Bayesian Approach for Randomization of Heuristic Algorithms of Discrete Programming

Jonas Mockus, Audris Mockus, and Linas Mockus

ABSTRACT. Discrete optimization problems are often solved using "heuristics" (expert opinions defining how to solve a family of problems). The paper is about ways to speed up the search by combining several heuristics involving randomization. Using expert knowledge a prior distribution of optimization results as functions of heuristic decision rules is defined and is continuously updated while solving a particular problem. This approach (BHA or Bayesian Heuristic Approach) is different from the traditional Bayesian Approach (BA) where the prior distribution is defined on a set of functions to be minimized.

The paper focuses on the main objective of BHA that is improving any given heuristic by "mixing" it with other decision rules. In addition to providing almost sure convergence such mixed decision rules often outperform (in terms of speed) even the best heuristics as judged by the considered examples. However, the final results of BHA depend on the quality of the specific heuristic. That means the BHA should be regarded as a tool for enhancing the best heuristics but not for replacing them.

The paper is concluded by a short discussion of Dynamic Visualization Approach (DVA). The goal of DVA is to exploit heuristics directly, bypassing any formal mathematical framework.

The purpose of the paper is to inform the authors inventing and applying various heuristics and about the possibilities and limitations of BHA hoping that they will improve their heuristics using this powerful tool.

1. General Ideas of Bayesian Heuristic Approach

The traditional numerical analysis considers optimization algorithms which guarantee some accuracy for all functions to be optimized. This includes the exact algorithms (that is the worst case analysis). Limiting the maximal error requires a computational effort that in many cases increases exponentially with the size of the problem. The alternative is average case analysis where the average error is made as small as possible. The average is taken over a set of functions to be optimized. The average case analysis is called the Bayesian Approach (BA) [**Dia88, Moc89**].

There are several ways of applying the BA in optimization. The Direct Bayesian Approach (DBA) is defined by fixing a prior distribution P on a set of functions $f(x)$ and by minimizing the Bayesian risk function $R(x)$ [**DeG70, Moc89**]. The risk function describes the average deviation from the global minimum. The distribution P is regarded as a stochastic model of $f(x)$, $x \in R^m$ where $f(x)$ might be a

1991 *Mathematics Subject Classification.* Primary 90C10,90C26; Secondary 62C10,65C05.

deterministic or a stochastic function. In the Gaussian case assuming (see [**Moc89**]) that the $(n+1)$th observation is the last one

$$R(x) = \frac{1}{\sqrt{2}\pi s_n(x)} \times$$

$$\int_{-\infty}^{+\infty} \min(c_n, z) e^{-\frac{1}{2}(\frac{y-m_n(x)}{s_n(x)})^2} dz. \tag{1.1}$$

Here $c_n = \min_i z_i - \epsilon,\ z_i = f(x_i)$, $m_n(x)$ is the conditional expectation given the values of $z_i,\ i = 1, ..., n$, $d_n(x)$ is the conditional variance, $\epsilon > 0$ is a correction parameter and

$$x_{n+1} = \arg\min_x R(x)$$

is a point minimizing the risk (called the Bayesian decission).

The objective of DBA (used mainly in continuous cases) is to provide as small average error as possible while keeping the convergence conditions.

The Bayesian Heuristic Approach (BHA) means fixing a prior distribution P on a set of functions $f_K(x)$ defining the best values obtained using K times some heuristic $h(x)$ to optimize a function $v(y)$ of variables $y \in R^n$ [**MEM+97**]. Usually the components of y are discrete variables. The heuristic $h(x)$ defines an expert opinion about the decision priorities. It is assumed that the heuristics or their "mixture" depend on some continuous parameters $x \in R^m$, where $m < n$. We start the detailed discussion with the Direct Bayesian Approach (DBA) because that is the main instrument of the BHA.

2. Direct Bayesian Approach (DBA)

The Wiener process is common [**Kus64, Sal71, TZ89**] as a stochastic model applying the DBA in the one-dimensional case $m = 1$.

The Wiener model implies that almost all the sample functions $f(x)$ are continuous, that increments $f(x_4) - f(x_3)$ and $f(x_2) - f(x_1)$, $x_1 < x_2 < x_3 < x_4$ are stochastically independent, and that $f(x)$ is Gaussian $(0, x)$ at any fixed $x > 0$. Note that the Wiener process originally provided a mathematical model of a particle in Brownian motion.

The Wiener model is extended to multi-dimensional case, too [**Moc89**]. However, simple approximate stochastic models are preferable if $m > 1$. These models are designed by replacing the traditional Kolmogorov consistency conditions because they require the inversion of matrices of nth order for computing the conditional expectation $m_n(x)$ and variance $d_n(x)$. The favorable exception are the Markov processes including the Wiener one. Extending the Wiener process to $m > 1$ the Markovian property disappears.

Replacing the regular consistency conditions by:
- continuity of the risk function $R(x)$
- convergence of x_n to the global minimum
- simplicity of expressions of $m_n(x)$ and $s_n(x)$

the following simple expression of $R(x)$ is obtained using the results of [**Moc89**].

$$R(x) = \min_{1 \le i \le n} z_i - \min_{1 \le i \le n} \frac{\|x - x_i\|^2}{z_i - c_n}.$$

The aim of the DBA is to minimize the expected deviation. In addition, DBA has some good asymptotic properties, too. It is shown in [**Moc89**] that

$$d^*/d_a = \left(\frac{f_a - f^* + \epsilon}{\epsilon}\right)^{1/2}, \ n \to \infty$$

where d^* is density of x_i around the global optimum f^*, d_a and f_a are average density of x_i and average value of $f(x)$, and ϵ is the correction parameter in expression (1.1). That means that DBA provides convergence to the global minimum for any continuous $f(x)$ and greater density of observations x_i around the global optimum if n is large. Note that the correction parameter ϵ has a similar influence as the temperature in simulated annealing. However, that is a superficial similarity since, using DBA the good asymptotic behavior should be regarded just as an interesting "by-product". The reason is that Bayesian decisions are applied for the small size samples where asymptotic properties are not noticeable.

Choosing the optimal point x_{n+1} for the next iteration using DBA one solves a complicated auxiliary optimization problem minimizing the expected deviation $R(x)$ from the global optimum (see Figure 1). That makes the DBA useful mainly for the computationally expensive functions of a few ($m < 20$) continuous variables. This happens in wide variety of problems such as maximization of yield of differential amplifier, optimization of mechanical system of shock absorber, optimization of composite laminates, estimation of parameters of immunological model and nonlinear time series, planning of extremal experiments on thermostable polymeric composition [**Moc89**].

Using DBA the expert knowledge is included by defining the prior distribution. In BHA the expert knowledge is involved by defining the heuristics and optimizing their parameters using DBA.

3. Bayesian Heuristic Approach (BHA)

Using DBA the expert knowledge is included by defining the prior distribution. In BHA the expert knowledge is involved by defining the heuristics and optimizing their parameters using DBA.

If the number of variables is large and the objective function is not expensive the Bayesian Heuristic Approach (BHA) is preferable. That is the case in many discrete optimization problems. As usual these problems are solved using heuristics based on an expert opinion. Heuristics often involve randomization procedures depending on some empirically defined parameters. The examples of such parameters are the initial temperature if the simulated annealing is applied or the probabilities of different randomization algorithms if their mixture is used. In these problems the DBA is a convenient tool for optimization of the continuous parameters of various heuristic techniques. That is called the Bayesian Heuristic Approach (BHA) [**MEM+97**].

The example of knapsack problem illustrates the basic principles of BHA in discrete optimization. Given a set of objects $j = 1, ..., n$ with values c_j and weights g_j, find the most valuable collection of limited weight.

$$\max_y v(y), \ v(y) = \sum_{j=1}^{n} c_j y_j, \ \sum_{j=1}^{n} g_j y_j \le g.$$

Here the objective function $v(y)$ depends on n Boolean variables $y = (y_1, ..., y_n)$, where $y_j = 1$ if object j is in the collection, and $y_j = 0$ otherwise. The well known greedy heuristics $h_j = c_j/g_j$ is the specific value of object j. The greedy heuristic algorithm: "take the greatest feasible h_j", is very fast but it may get stuck in some non-optimal decision.

A way to force the heuristic algorithm out of such non-optimal decisions is by taking decision j with probability $r_j = \rho_x(h_j)$, where $\rho_x(h_j)$ is an increasing function of h_j and $x = (x_1, ..., x_m)$ is a parameter vector. The DBA is used to optimize the parameters x by minimizing the best result $f_K(x)$ obtained applying K times the randomized heuristic algorithm $\rho_x(h_j)$. That is the most expensive operation of BHA therefore the parallel computation of $f_K(x)$ should be used when possible reducing the computing time in proportion to a number of parallel processors.

Optimization of x adapts the heuristic algorithm $\rho_x(h_j)$ to a given problem. Let us illustrate the parameterization of $\rho_x(h_j)$ using three randomization functions: $r_i^l = h_i^l / \sum_j h_j^l,\ l = 0, 1, \infty$ Here the upper index $l = 0$ denotes the uniformly distributed component and $l = 1$ defines the linear component of randomization. The index ∞ denotes the pure heuristics with no randomization where $r_i^\infty = 1$ if $h_i = \max_j h_j$ and $r_i^\infty = 0$, otherwise. In this case parameter $x = (x_0, x_1, x_\infty)$ defines the probabilities of using randomizations $l = 0, 1, \infty$ correspondingly. The optimal x may be applied solving different but related problems, too [**MEM+97**]. That is very important in the "on-line" optimization adapting the BHA algorithms to some unpredicted changes.

Another simple example of BHA application is by trying different permutations of some feasible solution y^0. In this case heuristics are defined as the difference $h_i = v(y^i) - v(y^0)$ between the permuted solution y^i and the original one y^0. The well known simulated annealing algorithm illustrates the parameterization of $\rho_x(h_j)$ depending on a single parameter x. Here the probability of accepting a worse solution is equal to $e^{-h_i/x}$, where x is the "annealing temperature".

The comparison of BHA with exact $B\&B$ algorithms solving a set of the flow-show problems shows the table 1 from [**MEM+97**]. where S is the number of

TABLE 1. Comparing the BHA and the truncated B&B

$R = 100$, $K = 1$, $J = 10$, $S = 10$, and $O = 10$					
Technique	f_B	d_B	x_0	x_1	x_∞
BHA	6.18	0.13	0.28	0.45	0.26
CPLEX	12.23	0.00	—	—	—

tools, J is the number of jobs, O is the number of operations, f_B, x_0, x_1, x_∞ are the mean results, d_B is the variance, and "CPLEX" denotes the standard MILP technique truncated after 5000 iterations. The table shows that in the randomly generated flow-shop problems the average make-span obtained by BHA was almost twice less that obtained by the exact $B\&B$ procedure truncated at the same time as BHA . The important conclusion is that stopping the exact methods before they reach the exact solution is not a good way to obtain the approximate solution.

The BHA has been used to solve the batch scheduling [**MEM+97**] and the clustering (parameter grouping) problems. In the clustering problem the only parameter x was the initial annealing temperature [**DS90**].

The main objective of BHA is improving any given heuristic by defining the best parameters and/or the best "mixtures" of different heuristics. The heuristic decision rules mixed and adapted by BHA often outperform (in terms of speed) even the best individual heuristics as judged by the considered examples. In addition, BHA provides almost sure convergence. However, the final results of BHA depend on the quality of the specific heuristics including the expert knowledge. That means the BHA should be regarded as a tool for enhancing the heuristics but not for replacing them.

Many well known optimization algorithms such as Genetic Algorithms (GA) [**Gol89**], GRASP [**MPPR97**], and Tabu Search (TS) [**Glo94**], may be regarded as generalized heuristics that can be improved using BHA. There are a number of heuristics tailored to fit specific problems. For example, the Gupta heuristic was the best one while applying BHA to the flow-shop problem [**MEM$^+$97**].

Genetic Algorithms [**Gol89**] is an important "source" of interesting and useful stochastic search heuristics. It is well known [**AV91**] that the results of the genetic algorithms depend on the mutation and cross-over parameters. The Bayesian Heuristic Approach could be used in optimizing those parameters.

In the GRASP system [**MPPR97**] the heuristic is repeated many times. During each iteration a greedy randomized solution is constructed and the neighborhood around that solution is searched for a local optimum. The "greedy" component constructs a solution, one element at a time until a solution is constructed. A possible application of the BHA in GRASP is in optimizing a random selection of a candidate to be in the solution because different random selection rules could be used and their best parameters should be defined. BHA might be useful as a local component, too, by randomizing the local decisions and optimizing the corresponding parameters.

In tabu search the issues of identifying best combinations of short and long term memory and best balances of intensification and diversification strategies may be obtained using BHA.

Hence it seems that the Bayesian Heuristics Approach may be considered when applying almost any stochastic or heuristic algorithm of discrete optimization. The proven convergence of a discrete search method (see, for example, [**And96**]) is an asset. Otherwise, the convergence conditions are provided tuning the BHA [**MEM$^+$97**] if needed.

4. Process Scheduling

4.1. Introduction. In [**MR96a**] a general approach to the short-term scheduling problem of multipurpose batch and continuous operations has been described. A mathematical programming formulation that takes into account a number of the features of realistic industrial problems has been presented. The formulation is based on a continuous time representation, in which the planning horizon is divided into a number of intervals of unequal and unknown duration resulting in a large mixed integer nonlinear program. In principle, the optimal solution of this MINLP can be obtained by standard exact Generalized Benders Decomposition, Outer Approximation, or Branch and Bound techniques. However, in practice, it is known that such problem structures often result in exponential growth of solution times with problem size, thus rendering most industrially-relevant problems intractable.

This work is an attempt to address the above problem. It was suggested in [**MR94**] that the Bayesian Heuristic approach could be employed for the solution of such problems. The BH framework allows us to adapt any specific heuristic for a given class of problems. Furthermore, by clever choice of a heuristic we can make the numerical algorithm more efficient. In [**MR96b**] a general simulated annealing heuristic is used. In this work we employ a specialized Material Requirements Planning heuristic tuned for a class of batch and continuous scheduling problems. Computational experiments suggest that the BH approach combined with nonuniform time discretization shows promise for the solution of batch and continuous scheduling problems. Section 4.2 summarizes the key aspects of the BH approach. Section 4.4 describes how the MRP heuristic [**Orl75**] is tuned to incorporate batch and continuous operations. Finally, section 4.8 illustrates the effectiveness of this technique vis-a-vis the standard branch and bound technique applied to the Uniform Time Discretization approach (UDM).

4.2. Bayesian Heuristic Approach in Process Scheduling. Algorithms of exponential complexity are usually required to obtain the exact solution of global and discrete optimization problems. Even in cases when an approximate solution which lies within some tight error bounds is acceptable, the exponential complexity often remains. The desire to guarantee satisfactory results for the worst case is an important factor in forcing exponential complexity. Therefore in practice the solution of many applied global and discrete optimization problems is often approached using heuristics.

Most decision processes consist of a number of steps. For example, in batch scheduling problems these steps are: selecting a task, selecting a suitable equipment unit to process the task, and determining the amount of material to be processed by this task. In each step, an object is selected from some decision set (for example, select a task from a given set of tasks which produce the necessary product). A heuristic is a set of rules used to perform a step. For the classical knapsack problem, for example, this heuristic might be to select an object with maximal specific price (price over weight ratio). It may be helpful to think of this process as descending the decision tree by a path predefined by heuristic rules. In Figure 1 (upper part) the descent in a decision tree using the rule of always choosing the leftmost node is illustrated.

The key idea of the BH approach is to randomize these heuristic rules. Instead of descending the decision tree only once, the solution is repeated many times by selecting different paths. Each path is selected by applying the heuristic rule with some parameterized probability. Since the set of parameters for each descent in the decision tree is different, each set represents a different solution replicate and thus the best of them can be chosen for retention. In Figure 14.2 for each replicate the left most node is chosen with some probability which is the same within a specific replicate but is different for each replicate.

The parameterization is another key feature of the BH approach. If we know or expect that some heuristic "works" well, then we may increase the efficiency of the search by randomizing the parameters of the heuristic. Instead of solving a multidimensional discrete optimization problem directly we tune the parameters of the randomized heuristic. This tuning process is a low dimensional continuous optimization problem. We propose to solve this tuning problem using the Bayesian method of global optimization [**Moc89**].

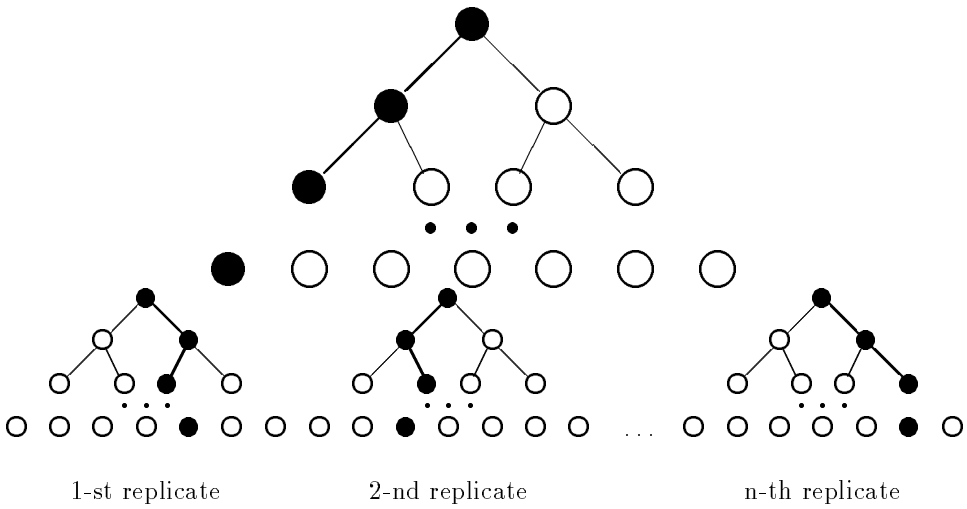

FIGURE 1. Decision tree
Deterministic descent (upper part). Randomized descents (lower part).

The main advantage of the worst case rigorous approach is that it yields error bounds on the solution. The main disadvantage is the focus on the worst possible case for a given class of problems. If this class is large, then in order to obtain sufficiently tight bounds many iterations may be required. This is the natural "cost" of such a guarantee.

The main advantage of the Bayesian approach is its focus on average case performance. An additional advantage of the BH approach is the possibility of including expert knowledge in a natural and convenient way. The potential ability to "learn" is also a positive feature of the BH approach. By learning we mean that the decision parameters which are optimal for some problems of the given class may be "good enough" for the rest of the class. The main disadvantage is that it is in general not possible to obtain and maintain guaranteed bounds on the quality of solution.

4.3. Penalty Function. The feasible region of the MINLP model defined in the first part of this series [**MR96a**] is quite complex. However, Bayesian global optimization methods are designed for a feasible region described by a hyper rectangle. Thus we need some device to transform the feasible region to a hyper rectangle. The penalty function is a convenient way to do this. Since both binary and continuous variables are present the penalty function must have two components: one for the violation of the binary constraints and one for violation of the continuous constraints. The right choice of penalty parameters is also important. If the values of the parameters are chosen to be too large, then the optimization problem may degenerate into the search for the "nearest" feasible decision. If the penalty parameters are chosen to be too small, then the constraints may be violated. We may reach some compromise by increasing the penalty parameters after each iteration. In general, the binary penalty parameter must be much greater than the continuous one, because while some violation of the continuous constraints may be permissible, the binary constraints must be satisfied. Of course, one may ignore the strict

nature of the binary constraints at the initial stages of global optimization, when the optimum is still far away but then one must satisfy these constraints exactly when approaching the global optimum.

4.4. Material Requirement Planning Heuristic. MRP is an inventory management and production planning technique [**Orl75**] which, given a delivery schedule for a final products, determines the initiating times for all raw materials orders, for the production runs needed to prepare all required intermediate products, as well as the starting times for the production of the final product itself. Given a delivery time for a product shipment, each branch of the product processing tree is traced from the product in question and each component requirement is calculated. If the inventory is inadequate, then a production order is issued for that component. The tracing of each branch of the processing tree continues until all raw materials requirements have either been met or ordered.

For scheduling problems relevant to the chemical industry we have to extend this heuristic to handle unit and task assignment, batch size determination, and other features. For purposes of this paper we will employ simple and natural heuristic rules. Of course, these rules could be enhanced or augmented to give further improvements in the results. The following is the list of heuristic rules:

- *Product selection rule.* The production run of each product in a batch and continuous processing system has some due date. We begin with a product which has the earliest due date. The rationale for this is that knowledge of due dates close to start of the scheduling horizon is more concrete. When production of this product is scheduled we recursively also schedule the production of the intermediates required to produce it.
- *Equipment item selection rule.* To schedule production of a given product or intermediate it is necessary to select a task and an equipment unit to process this task. When there are few tasks producing the same product we randomly assign the quantity of a product that the given task has to produce. Then we select an equipment item which is available during the time interval closest to the due date (a unit may become unavailable for processing because it processes another task, it is shutdown for maintenance etc.).
- *Task start selection rule.* Usually the size of the unit availability interval is different than the task processing time. The rule will start a given task to end exactly at the end of this interval.

  For example, assume that task2 processing time on unit1 is 1h. Furthermore, the interval chosen in the previous step is from 3pm to 5pm. Then task2 is started at 4pm to end at 5pm.

4.5. Schedule Generation. The process of schedule generation using the randomized heuristic is essentially the same as that described in section 4.4. The only difference is that instead of using the heuristic rules deterministically we make our selection with some probability. Consider, for example, the product selection rule. Instead of selecting a product with smallest due date, we select the product with some probability r which is a function of its due date D_s. The probability r_s of selecting the product s is expressed as

$$r_s = xa_0 + (1-x)a_1 h(d_s). \tag{4.1}$$

Here x is the parameter defined in the previous step of the BHA algorithm 1, M is the number of the products in the list , and the heuristic $h(d_s) = \exp\{-D_s\}$. We chosen an exponential function that the probability to select a product with late due date would be very small. As we see, the products with smallest due dates are chosen with highest probability, thus we preserve the character of the MRP heuristic while making it random. In a similar manner we deal with the other rules.

The generated schedule may be infeasible because of violations of the state capacity constraints. It is possible that the state capacity is exceeded or that a negative amount of material exists in a given state. These situations are handled by penalizing the variations from the minimal and the maximal state capacity values and adding this penalty to the objective function. The objective function is readily evaluated by calculating the storage and utility costs, the profit gained by satisfying demands less than raw material costs.

4.6. Bayesian Heuristic Algorithm. The key component of the BH framework is the randomized heuristic function. This function corresponds to the objective function in the global optimization. Once the randomized heuristic function is defined, we may optimize its parameters using the Bayesian global optimization method [**Moc89**] and as byproduct we obtain the optimum profit. For purposes of this paper we employ a parametric first order polynomial function of the Material Requirements Planning (MRP) heuristic as the randomized heuristic. The feasible region of this heuristic is quite complex, therefore a penalty function is used to transform the feasible region to a hyper rectangle. When using the MRP heuristic only the amount of material in the given state is allowed to exceed state capacity while all other constrains, binary and continuous, are not allowed to be violated. Thus we need only one penalty parameter for the continuous constraints. Theoretically, the value of the penalty parameter has to be much bigger than unit price of product to ensure a steep enough increase in penalty when we leave the feasible region. However, this value should not be too large so that the increase in penalty is quite smooth. Empirically we found the value of 500 to yield the best results for the test examples, although for other problems its value may be different.

In section 4.4 a version of the MRP heuristic for batch and continuous tasks was described . If d_i is some decision (select a product, a suitable equipment unit, or task start), then $h(d_i)$ is the heuristic function. The function $r(x, h(d_i))$ is a randomized heuristic function which gives the probability of the decision d_i. x is the randomization parameter. We used

$$r(x, h(d_i)) = xa_0 + (1 - x)a_1 h(d_i)$$

where

$$a_0 = \frac{1}{M},$$

$$a_1 = \frac{1}{\sum_{i=1}^{M} h(d_i)},$$

and M is the number of possible decisions. The heuristic rules are as defined in section 4.4. Thus the Bayesian Heuristic Algorithm can be represented by the following simple steps:

Step 1. Fix parameter x using the global Bayesian method [**Moc89**]. [1]
Step 2. Generate a schedule by using the randomized MRP heuristic (see sections 4.4 and 4.5);
Step 3. Evaluate the schedule for this parameter;
Step 4. If the schedule is feasible (there is no penalty) and the value of the best schedule so far did not increase for 10 iterations then go to step 6;
Step 5. Go to step 1;
Step 6. Fix binary variables to the values given by the best schedule. [2] Substitute the values of these binary variables into the model to reduce the problem to a linear program. Then the solution of the linear program gives the exact starting times, batch sizes, and processing rates.

We see that the scheduler generates sequences and assignments while the LP model is producing the exact schedule. The LP model is derived from the Non-Uniform Discrete-Time Model (NUDM) model (see [**MR94**]) by substituting the sequencing and assignment variables which are fixed by the scheduler. This disaggregation of the scheduling problem allows us to solve the large scale MINLP problem by using a combination of heuristic algorithm and an efficient LP solver.

The key issue in using the BH approach is that together with the best schedule we also acquire the best randomization parameter x, or, in other words, we tailor the heuristic for a given class of problems. Initially we are parameterizing the heuristic, thus giving it a few extra degrees of freedom. By varying parameters in an intelligent way, so that the expected outcome is maximized, we find the parameter values which yields the best profit.

4.7. Relationship of the Bayesian Heuristic Approach with Simulated Annealing and Genetic Algorithms. The Bayesian Heuristic Approach has a direct and deep relationship with simulating annealing and genetic algorithms. For example, consider (4.1). It indicates that products with smaller due date are selected with higher probability. However, there is a non zero probability of selecting a product with a higher due date. This flexibility which is allowed by the heuristic randomization effectively increases the search space. In the simulated annealing case, the same effect is achieved by accepting a point with lower objective value.

Genetic algorithms through an evolution process proceed from some initial population A to final population B (see Figure 2).

This evolution process is basically managed by the mutation and crossover probabilities which are constant throughout the entire evolution process. In the Bayesian Heuristic Approach, in effect multiple evolutions occur. Initial population A evolves to some intermediate population I_1 via evolution which depends on some evolution parameter x_1, i. e. mutation and crossover probabilities are defined by this parameter. The intermediate population I_1 evolves to some other intermediate

[1] This provides the asymptotic convergence with probability one. That is the first feature of the Bayesian Heuristic Approach. For the convergence proof see [**MEM+97**].

We have to note, however, that parameter x is fixed based on the value of the profit of the previously generated schedules. Thus the second feature of a given global Bayesian method is that it fixes the next x to a value for which it expects to get maximal profit. Of course, it may be that this new x value does not give the best profit, thus we need to use additional iterations.

These two features were key reasons for using the global Bayesian method. The second feature makes the search more efficient than a pure random Monte-Carlo search while the first enables a more thorough exploration of the decision space.

[2] These variables correspond to the sequencing and assignment of tasks.

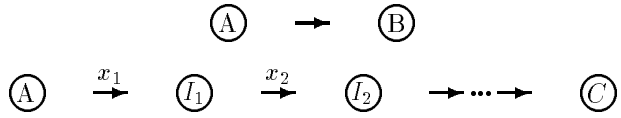

FIGURE 2. Bayesian approach versus genetic algorithms
Genetic algorithm. Single evolution (upper part). Bayesian approach. Multiple evolutions (lower part). Here x_i is chosen so that to maximize conditional expected fitness of i-th population.

population I_2 via evolution which depends on another evolution parameter x_2. This process is repeated many times. The best population is selected as the final population. Another important feature besides multiple evolutions is that each x_i is selected based on the fitness of the previous populations so as to maximize the conditional expected fitness of i-th population. In such a way, the process is not a purely random one but is aimed to maximize the fitness. By changing the mutation and crossover probabilities the tendency of genetic algorithms to produce local solutions is mitigated.

Another key difference of the BHA from simulated annealing and genetic algorithms is learning. In simulated annealing or genetic algorithms parameters, such as initial temperature or mutation probability etc., are found experimentally by studying a number of scheduling problems. However, in the Bayesian Heuristic Approach these parameters are tuned for each problem of a given class and do not need to be fixed for all classes. Thus, in contrast to simulating annealing and genetic algorithms, the Bayesian approach allows learning to occur for the problem at hand.

The further drawback of simulated annealing and genetic algorithms is that when applied to discrete or mixed integer problems they generate a large number of infeasible solutions. Thus a big percentage of the computation time is wasted in the generation of these infeasible solutions. The similar situation was observed when trying to employ simulation annealing heuristic within the Bayesian Heuristic Heuristic Approach [**MR96b**]. Because of this, only small size problems could be solved. Using the BHA with MRP heuristics one overcomes this problem and 70% percent of the generated schedules were feasible. This can be explained by the fact that MRP heuristic fits better to a given class of scheduling problems and thus generates mostly realizable schedules.

4.8. Scheduling Results. We compared the BHA algorithm with solution of the MILP uniform discretization model using branch and bound enumeration (B&B) [**ZPMR94**]. Results are reported for several test examples, both batch and continuous. The detailed numerical data for these examples is available electronically from the rcsplib account (rcsplib@ecn.purdue.edu) in the form of RCSPec language files.

We summarize the results in Table 2. Computational experiments were performed on a HP 9000/755 workstation using the commercially available CPLEX solver. The column "BH profit" shows the profit value obtained before performing the LP solution phase, i.e. the profit value given only by the statistical part of the algorithm. As we see, the LP solution does not significantly increase the value of

TABLE 2. B&B and BHA comparison for examples of batch and continuous processes

Problem	B&B Profit ($)	B&B Time (sec)	BHA Profit (%)	BHA Time (sec)	Number of replicates	BH profit (%)
batch1	3230	6	100	0.46	12	77.91
batch4	60534	1.7	99.99	2.90	13	93.79
batch3	105756	8.6	99.97	3.85	20	99.18
exIII	1400	66	100	3.65	15	100
cpctsp1	4724	0.6	100	0.06	11	100
cpctsp2	8122	188.5	100	0.20	11	100
cpctsp3	12015	*	99.60	3.01	16	99.6
cipac2	20879	*	100.54	31.8	29	100.39
cipac1	22800	824.6	107.77	27.3	29	107.77

the profit. However we can not draw definite conclusions about the need of the LP solution step from this case study alone. We have to note that while for batch processes (batch1, batch3, batch4, exIII, cpctsp) BHA gives solutions slightly worse than the branch & bound approach, for continuous processes BHA (cipac1, cipac2) gives better results than UDM. This can be explained by the inherent discrete nature of the UDM which is not suited for continuous tasks. Time variable can have only discrete values in the UDM case, while processing time of continuous tasks assumes continues values and thus can not be represented as discrete variables. NUDM works much better than UDM for problems with sequence dependent tasks (cipac2, cpctsp1, cpctsp2, cpctsp3) also. This is due to the fact that NUDM handles sequence dependent changeovers in a more efficient way, i.e. uses *resource-task* network representation as opposed to the *state-task* network representation used by the UDM. It is shown that constraints handling sequence dependent changeovers based on the *resource-task* network representation are tighter than those based on the *state-task* network representation . It is worth noting that with increasing number of sequence dependent tasks (cpctsp1, cpctsp2, cpctsp3) the solution time of B&B grows exponentially while the corresponding time of BHA grows only polynomial. * means that optimal solution for the examples cpctsp3 and cipac2 was not reached by B&B since the solution time was unreasonably large. The B&B tree was terminated after 20000 nodes. The computational time for the batch4 example is worse in BHA case due the fact that MRP heuristic is not very well suited for such processes (batch4 example contains zero wait states). It is possible to modify BHA account for the zero wait states by aggregating the two tasks connected by zero wait state together.

4.9. Scheduling Conclusions. In [**MR96a**] a general formulation of the short-term scheduling problem for complex multipurpose batch and continuous operations is presented. However, the size of the resulting MINLP raised serious concerns regarding the practical applicability of the NUDM. In this paper we provide an algorithm to overcome this limitations.

This algorithm is based on the BH approach to discrete optimization and a clever choice of heuristic is the key issue. If the heuristic is a very general one (as

simulated annealing in [**MR96b**]) then the class of problems solved is large as is the computational time. If the heuristic is a special one (as MRP in our case) then the class of problems solved is smaller as is the computational time. Of course, problems which do not fit a given heuristic are solved also but the computational effort required can be unpredictably high (batch4 is a good example). Thus one of the directions for future work may well be the expert system which recognizes the structure of the problem and suggests a heuristic to solve this problem.

Usually processing data is uncertain (task processing times fluctuate around the mean value due to the quality of feed, due dates are not well known in advance etc.). The BH approach is a statistical framework and thus theoretically accommodates stochastic data. Thus the other promising future direction can be identified as an application of the BH approach to scheduling problems with uncertainty. NUDM allows one to model various uncertainties in time and size parameters without modifications since time and size are continuous variables (for the UDM uncertainty in processing times require major modifications).

5. Software for Global Optimization

5.1. Background. The global optimization software was initiated considering the results of international "competition" of different algorithms of global optimization (see [**DS78**]). The experience in real life optimization problems and some recent results were also used selecting the set of optimization algorithms. The set of algorithms of global optimization includes

- four versions of the Bayesian search,
- a version of clustering,
- a version of uniform deterministic grid,
- a version of pure Monte Carlo search.

Usually it is reasonable to start optimization by a global method and to finish it by some local method. An exception is two global algorithms: the Torn version of clustering [**TZ89**], and the Zilinskas version of the Bayesian technique [**TZ89**]. Both of these algorithms contain some simple local search algorithms. The local search is not necessary for those two methods, but it may be useful.

There are three local optimization methods:

- a method of variable metrics type with Lagrangian multipliers and penalty functions for constrained optimization of smooth functions (see [**Sch86**]),
- a method of simplex type of Nelder and Mead with penalty functions for constrained optimization of non-differentiable functions.
- a method of stochastic approximation type for "noisy" functions (see [**Moc89**]).

5.2. Application Areas. Each subroutine represents a global or a local method. The choice of method has to follow the idea that the computational complexity of the method should roughly correspond to that of the objective function:

- For computationally "expensive" functions the Bayesian methods could be recommended. Those methods need a large amount of auxiliary calculations to make each observation more efficient.
- For "cheap" functions the simple grid methods, like Monte Carlo or a uniform deterministic grid (see [**Sob67**]), can be better. Here observations are not so efficient, but auxiliary calculations are negligible. This explains a relative efficiency of simple methods when optimizing simple functions.

- The clustering techniques (see [**TZ89**]) may be the best choice, if we expect the number of local minima to be small.
- A relatively simple Bayesian technique is available [**TZ89**] for global optimization of one-dimensional functions;
- There are optimization problems where objective functions can be roughly represented as a sum of components depending on different variables. Here the Bayesian method of line search along the coordinates usually shows very good results. This method globally optimizes one variable at a time by one-dimensional Bayesian search. The difference of this method from other methods of global optimization is that it depends on the starting point. Thus a deviation from the global minimum can be made as small as desired by applying a multi-start procedure with different uniformly distributed starting points.

5.3. Constraints. All the global methods optimize in a rectangular region. Therefore we represent the linear and non-linear inequality constraints as some penalty functions. The same applies to the local method of stochastic approximation type. In local methods of simplex and variable metrics type the linear and the non-linear constraints can be defined directly. This may be done by constraint subroutines, supplied by the user in addition to the objective function.

5.4. Software Versions. The global optimization software is in four versions:

- portable Fortran Library,
- interactive software for Turbo C compiler and DOS operating system,
- interactive software for C++ compiler and UNIX operating system and X-Window system.
- interactive software for Java

One may notice a cycle of portability in this sequence of software versions. The sequence is started from by the portable Fortran library and is concluded by Java language. The two systems in between are more difficult to port. The Turbo C system is for DOS-compatible operating systems and C++ is for the UNIX environment. Fortran, Turbo C and C++ versions are described in [**MEM+97**]. Now we briefly consider the Java version.

5.5. Java Version.

5.5.1. *Global Minimizer for Java (GMJ).* The GMJ system [**Gry98**] is a class framework for implementing and testing global optimization algorithms ($METHODS$), functions to be optimized ($TASKS$), and and visual representations classes ($ANALYSIS$).

5.5.2. *Running GMJ.* The GMJ system can be run as a Java Applet or as Java Application. The advantage of running it as Java Applet is obvious-it can be used over the Web. The advantage of the Java Application is that the most recent Java features supported by the Java Development Kit (JDK) can be used.

5.5.3. *Configuring Applet.* The sample file *gmj.html* shows how to set configuration parameters of the *gmj* applet. The $<$ *applet* $>$ tag looks like this:

```
<applet code="lt.ktu.gmj.ui.GMJ.class"
                codebase="Lib"
                align="baseline"
                width="450"
                height="450"
```

```
                archive="gmj.jar">

<param name="TASKS"
value= "lt.ktu.gmj.tasks.Sin|Undefined">

<param  name="METHODS"
value="lt.ktu.gmj.methods.Mig1| lt.ktu.gmj.Bayes1">

<param name="ANALYSIS"
value= "lt.ktu.gmj.analysis.Convergence|
lt.ktu.gmj.analysis.Spectrum|lt.ktu.analysis.Projection">

</applet>
```

Here:

code specifies the applet class and should not be changed

codebase specifies the relative URL of the applet class archive file and user class files. The URL is relative to the location of the HTML file where the $< applet >$ tag resides or is absolute path.

archive lists the class archive files

width and *height* specifies the applet size, as it appears in the HTML page.

The parameter $TASKS$ lists tasks which are available when the applet loads. Complete class names (package and class name) are separated by | symbol.

The parameter $METHODS$ lists methods which are supported when the applet loads.

The parameter $ANALYSIS$ lists visual analysis classes which are supported when the applet loads.

5.5.4. *Configuring Stand-Alone Application.* The Java Runtime Environment (JRE) is required when running GMJ as a stand-alone application.

Before running GMJ, a $CLASSPATH$ should be configured, so that the GMJ classes can be found by the JRE. For Windows NT the command might look like this:

```
SET CLASSPATH=#CLASSPATH#;c:\gmj\gmj.jar
```

the application is started by loading the $lt.ktu.gmj.ui.GMJ$ class:

```
java lt.ktu.gmj.ui.GMJ
```

Like the applet the application accepts three optional parameters: tasks, methods, and analysis objects.

5.5.5. *Display and Control.* GMJ displays a tab control that has three choices: method, task and operation. The appropriate pages can be selected by clicking on the page tabs. The detail discussion is in [**Gry98**].

5.6. Software Availability. The software .

The interactive UNIX C++ software (LINUX 1.2.8. version) and the library of portable Fortran subroutines (LINUX 1.2.8. and DOS versions) are included in [**MEM$^+$97**], and available on $ftp://optimum.mii.lt/pub$

6. Dynamic Visualization Approach (DVA)

We started the description of algorithms from the formal Direct Bayesian Approach (DBA). Then we considered semi-formal Bayesian Heuristic Approach

(BHA). An informal interactive optimization is needed if an optimization problem is not well defined. This may arise, for example, if the mathematical model, including the objective function, must be updated during the course of optimization process.

The informal interactive approach attempts to represent an optimization problem in a visual form that is domain specific and is intuitive to the domain expert. The visual representations can vary significantly across the domains. In the examples the dynamic visual representation of a smooth function in time and space turned out to be effective in several domains. The domain specific visual representation can efficiently convey information about a complex model and help make qualitative judgments about model's adequacy and optimality.

The efficiency of informal interactive optimization depends on dynamic visualization techniques. We regard dynamic visualization as an important tool using heuristics in an informal interactive way and thus will refer to it as a Dynamic Visualization Approach. Two basic techniques of dynamic visualization are considered in [**MEM$^+$97**]:

- space-time smoothing;
- image search.

Those techniques are explained in [**MEM$^+$97, EM96**] through real life examples. We mentioned Dynamic Visualization Approach (DVA) just to illustrate different ways of using heuristics, from pure formal Bayesian Approach (by priori distribution on a set of objective functions) to semi-formal Bayesian Heuristic Approach (by priori distribution on a set of parameters of randomized heuristics and their "mixtures") and pure interactive DVA (directly by the expert opinion using visual representation).

In the Java optimization system (see section 5.5) the possibility of the domain specific visualization is included by the $ANALYSIS$ classes. This way the Bayesian Heuristic Approach and the Dynamic Visualization are integrated. The simplest DV techniques such as convergence lines, projections and the objective function distributions are rather general. The more advanced visualization techniques are domain specific and should be developed while designing the $TASKS$ classes representing specific objective functions.

References

[And96] S. Andradottir. A global search method for discrete stochastic optimization. *SIAM Journal, Optimization*, 6:513–530, 1996.

[AV91] I.P. Androulakis and V. Venkatasubramanian. A genetic algorithmic: Framework for process design and optimization. *Computers in Chemical Engineering*, 15:217–228, 1991.

[DeG70] M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[Dia88] P. Diaconis. Bayesian numerical analysis. In *Statistical Decision Theory and Related Topics*, pages 163–175. Springer Verlag, 1988.

[DS78] L.C.W. Dixon and G.P. Szego. *Towards global optimisation 2*. North Holland, Amsterdam, 1978.

[DS90] G. Dzemyda and E. Senkiene. Simulated annealing for parameter grouping. In *Transactions. Information Theory, Statistical Decision Theory, Random Processes*, pages 373–383, Praque, 1990.

[EM96] William Eddy and Audris Mockus. Dynamic visualization in modeling and optimization of ill defined problems, case studies and generalizations. In C. A. Floudas and Panos M. Pardalos, editors, *State of the Art in Global Optimization: Computational*

Methods and Applications, volume 7 of *Non-convex Optimization and its Applications*, pages 499–520. Kluwer Academic Publishers, 1996.

[Glo94] F. Glover. Tabu search: improved solution alternatives. In *Mathematical Programming. State of the Art 1994*, pages 64–92. University of Michigan, 1994.

[Gol89] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[Gry98] M. Grybauskas. Global minimizer for java (gmj) version 1.1. Technical report, Institute of Mathematics and Informatics, Working Paper, MII, Akademijos 5, Vilnius, Lithuania, 1998.

[Kus64] H.J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J. of Basic Engineering*, 86:97–100, 1964.

[MEM+97] J. Mockus, W. Eddy, A. Mockus, L. Mockus, and G. Reklaitis. *Bayesian Heuristic Approach to Discrete and Global Optimization.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[Moc89] J. Mockus. *Bayesian approach to global optimization.* Kluwer Academic Publishers, Dordrecht-London-Boston, 1989.

[MPPR97] T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. A GRASP for the biquadratic assignment problem. *European Journal of Operations Research*, 1997.

[MR94] L. Mockus and G.V. Reklaitis. Mathematical programming formulation for scheduling of batch operations using nonuniform time discretization. In *AIChE annual meeting*, number 235d, San-Francisco, California, 1994.

[MR96a] L. Mockus and G. V. Reklaitis. Continuous time representation approach to batch and continuous process scheduling-ii. minlp formulation. submitted, 1996.

[MR96b] L. Mockus and G. V. Reklaitis. A new global optimization algorithm for batch process scheduling. In C. A. Floudas and Panos M. Pardalos, editors, *State of the Art in Global Optimization: Computational Methods and Applications*, volume 7 of *Non-convex Optimization and its Applications*, pages 521–538. Kluwer Academic Publishers, 1996.

[Orl75] J. Orlicky. *Material Requirement Planning.* McGraw-Hill, 1975.

[Sal71] V.R. Saltenis. On a method of multi-extremal optimization. *Automatics and Computers (Avtomatika i Vychislitelnayya Tekchnika)*, (3):33–38, 1971. in Russian.

[Sch86] K. Schittkowski. Nlpql: A fortran subroutine solving constrained nonlinear programming problems. *Annals of Operations Research*, 5:485–500, 1985/86.

[Sob67] I.M. Sobolj. On a systematic search in a hypercube. *SIAM Journal on Numerical Analysis*, 16:790–793, 1967.

[TZ89] A. Torn and A. Zilinskas. *Global optimization.* Springer-Verlag, Berlin, 1989.

[ZPMR94] M. G. Zentner, J. F. Pekny, D. L. Miller, and G. V. Reklaitis. RCSP++: A scheduling system for the chemical process industry. In *Proc. PSE'94*, pages 491–495, Kyongju, Korea, 1994.

Department of Optimization, Institute of Mathematics and Informatics, Akademijos 4, Vilnius 2600, Lithuania

E-mail address: jonas@optimum.mii.lt

Software Production Research Department, Bell Laboratories Lucent Technologies Naperville, IL 60566-1444

E-mail address: audris@research.bell-labs.com

R&D, Monsanto, Skokie, IL 60077

E-mail address: LINAS.MOCKUS@monsanto.com