_i$ could be improved if the optimal value of the component $x_1$ were cosiderabky more than zero. In such a case we go over to a second step of optimization.

At the second step we use the randomized procedure $r(x)$ with probabilities defined by expression (27). The optimization of parameters $x$ should make the search more efficient.

If we see that the optimization of parameters $x$ does improve the results significantly, then we go over to third step.

At the third step we repeat the randomized decision procedure $r(x)$ at fixed values of $x$, which we had obtained at the second step of optimization.

Some optimization steps can be omitted depending on specific conditions of the problem.

The number of repetitions $K$ may be different at different steps, that is $K_1, K_2$, *and* $K_3$. The total number of observations

$$K_T = K_1 R_1 + K_2 R_2 + K_3.$$

Here $R_1$ is the number of iterations at the first step of optimization and $R_2$ is the number of iterations at the second step.

The right choice of repetition numbers $K_1$, $K_2$, and $K_3$ depends on several factors. An important factor is the "cost" of one observation. We may set the repetition numbers $K_1 = K_2 = 1$, and $K_3 = 0$, if the cost of an observation is higher than that of auxiliary computations. We use the auxiliary computations to define the next value of $x$ at each iteration of optimization. The Bayesian methods need a lot of such computations.

If the observations are "cheap", then we ought to set larger numbers of repetitions $K_1, K_2$. The reason is to make the cost of observations at each iteration close to the cost of auxiliary computations . The auxiliary computations are defined by the optimization method, we do not control that. However we can increase the cost of observations in one iteration by increasing the number of repetitions. We may improve the result by

setting a large number of "post-optimization" repetitions $K_3$, if observations are cheap.

The described methods are of great importance when solving families of related discrete optimization problems. Here we may optimize $x$ for only one typical problem of the family, applying the optimal values of $x$ to many related problems. Thus, we can reduce the optimization "cost" by dividing it among many problems. In the examples of this paper we did optimize the parameters $x$ of each problem individually. Therefore we may see only a part of the advantages of the Bayesian approach.

**7. Methods of the global stochastic optimization.** There are several specific features of problem (6), (34) that make the direct optimization difficult:

i. The function $f_K(x)$ is stochastic,
ii. The function $f_K(x)$ is multi-modal, as usual,
iii. The function $f_K(x)$ is expensive (one observation requires a lot of calculations)

There exist the techniques of stochastic approximation (Ermoljev, Wets, 1988) that converge to some local minimum. There also exist methods of global stochastic optimization that converge to the global minimum. Thus, both conditions i. and ii. can be met.

The convergence of conventional methods of global stochastic optimization, such as the global stochastic approximation or simulated annealing, is usually very slow, see Žilinskas and Zygliavsky (1992). Sometimes the convergence condition appears to be almost irrelevant, if the number of iterations is not large enough. The reason is that those methods are meant for the optimization of inexpensive functions. If the function is not expensive, then the number of iterations is usually large and the asymptotics is important.

To meet all the three conditions we need a search technique such that tends to minimize the expected deviation from

the global minimum when the number of iterations (evaluations of the function $f_K(x)$) is small. We call such techniques the Bayesian methods of global optimization, see Mockus (1989).

We call the application of those methods to problem (6), (34) the Bayesian approach to the discrete optimization. In this definition we mean that the Bayesian methods of global optimization are efficient when solving general problem (6), (34). "Non-Bayesian" methods of global stochastic optimization could fit better for many special problems (6), (34).

The main advantage of the Bayesian approach to discrete optimization is that it shows how to involve the expert knowledge about the problem into a general mathematical framework. The expert knowledge we include into the heuristics $h_i$.

The mathematical part of the problem considers mainly the optimization of (6), (34). If the choice of heuristics $h_i$ is good, then the optimization of (6), (34) helps solve (22) more efficiently. If not, then the results of the optimization of (6), (34) can show that the heuristics $h_i$ is irrelevant to the discrete optimization programming problem (22).

**8. Mixed Boolean Bilinear Programming(MBBLP).** By the term "Bilinear" we mean the minimization of a sum of products of two different variables. By the term "Mixed Boolean" we mean that some of the variables are Boolean, just zero or one. We denote by $x = (x_i,\ i = 1,\dots,n_x)$, $w = (w_i,\ i = 1,\dots,n_w)$ the real valued variables. We denote by $y = (y_i,\ i = 1,\dots,n_y)$, $z = (z_i,\ i = 1,\dots,n_z)$ the Boollean variables.

We suppose that the real variables $w = (w_i, i, \dots, n_w)$ multiply only by real variables. The Boolean variables $z$ mutiply only by Boolean variables. This partition of variables is just for the convenience of formulation. We suppose, for simplicity of the description that $n_x = n_w = n_y = n_z = n$.

Then:

$$\min_{x,w,y,z}\Big(\sum_{i=1,n}(a_{0i}x_i + b_{0i}y_i + c_{0i}z_i + d_{0i}w_i) + \sum_{i,j=1,n}(a_{0ij}x_iy_j + b_{0ij}y_iz_j + d_{0ij}x_iw_j)\Big) \quad (39)$$

$$\sum_{i=1,n}(a_{ki}x_i + b_{ki}y_i + c_{ki}z_i + d_{ki}w_i) + \sum_{i,j=1,n}(a_{kij}x_iy_j + b_{kij}y_iz_j + d_{kij}x_iw_j) \geqslant 0, \quad k = 1,\ldots,m \quad (40)$$

$$x_i \geqslant 0, \quad w_i \geqslant 0, \quad y_i \in \{0,1\}, \quad z_i \in \{0,1\} \quad (41)$$

We assume that $a_{ki} \neq 0$, $b_{ki} \neq 0$, $d_{ki} \neq 0$, $c_{ki} \neq 0$, $k = 0,\ldots,m$, $i = 1,\ldots,n$. We also suppose that most of three-index parameters such as $a_{kij}$, $b_{kij}$, $c_{kij}$, $d_{kij}$ are zero. Otherwise, too much additional inequalitys could be needed later, when reducing MBBLP to the standard Mixed Integer Linear Programming (MILP).

**8.1. Solution of MBBLP as a sequence of Mixed Boolean Partly BiLinear Programming problems (MBPBLP).** We mean by the term "Partly Bilinear" that there are no products of real variables. That help us, when reducing MBPBLP to the standard MILP. Formaly we can define MBPBLP just by deleting the variables $w$ in (1)–(3)

$$\min_{x,y,z}\Big(\sum_{i=1,n}(a_{0i}x_i + b_{0i}y_i + c_{0i}z_i) + \sum_{i,j=1,n}(a_{0ij}x_iy_j + b_{0ij}y_iz_j)\Big), \quad (42)$$

$$\sum_{i=1,n} (a_{ki}x_i + b_{ki}y_i + c_{ki}z_i)$$
$$+ \sum_{i,j=1,n} (a_{kij}x_iy_j + b_{kij}y_iz_j) \geqslant 0,$$
$$k = 1, \ldots, m, \tag{43}$$
$$x_i \geqslant 0, \quad y_i \in \{0,1\}, \quad z_i \in \{0,1\}. \tag{44}$$

We solve MBBLP as a sequence of corresponding MBP-BLP problems by the following iterative algorithm:

1. Fix some initial value of $w = w^0$ in expressions (1)–(3).
2. Get the optimal value $x = x^0$, given $w = w^0$.
3. Get the optimal value $w = w^1$, given $x = x^0$.
4. Stop, if there is no improvement, go to step 1, otherwise.

The disadvantage of this technique is that for some initial $w = w^0$ there might be no feasible solutions. We may succeed if we carefully select $w^0$. We may relax the constraints and minimize the auxiliary objective, otherwise. The auxiliary objective includes the original one, plus the distance to the original feasible set. We describe the auxiliary objective later on, when considering the Combinatorial Linear Programming.

The alternative technique is the Outher Approximation algorithm, see Reklaitis and Mockus (1994). However the Outher Approximation is rather a general technique. So we can expect it to be less efficient as compared to the algorithm, designed specifically for MBBLP.

**8.2. Reduction of MBPBLP to MILP.** There are well documented algorithms for the MILP problems. We reduce here the MBPBLP problem to the MILP problem.

$$\min_{x,y,z,u,v} \Big( \sum_{i=1,n} (a_{0i}x_i + b_{0i}y_i + c_{0i}z_i)$$
$$+ \sum_{i,j=1,n} (a_{0ij}v_{ij} + b_{0ij}u_{ij}) \Big), \tag{45}$$

$$\sum_{i=1,n}(a_{ki}x_i + b_{ki}y_i + c_{ki}z_i) + \sum_{i,j=1,n}(a_{kij}v_{ij} + b_{kij}u_{ij}) \geqslant 0,$$
$$k = 1, m, \tag{46}$$

$$v_{ij} \leqslant Ky_j, \quad u_{ij} \leqslant y_i, \quad u_{ij} \leqslant z_j, \tag{47}$$

$$u_{ij} \geqslant y_i + z_j - 1, \tag{48}$$

$$x_i \geqslant 0,\ v_{ij} \geqslant 0,\ y_i \in \{0,1\},\ z_i \in \{0,1\},\ u_{ij} \in \{0,1\}, \tag{49}$$

where $K$ is a large number.

We can solve (7)–(11) using the MILP algorithm. However we have to stop before reaching the global minimum in many real problems. In such cases we may regard the "incumbant" as an approximate solution of MILP. We call such technique as the "truncated b&b", we stop the "branch-and-bound" procedure.

It looks like a very convenient deterministic technique. We proceed until we reach the time limit. Then we stop and get an approximate solution. However b&b is designed to get the exact solution. It is not clear how good is the "incumbant" as the approximate solution. The convergence to the exact solution does not necessarily means a good approximation, if we stop before reaching the exact solution.

If we wish to get the good approximation, then we shoud consider this objective when designing the algorithm. A convenient way to do that is designing randomized heuristics. The heuristics helps to involve the expert intuition. The randomization gives the flexibility of design. The optimization of the decision parameters adapts the algoritm to the given family of problems.

**8.3. Reduction of MBPBLP to Combinatorial Linear Programming (CLP).** We may consider the penalty

function as a sort of heuristics, if it is too difficult to keep inside the feasible region. We define the penalty function as follows. We fix the values of discrete variables $y, z$ and minimize the original linear objective (4) plus the sum of violations of constraints (5) multiplied by the factor $r > 0$. We denote the result as $L(y, z)$. From this definition and expressions (4)–(6) it follows that

$$L(y,z) = \min_x \Big( \sum_{i=1,n} (a_{0i}x_i + b_{0i}y_i + c_{0i}z_i) + \sum_{i,j=1,n} (a_{0ij}x_iy_j + b_{0ij}y_iz_j)) + r \sum_{k=1,m} s_k \Big), \quad (50)$$

$$s_k \geqslant -\Big( \sum_{i=1,n} (a_{ki}x_i + b_{ki}y_i + c_{ki}z_i) + \sum_{i,j=1,n} (a_{kij}x_iy_j + b_{kij}y_iz_j) \Big), \quad k = 1, \ldots, m, \quad (51)$$

$$x_i \geqslant 0, \quad s_k \geqslant 0, \quad y_i \in \{0,1\}, \quad z_i \in \{0,1\}. \quad (52)$$

The advantage of CLP is that all the boolean decisions $y, z$ are "feasible". Here the "feasibility" of $y, z$ means that we define some auxiliary objective $L(y, z)$ for any fixed $y, z$. We may see from (12) and (4) that the auxiliary objective (12) is the minimum of the original objective (4) plus the "distance" from original feasible set (5). We define the distance from set (5) as the sum of constraint violations $s_k$.

Now we apply some randomized heuristics to minimize $L(y, z)$ as a function of Boolean variables $y, z$. We consider the algorithm that solves CLP in five steps:

1. Fix a vector $(y^0, z^0)$.
2. Generate a "perturbation" vector $(y^l, z^l)$.
3. Define heuristics $h_l$ for the perturbation $(y^l, z^l)$. A simple and natural way is to assume the heuristics to be equal to the auxiliary objective $h_l = L(y^l, z^l)$.

4. Go ower to perturbation $l$ with probability $r_l$ defined as some function of heuristics $h_l$.
5. Repeat the randomized technique many times optimizing the parameters of the randomization function by Bayesian methods, see Mockus *et al.* (1994).

The simplest example of this algorithm is to perturb the Boolean variables $(y, z) = (y_i, z_i, i = 1, \ldots, n)$ one-by-one, then to use the simulated annealing, and to optimize the unknown parameters $(T_o, r)$ by the Bayesian techniques. Here $T_0$ is the initial temperasture of annealing. We can also use more complicated perturbation and randomization algorithms, described by Mockus *et al.* (1994).

We designed the heuristics $h_l = L(y^l, z^l)$ for a family of all MIPBLP problems. For more specific problems we can design simpler heuristics taylored to fit just those problems. For example, the longest remaining time for a given job is apparently the simplest heuristics in some scheduling problems.

It is very important to investigate what approximation is better: solving MBPBLP by the truncated b&b or solving CLP by the Bayesian optimization of randomized heuristics. To do that we have to consider different applied problems. Here we shall consider two of them: the scheduling of batch operations, and the optimization of neural networks.

## 9. Scheduling of batch operations.

**9.1. Scheduling problem.** Scheduling problems occur in a broad range of industries, including the chemical processing industries. Batch production has long been the accepted procedure for the manufacture of many types of chemicals, particularly those which are produced in small quantities and for which the production processes or the demand pattern are likely to change. These products are frequently fine chemicals of high commercial value and it is important that sufficient manufacturing flexibility should be available to avoid the loss

of potential sale by a failure to meet the customer's requirements at the due time. As a result there has been increasing an interest in the development of procedures for scheduling batch process operations.

We consider here a simplifyed version of the scheduling problem described by Reklaitis and Mockus (1994). We use the state–task–network (STN) representation of the batch process. Fig. 1 shows the simplest example. We consider three types of states: feed, intermediary, and product. We regard four types of events, correspondingly: start, finish, receipt, and delivery.

**9.2. Notation.** We denote independent variables by $x$, $y$, $z$, etc., the constants by $a$, $b$, $A$, $B$, etc.

$x^S_{ijk}$ is th input of material when starting the task $i$ in the unit $j$ at the time $t_k$;

$x^F_{ijk}$ is the output of material on finishing the task $i$ in the unit $j$;

$x^R_{sk}$ is the receipt of material in the feed state $s$ at the time $t_k$, usually $x^R_{sk} = a_{sk}$;

$x^D_{sk}$ is the delivery of material in the product state $s$ at the time $t_k$, usually $x^D_{sk} = d_{sk}$;

$y^S_{ijk} =$ 1, if unit $j$ starts the task $i$ at the time $t_k$; $y^f_{ijt} = 0$, otherwise;

$y^F_{ijk} =$ 1, if unit $j$ finishes the task $i$ at the time $t_k$; $y^f_{ijk} = 0$, otherwise;

$B_{sk}$ is the the amount of material stored in the state $s$ at the time $t_k$;

$N_{jk}$ is the number of units $j$ available at the time $t_k$;

$x^S_k = t_k$ is the the time of start event $k$;[6]

$x^F_k = t_k$ is the the time of finish event $k$;

$x^R_k = t_k$ is the time of receipt event $k$, usually $x^R_k = a^t_k$;

$x^D_k = t_k$ is the time of delivery event $k$, usually $x^D_k = d^t_k$;

[6] The time when we start some task.

$c_{sk}$ is the the unit price of material in the state $s$ at the time $t_k$;

$c^S_{sk}$ is the running cost of storing a unit of material in the state $s$ over unit interval[7] starting at the time $t_k$;

$c^U_{uk}$ is the unit cost of utility $u$ in the state $s$ over unit interval[8] starting at the time $t_k$;

$\rho^S_{is}$ is the proportion of input of task $i$ from state $s$;

$\rho^F_{is}$ is the proportion of output of task $i$ to the state $s$;

$\tau_{is}$ is the processing time for the output of task $i$ to the state $s$;

We order the events $k$ by the inequality

$$t_{k+1} \geqslant t_k, \quad k = 0, \ldots, K. \tag{53}$$

### 9.3. Material balance constraints.

$$B_{sk+1} = B_{sk} - d_{sk} + a_{sk} - \sum_i \rho^S_{is} \sum_j y^S_{ijk} xijk^S + \sum_i \rho^F_{is} \sum_j y^F_{ijk} x^F_{ijk},$$

$$\sum_{l \geqslant k} (t_l - t_k) y^F_{ijl} = y^S_{ijk} \tau_{is}. \tag{54}$$

Here $tau_{is}$ is the processing time for the output of task $i$ to state $s$. The constraints simply states that the nett increase $(B_{sk+1} - B_{sk})$ in the amount of material stored in the state $s$ at the time $t_k$ is given by the difference of the amount produced in this state and that used.

---

[7] We assume that the running cost does not change between the events.

[8] We assume that the unit cost does not change between the events.

**9.4. Capacity constraints.**

1. The amount of material stored in the state $s$ must not at any time exceed the maximum storage capacity $B_s$

$$0 \neq B_{sk} \neq B_s \tag{55}$$

2. The amount of material when starting task $i$ in unit $j$ at time $t_k$ is limited by the minimum and the maximum capacities of that unit:

$$y_{ijk}^S B_{ij}^{\min} \leqslant x_{ijk}^s \leqslant y_{ijk}^S B_{ij}^{max}. \tag{56}$$

**9.5. Allocation constraints.** At any time an idle item of equipment can only start at most one task. Of course, if the item does start performing a given task, then it cannot start any other task until the current one is finished, i.e,. the operation is nonpre-emptive. These requirements can be expressed in the following terms

$$\begin{aligned} &N_{jk+1}^A = N_{j,k}^A - \sum_i y_{ijk}^S + \sum_i y_{ijk}^F, \\ &\sum N_{jk} \leq 1, \quad \sum_j y_{ijk}^S \leqslant 1. \end{aligned} \tag{57}$$

**9.6. Objective function.** We maximize the profit, given both delivery and receipt:

$$\begin{aligned} \max_{xy} \Big( &\sum_{k \in D} c_{sk}^D d_{sk} - \sum_{k \in R} c_{sk}^R a_{sk} - \sum_k \sum_s c_{sk}^S B_{sk}(t_{k+1} - t_k) \\ &- \sum_k \sum_u U_{uik} c_{uk}^U \sum_j x_{ijk}(t_{k+1} - t_k) \Big). \end{aligned} \tag{58}$$

We denote by $U_{uik}$ the amount of utility $u$ needed for task $i$ in the interval starting at time $t_k$. We express it as follows

$$U_{uik} = \alpha_{uik} + \beta_{uik} \sum_j x_{ijk}.$$

We denote by $D$ and $R$ the set of delivery and receipt events, correspondingly. We keep to conditions (53), (54), (55), (56), and (57).

We optimize just the last two components of expression (58) describing the storage and the utility cost. The first two components are constants, since both the delivery and the receipt are given. We my use the same algorithm to optimize the delivery and the receipt, too.

**9.7. Illustrative example.** We consider a small example of the batch process described by the state task network in Fig. 1. This is a slightly modified version of the problem, described by Sahidis (1991). Fig. 2 shows the schedule diagram, and Table 1 shows the schedule data.

## 10. Neural network optimization.

**10.1. One-layer network.** We consider here only a part of the network optimization problem[9], just as an example of Boolean bilinear programming. We start from the simplest one-layer linear problem.

$$\min_x \sum_{k=1,n} (s^k)^2, \tag{59}$$

$$s_k = t^k - b^k, \tag{60}$$

$$t^k = 1/m \sum_{j=1,m} a_j^k x_j. \tag{61}$$

Here the number $b^k$ is the output, and the vector $a^k = (a_j^k, j = 1,..,m)$ is the input of the learning set $k = 1,\ldots,n$.

[9]The general problem is described by Hecht–Nielsen (1990).

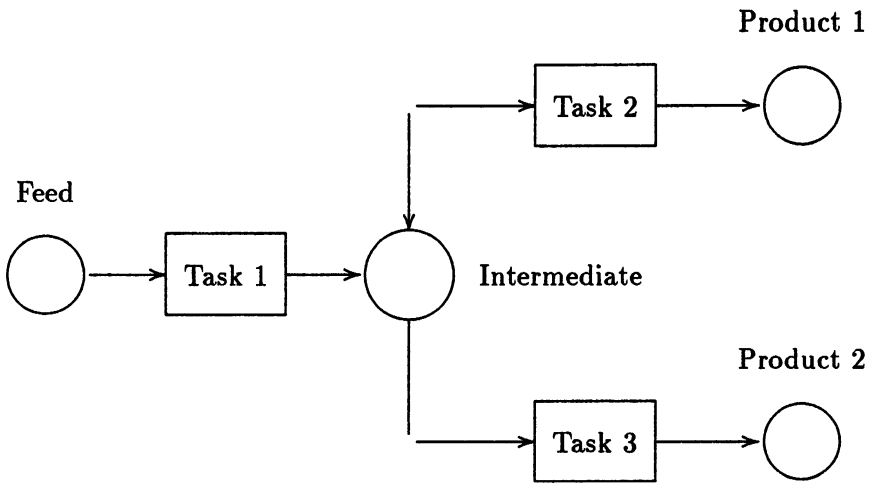


**Fig. 1.** State task network.

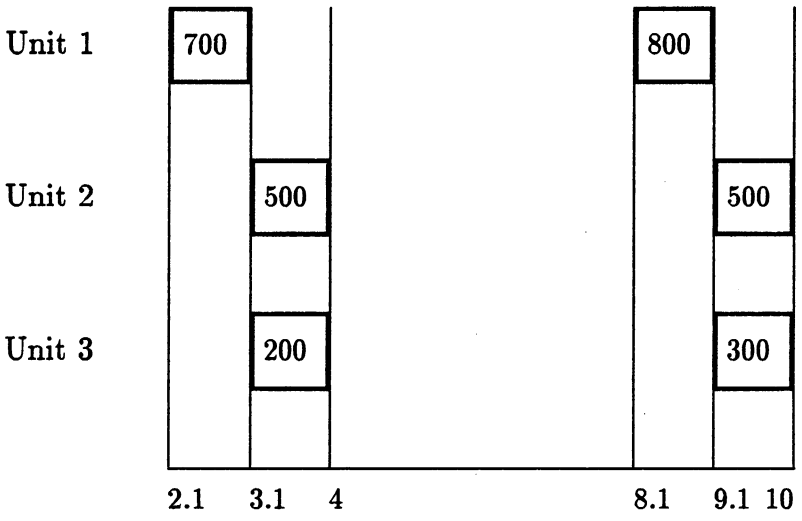


**Fig. 2.** Schedule diagram.

**Table 1.** Schedule data

| Units, tasks | | | |
|---|---|---|---|
| Units | Size | Units suitability | Processing time |
| Unit1 | 1500 | Task1 | 1 |
| Unit2 | 1000 | Task2 | 0.9 |
| Unit3 | 1000 | Task3 | 0.9 |

| States | | |
|---|---|---|
| States | Capacity limits | Prices |
| Feed | Unlimited | 5 |
| Intermediate | 5000 | |
| Product1 | Unlimited | 10 |
| Product2 | Unlimited | 8 |

| States | Demands | | | | | |
|---|---|---|---|---|---|---|
| | Time | | | | | |
| | 4 | 6 | 7 | 10 | 11 | 12 |
| Product1 | 200 | | 300 | 400 | 100 | |
| Product2 | 50 | 150 | | 200 | | 100 |

| Cost data | | |
|---|---|---|
| $\alpha_{u10} = 200$ | $\beta_{u10} = 0.6$ | $c^U_{u1} = 1$ |
| $\alpha_{u20} = \alpha_{u30} = 222.2$ | $\beta_{u20} = \beta_{u30} = 0.67$ | $c^S_{s1} = 0.18$ |

We optimize the vector $x = (x_j, j = 1, \ldots, m)$ to get the minimal square deviation (59). We assume that $x_j, a_j, b^k \in [0, 1]$. We can solve (59) by the standard quadratic programming methods.

If we prefer the linear programming, then we minimize

$$\min_{x,u} \sum_{k=1,n} (u^k), \tag{62}$$

$$u_k \geqslant s_k, \tag{63}$$

$$u_k \geqslant -s_k, \tag{64}$$

$$s_k = t^k - b^k, \tag{65}$$

$$t^k = 1/m \sum_{j=1,m} a_j^k x_j. \tag{66}$$

Here the auxiliary variable $u^k \geqslant 0$.

Both problems (59) and (62) do not take into account the "treshold of neuron sensitivity" factor which is important in neural networks. We can do that by the Boolean linear programming.

$$\min_{x,u,y} \sum_{k=1,n} (u^k), \tag{67}$$

$$u_k \geqslant s_k, \tag{68}$$

$$u_k \geqslant -s_k, \tag{69}$$

$$s_k = y^k - b^k, \tag{70}$$

$$Ly^k \geqslant 1/m \sum_{j=1,m} a_j^k x_j - t, \tag{71}$$

$$L(1 - y^k) \geqslant -(1/m \sum_{j=1,m} a_j^k x_j - t). \tag{72}$$

Here both the output $b^k \in \{0,1\}$ and the auxiliary variable $y^k \in \{0,1\}$ are Boolean, $L$ is the large number, and $t \in (0,1)$ is the threshold of neuron sensitivity. We can solve (67) by the standard mixed integer linear programming (MILP) methods.

**10.2. Two-layer network.** Now we extend (68) to the two-layer neuron network. In the simplest case we have a bilinear programming problem

$$\min_{x,u} \sum_{k=1,n} (u^k), \tag{73}$$

$$u_k \geqslant s_k, \tag{74}$$

$$u_k \geqslant -s_k, \tag{75}$$

$$s_k = t^k - b^k, \tag{76}$$

$$t^k = 1/m \sum_{j=1,m} a_j^k x_j, \tag{77}$$

$$a_j^k = 1/m \sum_i a_{ji}^k x_{ji}. \tag{78}$$

It is a more difficult problem as compared to (62). We may consider the threshold factor $t$, too, by the following mixed Boolean bilinear programming problem MBBLP.

$$\min_{x,u,y} \sum_{k=1,n} (u^k), \tag{79}$$

$$u_k \geqslant s_k, \tag{80}$$

$$u_k \geqslant -s_k, \tag{81}$$

$$s_k = y^k - b^k, \tag{82}$$

$$Ly^k \geqslant 1/m \sum_{j=1,m} a_j^k x_j - t, \tag{83}$$

$$L(1 - y^k) \geqslant -(1/m \sum_{j=1,m} a_j^k x_j - t), \tag{84}$$

$$a_j^k = 1/m \sum_i a_{ji}^k x_{ji}. \tag{85}$$

We can solve it using the algoritm described in the MB-BLP section.

**11. Conclusions.** Using the randomized heuristics we may reduce the discrete optimization to the problem of a continuous stochastic global optimization. We can solve this problem using the well known Bayesian techniques. This is a new approach which helps to involve in a natural way the expert knowledge about a specific problem of discrete optimization into the mathematical framework of optimization. The expert knowledge is very important, if we wish to get a fast approximate solution.

## REFERENCES

Baker, Kenneth R. (1974). *Introduction to Sequencing and Scheduling.* John Willey & Sons, New York.

Biegler, L. T., I. E. Grossmann and G. V. Reklaitis, (1988). Applications of operations reseach methods in chemical engineering. In R.R. Levary (Ed.), *Engineering Design: Better Results Through OR Methods,* North-Holland, Amsterdam.

Betro, B. (1991). Bayesian Methods of Global Optimization. *Journal of Global Optimization,* **1**, 1–14.

De Groot, M. (1970). *Optimal Statistical Decisions.* McGraw–Hill, New York.

Diaconis, P. (1988). "Bayesian Numerical Analysis, *Statistical Decision Theory and Related Topics.* Springer Verlag, pp. 163–175.

Ermakov, S.M., and A.A. Zigliavski (1983). On random search for global extremum. *Probability Theory and Applications,* **83**, 129–136 (in Russian).

Ermoljev, Yu., and R.J-B. Wets (1988). *Numerical Techniques for Stochastic Optimization.* Springer–Verlag, Berlin–New York–London.

Hecht–Nielsen R. (1990). *Neurocomputing.* Addison–Wesley.

Helman, P., B.M.E. Moret and H.D. Shapiro (1993). An Exact Characterization of Greedy Structures. *SIAM J.Disc. Math.,* **6**, 274–285.

Karp, R.M. (1976). The Probabilistic Analysis of some combinatorial search algorithms. In J.F. Traub (Ed.), *Algorithms and Complexity,* Academic Press, New York. pp. 1–20.

Kondili E., C.C. Pantelides and R.W.H. Sargent (1989). *A General Algorithm for Scheduling Batch Operations,* Department of Chemical Engineering, Imperial College of Science and Technology, London.

Kushner, H.J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J. of Basic Engineering,* **86**, 97–106.

Lee, B., and G.V. Reklaitis G.V. Optimal scheduling of batch processes for heat integration. *Computers & Chem. Eng.(to appear)*

Metropolis N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller (1953). Equations of state calculation by fast computing machines. *Journal of Chemical Physics,* **21**, 1087–1092.

Mockus, A., and L. Mockus (1990). Design of software for global optimization. *Informatica,* **1**, 71–88.

Mockus, J. (1967). *Multimodal Problems in Engineering Design.* Nauka, Moscow (in Russian).

Mockus, J. (1972). On Bayesian methods for search of extremum. *Automatics and Computer Technics,* **72**, 53–62 (in Russian).

Mockus, J. (1989). *Bayesian Approach to Global Optimization.* Kluwer academic publishers, Dordrecht–London–Boston.

Mockus, J., and L. Mockus (1991). Bayesian approach to global optimization and applications to multi-objective and constrained optimization. *J. of Optimization Theory and Applications,* **70**, July, No. 1.

Mockus, J. (1993). Application of Global Line Search in Optimization of Networks. In Ding Zhu Du and P.M. Pardalos (Eds.), *Network Optimization Problems,* World Scientific Publishing Co., pp. 169–175.

Mockus, J. (1994). Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization. *Journal of Global Optimization,* 29p. (to appear).

Mockus, A., J. Mockus and L. Mockus. Bayesian approach to discrete optimization. *Journal of Global Optimization,* 23p. (to appear).

Mockus A., J. Mockus and L. Mockus (1994). *Bayessian Approach to Global and Discrete Optimization.* Technical Report, Optimal Decisions Theory Department, Vilnius.

Musier, R.F.H., and L.B. Evans (1989). An approximate method for the production scheduling of industrial batch processes with parallel units. *Computers Chem. Engng,* **13**, 229–238.

Pardalos, P.M., and J.B. Rosen (1987). *Constrained Global Optimization: Algorithms and Applications.* Lecture Notes in Computer Science # 268, Springer–Verlag, Berlin.

Pekny, J.F., D.L. Miller (1991). Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristic methods. *Computers & Chem. Eng.,* **15**, 741–748.

Reklaitis, G.V. (1992). Overview of scheduling and planning of batch process operations. *Proc. NATO Advanced Study Institute on Batch Processing Systems Engineering,* Antalya, Turkey.

Reklaitis, G.V., and L. Mockus (1994). Mathematical Programming Formulation for Scheduling of Batch Operations Based on the Nonuniform Time Discretization. *Technical Report,* School of Chemical Engineering, Purdue University, West Lafayette. 28pp.

Rastrigin, L.A. (1968). *Statistical Methods of Search.* Nauka, Moscow (in Russian).

Sahidis, N.V., and I.E. Grossmann (1991). Reformulation of multiperiod MILP models for planning and scheduling of chemical processes. *Computers Chem. Engng,* **15**, 255–272.

Sobolj, I.M. (1967). On a systematic search in a hypercube. *SIAM Journal on Numerical Analysis,* **16**, 790–793.

Stoyan, Yu.G., and V.Z. Sokolowskij (1980). *Solution of Some Multiextremal Problems by the Method of Narrowing Domains.* Naukova Dumka, Kiev (in Russian).

Strongin, R.G. (1978). *Numerical Methods of Multiextremal Optimization.* Nauka, Moscow (in Russian).

Sukharev, A.G. (1971). On optimal strategies of search of extremum. *Computational Mathematics and Mathematical Physics,* **4**, 910–924.

Taha, H.A. (1975). *Integer Programming.* Academic Press, New York–San Francisco–London.

Torn, A., and A. Žilinskas (1989). *Global Optimization.* Lecture Notes in Computer Science # 350, Springer–Verlag, Berlin.

Van Laarhoven, P.J.M., C.G.E. Boender, E.H.L. Aarts and A.H.D. Rinnooy Kan (1989). A Bayesian Approach to Simulated Annealing. *Probability in the Engineering and Information Sciences,* **3**, 453–475.

Wellons, M.C., and G.V. Reklaitis (1991). Scheduling of multipurpose batch chemical plants. Part 1: Formation of single product campaigns. *Ind. Eng. Chem. Res.*, **30**, 671–688.

Wellons, M.C.. and G.V. Reklaitis (1991). Scheduling of multipurpose batch chemical plants. Part 1: Multiple-product campaign. Formation and production planning. *Ind. Eng. Chem. Res.*, **30**, 688–705.

Zabinsky, Z.B., R.L. Smith and J.F. McDonald (1990). *Improving Hit and Run for Global Optimization.* Working paper, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI.

Zentner, M.G. (1992). An Interval-based Framework for the Scheduling of Batch Chemical Processes. Ph.D. Thesis, Purdue University.

Zigliavskij, A.A. (1985). *Mathematical Theory of Global Random Search.* Leningrad University Press, Leningrad (in Russian).

Žilinskas, A. (1986). *Global Optimization: Axiomatic of Statistical Models, Algorithms and their Applications.* Mokslas, Vilnius (in Russian).

Žilinskas, A. (1986). *Global Optimization: Axiomatic of Statistical Models, Algorithms and their Applications.* Mokslas, Vilnius (in Russian).

Žilinskas, A., and A. Zygliavsky (1992). *Methods of Global Optimization.* Nauka, Moscow (in Russian).

Received March 1994

**A. Mockus** graduated the Moscow Physical–Technical Institute, USSR, in 1988. He is a postgraduate student at the Department of Statistics, Carnegie–Mellon University, Pittsburgh, U.S.A. His field of interest is software systems in statistics and engineering, including the theory and software of statistical animation.

**J. Mockus** graduated Kaunas Technological University, Lithuania, in 1952. He got his Doctor habilitus degree in the Institute of Computers and Automation, Latvia, in 1967. He is a head of Optimal Decision Theory Department, Institute of Mathematics and Informatics, Vilnius, Lithuania and professor of Kaunas Technological University.

**L. Mockus** graduated the Moscow Physical–Technical Institute, USSR, in 1984. He is a postgraduate student at the School of Chemical Engineering, Purdue University, West Lafayette, U.S.A. His field of interest is software systems in optimization and engineering, including the theory and software of the scheduling problems.