Evolutionary Real-World Shop Floor Scheduling using Parallelization and Parameter Coevolution

J. Käschel* ‡
Dept. of Economics
Technical University
D-09107 Chemnitz
Germany

Bernd Meier§
Dept. of Economics
Technical University
D-09107 Chemnitz
Germany

Marco Fischer¶
Dept. of Economics
Technical University
D-09107 Chemnitz
Germany

Tobias Teich‖
Dept. of Economics
Technical University
D-09107 Chemnitz
Germany

Abstract

Evolutionary algorithms possess promising potential for solving manufacturing control problems. While this has been shown during the last years real-world applications of these methods are still rare. There are two basic reasons for this. Firstly, a scheduling tool based on Evolutionary algorithms (EA) has to fit into the organizational structure of today's companies i.e. it has to be coupled with e.g. Enterprise Resources Planning-Systems (ERP-Systems) in such a way that the power of evolutionary search is used to a full extend. This topic is only addressed in short in the following. The focus in this article is on the second reason - the size of real-world shop floor scheduling problems. To deal with it the authors propose a continuously operating EA-scheduler which can easily adapt to sudden changes in a production system. Two different approaches to parameter adaptation by coevolution are discussed. To decrease the amount of calculation time needed to find a satisfying solution a simple parallelization approach is introduced. In the last section the results of a test of a real-world application based on these ideas are shown.

1 Introduction to the Problem

The task of shop floor scheduling is to find sequences in which n jobs are to be processed on a set of m resources (machines, staff, special tools etc.) so that several constraints (technological, organisational, etc.) are met and so that a given objective (e.g. mean flow time, mean lateness etc.) is satisfied as good as possible. Each job can consist of one or more operations. The technological sequence of the operations within a job is not necessarily set. Thus real-world shop floor scheduling comprises many different combinatorical optimization problems like Job Shop, Flow Shop, Open Shop, Parallel Machines Problems and many other. While in Operations Research models for these problems and heuristics to solve them had been developed it was still not possible to handle shop floor scheduling as a whole. Even today it is mostly done manually. With evolutionary methods and other meta-heuristics like Simulated Annealing or Threshold Accepting at hand there seemed to be hope for a change of this situation for the first time.

Within the last ten years a number of attempts have been made world-wide to solve hard combinatorical optimization problems in the field of production control with the help of Genetic Algorithms (GA) from the early work of Davis [Davis, 1985] over the interesting approaches of Nakano and Yamada [Nakano and Yamada 1995, 1996] or of Bierwirth and Mattfeld [Bierwirth et al. 1993, 1995] to recent developments e.g. by Mattfeld [Mattfeld, 1999] or van Bael [van Bael, 1999] to name only a few. However, real-world applications in the field of shop floor scheduling are still rare although the results shown in benchmark tests have been far better than those of the best priority rule heuristics. What is the reason for this?

There is of course the problem that there is a difference between theoretical research work and the development of a real-world application. But this is not the main point. More important is that benchmark problems are rather small compared to real-world ones and they do not contain those many, little extra problems and degrees of freedom that increase the complexity

*Email: j.kaeschel@wirtschaft.tu-chemnitz.de
‡phone: ++49-371-531 42 52
§Email: bernd.meier@wirtschaft.tu-chemnitz.de
¶Email: marco.fischer@wirtschaft.tu-chemnitz.de
‖Email: t.teich@wirtschaft.tu-chemnitz.de

of real-world shop floor scheduling by some powers of ten compared e.g. to the job shop benchmark problems that are often used for testing scheduling GAs. Regarding this high complexity the question of the time that is needed to calculate a satisfying solution is of highest interest for a practical use of EA-based scheduling tools. Often there are demands for calculation times no longer than five minutes as a study carried out by the authors in 1998 showed [Teich, 1998]. There is no reason to believe that the solution an Evolutionary Algorithm delivers after five minutes could not be improved further. Therefore it seems like a good idea to let the algorithm run continuously. There are two problems that have to be solved to do this.

Firstly, there has to be an organizational framework which allows companies to make use of a continuously running Evolutionary Algorithm for shop floor scheduling. Since in today's organizations scheduling is usually done once at the beginning of a planning period while afterwards only minor changes are allowed, companies are not prepared to handle new improved schedules delivered by an EA after the first schedule has been fixed and its sequencing information has been distributed to every worker. To change this the authors developed an organizational framework in which these new improved schedules can easily be handled. However, since these are matters which do not lie within the field of Evolutionary Algorithms they are not discussed further in this paper.

Secondly, there is the Evolutionary Algorithm itself. Since situations in production systems may change frequently no set parameter constellation seems appropriate. To adapt to changes parameters themselves should be the matter of a coevolution process. Last not least the computational resources companies can use for shop floor scheduling are often rather limited. Therefore a simple parallelization approach using intelligent terminals of a plant data acquisition system (PDA) has been developed and tested successfully.

2 Evolutionary Search Scheduling Algorithm

The shop floor scheduling EA introduced here consists of two parts - a Genetic Algorithm for the schedule optimization and a second evolutionary algorithm for the parameter coevolution. For schedule optimization a GA developed by the authors in 1998 for job shop problems was used [Käschel et al., 1999a]. It is build upon a genetic representation of permutations with repetition. This approach was developed by *Bierwirth et al.* in 1993 [Bierwirth et al., 1993]. Figure 1 gives an idea how it works.

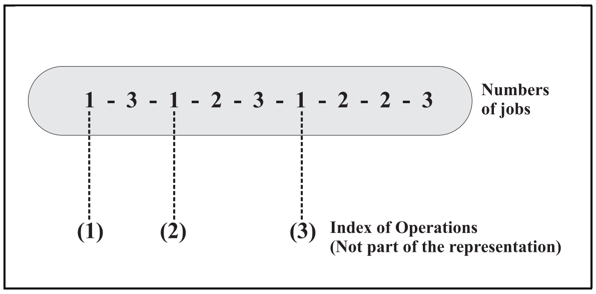

Figure 1: Genetic representation of the schedule optimization GA

Each job is occuring within a chromosome as often as it has operations. Thus it is possible to avoid invalid solutions without a complicated evaluation procedure. To build a semi-active schedule the chromosome has only to be read from the beginning to the end and each operation has to be placed on the necessary resources at the earliest possible starting date at which all resources have to be available and the predecessor of the operation within the same job must be finished. The latter condition could be weakened because sometimes there are operations whose sequence is not necessarily set. For crossover we used the Generalized Order Crossover (GOX) developed by Bierwirth et al. in 1993 and the Generalized Position Based Crossover (GPBX) developed by the authors [Teich, 1998]. Each operator is chosen with a certain probability p_{ox_i} with

$$\sum_i p_{ox_i} = 1.$$

Thus possible disadvantages of the use of just one crossover operator shall be overcome. Tests carried out by the authors [Teich, 1998] seem to justify this approach because the mean results reached by combined use of the two operators were better than the results reached by using single operators. Both GOX and GPBX are generalized versions of crossover operators which were originally developed by Syswerda [Syswerda, 1991] for the Travelling Salesman Problem. Syswerdas attempt to preserve order and absolute position information respectively is unfortunately partly undermined by Bierwirths genetic representation. The problem is that actually order/position information of operations shall be preserved whereas the chromosome contains job numbers to avoid invalid solutions. This dilemma on one hand and the high redundancy of the representation which increases the dimension of the search space by some additional powers of ten on the other hand let it seem worthwhile to use a more powerful scheduling GA in the future. Therefore the authors are currently comparing the concepts of van Bael [van Bael, 1999] and of Mattfeld [Mattfeld, 1999] with regard to their suitability for real-world shop floor scheduling. For mutation the scheduling GA uses a reservoir of no less than six operators which differ in their power to destroy the genetic information. A shift operator which takes only one operation out of the chromosome and moves it to another position and a scramble operator which changes the chromosome are the two opposite ends of a scale these operators can be placed on. For mutation one of them is chosen according to the same principle as the crossover operator. The same concept was applied with selection schemes. The idea behind the implementation of this variety of operators was to produce a robust algorithm which avoided a problem that GAs which use only some operators often have - they do well on one problem but fail to deliver satisfying solutions on another one. With its variety of operators our algorithm proved to be robust but at the cost of an inferior quality of solutions for benchmark tests. A first version of this algorithm was implemented in 1998 as generational replacement GA. Results of this algorithm can be found in [Käschel et al., 1999a]. For easier parallelization it was later changed to a steady state GA. New solutions are accepted if they are better than the worst individual or with a certain probability if they are worse. This concept was borrowed from Simulated Annealing [Davis, 1987, Kirkpatrick et al., 1983]. The probability is lowered during the course of the optimization according to what is called a cooling schedule in Simulated Annealing. The parameters which determine the cooling schedule can themselves be changed by coevolution. The idea of coupling the Simulated Annealing Concept with a steady state GA was born because of the very good results the authors reached with a simple Simulated Annealing procedure on the Fisher-Thompson benchmark problems for the JSP [Muth et al., 1963, Käschel et al., 1999b].

The schedule optimization GA uses a weighted combination of mean lateness and mean flow time as objective function. Job priorities can easily be taken into account thus. For the simulation tests whose results are displayed at the end of this paper both objectives were wheighted equally. Parameters for the schedule optimization are represented as integer values within the chromosome sketched in figure 2. The schedule optimization GA is run until there have been no improvements within the last 400 iterations. Then its parameters are mutated with a normal distributed mutation operator as part of a simple (1+1)-Evolution Strategy (ES) and the scheduling GA is run again with the new set of parameters until there have been no improvements throughout the last 400 iterations. Then the former parent and offspring are compared and the better is chosen for a new mutation.

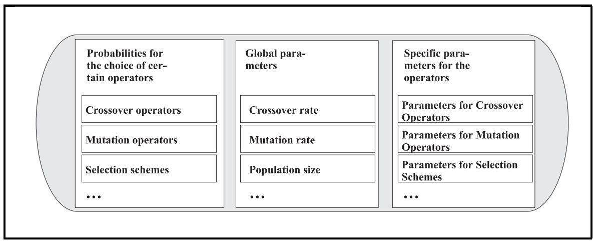

Figure 2: Parameter representation

As another way of parameter coevolution the authors tried to tie one parameter chromosome to each individual of the schedule optimization GA. The parameter chromosomes use a crossover operator that calculates the mean value of the two parent chromosomes. The mutation operator is the same as in the Evolution Strategy described above.

3 Parallelization

Today's companies usually have only very limited computational resources available for shop floor scheduling. Therefore the only practicable solution for a parallelization of our EA seemed to lie in the PDA-terminal computers. The only tasks these computers have to perform are to distribute work sequence information to the workers and to allow them to report the status of the job they are currently working on (e.g.: job in progress, job finished etc.). Naturally they are idle most of the time. Since in our case (and probably in many cases) these computers are only comparably slow (133MHz RISC-Processor, 32 MB memory) the best way to use them for parallelization seemed to be a single population master-slave approach [E. Cantu-Paz, 1998]. The idea is sketched in figure 3.

Each time a new individual has been created by crossover or mutation on the computer where the GA-scheduler is running (master) it is sent to one of the PDA-terminals (slave) which is currently idle. All the information needed for scheduling (capacity data, technological constraints etc.) is held in a shared data base. The terminals as well as the GA-scheduler load up this information when they are switched on and update it each time an operation is reported finished, a new job is released or some disturbance occurs. The slave computer is carrying out the time consuming

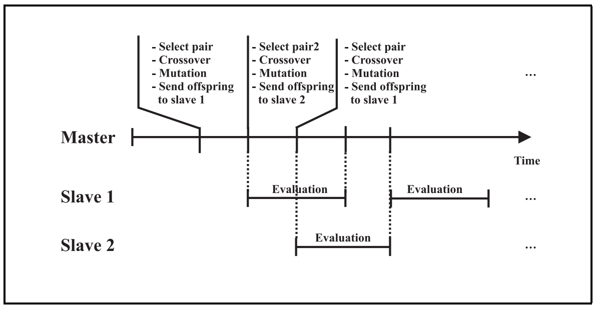

Figure 3: Asynchronuous Master-Slave Parallelization

evaluation procedure and sends the values of the objective and the fitness function of the schedule back to the master computer. To avoid that the GA gets stuck the evaluation procedure is carried out on the master computer if all slaves are busy. Since we use a steady state GA it is possbile to run the process asynchronuously so that it is not necessary to put individual A and B back into the population in the same sequence in which they have been sent away for evaluation.

4 Test results

The shop floor scheduling system was tested on real-world data of a German engineering company. For this purpose the release date, the finishing date and the due date of each job that passed the plant within a period of two months was recorded as well as the date and the duration of all disturbances of the manufacturing process (machine breakdowns, rejects because of material defects, illness of workers etc.). The plant is organized according to the workshop principle that is to say machines are grouped according to the operations that have to be carried out on them rather than according to the products that are manufactured on them. With this data it was possible to build a complete simulation model of the plant as shown in figure 4 on which the scheduler could be tested.

For comparisons we used the mean flow time and the mean lateness of jobs. The EA-scheduler was tested in four different real-time simulations with the Evolution Strategy for parameter coevolution (ESCO) that has been described above and with continuous coevolution within the GA (GACO). These two versions were additionally tested with parallelization (ESCOPA) and (GACOPA). For comparison we used the results reached by manual scheduling which could be obtained from the data recorded (MANUAL), and simulations with a priority rule-based scheduler (PRIORULE) and a GA-scheduler (NONCONGA) which

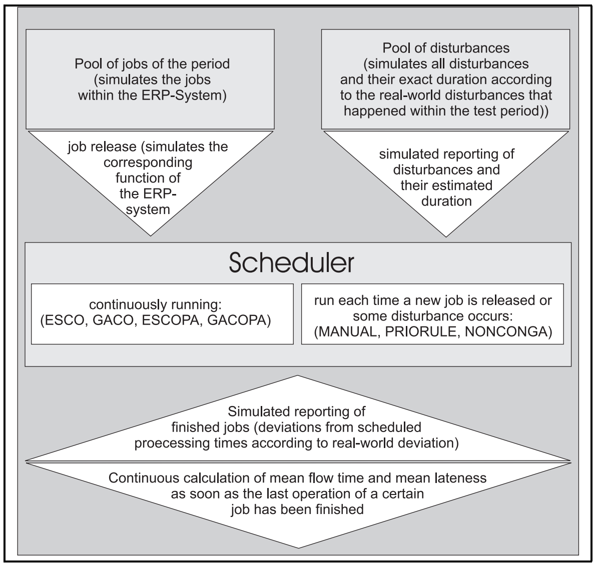

Figure 4: Simulation environment

were both run for five minutes each time rescheduling was necessary because of changes (new jobs, machine breakdowns etc.). The priority rule scheduler was allowed to calculate a number of single, standard priority rules (e.g.: Shortest Processing Time, Earliest Due Date and others) and combinations of these rules. The best result was used. For the NONCONGA the schedule optimization GA was used without parameter coevolution. The test results are displayed in table 1. With manual scheduling in practice a mean lateness of 2.4 days and a mean flow time 13.2 days was reached. These values were set to 100 % to ease the comparison.

Table 1: Relative results (manual scheduling = 100%)

scheduler	*mean lateness*	*mean flow time*
MANUAL	100.0 %	100.0 %
PRIORULE	95.3 %	97.9 %
NONCONGA	83.4 %	93.1 %
ESCO	64.2 %	88.2 %
GACO	55.8 %	85.1 %
ESCOPA	63.2 %	87.8 %
GACOPA	55.3 %	85.0 %

The results show clearly a superior quality of the solutions of the continuously running EA schedulers to the other approaches. Parameter coevolution within the GA does a little better than the ES approach. This we think is because of the larger amount of parameter sets that can be tried if every individual of the schedule optimization GA has its own parameter chromosome.

If the evolution strategy is used a change of parameters takes place only every 5-15 minutes as we found out during the simulation. It is remarkable though that this simple evolution strategy still does comparably well. The influence of parallelization is only small. This could be due to the relatively small number of 5 PDA-terminal computers that we were able to use. A more massive parallelization would possibly do better. There is still quite a number of jobs late. However, we see the main reason for this not within the scheduler but in the due dates the companie's sales officials negotiated for the jobs long before the manufacturing process started.

5 Conclusion

As a result it can be stated that EA-based scheduling tools for real-world shop floor scheduling allow companies to mobilize a lot of reserves in their production systems at comparably low cost. The author's future research will be directed towards a more powerful scheduling GA as has already been mentioned above and towards a useful determination of due dates which means to go beyond shop floor scheduling towards production planning.

6 Acknowledgements

This work has been supported by the German Research Community (DFG) as part of the special research area SFB 457.

References

C. Bierwirth, H. Kopfer, D. Mattfeld, T. Utecht (1993). *Genetische Algorithmen und das Problem der Maschinenbelegung*, Universität Bremen, Lehrstuhl für Logistik, Bremen 1993.

C. Bierwirth (1995). *A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms*, OR Spektrum vol.17, pp. 87-92, 1995.

E. Cantu-Paz (1998). *Designing efficient master-slave parallel genetic algorithms*, in: J. R. Koza et al. (editors): Genetic Programming 1998: Proceedings of the Third Annual Conference, pp. 455, San Francisco, CA, Morgan Kaufman, 1998.

L. Davis (1985). *Job shop scheduling with Genetic Algorithms*, in J. Grefenstette, editor, Proceedings of an International Conference on Genetic Algorithms and their Applications, pp. 136-140, Hillsdale, 1985, Lawrence Erlbaum Associates, 1985.

L. Davis (1987). *Genetic Algorithms and Simulated Annealing*, Morgan Kaufman, Los Altos, 1987.

J. Käschel, G. Köbernik, B. Meier, T. Teich (1999a). *Genetic Algorithm, Avoiding of Deadlocks and Gantt-Chart-Generation for the Job Shop Scheduling Problem*, in: W. Banzhaff et al. (editors), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), p. 792, Morgan Kaufman 1999.

J. Käschel, G. Köbernik, B. Meier, T. Teich (1999b). *Algorithms for the Job Shop Scheduling Problem - a comparison of different methods*, in: Proceedings of the European Symposium on Intelligent Techniques (ESIT'99), p. 74, Technical University of Crete, 1999.

S. Kirkpatrick, C. Gelatt, M. Vecci (1983). *Optimization by Simulated Annealing*, Science Vol.220, pp. 671-680, 1983.

D. Mattfeld (1999). *Scalable Search Spaces for Sche duling Problems*, in: W. Banzhaff et al. (editors), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), pp. 1616-1621, Morgan Kaufman 1999.

J. F. Muth, G. L. Thompson (1963). *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, New Jersey, 1963.

G. Syswerda (1991). *Schedule Optimization using Genetic Algorithms*, in L. Davis, editor, Handbook of Genetic Algorithms, pp. 332-349, Van Nostrand Reinhold, New York, 1991.

T. Teich (1998). *Optimierung von Maschinenbelegungsplänen unter Benutzung heuristischer Verfahren*, Verlag Josef Eul, Lohmar, Köln, 1998.

P. van Bael et al.(1999). *The Job Shop Problem solved with simple, basic evolutionary search elements*, in: W. Banzhaff et al. (editors), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), pp. 665-669, Morgan Kaufman 1999.

T. Yamada, R. Nakano (1995). *A Genetic Algorithm with Multi-Step Crossover for Job Shop Scheduling Problems*, in Proceedings of an International Conference on GAs in Engineering Systems: Innovations and Applications (GALESIA 1995), pp. 146-151, 1995.

T. Yamada, R. Nakano (1996). *Scheduling by Genetic Local Search with Multi-Step Crossover*, in Proceedings the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV), pp. 960-969, Berlin 1996.