Discovering Performance Bounds for Grid Scheduling by using Evolutionary Multiobjective Optimization

Christian Grimme Joachim Lepping Alexander Papaspyrou

Robotics Research Institute
Section Information Technology
Dortmund University of Technology
Otto-Hahn-Strasse 8, 44227 Dortmund, Germany
$\{firstname.lastname\}$@udo.edu

ABSTRACT

In this paper, we introduce a methodology for the approximation of optimal solutions for a resource allocation problem in the domain of Grid scheduling on High Performance Computing systems. In detail, we review a real-world scenario with decentralized, equitable, and autonomously acting suppliers of compute power who wish to collaborate in the provision of their resources. We exemplarily apply NSGA-II in order to explore the bounds of maximum achievable benefit. To this end, appropriate encoding schemes and variation operators are developed while the performance is evaluated. The simulations are based upon recordings from real-world Massively Parallel Processing systems that span a period of eleven months and comprise approximately 100,000 jobs. By means of the obtained Pareto front we are able to identify bounds for the maximum benefit of Grid computing in a popular scenario. For the first time, this enables Grid scheduling researchers to rank their developed real-world strategies.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*Computations on Discrete Structures*; G.2.1 [**Mathematics of Computing**]: Discrete Mathematics—*Combinatorics*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic Methods*; I.6.4 [**Computing Methodologies**]: Simulation and Modeling—*Model Validation and Analysis*

General Terms

Experimentation, Algorithms, Performance

Keywords

Multi-Objective Optimization, Scheduling, Grid Computing

1. INTRODUCTION

Since Foster and Kesselman [13] introduced the concept of Grid Computing, it has become a common term for the collaborative usage of heterogeneous and geographically distributed resources. The main goal is to enable users to access these resources as services in a transparent way. Although the nature of provisioning is not limited to bare computational power, it is still the most common case for production Grid environments. These High Performance Computing (HPC) Grids comprise several Massive Parallel Processing[1] systems, so called *Grid sites*, each of which capable of running highly demanding applications. Examples of such are simulations in the area of earth system science [3], biology [14], or high energy physics [9].

One of the most important challenges in current Grid research is to find scheduling strategies that offer a good job-to-resource assignment. However, since resources are usually owned and maintained by different institutions or organizations and, as such, situated in different administrative domains, the problem is far more complex than parallel job scheduling [12]: in addition to the assignment of jobs to local resources (which already poses an NP-complete problem), a suitable Grid site has to be chosen in the first place.

Most theoretical and experimental works do not consider these two problems separately: heuristic strategies handle both job interchange between computing sites as well as local scheduling and assume an unrealistic model by building upon complete knowledge regarding the systems' dynamic performance. By doing so, they ignore real world requirements: current HPC Grid installations usually run autonomous local batch management systems with scheduling strategies tailored to the needs of the funding organization's individual user communities. In addition, most resource providers consider data on the performance of their systems confidential, usually denying access to metrics such as machine utilization, number of current jobs and the like.

Thus, a major concern should be the optimization of job interchange mechanisms between autonomous and equitable sites, leaving local systems and strategies untouched as far as possible. Additionally, there is a strong need for perfor-

[1]Such as parallel or cluster computers.

mance values regarding the attainable frontiers of scheduling heuristics in general and distribution decision algorithms in particular. This could provide a solid foundation for performance comparison during the development of such strategies.

In this paper, we propose—using the Non-dominated Sorting Genetic Algorithm (NSGA-II) [5] as one standard method for multi-objective optimization—a methodology to determine near optimal exchange of jobs between heterogeneous Grid sites. To this end, we consider the problem of finding an optimal partition of the overall workload submitted to all sites. In order to model real world scenarios and user behavior as exact as possible, we consider real machine configurations, common local scheduling strategies and real workload recordings consisting of 50,000 to 150,000 jobs. The perceived results point out the benefit of Grid computing, and—together with a realistic infrastructure model—form an early test suite for future strategy development.

The remainder of this paper is organized as follows: in Section 2 we review the problem of job interchange and Grid scheduling in more detail. Then, we explain the considered model in Section 3 and the applied methodology in Section 4. Following, we present the reference sets that quantify the benefit of Grid job interchange in Section 5 and conclude the paper in Section 6.

2. BACKGROUND

The problem of job distribution between federated compute clusters has been continuously studied since the emergence of Grid computing; today's research however focuses on the behavior of centralized Grid scheduling services [2, 10] and yields only limited innovation regarding efficient algorithms for resource allocation. Ernemann et al. [8] point out advantages of hierarchical scheduling in general by considering the Average Weighted Response Time objective. In another work, they propose an adaptive algorithm which decides—based on the overall system state—whether to split up a job to execute it on different Grid sites (Multi-site computing) [7]. Kurowski et al. [19] identify multiple objectives for efficient job scheduling in Grids and propose a strategy based on prediction mechanisms and resource reservation. Of course, due to the highly dynamical character of the problem, nature-inspired techniques for adaptive scheduling have also been proposed during the last years: besides GA-driven Metaschedulers [4], hybrid heuristics were applied [1, 18].

For the decentralized scenario, only few results that support the application of job delegation in Grids have been published so far. England and Weissman [6] give an estimation of benefits and costs but base their results on synthetic workloads and review the average slowdown only. Hamscher et al. [17] use different job sharing strategies on two real workloads but give no quantification of the resulting benefit. Recently, Grimme et al. [15] analyze the prospects of collaborative job sharing in Grids. They compare their results to the non-cooperative scenario of the same machines but do not give a quantitative estimation of possible collaborative benefits.

Here, we adapt the model of Grimme et al. by omitting any central component and allow direct communication between sites. Unlike the before cited works, we do not aim to propose a new scheduling heuristic, but to set a cornerstone for further algorithm development by highlighting the obtainable benefit of Grid computing.

3. SCENARIO

The considered Grid scenario setup consists of K independent HPC sites, each of which representing a parallel machine with m_k, $k \in \{1, \ldots, K\}$ identical processors. It is assumed that submitted jobs (a) can be executed on any subset of the m_k processors and (b) that the machine configurations of all sites only differ in the amount, but not in the speed of available resources. The current trend in large MPP systems however shows that, in most cases, systems consist of CPUs with more or less equal speed ratings[2], making this simplification justifiable.

Further, we assume the workload of each site to consist of n_k rigid[3] parallel batch jobs, which is the common case in such systems [21]. These jobs need exclusive access to $m_j \leq m_k$ processors during their execution. The parallelism value of a job j is provided at its release date r_j, while the processing time p_j of the job is unknown until its completion time C_j. Until then, only a user-provided, usually inaccurate estimate $\bar{p}_j$ is known to the system. As no overuse of resources is allowed, the processing time is determined as $p_j = \min\{p_j, \bar{p}_j\}$, following the common policy of canceling jobs whose real processing time exceeds the user's estimation.

Each site comprises two elements that influence job-to-resource mapping decisions: the batch management system with its local scheduling heuristics and a decision maker component that handles the acceptance and distribution of jobs between participating Grid sites.

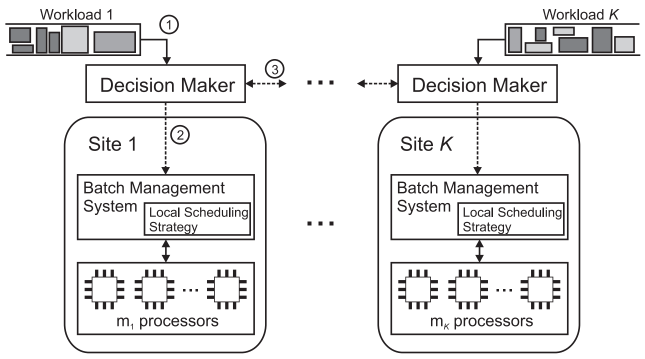

Figure 1: Schematic depiction of the considered Grid scenario with independent sites using decentralized scheduling. The decision maker (1) accepts workload from a local user community and either (2) assigns jobs to the local site's resource management system or (3) offers them to another site's decision maker.

Figure 1 depicts this process: jobs are submitted by a local user community to the site's scheduling system. There, it is enqueued (1) and—depending the decision maker strategy—either started locally (2) or moved to another site (3). The on-site assignment to processors is finally done by the local batch management system.

The process of job exchange can be realized on the basis of two different protocols: in *active delegation* the local de-

[2]Such as MIPS (million floating point operations per second), see `http://top500.org`.

[3]The jobs are neither moldable nor malleable concerning processing time or resource consumption.

cision maker offers jobs to remote sites in a proactive way, while in *passive delegation* remote sites request jobs that the local decision maker is willing to dispatch. In either case, the local decision maker has to publish information on the contents of its local waiting queue. Depending on the policy of its administrative domain, this data can be restricted to a subset. The decision on how many and which jobs to disclose to remote sites is in any case left to the local decision maker.

As the workload is arriving over time, Grid scheduling belongs to the class of online problems and poses two algorithmic challenges:

1. Find a decision strategy that accepts submitted jobs and determines whether to hand over a job to the local resource management system or to a remote site while keeping the performance of the local systems high.

2. Assign ultimately accepted jobs that are waiting in the queue to local resources in an efficient way.

While different heuristics for the second step are currently in use on real HPC installations, only few strategies have been proposed for the decision of job acceptance and delegation. As such, the latter is still an open research problem.

4. METHODOLOGY AND ALGORITHM

In this paper, we utilize a multi-objective optimization method to determine the maximum profit the application of job interchange in the afore described model may yield. Of course, the here described approach will not identify the overall optimal interchange results for all site setups; moreover, we develop a methodology for generating reference values to rate newly designed decision strategies not regarding the non-cooperative case but the best achievable interchange behavior.

To this end, we modify the decision maker component in Figure 1 to allow submission only to the local resource management system. The interchange of jobs is modeled in an offline manner by workload partitioning, leading to differing amounts of workload on each site. Now, if a huge amount of the workload from site A is migrated to site B the performance of site A typically improves at the expense of the performance of site B. Thus, we obtain a multi-objective problem of finding best trade-offs for workload distribution for all sites.

The following sections introduce the representation of the problem and the applied methodology: first, we describe the encoding scheme for a Grid consisting of two connected sites. Note, however, that this methodology is also applicable for an arbitrary number of Grid participants. Then, we present the recombination and mutation operators that have been adapted to the problem. After introducing the actual computation of the objectives, we discuss issues like search space limitation, constraint handling, and creation of a start population.

4.1 Encoding Scheme

The genotype of an individual $\mathbb{I} = [a_1, a_2, \ldots, a_{l(\mathbb{I})}]$ has to respect the partitioning of the original workload traces with n_k jobs each into K sets, where K is the total number of grid sites and $l(\mathbb{I}) = \sum_{k=1}^{K} n_k$ determines the length of individual $\mathbb{I}$.

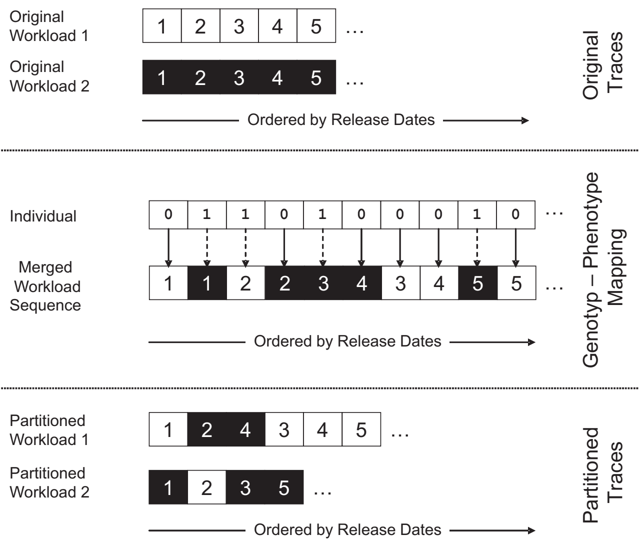

Figure 2: Encoding scheme of an individual an the resulting workload partition.

By merging the original workload traces, we create *one* job sequence that is sorted by the jobs' release dates r_j. This job sequence is considered as reference workload and its partition is now represented by the individuals' allele values. To this end, an individual must assign each job to a Grid site and requires consequently $l(\mathbb{I})$ genes.

Each grid site is then represented by a number from the interval of possible alleles $a \in [0, K-1]$. For the here examined examples, we set $K = 2$ which reduces the genotype to a binary representation. The mapping between geno- and phenotype results in the desired two workloads. Still, the order by release dates is preserved. Due to this encoding scheme we have to deal with a very high dimensional search space as one gene position for each job is required.

The whole process is depicted in Figure 2. It is noteworthy that the assignment of a job j to a machine during the manipulation of a certain gene can be vetoed: if the job requires more processors than the site can provide, $m_j > m_k$, it is reassigned to its originating site.

4.2 Mutation Operators

Commonly, the NSGA-II algorithm is applied with mainly recombination while mutation plays a minor role. During simulations, we first applied a *random mutation* in which each gene $a_i \in \mathbb{I}$ is changed with probability P_{rand} to

$$a_i = [a_i + \lfloor (K+1) \cdot \mathcal{U}(0,1) \rfloor] \bmod K \qquad (1)$$

where $\mathcal{U}(0,1)$ is a uniformly distributed number between 0 and 1. This mutation exclusively favored the exchange of jobs between different partitions and led to a beneficial assignment of better fitting jobs.

However, our experiments showed that it is crucial to also vary the partition size: not doing so resulted in a limited coverage of the Pareto front. Consequently, it was necessary to introduce an additional mutation which realizes a one-sided shift of jobs.

This *shift mutation* is described in Algorithm 1: first, the site k_s where jobs will be shifted to is determined randomly (Line 1). Then, the number of jobs to be shifted n_s is deter-

Algorithm 1 Job Shift Mutation

```
Require: Individual $\mathbb{I}$, Shift step-size $\sigma$, Grid size $K$
1: $n_s = \lfloor |\mathcal{N}(0, \sigma)| + 0.5 \rfloor$
2: $k_s = \lfloor (K + 1)\mathcal{U}(0, 1) \rfloor$
3: while $n_s > 0$ do
4:     $p$ := uniform random position in individual $\mathbb{I}$
5:     if $\mathbb{I}[p] \neq k_s$ then
6:         $\mathbb{I}[p] = k_s$
7:         $n_s = n_s - 1$
8:     end if
9: end while
```

mined (Line 2) and, until this number is reached (Line 3), shift operations are performed. Herefor, a uniform random position in the individual is selected (Line 4) and, if a shift is possible[4] (Line 5), the corresponding gene is modified to reflect the change towards the target site (Line 6).

The algorithm requires three parameters:

1. the individual $\mathbb{I}$ to modify,
2. the step-size σ which specifies the standard deviation for normal distribution, and
3. the number K of sites participating in the Grid.

In order to incorporate both variation methods we switch between both operators with probability P_{shift}. In each generation we mutate every individual with probability P_{shift} using the shift algorithm and with probability $(1 - P_{shift})$ with random mutation. For the latter, we change each gene with the above mentioned probability P_{rand}.

4.3 Recombination Operator

The typical application of NSGA-II implies the dominant influence of the recombination operator. Since the effect of its properties to scheduling problems is not sufficiently studied, we resort to choosing among the standard operators [22]. Uniform Crossover (UCX), where each bit is swapped between two parents with fixed probability, is not appropriate as it results in an exchange of jobs only, but almost no variation in partition size. As mentioned above this is required in order to discover a diverse front. A swap between parents would result on average in the same number of assigned jobs to each partition.

n-Point Crossovers, in contrast, are capable to combine both desired variation characteristics and to enable an exchange and balance variation between the partitions at the same time. Therefore, we apply the Two-Point Crossover (TPX) operator with a probability of P_{recomb}.

4.4 Objective Functions

As we aim to discover the potential benefits of job exchange in Grid computing, we have to formulate appropriate objective functions. In theory of scheduling [23] as well as in practical evaluation of algorithms [11], the user-centric perspective is commonly reflected by the response time $\mathrm{RT}_j = C_j - r_j$ objective of a job j. This objective is defined as the timespan between the submission of job j at its release date r_j and the actual completion time C_j at site k. The shorter the response time the less the user has to wait for the results of the submitted job.

$$\mathrm{AWRT}_k = \frac{\sum\limits_{j \in \pi_k} p_j \cdot m_j \cdot (C_j - r_j)}{\sum\limits_{j \in \pi_k} p_j \cdot m_j} \tag{2}$$

In order to determine the schedule quality for all jobs $j \in \pi_k$ that are executed at site k, we compute the Average Weighted Response Time AWRT_k, see Equation (2).

Here, each job is weighted with its resource consumption $(p_j \cdot m_j)$. Following Schwiegelshohn et al. [25], this ensures that neither splitting nor combination of jobs can influence this objective function in a beneficial way.

However, a shorter AWRT for users at a certain site comes along with longer AWRT values for users at other sites. This reveals the conflicting nature of Grid collaboration. In this paper, we determine the Pareto front of AWRT values that can be achieved between different sites. In other words, we solve the multi-objective optimization problem

$$\mathrm{MOP} := \min \begin{pmatrix} \mathrm{AWRT}_1 \\ \vdots \\ \mathrm{AWRT}_K \end{pmatrix} \tag{3}$$

by finding optimal assignments of jobs to Grid sites.

4.5 Limitation of the Search Space

The multi-objective search space of the tackled problem is extraordinarily large and therefore hard to discover: when most jobs are migrated to one site, leaving the other site with almost none, the AWRT values deteriorate extremely, since even the most efficient local scheduling system is unable to compensate for an oversized workload. Heavy load on one site causes a general congestion of jobs resulting in very long AWRT values. Such results, however, are not acceptable in practice and therefore of minor interest. In fact, AWRT values that exceed the results obtained when completely obviating collaboration by 30% and more obviously foil the participation in HPC Grids on the whole. For the used workloads, see Table 1, we consequently limited the search space by restricting possible response time values using

$$\mathrm{AWRT}_k \in [0 \ldots 100{,}000] \; \forall \; k = 1 \ldots K \tag{4}$$

such that if an individual achieves a higher AWRT, its objective is assigned an infinite value, effectively discarding it from the prospective evolutionary process.

4.6 Generation of an Initial Population

Due to the size of the search space it is further necessary to start the evolution in an area of interest. As we aim to find Pareto optimal solutions better than exclusive single site execution, we take the results from a well performing and highly popular local scheduling heuristic as the start solution.

This algorithm, the Extended Argonne Scheduling System [20] (EASY) works as follows: First, the job at the head of the waiting queue is examined. If this job can be started immediately on the locally available nodes, it is removed from the waiting queue and executed directly. Otherwise, its earliest possible start time—calculated on the basis of the users' runtime estimations of the already running jobs—is

[4]The job was not already assigned to the target site, that is.

assumed as a reservation. Then, for every other waiting job the following two conditions are tested: (1) it will terminate before the first job is expected to start and (2) it will not interfere with the nodes reserved for the first job. The first candidate that meets either condition is used as a backfill and started immediately.

In order to create a sufficient amount of start solutions, we used the workloads from two original recordings, simulated their processing and and shuffled them by uniformly swapping the allocation of 5,000 jobs. This is achieved flipping their corresponding genes in each individual, guaranteeing a certain degree of diversity close to the EASY reference solution.

5. EVALUATION

As we aim to evaluate our methodology and its application on realistic data originating from real-world Grid setups, we first introduce the data sources used for our experiments. This is followed by a detailed explanation of four different Grid site setups. Before discussing our results in detail, we describe two heuristic approaches used for the matter of comparison.

5.1 Used Real Workload Traces

The Parallel Workloads Archive[5] provides job submission and execution traces recorded on real-world HPC sites, each of which containing information on relevant job characteristics. For our evaluations, we restrict the set of used workloads to those which contain valid runtime estimations, since the EASY algorithm depends on this data. This results in the following five traces:

1. the KTH trace which contains records from a 100 processor IBM RS/6000 SP system at the Swedish Royal Institute of Technology in Stockholm over a period of eleven months,

2. the CTC trace from a 430 processor IBM RS/6000 SP system at the Cornell Theory Center in Ithaca, NY, which also covers a timespan of eleven months,

3. the SDSC*xx* logs recorded at the San Diego Supercomputer Center in La Jolla, CA; they comprise

   (a) the SDSC00 trace containing submissions to a 128 processor IBM RS/6000 SP system over 24 months,

   (b) the SDSC03 "Blue Horizon" trace of a 1152 processor IBM RS/6000 SP system over 24 months, and

   (c) the SDSC05 "DataStar" trace recorded on a IBM *e*Server pSeries 655/690 system with a total of 1664 processors during 13 months.

The original workloads record time periods of different length. In order to be able to combine different workloads in a multi-site simulations we shortened the workloads to an 11-month recording, the minimum required length of all participating workloads. Details on the used traces are given in Table 1.

[5]`http://www.cs.huji.ac.il/labs/parallel/workload/`

5.2 Simulation Setup

We evaluate four different setups with two sites each, see Table 1. Our first setup comprises two smaller sites (100 and 128 processors) that represent typical department-size cluster installations, see Setup I. As an example of one small and one medium-sized machine, we evaluate Setup II, a collaboration between 100 processors and 430 processors; the latter representing a fairly standard-sized university compute center. Another configuration combines a small and a large site (100 and 1,152 processors), reflecting one small/medium business system which acquires additional compute cycles from a large HPC center. Finally, we investigate the effects when connecting two large HPC centers (1,152 and 1,664 processors), each attempting to balance their local load.

Regarding the configuration of the NSGA-II algorithm, we use a population size of $\mu = 70$ individuals with a total of 200 generations for each simulated setup. We apply tournament selection without replacement and a tournament size of 2. In detail, we randomly select two individuals without replacement and copy the best individual to the mating pool. This process is repeated until μ individuals are selected. As the crossover operator, we use TPX with a probability of $P_{recomb} = 0.9$. For mutation, we apply shift and random mutation with equal probability of 50%, using a shift step size of $\sigma = 2,000$. The random mutation flips each gene with a probability of $P_{rand} = 0.1$.

The evaluations where performed on a 120 processor cluster comprising standard PCs with Pentium IV 2.4 GHz processors running on Linux. On this setup, each generation took approximately 15 minutes to evaluate in parallel, leading to a simulation time of about two days per setup. All simulations were based on a variant of Sastry's [24] C++ implementation of the GA toolbox for MATLAB and the Java-based TEIKOKU Grid Scheduling Framework [16].

5.3 Reference Values

In order to verify the approximation results produced by our methodology, we generate a solution for job interchange between two sites using a request heuristic (RQ) with complete insight into each site's waiting queue which operates as follows:

- When a job arrives at the decision maker component that has been submitted from the local user community, it is added to the end of the local waiting queue.
- On every modification of the local waiting queue, the local scheduling algorithm (in our case EASY) is invoked. During this step, either the first job in the local waiting queue is started, or one other job from the local waiting queue may or may not be backfilled.
- The decision maker requests all remote queues from all sites and—subsequently iterating over all remote jobs—adds each job that could be started immediately to the end of the local waiting queue. During this step, the number of potentially free resources at the current instant is reduced by the number of requested resources of each added remote. Note however, that the real number of currently unused resources is left untouched.

Due to the combination of requesting potential backfilling candidates from remote sites and the inherent characteristics of the EASY algorithm, we anticipate the RQ heuristic

Setup	Workload	Machine Size	Number Jobs	Total Jobs	AWRT (EASY)	AWRT (POOL)	AWRT (RQ)
I	KTH-11	100	28,479	58,289	75,157.63	63,011.46	61,448.95
	SDSC00-11	128	29,810		73,605.86	58,772.48	57,918.61
II	KTH-11	100	28,479	105,678	75,157.63	58,056.65	55,462.28
	CTC-11	430	77,199		52,937.96	51,742.31	51,347.65
III	KTH-11	100	28,479	94,063	75,157.63	58,497.60	55,527.87
	SDSC03-11	1,152	65,584		50,772.48	50,809.59	50,601.24
IV	SDSC03-11	1,152	65,584	140,487	50,772.48	43,731.50	44,264.10
	SDSC05-11	1,664	74,903		54,953.84	48,628.57	46,498.92

Table 1: Properties and reference results for the examined Grid setups. The AWRT values (in seconds) are given for exclusive single site execution (EASY), the job pool exchange mechanism (POOL), and the request heuristic (RQ).

to perform near optimal for our use case. Note that the assumption of a holistic view on the waiting queues of remote sites is unrealistic, since the common real-world policy for publishing information, see Section 3, on the local waiting queue is restrictive, allowing no direct access at all. Still, this approach is justifiable against the background of our stated goal to discover a theoretically achievable result.

Additionally, we use results determined by Grimme et al. [15] to exemplary show the application of our proposed methodology for the assessment of developed Grid job interchange algorithms. In their approach (POOL) a central job pool is used to migrate jobs between Grid sites: in case a considered job cannot be executed locally, the decision maker publishes it into a central pool. At the same time, each site includes the pooled jobs into its local scheduling process and thus enables job exchange between independent sites.

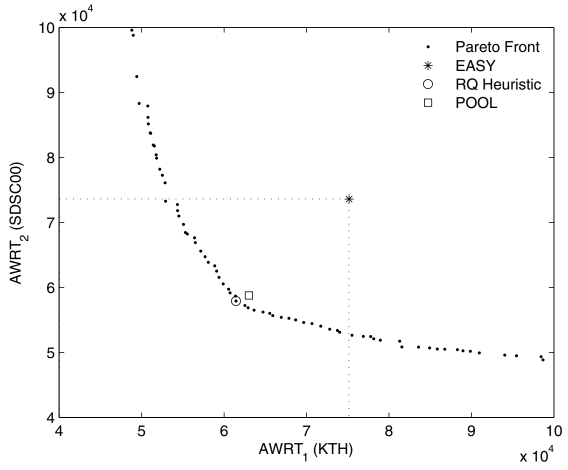

Figure 3: Results for Setup I with KTH ($m = 100$) and SDSC00 ($m = 128$) workloads after 200 generations.

5.4 Results

The obtained results are depicted in four figures for each setup separately. In addition to the Pareto front, each figure contains the results of the POOL strategy, the RQ heuristic, and the non-collaborative EASY algorithm for both objectives. Further, we mark the *area of benefit* limited by the EASY algorithm results and the coordinate system's origin.

In all examined cases, it is possible to find a diverse front within the specified search interval. Obviously, there exists a large set of trade-off solutions for job exchange apart from the already known heuristics results. In order to judge on the quality of the NSGA-II generated Pareto front we can only refer to the single RQ heuristic reference result, see Section 5.3. As all Pareto fronts contain this solution we conjecture that also the rest of the front is approximated appropriately. Further, this indicates that the application of NSGA-II to the presented problem of job exchange in HPC Grids works fine.

In detail, Figures 3 and 6 show results for the two similar-sized Setups I and IV. In both cases, a convex Pareto front is obtained which covers the whole range of the search space. The heuristic solutions produce results in the center of the front while it is actually reached by the RQ heuristic.

Figures 4 and 5 yield similar results for the front. However, compared to the non-collaborative EASY reference result, the potential of AWRT improvement is much smaller for the larger site than for the smaller site. Note that the axes are scaled in a smaller range for $AWRT_2$ in both figures.

Using our proposed methodology we discover two bounds for the benefit of collaborative Grid computing: (a) the non-

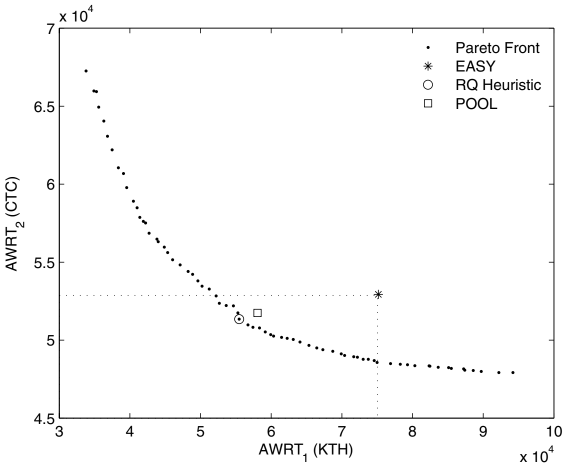

Figure 4: Results for Setup II with KTH ($m = 100$) and CTC ($m = 430$) workloads after 200 generations.

collaborative case where no jobs are exchanged marks the upper bound, and (b) the Pareto solutions within the area of benefit which marks approximated lower bounds among the

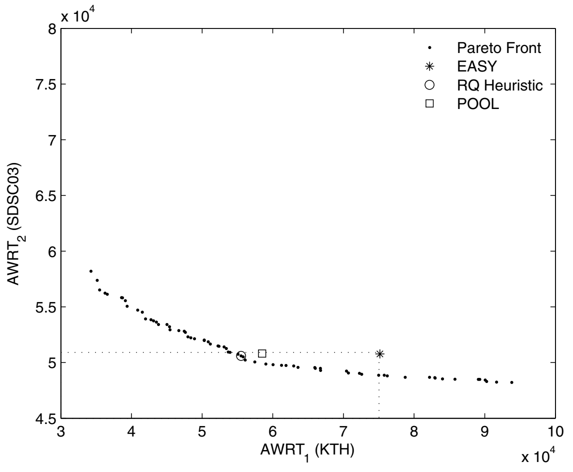

Figure 5: Results for Setup III with KTH ($m = 100$) and SDSC03 ($m = 1,152$) workloads after 200 generations.

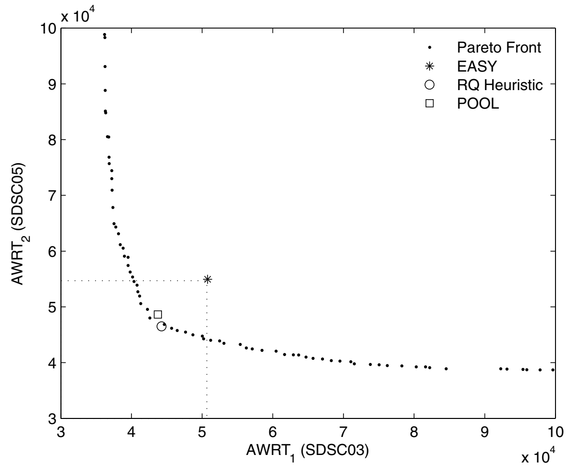

Figure 6: Results for Setup IV with SDSC03 ($m = 1,152$) and SDSC05 ($m = 1,664$) workloads after 200 generations.

beneficial trade-off solutions. Reasonable real-world heuristics must yield performance results within these bounds in order to be applicable for Grid scheduling in an advantageous means. Further, those heuristics become comparable as they can be ranked with respect to both the Pareto front and the EASY results.

Exemplarily, we analyze the performance of the POOL strategy for the here given setups, although it ignores real-world restrictions. By utilizing jobs in a central pool, this strategy is enabled to reach an almost perfect compromise between partners.

Furthermore, knowledge about the Pareto front allows more advanced evaluations of a heuristic's quality. In our case studies, it is possible to show for the first time that the common heuristics tend to balance the load between sites. As a consequence, the resulting AWRT values range in the same order of magnitude while representing the most balanced trade-off solutions in the whole front. However, for the size-wise heterogeneous Setups II and III, see Figures 4 and 5, these solutions seem to be fair when the whole Pareto front is taken into account, but considering only the area of mutual benefit reveals that those "fair" solutions cannot develop the full potential of trade-offs disclosed within this area. In other words, a good Grid job exchange heuristic should not only lead to a single compromise within the whole Pareto front but should achieve balanced results in the whole area of benefit.

6. CONCLUSION

In the work at hand, a methodology for the approximation of compromise solutions for real-worlds job exchange strategies between HPC systems has been described and exemplarily applied. The NSGA-II based approach explored the bounds of maximum achievable benefit in Grid scheduling by generating reference results for selected Grid setups. We compared existing Grid scheduling heuristics with respect to the Pareto front and found new insights in quality assessment, allotment of profit, and fairness aspects. With these concepts and the obtained results, any kind of Grid job exchange strategy can be compared and ranked.

For future research a generalization of our methodology may include various aspects: The used NSGA-II might be substituted by another multiobjective optimization algorithm or alternative nature inspired concept. Further, our case studies can be regarded as a first effort that leaves ample room for future research: the investigation of more complex Grid scenarios of multiple partners, for example, would require the consideration of more than two objectives.

For Grid scheduling research, the identified results motivate the design of new exchange strategies that are capable to reach desired points in the Pareto front by handling the job exchange more flexibly. Within this context, our methodology and the achieved results can be seen as a cornerstone for such future development endeavors.

Acknowledgement

Joachim Lepping is member of the Collaborative Research Center 531, "Computational Intelligence", at the Dortmund University of Technology with financial support of the DFG (Deutsche Forschungsgemeinschaft).

7. REFERENCES

[1] A. Abraham, R. Buyya, and B. Nath. Nature's heuristics for scheduling jobs in computational grids. In P. Sinha and R. Gupta, editors, *Proceedings of 8th IEEE International Conference on Advanced Computing and Communications*, pages 45–52, New Delhi, India, 2000. Tata McGraw-Hill Publishing.

[2] P. Andreetto, S. Borgia, A. Dorigo, et al. Practical Approaches to Grid Workload and Resource Management in the EGEE Project. In *Proceedings of the Conference on Computing in High Energy and Nuclear Physics (CHEP)*, Interlaken, Switzerland, September 2004. Organisation Européenne pour la Recherche Nucléaire (online).

[3] D. Bernhold, S. Bharathi, D. Brown, K. Chanchio, et al. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. *Proceedings of the IEEE*, 93(3), March 2005.

[4] J. Carretero, F. Xhafa, and A. Abraham. Genetic algorithm based schedulers for grid computing systems. *International Journal of Innovative Computing, Information and Control*, 3(6):1–19, 2007.

[5] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer et al., editors, *Parallel Problem Solving from Nature VI*, volume 1917 of *Lecture Notes in Computer Science (LNCS)*, pages 849–858. Springer, 2000.

[6] D. England and J. B. Weissman. Cost and Benefits of Load Sharing in the Computational Grid. In D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proceedings of Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science (LNCS)*, pages 160–175. Springer, 2004.

[7] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Enhanced Algorithms for Multi-Site Scheduling. In M. Parashar, editor, *Proceedings of the 3rd International Workshop on Grid Computing*, volume 2536 of *Lecture Notes in Computer Science (LNCS)*, pages 219–231. Springer, 2002.

[8] C. Ernemann, V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. On Advantages of Grid Computing for Parallel Job Scheduling. In *Proceedings of the 2nd International Symposium on Cluster Computing and the Grid*, pages 39–46. IEEE Press, May 2002.

[9] M. Ernst, P. Fuhrmann, A. Papaspyrou, M. Radicke, L. Schley, and R. Yahyapour. A Computational and Data Scheduling Architecture for HEP Applications. In *Proceedings of the Conference on High Energy Physics (CHEP)*, Mumbai, India, February 2006.

[10] D. W. Erwin and D. F. Snelling. UNICORE: A Grid Computing Environment. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the 7th International Euro-Par Conference*, volume 2150 of *Lecture Notes in Computer Science (LNCS)*, pages 825–834, Manchester, UK, August 2001. Springer.

[11] D. G. Feitelson. Metrics for parallel job scheduling and their convergence. In D. G. Feitelson and L. Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 2221 of *Lecture Notes in Computer Science (LNCS)*, pages 188–206. Springer, 2001.

[12] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel Job Scheduling – A Status Report. In D. Feitelson, L. Rudolph, and U. Schwiegelshohn, editors, *Proceedings of Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science (LNCS)*, pages 1–16, Boston (MA), USA, June 2005. Springer.

[13] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufman, 1st edition, 1998.

[14] J. Garzon, E. Huedo, R. Montero, I. Llorente, and P. Chacon. Adaptation of a Multi-Resolution Docking Bioinformatics Application to the Grid. *Journal of Software*, 2:1–10, 2007.

[15] C. Grimme, J. Lepping, and A. Papaspyrou. Prospects of Collaboration between Compute Providers by means of Job Interchange. In E. Frachtenberg and U. Schwiegelshohn, editors, *Proceedings of Job Scheduling Strategies for Parallel Processing*, volume 4942 of *Lecture Notes in Computer Science (LNCS)*, pages 132–151. Springer, June 2007.

[16] C. Grimme, J. Lepping, A. Papaspyrou, P. Wieder, R. Yahyapour, A. Oleksiak, O. Wäldrich, and W. Ziegler. Towards a standards-based Grid Scheduling Architecture. CoreGRID Technical Report TR-0123, Institute on Resource Management and Scheduling, December 2007.

[17] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Evaluation of Job-Scheduling Strategies for Grid Computing. In R. Buyya and M. Baker, editors, *Proceedings of the 7th International Conference on High Performance Computing*, volume 1971 of *Lecture Notes in Computer Science (LNCS)*, pages 191–202. Springer, 2000.

[18] W. Jakob, A. Quinte, K.-U. Stucky, and W. Süss. Optimised Scheduling of Grid Resources Using Hybrid Evolutionary Algorithms. In R. Wyrzykowski et al., editors, *Proceedings of the 6th International Conference on Parallel Processing and Applied Mathematics*, number 3911 in Lecture Notes in Computer Science (LNCS), pages 406–413, 2005.

[19] K. Kurowski, J. Nabrzski, A. Oleksiak, and J. Weglarz. Scheduling Jobs on the Grid - Multicriteria Approach. *Computational Methods in Science and Technology*, 12(2):123–138, 2006.

[20] D. A. Lifka. The ANL/IBM SP Scheduling System. In *Proceedings of Job Scheduling Strategies for Parallel Processing*, volume 949 of *Lecture Notes in Computer Science (LNCS)*, pages 295–303. Springer, 1995.

[21] U. Lublin and D. G. Feitelson. The Workload on Parallel Supercomputers: Modeling the Characteristics of Rigid Jobs. *Journal of Parallel and Distributed Computing*, 63(11):1105–1122, 2003.

[22] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 2 edition, November 1998.

[23] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice-Hall, New Jersey, second edition, 2002.

[24] K. Sastry. Single and Multiobjective Genetic Algorithm Toolbox for Matlab in C++. Technical Report 2007017, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 117 Transportation Building, 104 S. Mathews Avenue Urbana, IL 61801, 2007.

[25] U. Schwiegelshohn and R. Yahyapour. Fairness in Parallel Job Scheduling. *Journal of Scheduling*, 3(5):297–320, 2000.