EFFICIENT APPLICATION OF OPTIMIZATION METHODS BY USING CONCURRENT AND SIMULTANEOUS OPTIMIZATION

Wünsch, Andreas; Vajna, Sandor
Otto-von-Guericke-University Magdeburg, Germany

Abstract

The present paper addresses the research question of the efficient execution of optimization tasks in product development by using parallelization methods and by considering the existing resources within an organization. Based on analogies between optimization processes and the execution of tasks in interdisciplinary teams, a method for efficient parallelization of evaluation processes of optimizations is introduced and implemented prototypically into a framework for distributed optimization. Thereby the parallelization of evaluation processes is based on the decomposition of both the optimization model and the evaluation model. Since the processing of these process elements depends only on the availability and the stability of required information and resources, high flexibility in the application and dynamic response to spontaneous changes in the resources can be ensured. Within this environment resources include available hardware, installed software, and available licenses. Furthermore, we show how the efficiency of an optimization can be further increased by using priority-based processing of the process elements of the evaluation process.

Keywords: Concurrent Engineering (CE), Design informatics, Integrated product development, Optimisation, Simulation

Contact:
Dr.-Ing. Andreas Wünsch
Otto-von-Guericke-University Magdeburg
Chair of Information Technologies in Mechanical Engineering
Germany
andreas.wuensch@ovgu.de

Please cite this paper as:
Surnames, Initials: *Title of paper*. In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 2: Design Processes | Design Organisation and Management, Vancouver, Canada, 21.-25.08.2017.

1 INTRODUCTION

Analysis and simulation enable engineers to evaluate new products without the need for physical prototyping. The area of computer aided engineering has established itself as an important part of the product development process. Based on virtual prototypes the relevant properties of a product can be determined and optimized during early phases of the product development process. Especially when a large number of requirements have to be considered and the requirements contradict each other multiobjective and multidisciplinary optimization methods can be used to support the development process and to reduce the number of physical prototypes which leads to cost and time savings.
However, these methods often require a lot of computational resources. Thereby the optimization process has to be constantly efficient. Bottlenecks should be avoided and available resources should not become idle unless the optimization is not finished.
The present paper addresses the research question of an efficient execution of optimization tasks in product development by using parallelization methods and by considering the existing resources within an organization.
The structure of this paper is as follows. Firstly, we present concurrent and simultaneous optimisation as a parallelization method to solve the research question. Subsequently, we show how the efficiency of an optimization can be further increased by using priority-based processing of the process elements of the evaluation process. Furthermore, we present a framework for distributed optimization, which was developed for the prototypical implementation of the parallelization method, and its validation based on two case studies. This paper ends with some concluding remarks.

2 CONCURRENT AND SIMULTANEOUS OPTIMIZATION

Using parallelization is a common used method to increase the efficiency of solving a problem. Thereby the problem is split into independent subproblems that are solved in parallel. The ability to solve a certain problem in parallel depends strongly on the problem's structure. If the problem consists of mutually independent partial problems, the problem can be decomposed into its subproblems and processed independently. This is called inherent parallelism (Bengel et al., 2015).
Considering product optimization problems the involved evaluation processes have to be inherent parallel at least in a certain domain, which is called *parallelization domain*. Due to the inherent parallelism in this domain the evaluations can be processed in parallel. There are two kinds of model decomposition that yield to inherent parallel evaluation processes (Schumacher, 2013):

- **Decomposition of the optimization model:** The optimization model is decomposed into independent evaluations of independent variants of the product. Usually stochastic optimization methods, e.g. Evolutionary and Genetic Algorithms, Particle Swarm, and Simulated Annealing use independent variants of the optimized product. Considering Genetic Algorithms the population represents the parallelization domain, where the included individuals are mutually independent. Design of Experiment studies, which are used to create meta models that represent the product behaviour, also contain independent samples of the product. These samples can be evaluated in parallel, too (Cavazzuti, 2013).
  Since only the optimization model is decomposed the evaluation model is not affected by this decomposition and does not require any adaption.
- **Decomposition of the evaluation model:** The evaluation model is decomposed and processed in parallel. The optimization itself can be processed sequentially. The decomposition of the evaluation model can be done at different sites, e.g. to parallelize different load cases of a complex multidisciplinary simulation using the All-at-Once approach (Martins and Lambe, 2013) or to parallelize large finite element analyses such as crash simulation using domain decomposition.
  Since only the evaluation model is decomposed, the optimization model is not affected by this decomposition. Due to this fact any optimization method can be used even deterministic optimization methods which use dependent variants of the product during the optimization, e.g. search direction methods.

These decomposition methods represent the state of the art of decomposition and parallelization of optimization methods. Both decomposition methods are also the underlying principles of several

multidisciplinary optimization architectures, which were collected and summarized by Martins and Lambe (2013).
Considering these decomposition methods the optimization can either be processed fully parallel while the optimization model is decomposed or it is processed sequentially with load case parallelization while only the evaluation model is decomposed. The major advantage of both methods is the relatively low administrative effort. Thereby the most efficient way is to use the same number of computing resources as the number of evaluation processes, which leads to high computational costs in hardware, software, and available licenses since the number of number of evaluation process becomes high. The risk of both methods is the possibility of bottlenecks in the evaluation process, especially when it consists of many different load cases or simulations. The whole process has to wait for the slowest simulation and some resources become idle (Wünsch et al., 2015).
However, in real world applications resources play a major role. This includes hardware, software and licenses. In our opinion a flexible resource management system has to be considered in any parallelization method while continuously avoiding idleness of available resources. Due to this fact, we developed a dynamic parallelization method to ensure the efficient processing of an optimization while dynamically reacting on changing resources. Thereby the efficiency and the parallelization of optimization methods are characterized by three terms (Wünsch, 2017):

- **Effectiveness:** The effectiveness of an optimization describes the quality of the achieved Solution or the determined optimum. High efficiency is achieved when the global optimum of an optimization problem is determined or if there is a high probability of finding the global optimum.
- **Resources:** The resources include available hardware, software, and licenses. The smallest unit of a considered hardware resource is called slot. This can represent a single CPU or a whole workstation. Since in the studies of this paper a workstation cluster is used, in the subsequent remarks of this paper a slot represents a workstation.
- **Processing time:** The processing time signifies the overall time for processing and optimization.

These terms characterize the **efficiency** of an optimization. It indicates how quickly the solution of the optimization problem is achieved. Thereby the number of required evaluations and the possible degree of parallelization with respect to the available resources are taken into account.
Similar to the triangle of stakeholder expectations in project management, which counteract the dependencies of project objectives (Kuster et al., 2011); these terms represent the objectives of a product optimization. If there is no fundamental change in the optimization strategy, e.g. by changing the optimization method, none of these terms can be changed without affecting one of the other terms (see Figure 1).

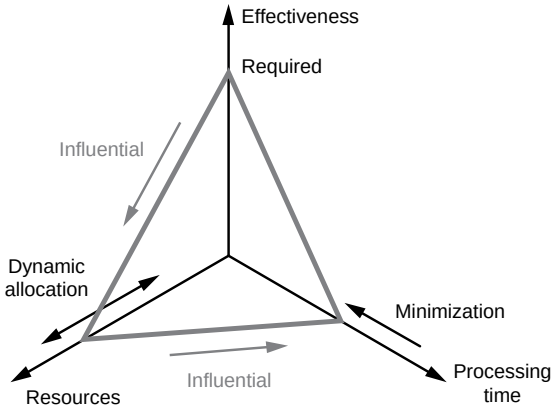

Figure 1. Relation of effectiveness, resources and processing time of an optimization using dynamic parallelization (Wünsch, 2017)

If, for example, the effectiveness of an optimization should be increased because the global optimum was not found, a larger number of evaluations is usually required, which increases the processing time of the optimization using an equal number of resources. If more resources are spontaneously available during the optimization, it is possible to increase the degree of parallelization and thus to reduce processing time. The major goal in determining the terms should always be based on the effectiveness of the optimization, which is required by the user and results in the parameters of the optimization, e.g. the population size of a GA. By dynamically allocating the evaluation processes, the maximum available

resources should always be used with the aim of minimizing processing time and thus increasing the efficiency of the optimization.
To realize this approach, we developed Concurrent Optimization (CO) and Simultaneous Optimization (SO), which represents a very flexible and efficient method to parallel optimization processes (see Figure 2). Since both optimization processes and product development processes can be seen as continuous iterative improvement according to the TOTE scheme (Miller et al., 1960; Ehrlenspiel, 1995), this method is based on analogies between processing evaluation processes and working on tasks in interdisciplinary product development teams using Concurrent Engineering (CE) and Simultaneous Engineering (SE). Thereby both the optimization model and the evaluation model are decomposed regarding to the required resources. Similar to CE and SE, CO and SO are only based on the availability and stability of the required information and resources. Thus, a great flexibility in parallel processing of the process elements and dynamically reaction to bottlenecks in the resources can be achieved.

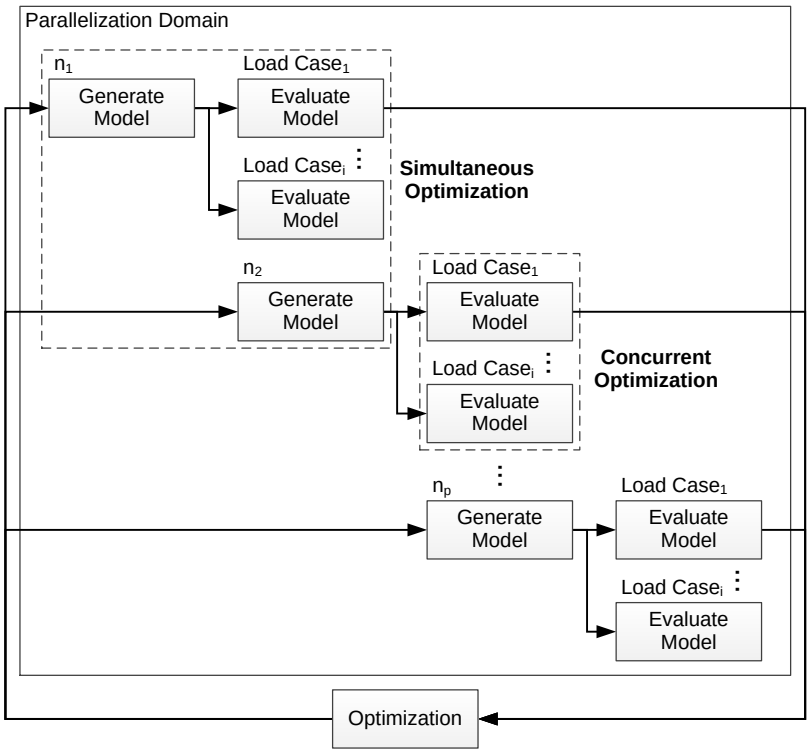

Figure 2. Concurrent and Simultaneous Optimization (Wünsch, 2017)

In this method the slots of the distributed computing environment behave similarly to members of an interdisciplinary development team who have different qualifications and thus can execute different tasks. If a member of the team spontaneously drops out or a new one joins the team, the tasks are redistributed by means of dynamic allocation, with the aim of processing the overall task as efficiently as possible and minimizing the overall processing time. The decomposition of the evaluation model should be done according to the required resources, e.g. according to the required software or hardware.

3 PRIORITY-BASED PROCESSING OF EVALUATION PROCESSES

Using the presented method of CO and SO to parallel optimization processes the efficiency of the optimization decreases if resources become idle while other resources are busy. This happens if the queue is non-homogeneous. The queue contains the process elements of the decomposed evaluations process. To avoid idleness of available resources priority-based processing of these process elements can be used while ensuring the advantages of dynamic allocation (by reacting dynamically to changing resources).
Finding the optimal priority values of the single process elements represents a scheduling problem which can be solved by different scheduling methods (Topcuoglu et al., 2002; Dong and Akl, 2006):

- **Complete enumeration:** All possible combinations of the problem are evaluated. The optimal solution of the problem is found by comparing all solutions. Due to the high computational effort, complete enumeration is only suitable in practical use for small problems.
- **Heuristics:** Heuristics describe reliable solutions based on the underlying complex structure of a problem. In contrast to optimization methods, the approximation solution is not determined iteratively but directly. Commonly used scheduling heuristics are *First In - First Out* (FIFO) and *Longest Path Following* (LPF). A summary of common heuristics can be found in Fündeling (2006) and Thonemann and Albers (2010).
- **Optimization methods:** The problem is solved by an iterative optimization process. Since scheduling problems have a combinatorial problem structure and are NP-complete different optimization algorithms exist to solve different kinds of problems. Algorithms can be highly specialised to particular scheduling problems or work more in a general way to solve various kinds of scheduling problems. Thereby stochastic optimization can be used to solve different kinds of problem while providing a high probability to find an optimal solution in an appropriate amount of time, e.g. Genetic Algorithms, Particle Swarm or Simulated Annealing.

To investigate these methods regarding to the determination of the optimal priority values of the single process elements we used different test scenarios which represent different evaluation processes. Two of these test scenarios are shown in Figure 3. The structure of the evaluation process and the dependencies of the single process elements are described as a directed acyclic graph (DAG). In this All-at-Once approach every process element represents a multidisciplinary load case that results in a state variable or a process that provides variables or model data to following processes. The last process element (Z) represents the calculation of the objective function. This element is not considered for the scheduling because this calculation is very fast and will be processed locally. In the DAG definition of the evaluation process cycles are not considered.

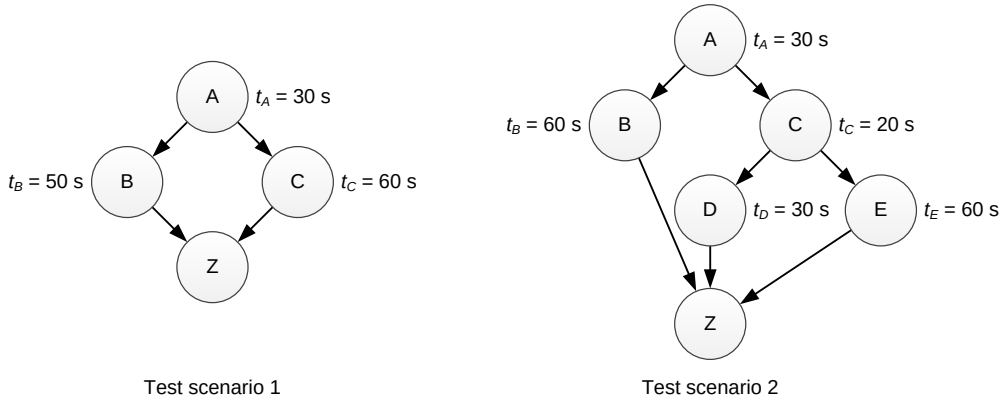

Figure 3. Test scenarios for determining the method for generating the optimal priority values of the process elements (Wünsch, 2017)

For investigating the scheduling methods we developed code that simulates the allocation process. This allows determining the entire solution space by complete enumeration to assess the quality and suitability of a method. The solution space contains all possible solutions. Furthermore, we used the FIFO heuristic, which yield to priority values of 0,0,0 in test scenario 1 and the LPF heuristic, which yield to values of 0,1,2. The priority values are in alphabetical order of the DAG in Figure 3. Additionally an optimization method was used to determine the optimal priority values. Is this study 4 slots ware used (ns = 4).

The results in Figure 4 show that heuristics are not suitable to determine the optimal priority values while varying the size of the parallelization domain. Only the optimized priority values result constantly in the shortest processing time because the optimisation considers the conditions of the problem individually instead of generally, e.g. the size of the parallelization domain.

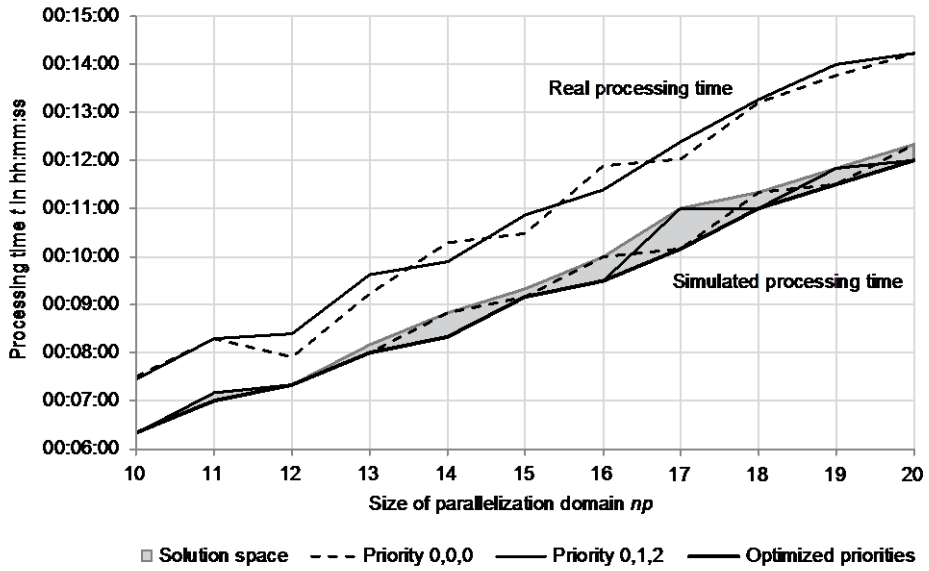

Figure 4. Comparison of real and simulated processing time for test scenario 1 (ns = 4)

The investigation of test scenario 2 yields to the same results (see Figure 5). The shortest processing time can only achieved constantly by using optimization methods to determine the optimal priority values.

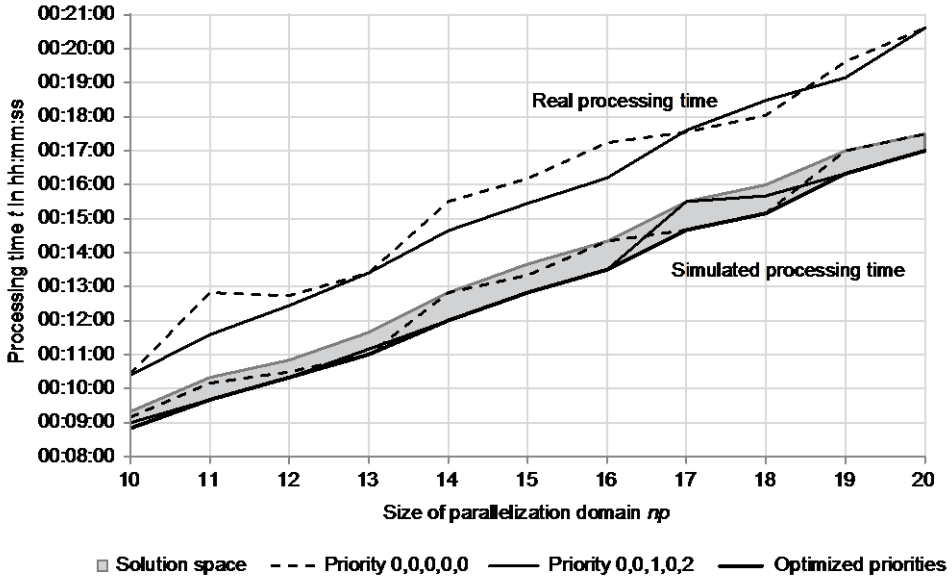

Figure 5. Comparison of real and simulated processing time for test scenario 2 (ns = 4)

For the investigation of a suitable optimization algorithm we selected the most adverse condition of the scheduling problem of test scenario 2. The size of the parallelization domain is set to *np* = 4; 15. With these values, only 4.7 % of the possible combinations of the priority values lead to the minimum processing time, which is the smallest percentage of the optimum priority values. We compared the quality of the solution and the required number of iterations of the following stochastic algorithms of the *inpspyred framework* (Inspired Intelligence Initiative, 2016):

- Nondominated Sorting Genetic Algorithm (NSGA-II).
- Simple Genetic Algorithm (GA).
- Genetic Algorithm that is fitted to the Traveling Salesman Problem (GA-TSP).
- Simulated Annealing (SA).

To consider stochastic dispersion we performed 5 optimization runs with each algorithm. The results show, that the NSGA-II algorithm, which represents a state of the art GA, yields constantly to the optimal priority values and thus to the shortest processing time (see Figure 6).

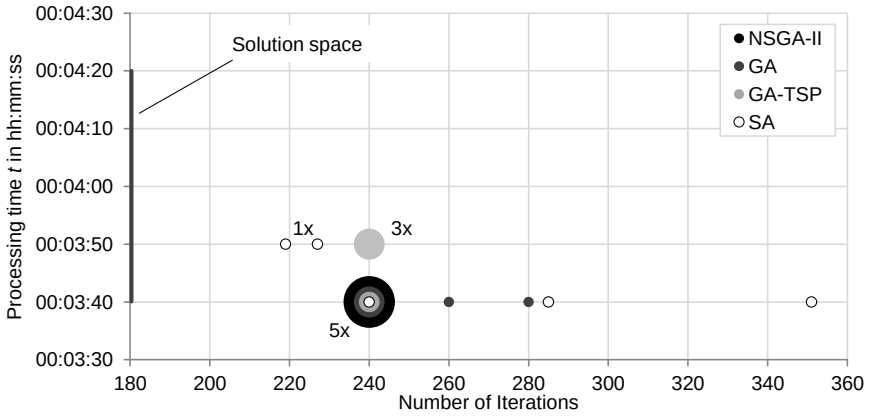

Figure 6. Comparison of different optimization algorithms for optimizing the process priority values of test scenario 2 (np = 4, ns = 4) (Wünsch, 2017)

The results were validated by a second study using a parallelisation domain of np = 15. In this study the NSGA-II algorithm also shows the smallest dispersion and yields constantly to the optimal priority values. Due to this fact this algorithm was selected for the prototypical implementation.

4 PROTOTYPICAL IMPLEMENTATION

For the prototypical implementation a framework was developed which realizes the described method of CO and SO for the efficient and dynamic parallelization of evaluation processes of optimizations in product development. The framework includes various functions and procedures for the efficient execution of optimizations and DoE studies in a distributed computing environment. The framework consists of 3 tiers: the front-end, the modules, and the resources. These tiers and the integrated systems including their specific tasks and interactions are shown in Figure 7.

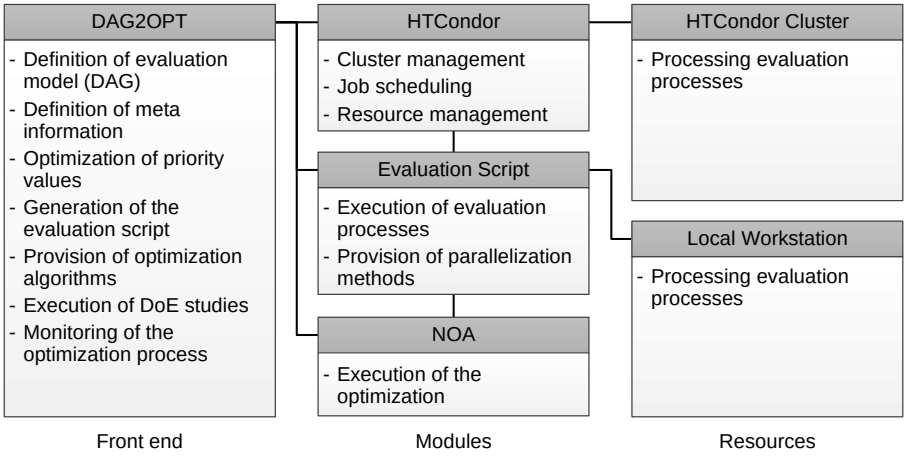

Figure 7. Tasks and interactions of the systems integrated into the framework (Wünsch, 2017)

The front end of the framework is represented by the optimization system DAG2OPT. Except the creation of the scripts for the automation of the process elements, all activities for preparing and executing an optimization can be done by the user via a graphical user interface, which enables a very easy setup of a distributed optimization.

The second tier contains the modules. The modules include the cluster management system HTCondor (Center for High Throughput Computing, 2015), which controls the resource management and the job scheduling. Furthermore the modules include the evaluation script and the optimization system NOA

(Jordan and Clement, 2004). Based on meta information of the process elements, which are set up by the user, the evaluation script contains the complete logic for the control of the evaluation processes and for the dynamic parallelization of the process elements. Thus it can also be integrated into external superordinate processes after its creation. NOA provides several EAs and GAs. It also starts the evaluation script, interprets the result of the evaluations, the objective function value, and determines the fitness value. NOA is controlled entirely by DAG2OPT. DAG2OPT also realizes the generation of the evaluation script and the communication to the modules.
A workstation cluster is used to provide the computing resources, since this kind of cluster represents a very cost efficient distributed computing environment. The advantages of workstation clusters are described by Wünsch et al. (2015). Based on their meta information the process elements are either transmitted to the HTCondor cluster or are executed on the local master workstation that also runs DAG2OPT and the modules. This enables a very easy integration of software tools that require their execution in GUI mode in an automated and distributed evaluation process. It also enables the integration and efficient execution of process elements that have a very short processing time below the overhead time (10 s). The overhead contains the extra amount of time for transferring data within the cluster and synchronizing results. Usually in common optimization frameworks the distributed processing of these kinds of elements is not efficient since the communication takes more time than the execution.

5 CASE STUDIES

The developed framework was validated in two case studies which represent multidisciplinary optimization tasks using the All-at-Once approach and address different properties of the framework. For this a heterogeneous workstation cluster in a university computer lab was used. The cluster consists of the following workstations:

- 24 x Dell T1600 (Windows 7 64-bit, CPU: Intel Xeon E3-1290 3.60 GHz, 16 GB RAM).
- 4 x Dell T5810 (Windows 7 64-bit, CPU: Intel Xeon E5-1620 v3 3.50 GHz, 8 GB RAM).

Similar clusters may be built up at most institutes and companies, where a large number of free standing workstations is available at very little cost.

5.1 DoE Study of a Crank Arm

The focus of this case study is the flexible use of the optimization system and the dynamic allocation of the available resources. The case study represents a DoE study of the crank arm. Thereby fully parallel processing and simultaneous optimization are compared while the number of slots (workstations) is varied ($1 \leq ns \leq 16$). Furthermore the processing time is compared to the ideal parallelization which is the theoretical optimum. The evaluation process consists of 3 process elements (see Figure 8).

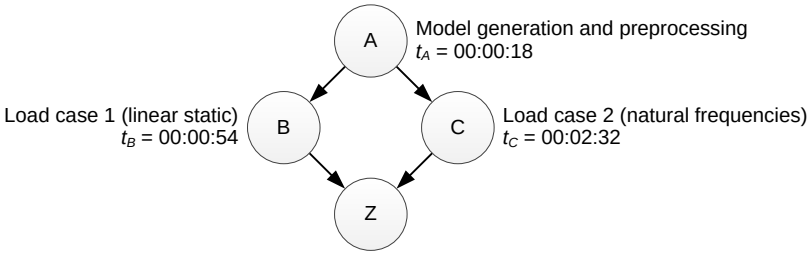

Figure 8. DAG representation of the evaluation process (Wünsch, 2017)

The results show that up to a slot number of $ns = 7$ simultaneous optimization leads to shorter processing times (see Figure 9). This can be explained on the one hand with the additional resource of the master workstation, which processes element A. On the other hand, the overhead is reduced since there is no allocation of process element A in the cluster.
When the slot number becomes greater or equal $ns = 8$, the fully parallel processing leads to shorter processing times because each process element of the parallelization domain is assigned its own resource and thus all processing elements are processed in parallel ($ns = np$). By a further increase in the number of slots, only a small reduction of the processing time can be achieved. The ideal processing times of the two parallelization methods show similar behaviour.

Comparing the real processing times with the ideal processing times, which were calculated analytically and represent the ideal theoretical parallelization, it can be seen that the curves are almost parallel. This indicates that the framework is able to react dynamically on a changing number of resources in an ideal way. The distance between the ideal and real curves is almost constant and results from the overhead. It can be seen that this overhead is almost constant and independent of the slot number. Thus the provided cluster is fully scalable.

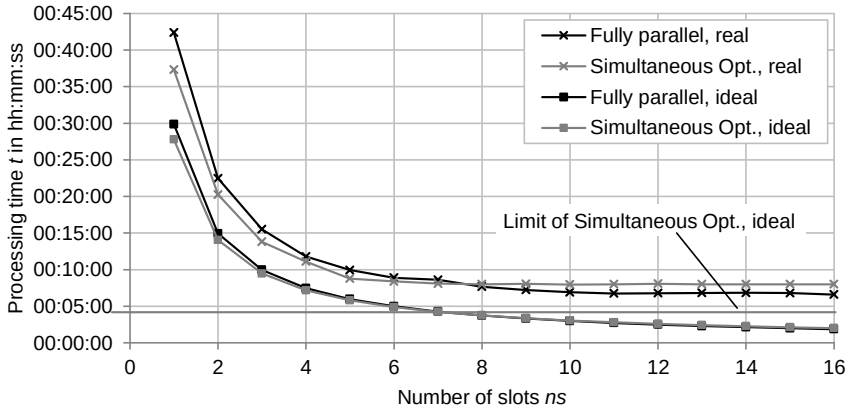

Figure 9. Real and ideal processing time using different parallelization methods (Wünsch, 2017)

5.2 Optimization of a Lever Mechanism

The second case study represents a multidisciplinary optimization of a lever mechanism and focuses on the robustness of the framework. Since a university computer lab was used for this study the workstations are frequently used for interactive work by students. The results show that the optimisation runs very stably and robustly. Thereby the framework can autonomously compensate a fluctuating number of resources (see Figure 10).

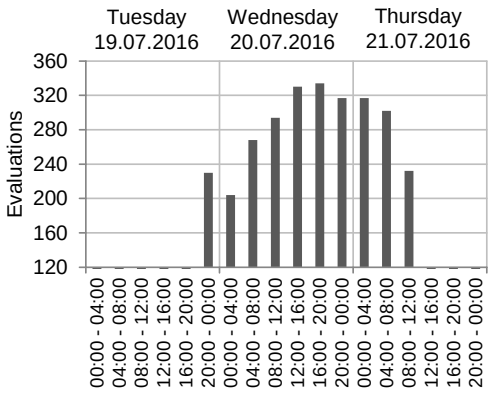

Figure 10. Evaluations of an optimization run separated by time ranges (Wünsch, 2017)

The optimization run contains 2828 evaluations in total. The overall processing time of this optimization was 38 h 35 min (≙ 1.61 d). Compared to the sequential processing of these evaluations which lead to overall processing time of 671 h 39 min (≙ 27.99 d) an overall speedup of Sp = 17.4 and an overall efficiency of Ep = 0.7 could be achieved. Considering only the time range 16:00 – 20:00 on 20.07.2016 where the most evaluations where processed (334), a speedup of Sp = 20.9 with an efficiency of Ep = 0.84 could be obtained.

6 CONCLUSION

The present paper focuses the efficient execution of optimization processes in product development by using dynamic parallelization considering available resources within an organisation. Based on analogies between optimization processes and the execution of tasks in interdisciplinary teams, a method

for efficient parallelization of evaluation processes of optimizations was presented using the terms of concurrent and simultaneous optimization. The method is based on the decomposition of both the optimization model and the evaluation model.
The efficiency of an optimization can be further increased by priority-based processing of the process elements, since unnecessary idling of existing resources is thus avoided. Thereby, the optimal priority values can by generated best by using the genetic algorithm NSGA-II.
The parallelization method was implemented prototypically into a framework for distributed optimization which was validated in two case studies, which represent different typical multidisciplinary optimization tasks using an All-at-Once approach. The case studies reflect an industrial environment where free workstations can be available as resources in a cluster during breaks, at night, or at weekends. The framework can be used in both homogeneous and heterogeneous workstation clusters, which can be implemented very cost-efficiently from an arbitrary number of workstations connected by a network. Additional hardware is not required. Since data is transferred directly between these workstations, there is no need for a shared memory environment. Thereby overhead is almost constant and independent of the number of workstations in the cluster. Thus the provided cluster is fully scalable while ensuring its efficiency. Workstations of the cluster are available for interactive work at any time. Interactive work is not affected by a running optimization.

REFERENCES

Bengel, G., Baun, C., Kunze, M. and Stucky, K.-U. (2015), *Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicoreprozessoren, Multiprozessoren, Cluster und Grid*, 2nd ed., Springer, Wiesbaden. ISBN 978–3–83481–671–9

Cavazzuti, M. (2013), *Optimization Methods: From Theory to Design: Scientific and Technological Aspects in Mechanics*, Springer, Berlin, New York. ISBN 978–3–64231–186–4

Center for High Throughput Computing (2015), *HTCondor Version 8.4.3 Manual*,University of Wisconsin-Madison.

Dong, F. and Akl, S. G. (2006), *Scheduling Algorithms for Grid Computing: State of the Art and Open Problems*, Technical Report No. 2006-504, Queen's University, Kingston, Ontario.

Ehrlenspiel, K. (1995), *Integrierte Produktentwicklung: Methoden für Prozeßorganisation, Produkterstellung und Konstruktion*, München, Hanser, Wien. ISBN 3–446–15706–9

Fündeling, C.-U. (2006), *Ressourcenbeschränkte Pojektplanung bei vorgegebenen Arbeitsvolumina*, Dissertation, University Karlsruhe.

Inspired Intelligence Initiative (2016), *Inspyred: Bio-inspired Algorithms in Python* [online], Available at: http://pythonhosted.org/inspyred/ (18.04.2016).

Jordan, A. and Clement, S. (2004), *Handbuch für das Konstruktionssystem NOA*, Lehrstuhl für Maschinenbauinformatik, Otto-von-Guericke-Universität Magdeburg.

Kuster, J., Huber, E., Lippmann, R., Schmid, A., Schneider, E., Witschi, U. and Wüst, R. (2011), *Handbuch Projektmanagement*, Springer, Berlin, Heidelberg. ISBN 978–3–642–21242–0

Martins, J. R. and Lambe, A. B. (2013), "Multidisciplinary Design Optimization: A Survey of Architectures", *AIAA Journal*, Vol. 51, No. 9, pp. 2049-2075. DOI 10.2514/1.J051895

Miller, G. A., Galanter, E. and Pribram, K. H. (1960), *Plans and the Structure of Behavior*, Holt, New York. ISBN 0–03010–075–5

Schumacher, A. (2013), *Optimierung mechanischer Strukturen: Grundlagen und industrielle Anwendungen*, 2nd ed., Springer, Berlin, Heidelberg. ISBN 978–3–64234–699–6

Thonemann, U. and Albers, M. (2010), *Operations-Management: Konzepte, Methoden und Anwendungen*, 2nd ed.,Pearson Studium, München. ISBN 978–3–8632–6559–5

Topcuoglu, H., Hariri, S. and Min-You Wu (2002), "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13 No. 3, pp. 260–274. DOI 10.1109/71.993206

Wünsch, A., Jordan, A. and Vajna, S. (2015), "Simultaneous Optimisation: Strategies for Using Parallelization Efficiently", *Proceedings of the 20th International Conference on Engineering Design (ICED 15), Vol. 6: Design Methods and Tools-Part 2*. Milan, Italy, 27.–30.07.2015, pp. 133–142. ISBN 978–1–90467–069–8

Wünsch, A. and Vajna, S. (2015), "Effiziente Parallelisierung bei Optimierungsproblemen in der Produktentwicklung", *12. Magdeburger Maschinenbautage*, Magdeburg, Germany, 30.09.–01.10.2015. ISBN 978–3–94472–226–9

Wünsch, A. (2017), *Effizienter Einsatz von Optimierungsmethoden in der Produktentwicklung durch dynamische Parallelisierung*, Dissertation, Otto von Guericke University Magdeburg. ISBN 978–3–941016–10–1