18th ICPR paper: Optimisation of manufacturing processes by distributed simulation

Gerald Weigert, Sven Horn, Sebastian Werner

► To cite this version:

Gerald Weigert, Sven Horn, Sebastian Werner. 18th ICPR paper: Optimisation of manufacturing processes by distributed simulation. International Journal of Production Research, 2006, 44 (18-19), pp.3677-3692. 10.1080/00207540600851774 . hal-00512934

HAL Id: hal-00512934

<https://hal.science/hal-00512934>

Submitted on 1 Sep 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**18th ICPR paper: Optimisation of manufacturing processes
by distributed simulation**

Journal:	<i>International Journal of Production Research</i>
Manuscript ID:	TPRS-2005-IJPR-0518.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	06-Jun-2006
Complete List of Authors:	Weigert, Gerald; Dresden University of Technology, Electronics Packaging Laboratory Horn, Sven Werner, Sebastian
Keywords:	OPTIMIZATION, SIMULATION, SCHEDULING, MANUFACTURING PROCESSES
Keywords (user):	Distributed simulation

Optimisation of manufacturing processes by distributed simulation

G. WEIGERT*†, S. HORN‡ and S. WERNER§

† Dresden University of Technology
Electronics Packaging Laboratory
D-01062 Dresden, Germany

‡ Dresden University of Technology
Centre of Microtechnical Manufacturing
D-01062 Dresden, Germany

§ Dresden University of Technology
Centre of Microtechnical Manufacturing
D-01062 Dresden, Germany

The paper describes a simulation-based optimisation system for manufacturing processes. The principle suitability of simple Threshold Accepting and Great Deluge Algorithms for optimising practical manufacturing systems is shown as well as the possibility of parallelising of these algorithms. A special IP Multicast architecture is introduced which allows to spread the simulation model to multiple clients and to increase so the efficiency of the optimisation algorithms by concurrent simulation. Each client can log on or log out at an optimisation server at any time so that it should be possible to use idle computer capacity of a single production department or a whole company. The optimisation server is implemented as an experimental option in the simulation system simcron MODELLER® which is particularly used in electronics and semiconductor manufacturing.

Keywords: Scheduling; Distributed simulation; Optimisation; Manufacturing

1 Simulation aided optimisation

Frequently, the complexity of practical problems of production control is too high to be solved by usual analytical methods. Heuristic optimisation algorithms in connection with simulation systems are a suitable alternative in these cases.

Figure 1: Basic optimisation cycle.

* Corresponding author. Email: weigert@avt.et.tu-dresden.de
<http://mc.manuscriptcentral.com/tprs> Email: ijpr@lboro.ac.uk

Figure 1 shows the basic principle of a simulation aided optimisation system. The problem is described by a simulation model which includes a set of control variables and responds with an objective value C after a simulation run is completed. In the case of multi-criterial optimisation problems the individual objectives can be combined to a scalar replacement, the so called fitness. The control vector x consists of several variables which influence the behaviour 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 x in the following formulas. So the optimisation system is divided into an assessment part on one hand and a separate optimisation algorithm on the other hand. Both subsystems communicate by x and C while the optimisation cycle is running. In the simulation system *simcron* MODELLER® this optimisation cycle is implemented as an experimental feature (*simcron* 2005).

2 Heuristic optimisation algorithms

2.1 Application in production planning

Most of the practical problems of production planning and scheduling are NP-hard, this means that no efficient algorithms are available. In connection with simulation methods some heuristic algorithms have been established instead. First of them was probably the Simulated Annealing (SA) which was already proposed by Metropolis in 1953 (Metropolis *et al.* 1953). The Simulated Annealing was and is one of the standard algorithms for solving discrete optimisation problems for a long time until today (see Lee, D.S. *et al.* 2002), supplemented more and more by the more recent Genetic Algorithms (GA) in the meantime. They are based on a large number of possible solutions (individuals) which are put together in a so called population. One of the advantages of GA is that the fitness of the individuals of a population can be computed in parallel. The movement through the search space is caused by gene crossing, mutation and selection. In the case of sequence optimisation problems a special repair mechanism avoids unfeasible permutations, how described e.g. by Hampel (Hampel 1997). A lot of practical applications, particularly scheduling systems, use GA or offer at least a GA-based option for optimisation. So Pongcharoena developed a GA-based scheduling tool for complex products with multiple resource constraints and deep product structure (Pongcharoena *et al.* 2002). An Advanced Planning System was developed by Lee which selects the best machine for each operation in a manufacturing supply chain by using GA (Lee, Y.H. *et al.* 2002). As a result they could minimise the makespan (regarding to the due date) for each order.

Gonçalves presents a hybrid algorithm for Job Shop scheduling problems where the schedules are constructed using a priority rule in which the priorities are defined by the GA and an additional local search heuristic is applied to improve the solution (Gonçalves *et al.* 2002). This already leads over to a new class of algorithms which are simpler structured as SA or GA but which are successfully used for several problems of production planning and scheduling. Two of them, the Threshold Accepting (TA) and the Great Deluge (DA) are described in the following. The TA algorithm was proposed by Dueck and Scheuer in 1990 (Dueck and Scheuer 1990) and since then it was successfully used for several discrete planning problems. Schneider (Schneider 1999) compared TA and the similar Great Deluge algorithm (GD) with Simulated Annealing in his PhD thesis and he could show there that these heuristics were suitable for solving Travelling Salesman Problems (TSP). Later he introduced a case study for assembly lines in car production and compared again TA with SA (Schneider *et al.* 2000). Lee successfully applied the TA for small- to moderate-size Zero-Wait scheduling problems and compared it with Simulated Annealing, too (Lee, D.S. *et al.* 2002). Furthermore he modified the classical TA to the so called List Based Threshold Accepting (LBTA) where the threshold parameter is controlled by a list. The LBTA was used for solving Job Shop scheduling problems (Lee, D.S. *et*

al. 2004). Feldmann and Biskup investigated 140 benchmark problems for scheduling jobs on a single machine against restrictive common due dates, among them problems up to 1 000 jobs. He could demonstrate that TA was efficient in comparison to evolutionary strategies as well as SA (Feldmann and Biskup 2003). Schmidt and Thierauf proposed a hybrid heuristic for optimisation of continuous and/or discrete variables based on Genetic Algorithm and Threshold Accepting (Schmidt and Thierauf 2005). Before a new population is created by GA, each individual in the given population is sequentially optimised by TA. Especially for sequence problems the Tabu Search Algorithm (TS) is also suitable. In contrast to the other introduced heuristics it is a deterministic algorithm based on the definition of neighbourhood of a permutation. Here the search is restricted to a comparatively small part of the whole space around the current point. The algorithm goes on while better solutions are found. Tabu rules should avoid undesired oscillation between a few points. A disadvantage of TS is that it is difficult to consider additional scalar control values to the optimisation problem, like machine capacities or job priorities. (Schulz 1996) This short but not at all complete overview should show that simple heuristics like Threshold Accepting become more and more important in practical applications. TA as well as GD, in parts in contrast to the more traditional algorithms SA and GA, are easy to implement, are very effective and are distinguished by a high flexibility.

2.2 The basic algorithm

Most of the heuristic optimisation algorithms can be traced back to the following general basic procedure: Starting from a given point x in the search space. A new point x' is generated by a specific function f . In the iteration i the point x is substituted by x' with the acceptance probability of p_i (the point is moved from x to x'), otherwise the location is kept. This process is repeated until the predefined number N of iterations is reached or a termination condition was met before.

```
M := 0
m := i := 0
xi,m := xstart
while (i < N and not termination condition) do {
    i := i+1
    xi,m := f(xi-1,m)
    if (random (0,1) < pi) then {
        xi,i := xi,m
        m := i
        M := M+1
    }
}
```

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

G. Weigert et al.

The diverse search strategies differ in the definition of the search function f or the acceptance probability p_i . In addition, it is assumed that a single objective or a cost function $C_i = C(x_i)$ is defined and it is asked for the minimum $C_{\min}$ of this function and for the related value of the control variable $x_{\min}$ (the formulas are applicable to maximum problems in general sense).

$$C_{\min} = C(x_{\min}) \quad (1)$$

Heuristic algorithms find the theoretically reachable minimum only at random, usually the result is just a suboptimal solution $\tilde{C}_{\min} = C(\tilde{x}_{\min})$ with $\tilde{C}_{\min} \geq C_{\min}$. Depending on the acceptance probability p_i , one can distinguish between two basic strategies. The first strategy is the so called Random Walk (RW) where each of the iterations is also a move in the search space ($M = N$):

$$p_i = 1 \quad \forall i \quad (2)$$

$$\tilde{x}_{\min} = x_{k,k} \quad \text{with} \quad C(x_{k,k}) \leq C(x_{m,m}) \quad \forall m \quad (3)$$

The second strategy is the Greedy¹ (GY) which accepts moves only in event of improvement of the objective function ($M \leq N$):

$$p_i = \Theta(-\Delta C_i) \quad (4)$$

$$\Delta C_i = C(x_{i,m}) - C(x_{m,m}) \quad (5)$$

$$\tilde{x}_{\min} = x_{M,M} \quad (6)$$

Θ is the Heaviside function, with $\Theta(x) = 0$ for $x \leq 0$ and otherwise 1.

Figure 2: Heaviside function.

The GY gives better results when the search point is located close to the optimum. But otherwise it can be attracted by a suboptimum, from where it will never find back. The advantage of the RW is that it can search in the whole space over all iterations. So it cannot be permanently caught by a suboptimum, but usually it needs more time for searching than GY.

The idea of most of the heuristic algorithms is to combine both strategies. Simulated Annealing, the first applicable algorithm of this type, simulates a melting pot where the mobility of the atoms increases for higher temperature. Like this, the acceptance probability is driven by an additional temperature variable, so that the algorithm starts as a RW and ends as a GY, when we assume that the search point probably approaches the optimum at the end of the search. Threshold Accepting algorithm as well as the Great Deluge algorithm has

¹ The Greedy is also often denoted as Hill-climbing algorithm.

the same basic behaviour than Simulated Annealing. Because of their easier structure, they could help to save computation time. TA and GD are shortly introduced in the following.

2.3 Threshold Accepting algorithm

In opposite to the pure GY the TA accepts not only improvements. The probability of acceptance of a move can be calculated as follows.

$$p_i = \Theta(Th_i - \Delta C_i)$$
 (7)

This means, a move is executed if $\Delta C_i \leq Th_i$, otherwise it is a simple iteration without moving. The Threshold Th is the control parameter which drives the algorithm. For high values of Th it reacts like RW and it changes to GY for a lower threshold. It is assumed that the algorithm starts with Th_0 whereas two basic strategies for threshold adaptation exist:

- Arithmetical: $Th_i = Th_0 - b \cdot i$ with $b > 0$
- Exponential: $Th_i = b \cdot Th_{i-1}$ with $0 < b < 1$

With the help of the parameters Th_0 and b , the optimisation process can be controlled. The precise values of them strongly depend on the problem itself and, last but not least, are influenced by the experience of the optimisation expert.

2.4 Great Deluge algorithm

This algorithm got its name from the history from the Bible about the Great Deluge. But in difference to the well known story Noah's Ark is missed so that the walker is driven higher and higher by the increasing water level. Usually every place can be reached at the beginning but the landscape will be divided into several isles while the time proceeds.

$$p_i = \Theta(L_i - C_i)$$
 (8)

$$L_i = L_{i-1} - \Delta L$$
 (9)

L is the water level and ΔL the rain quantity. Here, for the described minimum problem, it should be called the 'Great Drought Algorithm'. In this case ΔL is actually the evaporation quantity and the walker must be replaced by a fish which is always looking for the deepest place in the water.

Water level as well as the rain quantity is the control parameter of this algorithm. For GD the probability of acceptance is independent to ΔC and depends only on the absolute value of the objective function C . Similar to TA the probability decreases for higher cycle numbers. In opposite to TA the GD algorithm cuts off an increasing part of the search space with progressing time so that the danger of isolation is higher than for TA.

2.5 TA and GD in comparison to Simulated Annealing

Metropolis already proposed in the year 1953 a heuristic which was called Simulated Annealing (SA) (Metropolis *et al.* 1953). Since this time SA is successfully used for solving several discrete optimisation problems, among them are also sequence optimisation problems. The algorithm follows an analogy of a melt where the atoms are more movable for higher temperature T than for lower one. Assigned to an optimisation problem the probability of acceptance depends on the control parameter T as shown in equation (10).

G. Weigert et al.

$$p_i = \min \left\{ 1, \exp \left(-\frac{\Delta C_i}{T} \right) \right\} = \begin{cases} \exp \left(-\frac{\Delta C_i}{T} \right) & \text{if } \Delta C_i > 0 \\ 1 & \text{else} \end{cases} \quad (10)$$

The SA algorithm starts with high “temperature” so that - similar to TA - the probability of acceptance p_i is high. For very high temperature the SA behaves like the RW strategy, with a probability value close to 1 and nearly independent to ΔC . At the end of the optimisation process the temperature should be so low that the algorithm behaves like the GY strategy, as shown in figure 3. Then the degradation of the objective function is very improbable. The main difference between the Simulated Annealing and the Threshold Accepting is, that the value of p_i for TA is switched only between 0 and 1 while for SA all values between 0 and 1 are possible. In other words, the TA (and in an extended sense GD too) is a more deterministic algorithm than SA. But the basic principle is the same, maybe this explains the surprising success of these simple algorithms.

Figure 3: Probability of acceptance depending on ΔC and cycle number i for Simulated Annealing (left) and Threshold Acceptance (right).

3 Parallel optimisation system

Complex models with a lot of constraints cause a higher computational effort of the simulation. In such cases it is difficult to use heuristic algorithms with cyclic simulation runs. One solution to this issue is the distribution of the simulation model to several client computers, and optimisation in parallel.

The simcron MODELLER is a powerful and proved discrete event simulation system which was essentially developed by the authors, too. In the last years we expanded the simulator by an additional optimisation tool including several heuristic algorithms. The simcron MODELLER engine includes a protocol stack for the recurrent requirements - the distribution of the model - and the interchange of the parameters and results. This protocol stack acts as an extension to the Internet Protocol Stack (IP) and is based on IP Multicast. The IP Multicast architecture is an ideal platform for distributing model data over the network. It reduces the network overhead for the sending of the model to different clients and simplifies also the initial setup between client and server. The client and the server are able to find each other automatically by preconcerted multicast channels. Only the predefined multicast group and the IP port must be known to both sides.

The concept and the protocol give the possibility to bundle the productivity of a number of computers in a work group to optimise simulation models and realise online-optimisation tasks. These tasks typically occur in real-time scheduling environments in production control scenarios (Hampel 1997). In such cases short and fixed time constraints are also typical to find an optimised solution. Here, the distribution to several clients can improve the optimisation results significantly.

Figure 4: Distributed simulation and parallel optimisation system.

Figure 4 shows the approach and the schematic data flow of the model distribution and result data flow, how we have realised it in the simcron MODELLER. With this application it is possible to distribute simulation models over the network and parallelise optimisation cycles of different heuristic algorithms. A special Competition Mode is realised, too. It makes it possible to start a contest between several clients as for example the ‘SETI@home’ project does. In this mode the optimisation runs for a longer time, typically for several days. Clients can leave and join at any moment and their count of calculated cycles is accumulated during the whole time.

Figure 5: Speedup of the optimisation system under several conditions.

In performance tests with this system the speedup was measured up to 8 clients. Figure 5 shows these results as diagram. In the measured situation only the clients optimise the model, the server system was just waiting and coordinating. The higher speedup using an optimisation time of 360 seconds is a major result of the initial

G. Weigert et al.

overhead distributing the model over the network. 360 and 120 seconds are chosen to come close to typical cases in production environments, where often a fast optimisation result is required. Anyway, the distribution of optimisation tasks is suitable down to a number of two available clients. (Horn *et al.* 2005, Fachat 2000)

4 Experiments and results

In the following the optimisation system is used for 2 models which are derived from real systems of electronic production. The first model (Model 1) is quite simple so that the simulation time is short and a distributed simulation is not necessary. The second model (Model 2) is much more complex. In this paper the complicated control rules are not described in detail but they are completely realised in the simulation model, partly as scripts. These scripts extensively extend the simulation time so that the optimisation process must be accelerated by concurrent simulation of multiple models. We consciously did not examine so called benchmark models because their structure is mostly simple. Hence the time needed for simulation runs is short and the advantages of distributed simulation do not stand out in this case. Particularly the second model includes a lot of restrictions and special setup and dispatch rules. So we can better predict the behaviour of the optimisation algorithms under practical conditions.

4.1 Model 1

Model 1 is structured like a Flow Shop with nine machines and 65 jobs, waiting in the input queue. A conveyor transfers the jobs to the machines automatically. If the requested machine is busy, the job is placed into the related buffer in front of the machine with first-in-first-serve discipline. Because of the limited capacity of two jobs of each buffer blocking effects have to be considered. There are 10 different routes (T1, T2, ..., T10) through the system which are distinguished by skipping different machines. In addition the processing time depends on the machine and on the job respectively the route. The simulation starts with an empty system and ends when all jobs are completed. The objective is to minimise the cycle time by changing the input sequence of the jobs for non-delay schedules. The cycle time for the initial sequence is 215 160 seconds (about 2.5 days); this is approximately 130% of the best known value of 165 780 seconds. By the way, the longest cycle time which was found is 259 980 seconds or 157% of the shortest one. The system does not fit the scheduling theory with all its parts. It looks like a normal Flow Shop but is more similar to a Job Shop problem which is described as J9|block|C_{max}. (Pinedo 1995)

Figure 6: Job Shop with the example routes T1, T2 and T10.

Optimisation of manufacturing processes

The model generates 1 075 events and needs 0.03 seconds at a 1.7 GHz Pentium 4 processor for a complete simulation run. An optimisation process of 1 000 iterations is shorter than 2.5 minutes.

Figure 7: Typical convergence curve for RW and GY.

Figure 8: Typical convergence curve for GD and TA.

Figure 7 shows the typical convergence curve for RW in comparison to the GY, the two extreme variants of the heuristic search strategies. How expected, GD and TA are arranged between these extreme points as shown in figure 8. GD was started with a water level of $L_0 = 250\,000$ and the rain quantity was amounted -250. For TA the initial threshold Th_0 was set to 20 000, approximately the largest degradation to be ever observed between two successive optimisation cycles. An arithmetical threshold adaptation was selected with $b = 20$. GD, as well as TA, starts like Random Walk and ends with behaviour of Greedy after 1 000 iterations. The search function f generates new points based on the last one at random whereas the increment is limited by a constant maximal number of permutations for all algorithms.

From the start value of cycle time to the best known value so far is a range of 49 380 seconds. Only a few optimisation runs end close to the best value within 1 000 iterations. But most of them fall below the 50% level of the full range after some ten or hundred iterations. The iteration, in which this occurred, is a random number. Table 1 show this 50% iteration number for RW, GD, TA and GY and for every sequent optimisation run. The average iteration number is a characteristic for the algorithm and describes its speed of convergence in the start phase. The results approve the assumption that the GY is the best algorithm for this special problem. The reason for this observation could be that there are a lot of homogenous suboptimal points which are evenly spread in the search space. In this case the start point would have some suboptimal points in its

G. Weigert et al.

neighbourhood so that GY often succeeds. This is similar to the situation of the end of search where the search point is usually close to the optimum.

Table 1: Iterations until 50% level for 10 runs.

Run	RW	GD	TA	GY
1	58	102	4	11
2	199	23	109	29
3	642	169	14	97
4	25	127	28	152
5	124	98	29	29
6	42	74	191	34
7	177	12	14	18
8	7	109	127	23
9	81	95	152	15
10	26	163	59	18
Average	138	97	73	43

In practical applications the termination condition is not derived from the convergence curve but it is just the time which is available for optimisation. One is satisfied if the cycle time is reduced for a particular percentage or the optimisation run is aborted after a determined time. Under this condition the GY would give the best results.

For comparison some experiments with Genetic Algorithms were accomplished. In figure 9 three typical convergence curves of GA and GY are chosen and contrasted. The three thin upper curves belong to a usual GA with a population size of 50 individuals. The iteration variable was increased for each modified individual within the population so that the computation time for the same iteration value is comparable. It is shown that the convergence rate until the 1 000th iteration of the GA is not better, in most of the cases even worse, than the simple GY. This does not considerably change by using other population sizes, mutation rates or selection strategies.

Figure 9: Comparison between GA and GY.

In general, of course, the GA leads to better results than TA or GD in longer runs. In addition it does not tend to stick in a local optimum, how particularly the GY does. This could also be demonstrated with our own experiments with several benchmark models, such as the well known Muth&Thompson 10x10 Job Shop model (Fisher and Thompson 1963). But under the condition of the industrial application, especially for online scheduling, the short term convergence behaviour is much more interesting than the ultimate optimum, which is found not until a long time after the optimisation run was started.

4.2 Model 2

The second model is a snap shot of a running manufacturing process where 67 jobs have already been started and have been distributed over the machines and buffers and 40 jobs wait in an input queue in front of the system. The ratio between the shortest and the longest throughput time of all jobs is approximately 1:10. The machine system is organised like a Flexible Flow Shop with 10 stations (Pinedo 1995). Each station consists of a different number of parallel machines. The stations themselves are decoupled from each other by buffers with unlimited capacity so that blocking effects are not to be considered. Station 4 is a bottleneck with high utilisation but it is not a distinctive one, because at least two other stations can be highly utilised from time to time, too. The manufacturing process is controlled by job priorities whereas the following general dispatch rule is used for all queues inside the system: The job with the highest priority is served first in the next machine.

Figure 10: Flexible Flow Shop with bottleneck.

To minimise the average throughput time of the jobs (the number of Jobs is J) is one of the objectives of this scheduling problem. The throughput time of a job is defined here as the time difference between the date of completion of its last processing step $t_{j,end}$ and its supply date $t_{j,start}$.

$$C_1 = \frac{1}{J} \sum_{j=1}^J (t_{j,end} - t_{j,start}) \tag{11}$$

As a second objective, the continuous work of the M bottleneck machines of station 4 is aimed. This can be expressed by the average cycle time of the machines which is measured from start date $t_{m,start}$ to the end of last busy state $t_{m,last}$. This value has to be minimised, too.

G. Weigert et al.

$$C_2 = \frac{1}{M} \sum_{m=1}^M (t_{m,last} - t_{m,start}) \quad (12)$$

Figure 11: Normalised objective system with correlation function.

Objectives, the job throughput time and the machine cycle time are just weakly correlated, as shown in figure 11. Both of the objectives have the same order of magnitude so that the simple sum $C = C_1 + C_2$ is a feasible substitutive objective.

Unfortunately, it is impossible to regard this system just as a simple Flexible Flow Shop because of a lot of restrictions and special features. Not all of the jobs have the same route. So, some of them branch for final processing after the 8th station to a subsection of the system with 4 additional stations. Difficult setup rules have to be considered as well as batching, splitting and machine scheduling rules. That is why the known mathematical scheduling approaches are not suitable in this case and a simulation model is necessary.

Figure 12: GD convergence curves of 8 clients.

A simulation run needs approximately 6 seconds until all the jobs are completed. An optimisation process of 1 000 iterations would take too long for an application in a manufacturing control system. In addition, if a simulation run is started you cannot predict its speed of convergence so that in the worst case the result does not satisfy and the run must be repeated. For this reason parallel optimisation runs are advantageous. Figure 12 shows a set of convergence curves of a GD algorithm, where the simulation task was distributed to eight parallel clients. It is shown that some clients descend very quickly but remain static later. Other clients

reach better results but descent more slowly at the beginning. From the viewpoint of the server one get the resulting convergence curve as shown in figure 13 which is steeper at the beginning as the single curves and reach the best value of all at the end.

The optimisation process starts with an objective value of 128% of the minimal value C_{min} which was ever found. The worst known value is close to the start value and amounts 135% of the minimal value. Table 2 shows like table 1, for which iteration each of the eight clients has covered more than half of the range from the start value to the possible minimal value of the objective function. The GD and the GY bring the best results how one can see from the average of iterations.

Figure 13: Resulting convergence curve for GD.

The GD starts in a completely flooded ‘landscape’ ($L_0 > C_{max}$), i.e. the whole search space is available at the beginning. The evaporation rate $-\Delta L$ (Equation 9) was adjusted in a way that the minimum was dried out before 100 iterations. These parameter settings, determined by some experiments before, brought the best results.

Table 2: Iterations until 50% level for eight clients.

Client	RW	TA	GD	GY
1	6	19	20	2
2	11	8	11	13
3	8	14	18	4
4	47	10	6	8
5	24	13	8	13
6	42	39	6	11
7	32	17	7	12
8	42	9	18	50
Average	27	16	12	14

5 Conclusion

It is often asserted that Genetic Algorithms are specifically suitable for solving difficult scheduling or other NP-hard problems. A further advantage of them is that individuals of a population can be calculated concurrently. So there is a potential of acceleration if a number of computers are available and distribution of the calculation is possible. It was shown that the heuristic algorithms Threshold Accepting and Great Deluge are parallelisable, too. Surprisingly, they work very efficiently for most of usual scheduling problems. But in

G. Weigert et al.

difference to the Genetic Algorithms they are easier to implement and because of just one respectively two parameters which control these algorithms, they are very easy to use, too.

In addition, the Genetic Algorithms properly work only for a population size of at least 20 or more individuals. According to experience not less than 10 generations are necessary for an acceptable result of optimisation; each individual activates a complete simulation run. This is a substantial overhead in comparison to TA or GD, where the optimum was already found with less simulation runs.

Of course, it is difficult to compare several heuristic algorithms with each other. The strong stochastic influence prevents the reproduction of the results. In addition all these algorithms are very sensitive to changing parameters so that the experience of the optimisation expert is an important factor for the efficiency of the algorithm, too. However it could be shown that satisfactory results are obtainable with the examined algorithms. An explanation for this observation is that practical scheduling problems often have a lot of suboptimal solutions which are possibly homogeneously distributed over the search space, like the surface of a stormy sea. This would explain the relatively good results of the GY.

The approach of distributed simulation increases the performance of optimisation. It is usable in industrial environments because it is easy to install the system. Many available computers which are usually idle most of the time can help to bring scheduling problems to success in shorter time.

In the present approach the clients work independently from each other after the optimisation process was started. The information streams are joined at the server and are used only for the administration of results. For the further development interactions between the clients are planned. Then the server can stop clients with bad solutions ahead of time and start them from a more promising point. So similar to the Genetic Algorithms bad solutions would be eliminated with higher probability and better solutions become more accepted.

References

- Dueck, G. and Scheuer, T., Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, 1990, **90**, pp. 161-175.
- Fachat, A., A comparison of random walks with different types of acceptance probabilities. PhD thesis of Chemnitz University of Technology, 2000.
- Fisher, H. and Thompson, G.L., *Probabilistic learning combinations of local job-shop scheduling rules*, (Prentice Hall, Englewood Cliffs, New Jersey), 1963.
- Feldmann, M. and Biskup, D., Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Journal of Computers & Industrial Engineering*, 2003, **44**, pp. 307–323.
- Gonçalves, J.F., Mendes, J.J.M. and Resende, M.G.C., A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem. AT&T Labs Research Technical Report TD-5EAL6J, 2002.
- Hampel, D., Simulationsgestützte Optimierung von Fertigungsabläufen in der Elektronikproduktion. PhD thesis of Dresden University of Technology, 1997.
- Horn, S., Weigert, G. and Werner, S., Real time scheduling by parallel and distributed simulation over IP Multicast, in *15th International Conference on Flexible Automation & Intelligent Manufacturing*, 2005, pp. 305-312.
- Lee, D.S., Vassiliadis, V.S. and Park, J.M., List-Based Threshold-Accepting Algorithm for Zero-Wait Scheduling of Multiproduct Batch Plants. *Journal of Ind. Eng. Chem. Res.*, 2002, **41**, pp. 6579-6588.
- Lee, D.S., Vassiliadis, V.S. and Park, J.M., A novel thresholdaccepting meta-heuristic for the job-shop scheduling problem. *Journal of Computers & Operations Research*, 2004, **31**, pp.2199–2213.

Optimisation of manufacturing processes

Lee, Y.H., Jeong, C.S. and Moon, C., Advanced planning and scheduling with outsourcing in manufacturing supply chain. *Journal of Computers & Industrial Engineering*, 2002, **43**, pp. 351-374.

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

Pinedo, M., *Scheduling – theory, algorithms and systems*, (Prentice Hall: Englewood Cliffs, New Jersey), 1995.

Pongcharoena, P., Hicksa, C., Braidena, P.M. and Stewardson, D.J., Determining optimum Genetic Algorithm parameters for scheduling the manufacturing and assembly of complex products. *Int. J. Production Economics*, 2002, **78**, pp. 311–322.

Schmidt, H. and Thierauf, G., A combined heuristic optimization technique. *Journal of Advances in Engineering Software*, 2005, **36**(1), pp. 11-19.

Schneider, J.J., Effiziente parallisierbare physikalische Optimierungsverfahren. PhD thesis of University of Regensburg, 1999.

Schneider, J., Britze, J., Ebersbach, A., Morgenstern, I. and Puchte, M., Optimization of production planning problems - a case study for assembly lines. *International Journal of Modern Physics C*, 2000, Vol. 11, No. 5, pp. 949-972.

Schulz, M., Simulation based optimisation of scheduling problems using tabu search. ASIM Mitteilungen aus den Arbeitskreisen. 54(2), pp. 361-365, 1996.

simcron, Homepage of the simcron GmbH. Available online at: www.simcron.de (accessed 25 November 2005).