(19th ICPR) DESIGN AND VALIDATION OF HEURISTIC ALGORITHMS FOR SIMULATION-BASED SCHEDULING OF A SEMICONDUCTOR BACKEND FACILITY

Gerald Weigert, Klemmt Andreas, Horn Sven

(19th ICPR) DESIGN AND VALIDATION OF HEURISTIC ALGORITHMS FOR SIMULATION-BASED SCHEDULING OF A SEMICONDUCTOR BACKEND FACILITY

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0154.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	20-Dec-2008
Complete List of Authors:	Weigert, Gerald; Technische Universität Dresden, Electronics Packaging Laboratory Andreas, Klemmt; Technische Universität Dresden, Electronics Packaging Laboratory Sven, Horn; Technische Universität Dresden, Electronics Packaging Laboratory
Keywords:	SCHEDULING, SIMULATION, SEMICONDUCTOR MANUFACTURE, OPTIMIZATION
Keywords (user):	

DESIGN AND VALIDATION OF HEURISTIC ALGORITHMS FOR SIMULATION-BASED SCHEDULING OF A SEMICONDUCTOR BACKEND FACILITY

G. Weigert, A. Klemmt, S. Horn
Technische Universität Dresden
Faculty of Electrical Engineering and Information Technology
Electronics Packaging Laboratory
01062 Dresden, Germany

Abstract
A simulation-based scheduling system is discussed which was developed for a semiconductor Backend facility. Apart from usual dispatching rules it uses heuristic search strategies for the optimization of the operating sequences. In practice hereby multiple objectives have to be considered, e. g. concurrent minimization of mean cycle time, maximization of throughput and due date compliance. Because the simulation model is very complex and simulation time itself is not negligible, we emphasize to increase the convergence of heuristic optimization methods, consequentially reducing the number of necessary iterations. Several realized strategies are presented.

Keywords:
simulation-based scheduling, heuristic algorithms, multi-objective optimization, visualization.

1 INTRODUCTION

Semiconductor manufacturing is one of the most complex manufacturing tasks (Potoradi et al 2002). Basically, it consists of four distinct stages. These are namely the Wafer Fabrication and Wafer Test, Preassembly, Assembly and functional (electrical) Test (Sivakumar 1999). The wafer fabrication is known as the Frontend. Preassembly, Assembly and Test are known as Backend Semiconductor manufacturing (see Figure 1).

Figure 1: Simplified Backend workflow

Backend facilities were studied by many researchers (e.g. Kuhn and Quadt 2002). Also, different scheduling strategies were investigated. However, in most Backend areas often the control is whether designed intuitively or utilizing more or less effective local dispatch rules. The aim is to find an optimized Backend manufacturing strategy for a short period (1 or 2 shifts) on the basis of an exact forecast generated by a simulation-based approach. Therefore, the question: 'What means optimal?' is asked over and over again. Of course, there are basic objectives like cycle time reduction, throughput maximizing etc. but these can be overlaid by currently higher prioritized objectives (for example lots or products with critical due dates). Furthermore, it is necessary to consider more than one objective which leads to a multi-objective optimization problem.

A semiconductor Backend is typically organized as a flexible flow line, which is a flow line with parallel machines on some or all stages (Quadt 2005). In the Preassembly steps, the wafer is sawn and individual chips are produced. Technological steps are Dice and Grind which makes the separated chips available on framed foils. It follows Die Bond where machines mount the chip on lead frames or substrates. Wire Bond processes attach ultra-thin golden wires between bonding pad on the die and a connector on the lead frame, to create the electrical path. Next step is the encapsulation of the chip which is called Mold. It is aim to protect the chip from the environment. Important and cost expensive is the now following Burnin where early-fails are separated and also the final Test where each chip is tested and classified into speed categories (see Figure2).

Figure 2: Technological Backend steps

All products have to visit all steps. The number of parallel machines for each step may vary. For purposes of modeling infinite buffers can be assumed between the steps. There is no transportation time between the steps. The production runs 24 hours a day and seven days a week (Kuhn and Quadt 2002).

The focus of this paper is on Backend operations. We will describe a simulation-based scheduling system which uses heuristic algorithms for optimizing the processes of this facility.

2 RELATED WORK AND PAPER STRUCTURE

For semiconductor manufacturing several simulation-based optimization approaches exist. Sivakumar (1999) was one of the first who used deterministic online simulation of the facility and automatic model creation, plus optimization applied to a Backend test facility. An online parameterized model is also described by Potoradi et al (2002). Here the scheduling of the wire bonder equipment group as the current bottleneck was accomplished with the simulation system Factory Explorer. A further heuristic approach for wire bonder scheduling is described by Tovia et al (2004). Also heuristic methods are explained which can maximize the throughput of the equipments. Quadt (2005) recommends the use of the scheduling system Asprova, extended with a customized scheduling logic, for optimizing the allocation of parallel machines in

semiconductor Backends (flexible flow lines). Several publications are available about the examination of batching rules, especially for the burn in ovens in Backend, e.g. (Sung et al 2002). Most of these papers focus only on this special task, often using methods of simulation-based scheduling (SBS). All these publications are especially focused on the special behavior of Backend process flows. A wide experience of using SBS for semiconductor production flows can also be taken from similar articles about semiconductor Frontend processes, e.g. (Mönch and Zimmermann 2004) or (Rose 2003).

The key for the successful application of SBS systems is a powerful heuristic algorithm for process optimization. Unfortunately, such algorithms are mostly tested only for so-called benchmark models (simple flow shop or job shop problems) which are not related to the requirements of a real industrial environment (Ponnambalam et al 2000). In this way newly designed heuristics can simply be compared with already known algorithms and their efficiency can be established (Rajendran and Ziegler 2004). Thereby, the objective is generally the makespan minimization of an initially empty, run-out model. The lot release sequence is used as control variable. In contrast to this we practically have to deal with already filled, non-run-out models. These should be planned over only a short period of time (rolling production scheduling (Scholl et al 2003)). For optimizing such models, an adaptation or redefinition of control variables is necessary to effect changes in the whole model (not only lot release sequence) (Beier 2006). Also an adaptation of the objective function is required because not all objectives (like makespan) are calculable on the basis of short planning periods.

Because the used simulation-based optimization system can only handle with scalar objective functions, it is necessary to transform a multi-criteria optimization problem into a single-criteria optimization problem. Thereby the single objective scaling becomes very important (Gupta and Sivakumar 2002, Beier 2006). Approaches for implementing multi-criteria optimization problems are discussed in section 4.

For the optimization of the Backend model, local search heuristics were used which are also explained in section 4. Investigations of Weigert et al (2005) have shown that significant objective function improvements can be reached already by the usage of such simple local search algorithms. Thereby, the differences between the algorithms were not essential. Instead of that, the distance measurement used for neighborhood calculation is investigated in more detail. In (Horn et al 2006) and (Sevaux and Sörensen 2005) different powerful alternatives to the traditional used Hamming distance are shown. This will be adapted to the Backend model in section 5.

For optimizing a highly complex manufacturing model, like the Backend, generally it is required a high number of iterations to obtain a significant objective function improvement. By partitioning the model into several small single models the global optimization time can be reduced drastically. This is efficiently possible, especially for model partitions made between the system bottlenecks (Beier 2006).

The above listed items, like neighborhood calculation, redefinition of control variables and mulit-step optimization are specific contributions of approaches to increase the efficiency of simulation-based Backend optimization. Even for highly complex manufacturing models iterative heuristic search algorithms are more suitable than mathematical solvers (i.e. mixed integer programming MIP), which are just applicable for comparable small problem dimensions. Klemmt et al (2008) demonstrate up to which number of machines and/or jobs the solver based methods are in advantage to simulation based heuristics, for simple benchmark models. The principle of heuristic methods is explained more detailed in section 5.

The results of the Backend optimization approach by using the new heuristics are presented in section 6. For the better evaluation of effectiveness algorithms, we have developed the graphical tool OptVis3D (Klemmt and Weigert 2008). This tool displays up to 3 arbitrary objective functions. It offers a lot of visualization options, such as emphasizing certain points in the solution space, plotting of search paths, denoting of solution space structure and the visualization of objective depending Pareto-fronts. This provides a view inside the heuristics and enables us to fine-tune them.

3 A SIMULATION-BASED SCHEDULING SYSTEM

3.1 The simulation model

For the creation of the simulation model we use the simcron MODELLER. This is a simulation tool which has already proved itself in practical use. The simulator has only a small number of different module types but it is characterized by its high simulation speed. This is especially beneficial in the case of simulation based optimization. Object oriented event handlers, programmable objective functions and the possibility to include arbitrary scripts in the simulation model also allow the modeling of highly complex manufacturing processes. The model itself is generated completely automatically from the corresponding ERP-System data. This guarantees the up-to-dateness of the simulation model.

On average one model includes more than 300 machines and queues as well as more than 1500 jobs. For one simulation a usual computer (Intel Centrino Duo, 1.83 GHz, 2GB RAM) needs approximately 20 seconds. The planning horizon is set to 2 days.

3.2 Simulation-based optimization

Often, the complexity for practical problems of production control is too high to be solved by usual analytical methods. Heuristic optimization algorithms combined with simulation systems are a suitable alternative in these cases. Figure 3 shows the basic principle of a simulation-aided optimization system. The problem is described by a simulation model which includes a set of control variables x and responds with an objective value C after a simulation run is completed. The control vector $\mathbf{x}$ consists of several variables x which influence the behavior of the simulation model, e.g. job permutations, buffer or machine capacities and release dates. Because of simplification there is no distinction between a single control variable x and a control vector $\mathbf{x}$ in the following formulas. So the optimization system is divided into an assessment part on the one hand and a separate optimization algorithm on the other hand. Both subsystems communicate by $\mathbf{x}$ and C while the optimization cycle is running. In the simulation system simcron MODELLER, this optimization cycle is implemented as an experimental feature.

Figure 3: Basic optimization cycle.

4 OPTIMIZATION ALGORITHMS

4.1 Basic algorithm

For optimizing the Backend model we use iterative local search algorithms which are mainly variances of Threshold Accepting. Their basic algorithm was formulated by Weigert et al (2005) as follows:

$M := 0$
$m := i := 0$
$x_{i,m} := x_{start}$
while ($i < N$ **and not** termination condition) **do** {
    $i := i+1$
    $x_{i,m} := f(x_{i-1,m})$
    if (random (0,1) < p_i) **then** {
        $x_{i,i} := x_{i,m}$
        $m := i$
        $M := M+1$
    }
}

N	number of iterations
M	number of moves in the control space, $M \le N$
i	iteration index, $i = 0 \dots N$
m	iteration index of last move, $m = 0 \dots M$
$x_{i,m}$	point in the control space, $m \le i$
p_i	probability of move in iteration i (probability of acceptance)
f	search function which generates a new point in the search space

Several search strategies like Threshold Accepting (TA), Simulated Annealing (SA) (Metropolis 1953), Greedy Search (GY), Great Deluge (GD) etc. differ only in the definition of the search function f or the acceptance probability p_i (Weigert et al 2005). First we want to describe the Threshold Accepting algorithm:

Threshold Accepting (TA)

The probability of acceptance of a move can be calculated as follows.

$$p_i = \Theta(Th_i - \Delta C_i) \tag{1}$$

Here Θ is the Heaviside function

$$\Theta(x) := \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \Delta C_i := C(x_{i,m}) - C(x_{m,m}). \tag{2}$$

This means, a move is executed if $\Delta C_i \le Th_i$, otherwise it is a simple iteration without moving. The Threshold Th_i is the control parameter which drives the algorithm. For high values of Th_i it reacts like Random Walk (RW) and it changes to Greedy for a lower threshold (see also Weigert et al 2005). It is assumed that the algorithm starts with Th_0 and is adapted by the following strategy.

$$Th_i = Th_0 - b \cdot i \quad \text{with} \quad b > 0\,. \tag{3}$$

Figure 4 illustrates the functionality on a simple continuous example.

Figure 4: Functionality of Threshold Accepting

The disadvantage of TA is that we need some knowledge about the spread and dimension of the objective function C to set Th_i efficiently. To overcome this problem, algorithms with a dynamic threshold calculation are required. The following two algorithms include this.

Old Bachelor Acceptance (OBA)

The OBA algorithm was presented by Hu et al. in 1995. It is an advancement of TA. Here the probability of acceptance of a move is also calculated by (1). In contrast to TA the threshold Th_i is set automatically (depending on model parameters) and is changed during optimization by the following formula.

$$Th_i = \left(\left(\frac{age}{J}\right)^b - 1\right) \cdot \frac{C(x_{\min})}{J} \cdot \left(1 - \frac{i}{N}\right)^c \tag{4}$$

$C(x_{\min})$ Currently best found solution

J Number of jobs in the model

b allows a power law growth rate (set to 2)

c tunes heuristic "damping" factor $\left(1-\frac{i}{N}\right)$ (set to $\frac{1}{2}$)

age Threshold growth rate $\begin{cases} 0 & \text{if step is accepted} \\ incr(age) & \text{otherwise} \end{cases}$

In OBA the threshold Th_i is recalculated in every step. Thereby Th_i is increased after every not accepted step. After every accepted step the threshold becomes negative. The amplitude of threshold oscillation becomes smaller with a progressing simulation time. Figure 5 illustrates this functioning exemplarily.

Figure 5: Threshold behavior OBA

The advantage of OBA is in the not needed knowledge about the spread and the dimension of the objective function C.

Record-to-Record Travel (RRT)

This heuristic was presented by Dueck (1989). The probability of acceptance of a move is calculated as follows.

$$p_i = \Theta\left((1+\alpha)\cdot C(x_{\min}) - C(x_{i,m})\right) \quad \text{with} \quad \alpha > 0 \tag{5}$$

Here $C(x_{\min})$ is the currently best found solution. This means, a move is executed if

$$C(x_{i,m}) < (1+\alpha)\cdot C(x_{\min}) \tag{6}$$

otherwise it is an iteration without moving. So with each improvement of the objective function the tolerated deterioration (from optimum) of the next steps is decreased. This algorithm also has the advantage that we don't need to know something about the spread and the dimension of the objective function C.

For the sake of completeness an overview about some more investigated threshold algorithms is given in Table 1. The table also includes the formula for the calculation of the probability of acceptance and the most important control parameters. Especially the Simulated Annealing (SA) and the Great Deluge (GD) also achieve good results for the related problems (Weigert et al 2005).

Table 1: Algorithm overview

4.2 Multi-objective optimization

The objective function C is basically a scalar function which depends on a vector $\mathbf{x}$. The vector includes all control variables of the scheduling problem. A multi-objective optimization problem has several single objective functions C_k, which can be combined to an objective vector $\mathbf{C}$.

$$\mathbf{C}(\mathbf{x}) = \left(C_1(\mathbf{x}), C_2(\mathbf{x}), C_3(\mathbf{x}), \ldots\right)^T \tag{7}$$

Usually these single objectives are converted by a weighted sum into a scalar substitute function.

$$C(\mathbf{x}) := w_1 C_1(\mathbf{x}) + w_2 C_2(\mathbf{x}) + w_3 C_3(\mathbf{x}) + \ldots \tag{8}$$

Since the weighting can be combined to a vector $\mathbf{w}$ as well as the objective vector $\mathbf{C}$, the sum of equation (8) can be written as a scalar product of the weighting vector and the objective vector.

$$C(\mathbf{x}) := \mathbf{w}^T \mathbf{C}(\mathbf{x}) \tag{9}$$

The weight w_k describes the importance of the objective C_k, where important objectives get a high weight and less important ones a lower weight. Unlike the objective values the weight itself is subjectively determined. The real influence of a certain objective does not only result from the weight solely but from the product of the objective value and its weight. This means, objectives with high numerical values have a priori a larger self-weight as such objectives with lower numerical values. To avoid this undesirable effect, a normalization of the objectives is necessary.

The basic normalization principle is to derive the substitute scalar objective function C from a normalized objective vector $\mathbf{N}$, where the numerical values of all components N_k have the same order of magnitude. So the subjective weight factors display their correct impact.

$$C(\mathbf{x}) = \mathbf{w}^T \mathbf{N}(\mathbf{x}) \tag{10}$$

The most usual normalization method is described in equation (11), where C_k is the component k of the original objective vector of an arbitrary iteration.

$$N_k := \frac{C_k - C_k^{\min}}{C_k^{\max} - C_k^{\min}} \tag{11}$$

The minimum of every normalized component is 0 and the maximum is 1. The area of the normalized objective region is always 1, independent of the original objective area. In addition, the normalization is invariant to the position of the start point, but for calculation we need the absolute minimum (maximum) of the related original objective.

Figure 6: Original and normalized objective area (method 1, qualitative)

Actually, this is only known if the optimization process was definitely finished. During a running optimization, we cannot be sure to find a new minimum (maximum) of any objective variable. If this happens, the parameters of equation (11) are changed. Also, every point found up to this time has to be recalculated. Figure 6 shows an original objective area of two objectives which are enlarged by finding a new minimum (maximum) and its effect on the normalization area. As a side effect the normalized start point is shifted, exemplary for the other points.

The second normalization method (Beier 2006) is very simple and avoids some disadvantages of the method described before; especially it is suitable not only for a completed set of objective vectors but also for a still running optimization process. In difference to the first method its scale basis is exclusively the start value.

$$N_k := \frac{C_k}{C_k^0} \tag{12}$$

For practical application it is assumed that every component has a minimum value greater than 0:

$$C_k^{\min} > 0 \quad \forall k \tag{13}$$

For this method of normalization it is not necessary to know the minimum or maximum value of the original set. The start point has always the coordinates (1,1) and holds - as well as any other point - its position, even if the original area is enlarged. In other words, this normalization method is insensitive to changes of minimum or maximum value, as shown in Figure 7.

Figure 7: Original and normalized objective area (method 2, qualitative)

Unlike method 1 this normalization depends on the position of the start point. The normalized objective area is moved and changed if the start point is changed, as shown in Figure 8.

Figure 8: A new start point moves and changes the normalized objective area (method 2, qualitative)

Normalization means that all normalized objectives are of the same order of magnitude. The start point (1,1) satisfies this requirement by definition. In addition, we should assure that the normalized minimum values (and maximum values alike) are approximately equal for all components.

$$N_k^{\min} \approx N_l^{\min} \quad \forall k,l \tag{14}$$

This is equivalent to the requirement that the start values of an arbitrary objective pair are proportional to their minimum values and vice versa. Fortunately, this condition applies for our practical case, where the objectives have a similar dimension.

$$\frac{C_k^{\min}}{C_l^{\min}} \approx \frac{C_k^0}{C_l^0} \quad \forall k,l \tag{15}$$

5 ACCELERATION OF THE ALGORITHMS

In section 3 we have shown the basic approach of simulation-based optimization. With a model simulation time of 20 seconds the resulting optimization time is not viable. Thus the goal is now to describe possibilities for accelerating the optimization process without changing the model accuracy. This mainly could be realized by the reduction of search space or model segmentation between the system bottlenecks and an embedding of model-specific parameters in algorithm control parameters.

5.1 The distance measurement

One possibility to increase algorithm convergence is a closer investigation of the search function f of the basic algorithm shown in 4.1. This function generates a new point $x_{i,m} := f(x_{i-1,m})$ in the neighborhood of the last point $x_{i-1,m}$. In the situation of permutation control variables the function f uses a distance measurement for neighborhood calculation. Therefore in general the so called Hamming distance is used. This is defined as follows (thereby x, x' are two arbitrary permutations of the same length):

$$d(x,x') = \sum_{i=1}^{n} b_i \quad \text{with} \quad b_i = \begin{cases} 0 & \text{if } x_i = x_i' \\ 1 & \text{if } x_i \neq x_i' \end{cases} \tag{16}$$

Horn et al (2006) have shown that this classical way isn't very efficient because there is only a small correlation between objective function and distance to optimum (with usage of distance measurement). This situation is illustrated on a benchmark flow shop (7 jobs, 5 machines) in the following Figure 9.

Figure 9: Correlation on Hamming distance

In this case, the search space is completely enumerated ($7!=5040$ possible job sequences). The best found objective function value lies in 0. Every solution is displayed as a dot in the diagram. Thereby, the objective function difference is displayed on the ordinate. On the abscissa the distance to optimum (calculated with the used distance measurement) is plotted. It can be seen that the Hamming distance is badly structured and assumes only a few different values. There is no real correlation between objective function and distance to optimum. For the algorithm this has the consequence that sequences x' lie in the direct neighborhood of sequence x with highly distributed objective function values. This makes the simulation-based optimization approach inefficient and implicates that a high number of iterations is needed to come close to the optimum.

An overview of existing distance measurements is given by Sevaux and Sörensen (2005). Horn et al (2006) also investigated model-specific distance measurements by using significant model information (like processing times) in distance calculation. For the above shown benchmark flow shop we reached good results concerning the correlation between objective function and distance to optimum (Figure 10).

Figure 10: Correlation on model-specific distance

This leads to an increased efficiency of simulation-based heuristic algorithms.

5.2 Construction of control variables

A further possibility to increase convergence is to reduce the search space reasonably already in the construction of the control variables. Typically, these are job sequences or job priorities.

A simple example illustrating the effect of the permutation from job sequence is shown in Fig. 11 and Fig. 12. The example has three jobs which have to be processed successively on two machines (M1→M2). The processing time of each job is reflected by the length of the job-bar in the so called Gantt chart. In the initial situation there is a job $x = (1, 2, 3) = (\text{Job1}, \text{Job2}, \text{Job3})$.

Figure 11: Initial situation

If we use x as control variable, the algorithm generates -starting from $x_{0,0} = (1,2,3)$- a new sequence $x_{1,0}$ (→ $x_{i,m} := f(x_{i-1,m})$). Depending on the function f this could be a simple exchange of two jobs, e.g. $x_{1,0} = (3,2,1)$. The effect of this permutation is shown in Fig. 12. The cycle time is reduced drastically and the throughput of machine M2 has increased.

Figure 12: Effect of job permutation

Unfortunately, now there are $J!$ possible control variable settings exiting (J is the number of jobs in the model) whose enlarging the search space extremely. So it is to reduce this search space by using model specific information.

Therefore, the special structure of the Backend model is exploited. Generally every job needs a special tool for processing on a machine. The machines have to be retooled if the product of two sequent jobs is not equal. Therefore it is desired not only to reduce the machine idle times but also the machine setup times which are defined by the product mixture.

In the following we describe a special designed control variable witch enables us not to change conventional job sequences but rather product mix sequences.

Generally, every job has its own calculated priority which is placed in the actual process step queue. In the simplest case the job priority is a number (rocket lot, prioritized job, normal lot, etc.). Furthermore, it is often necessary to compare constraints according to further attributes, such as ordering jobs of the same priority by their wait time. Figure 13 shows the calculation of a job priority key by reservation of definite decimal places for any constraint.

Figure 13: Job priority calculation (decimal places)

The following example (Fig. 14) illustrates the allocation of six jobs to a set of machines (M3 or M4) by adherence to tool setups.

Figure 14: Resulting Gantt chart

Obviously, it is guaranteed that a low prioritized job can never be processed before a high prioritized one. Also, all jobs of the same priority are exactly ordered by their wait time. The job priority key now can be increased arbitrarily. Figure 15 shows the additional insertion of a product priority (colored - 3) in the priority key. So it is possible to prioritize the products differently.

Figure 15: Extended job priority calculation (Job3)

Thereby, the permutation of the product prioritization represents the new control variable. All lots are furthermore exactly sorted by their priority and all lots of the same product (necessary for tool setup) are still sorted by their wait time. The effect of this permutation is shown in Fig. 16: The higher prioritized product releases the machine setup on machine M3. So in some optimization runs machine M3 will be retooled on product C, in some others on product D. As a result, the tool C and D are available for other machines in different time intervals.

Figure 16: Effect of product priority permutation (C↔D)

Especially in the case of a limited number of tools, this influences the global product mixture and consequently all optimization objectives. This simple example shows how the number of possible solutions can be reduced from *J*! to *P*! (*P* is the number of different products).

5.3 Multistage optimization

The duration of one optimization cycle is nearly equivalent to the simulation time of the underlying manufacturing model. For optimizing a highly complex manufacturing model, like the Backend, a large number of iterations is needed to get a significant objective function improvement. By segmentation of the initial model in several small models and their sequential optimization, the global optimization time can be reduced distinctly. Especially, this is very efficient for dissections made between the systems bottlenecks. The following investigations of the semiconductor Backend model shall illustrate that in detail. The initial situation is shown in the Figure 17. Two bottlenecks (Die Bond in Assembly and Test) could be detected in the Backend model.

Figure 17: Initial situation of single step optimization

All steps and stations not belonging to these bottlenecks are not shown in this illustration (dotted line). There are two control variables x_1 and x_2 influencing the lot sequence by their respective bottleneck. The optimization goal is to find a good lot sequence on both bottleneck machines (throughput maximization).

If one of the algorithms described in section 4 is now adapted for optimizing the model (Figure 17), generally a high number of optimization cycles is needed to get a significant objective function improvement. The reason is that only both control variable settings can be accepted or rejected during one optimization cycle by the algorithm. This means that a good lot sequence on the first bottleneck machine can be rejected if the sequence is poor on the second bottleneck machine or vice versa. This leads to the following scheme:

Figure 18: Multi-step optimization

In Figure 18 the Backend model is dissected into two single models (single model 1 and single model 2). This segmentation was carried out between the dedicated bottlenecks in the Assembly-Test passage. Thereby, the single model 1 contains the first bottleneck (Die Bond) and the single model 2 contains the second bottleneck (Test). Also the objective function is adapted to the single models.

At first we optimize the single model 1 and obtain an optimal configuration for x_1. The completion dates of the jobs in single model 1 are set as the earliest supply dates for the same jobs in single model 2. It follows an optimization of single model 2. The optimization of both single models is thereby much faster, because of the decreased simulation time per single model and decreased the number of needed optimization cycles. The descried reduction of the search space has the disadvantage that the global optimum may not be in the current search space. But the advantage of shorter optimization time outweighs this disadvantage.

6 RESULTS AND VISUALIZATION

In the following, some results and further investigations are introduced. Thereby, it was used a multi-objective function *C* consisting of idle time minimization (for bottleneck machine groups Die Bond and Test, weighted by $\omega_1 = \omega_2 = 0.3$) and cycle time minimization (Assembly and Test, weighted by $\omega_3 = \omega_4 = 0.2$). In order to hide the real manufacturing parameters, the substitute objective function C is normalized by method 2 described in 4.2. So the initial objective function value (simulation of the manufacturing dispatch strategy) is 1.

In a first scenario, the Backend model was investigated by a random walk with 2500 iteration steps to get an impression of the spread and the dimension of the single objectives and the resulting substitute objective function. In every case, the introduced control variable (section 5) was used. Correlation coefficients were calculated in order to compare several distance measurements. Table 2 shows the result for the well known Hamming distance and the so called Deviation distance[1]. The latter should be more suitable for permutations. Principally, this could be approved of. However, the correlation coefficient is still not high enough to generalize this statement. Never the less, the Deviation distance was used for all of the later investigations.

Table 2: Correlation coefficients in dependence to the used distance measurement

In a next step the Backend model was optimized by single step heuristics. The RW of 2500 iterations was used as a reference. It reached the best object value:

$C_{RW}(x_{\min}) = 0.88$

To ensure "Online" time constraints, only 100 steps were evaluated by the single step algorithms. However, the results (see Table 3) are acceptable. An overall objective function improvement of 10-12% is possible. Thereby, the Backend model was optimized 10 times for every displayed algorithm. The best found objective function value $C(x_{\min})$ and the experimental standard deviation σ are denoted per algorithm.

Table 3: Results of single-step algorithm (100 steps)

As a next step the multi-step optimization approach (section 5.3) was investigated. This means, every single model was optimized by a 100-step-search heuristic. This approach shows the fastest performance, because of its smaller simulation models. The results are shown in Table 4. It can be seen that an overall objective function improvement of max. 14% is possible. All optimization runs were also repeated 10 times to exclude stochastic influences.

Table 4: Results of multi-step algorithm (100/100 steps)

One conclusion of this investigation is the possibly significant improvement of the objective function by adapting the described heuristics to the simulation-based optimization approach. The new heuristics provide fast and good results. There are several reasons for this: the control variable has significant influence on all objectives; the distance measurement allows a target-oriented optimization; the multi-step optimization reduces the search space in every step.

For further studies and investigations of the optimization algorithms' behavior and its parameter influences, the visualization tool OptVis3D (Klemmt and Weigert 2008) was developed. As an example, a three-dimensional objective space of the backend model is shown in Figure 19 (labeling dropped because of nondisclosure).

Figure 19: Visualization of optimization processes

[1] For more details see [15]

It can be seen the visualization of three different optimization. All accepted optimization steps of each algorithm are connected by a solid line. The small cumulation of dots is the search space structure, denoted by Random Walk. Also the visualization of the best found value (squares) or the Pareto front (grey surface) of the particular algorithm is possible. The Pareto front is thereby the set of all optimal points (concerning Random Walk) by different single objective weighting. In the diagram there is a single step GY (100 steps), a 2-step RRT (100/100 steps) and 2-step OBA (100/100 steps) displayed (representing some results of Table 3 and 4). All algorithms break through the Pareto front. This means that the algorithms found a better solution after less than 100 steps compared to the Random Walk after 2500 steps. For most of the researched cases the statistical analysis shows better results for 2-step algorithms than for 1-step algorithms, regardless of the search space reduction. As an additional advantage, multi-step algorithms are faster; they need approximately 50% of the run time of the related 1-step algorithms.

7 CONCLUSION AND OUTLOOK

Simulation-based scheduling is a suitable method for optimizing manufacturing processes. Even for highly complex manufacturing models iterative heuristic search algorithms are applicable and have advantages over mathematical solvers.

The application of heuristics for the online optimization under practical conditions is generally complicated by their long optimization time. But the described methods of search space reduction, model segmentation and the usage of non-standard distance measurement increased their efficiency clearly. So, depending on model-specific parameters, it is possible to design highly efficient methods for improving the convergence of the algorithm. Several visualization options to analyze the optimization process in detail are also described. The shown methods can also be used in other simulation based optimization approaches.

For the future work it is planned to use these methods for optimization of several work centers in the Frontend (the wafer fabrication) of a semiconductor manufacturer as well as the coupling of simulation based heuristics with mathematical solvers.

8 ACKNOWLEDGMENTS

The work for this paper was supported by Qimonda Dresden and by the EFRE fund of the European Union and funding of the State Saxony of the Federal Republic Germany (project number 8007/1289).

9 REFERENCES

Beier, E. : Entwicklung und Bewertung simulationsgestützter Optimierungsverfahren für die Halbleiterfertigung, Diploma thesis, Technische Universität Dresden, 2006.

Dueck, G., 1989. New Optimization Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel, IBM Germany, Heidelberg Scientific Center.

Gupta, A.K. and Sivakumar, A.I., 2002. Simulation Based Multiobjective Schedule Optimization in Semiconductor Manufacturing, Winter Simulation Conference, 1862-1870.

Horn, S.; Weigert, G.; Beier, E., 2006. Heuristic optimization strategies for scheduling of manufacturing processes, Papers of the 29th International Spring Seminar on Electronics Technology.

Hu, T.C.; Kahng, A.B.; Tsao, C.A., 1995. Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods, ORSA J. Comput. 7, No.4, 417-425.

Klemmt, A. and Weigert, G., 2008. 3D-Visualisierung heuristischer Optimierungsalgorithmen, 19th Conference on Simulation and Visualization, Magdeburg, Germany, 167-180.

Klemmt, A.; Horn, S.; Weigert, G., 2008. Analysis and Coupling of Simulation-based Optimization and MIP Solver Methods for Scheduling of Manufacturing Processes, 18th International Conference on Flexible Automation and Intelligent Manufacturing, Skövde, Sweden, Proceedings 1146-1153.

Kuhn, H. and Quadt, D., 2002. Lot sizing and scheduling in semiconductor assembly - a hierarchical planning approach, International Conference on Modeling and Analysis of Semiconductor Manufacturing, 211-216.

Metropolis, N.; Rosenbluth, A.; Rosenbluth, M.; Teller, M.; Teller, E., 1953. Equations of state calculations by fast computing machines, Journal of Chemistry Physics, 21, 1087-1092.

Mönch, L. and Zimmermann, J., 2004. Improving the performance of dispatching rules in semiconductor manufacturing by iterative simulation. Proceedings of the 2004 Winter Simulation Conference. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Quadt, D., 2005. Scheduling a semiconductor backend: from theory to practice. Proceedings of the 3rd International Conference on Modeling and Analysis of Semiconductor Manufacturing, MASM 2005, Singapore Institute of Manufacturing Technology.

Ponnambalam, S.G.; Aravindan, P.; Rajesh, S.V., 2000. A Tabu Search Algorithm for Job Shop Scheduling, The International Journal of Advanced Manufacturing Technology Vol. 16, 765-771.

Potoradi, J., Mason, S., Fowler, J., 2002. Using Simulation based Scheduling to maximize Demand Fulfillment in a Semiconductor Assembly Facility, Winter Simulation Conference, 1857-1861.

Rajendran, C. and Ziegler, H., 2004. Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs, European Journal of Operational Research, Nr. 155, 426–438.

Rose, O., 2003. Accelerating products under duedate oriented dispatching rules in semiconductor manufacturing. Proceedings of the 2003 Winter Simulation Conference. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Scholl, A.; Klein, R.; Häselbarth, L., 2003. Planung im Spannungsfeld zwischen Informationsdynamik und zeitlichen Interdependenzen, Friedrich-Schiller-Universität Jena, ISSN 1611-1311.

Sevaux, M. and Sörensen, K., 2005. Permutation distance measures for memetic algorithms with population management, Proceedings of MIC, 6th Metaheuristics International Conference, Vienna.

Sivakumar, A. I., 1999. Optimization of Cycle Time and Utilization in Semiconductor Test Manufacturing using Simulation Based, On-Line, Near-Real-Time Scheduling System, Winter Simulation Conference, 727-735.

Sung, C.S.; Choung, Y.I.; Fowler, J.W., 2002. Heuristic algorithm for minimizing earliness-tardiness on a single burn-in oven in semiconductor manufacturing. Proceedings of the International Conference on Modeling and Analysis of Semiconductor Manufacturing (MASM 2002), Tempe, AZ, 217-222.

Tovia, F.; Mason, S.; and Ramasami, B., 2004. A scheduling heuristic for maximizing wirebonder throughput. IEEE Transactions on Electronics Packaging Manufacturing, (27):2, 145-150.

Weigert, G.; Horn, S.; Werner, S., 2005. Optimization of Manufacturing processes by distributed simulation, 18th International Conference on Production Research.

Hamming distance	**Deviation distance**
0.104	0.202

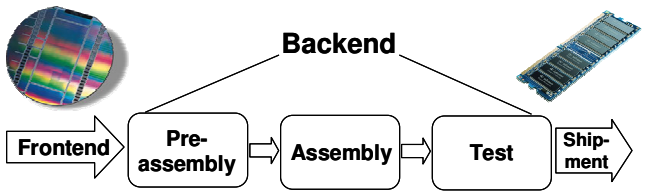

Figure 1: Simplified Backend workflow

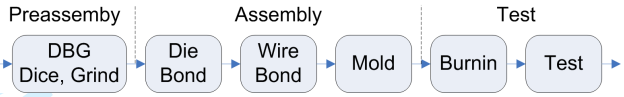

Figure 2: Technological Backend steps

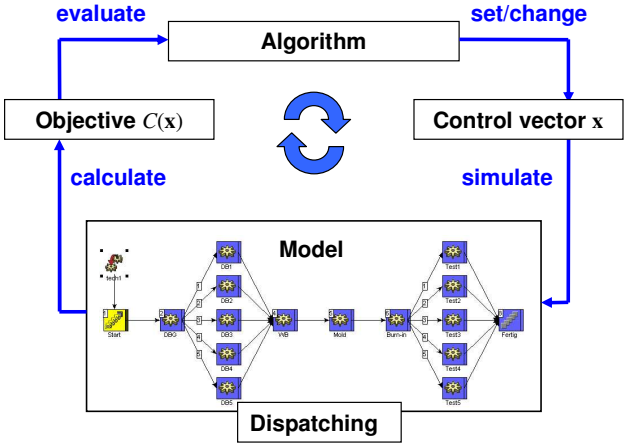

Figure 3: Basic optimization cycle.

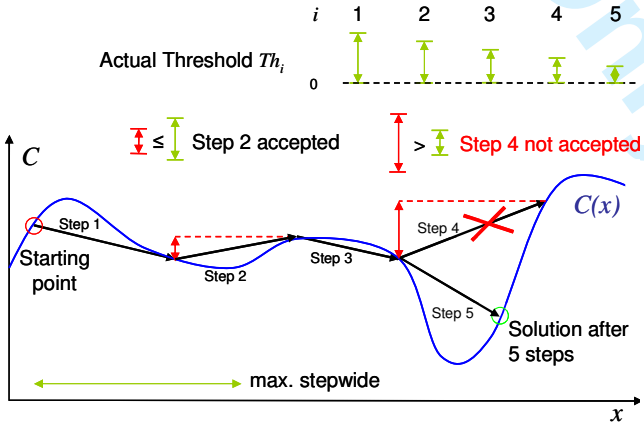

Figure 4: Functionality of Threshold Accepting

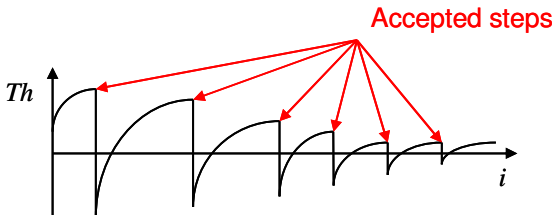

Figure 5: Threshold behavior OBA

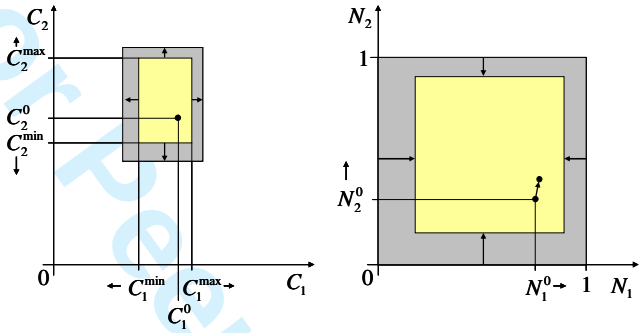

Figure 6: Original and normalized objective area (method 1, qualitative)

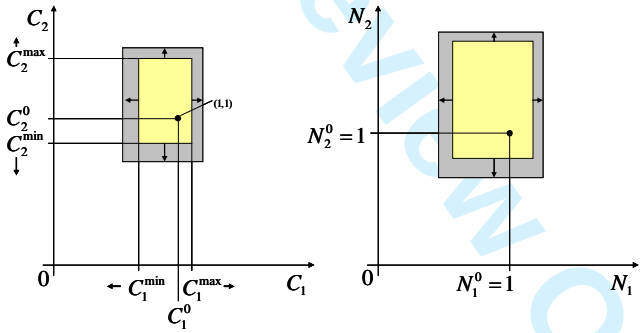

Figure 7: Original and normalized objective area (method 2, qualitative)

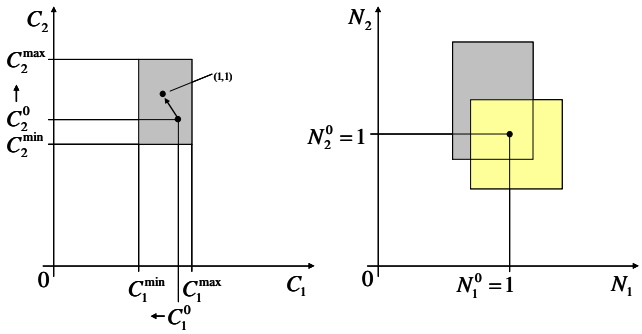

Figure 8: A new start point moves and changes the normalized objective area (method 2, qualitative)

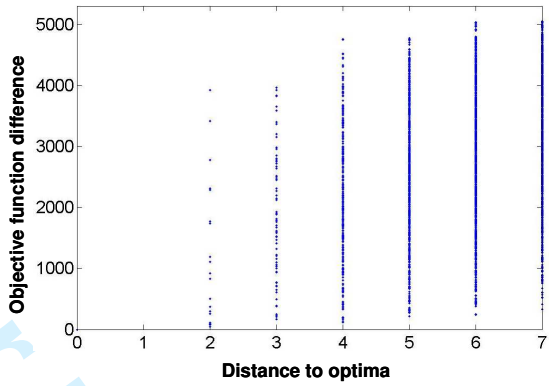

Figure 9: Correlation on Hamming distance

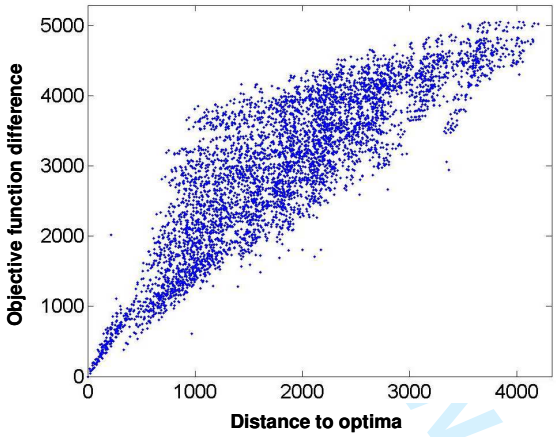

Figure 10: Correlation on model-specific distance

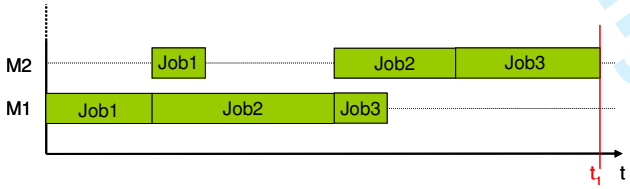

Figure 11: Initial situation

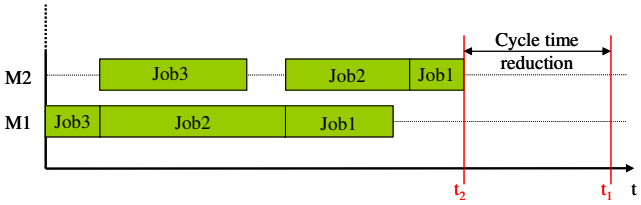

Figure 12: Effect of job permutation

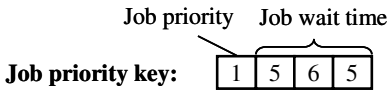

Figure 13: Job priority calculation (decimal places)

Job	Priority	Wait time	Product	Priority key
Job1	3	103	A	3103
Job2	2	045	B	2045
Job3	1	565	C	1565
Job4	1	455	D	1455
Job5	1	321	C	1321
Job6	1	003	D	1003

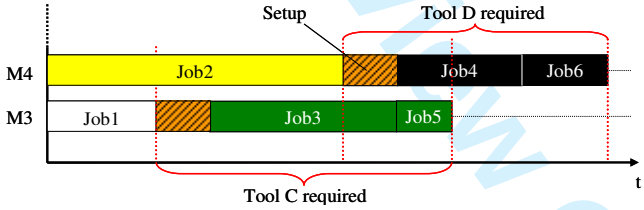

Figure 14: Resulting Gantt chart

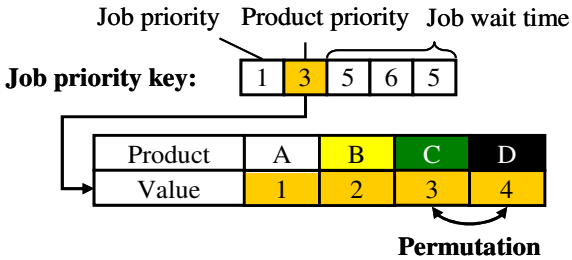

Figure 15: Extended job priority calculation (Job3)

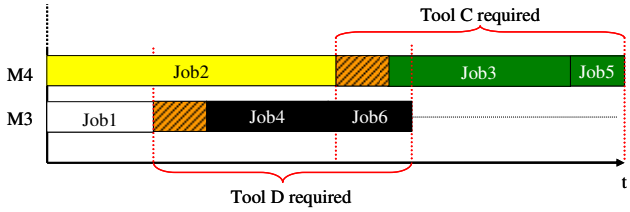

Figure 16: Effect of product priority permutation (C↔D)

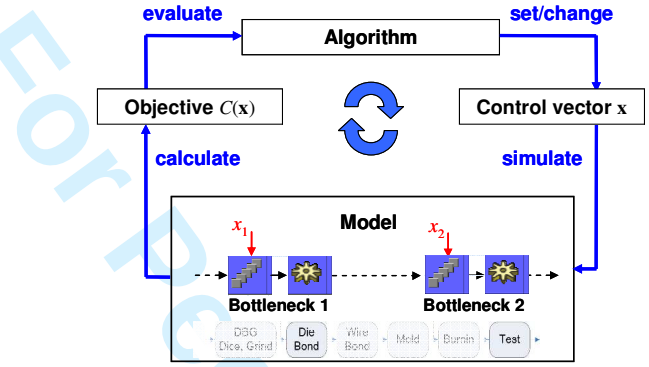

Figure 17: Initial situation of single step optimization

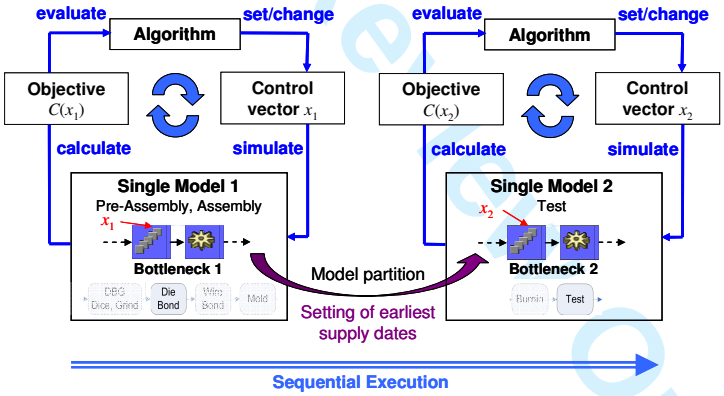

Figure 18: Multi-step optimization

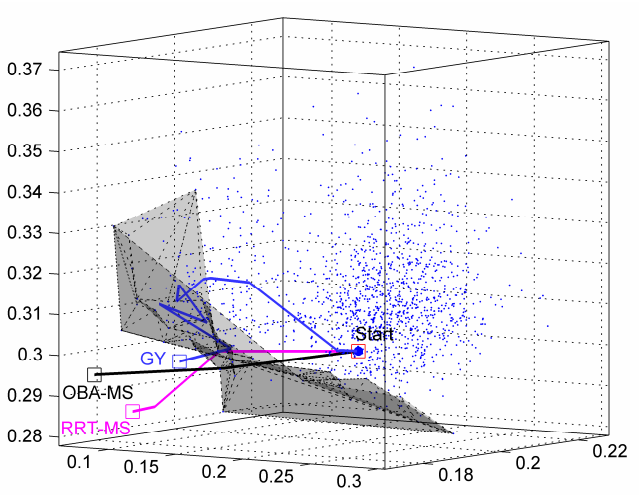

Figure 19: Visualization of optimization processes

Algorithm / Literature	Probability of acceptance	Algorithm control parameter
RW / (Weigert et al 2005)	$p_i = 1$	
GY / (Weigert et al 2005)	$p_i = \Theta(\Delta C_i)$	
TA / (Weigert et al 2005)	$p_i = \Theta(Th_i - \Delta C_i),\ Th_i = Th_0\left(1 - iN^{-1}\right)$	Th_0 initial threshold (Th_0 >0)
RRT / (Dueck 1989)	$p_i = \Theta\left((1+\alpha)\cdot C(x_{\min}) - C(x_{i,m})\right)$	α worsening percentage ($\alpha > 0$)
GD / (Weigert et al 2005)	$p_i = \Theta\left(L_i - C(x_{i,m})\right),\ L_i = L_0 - i\Delta L$	ΔL rain quantity L_0 initial water level
SA / (Metropolis 1953)	$p_i = e^{\frac{-\Delta C_i}{Th_0\left(1-iN^{-1}\right)}}$	Th_0 initial temperature
OBA / (Hu et al 1995)	$p_i = \Theta(Th_i - \Delta C_i)$, $Th_i = \left(\left(\frac{age}{J}\right)^b - 1\right)\cdot\frac{C(x_{\min})}{J}\cdot\left(1-\frac{i}{N}\right)^c$	J number of jobs in the model b allows a power law growth rate c tunes heuristic "damping" factor $\left(1-\frac{i}{N}\right)$ age threshold growth rate $\begin{cases} 0 \text{ if step is accepted} \\ incr(age) \text{ otherwise} \end{cases}$

Algorithm	$C(x_{min})$	**σ**
RRT	0.881	0.007
GY	0.893	0.049
OBA	0.879	0.010

Algorithm	$C(x_{min})$	**σ**
TA	0.893	0.027
GD	0.875	0.011
GY	0.862	0.005
RRT	0.871	0.004
OBA	0.863	0.006