Increasing Waiting Time Satisfaction in Parallel Job Scheduling via a Flexible MILP Approach

Stephan Schlagkamp, Matthias Hofmann
Robotics Research Institute
TU Dortmund University
{stephan.schlagkamp, matthias.hofmann}@udo.edu

Lars Eufinger
Institute of Transport Logistics
TU Dortmund University
eufinger@itl.tu-dortmund.de

Rafael Ferreira da Silva
Information Science Institute
University of Southern California
rafsilva@isi.edu

***Abstract*—Scheduling of jobs in parallel computing is crucial to efficiently use shared resources, while attaining user satisfaction. In this paper, we evaluate how mixed-integer linear programming (MILP) can be applied for the online parallel job scheduling problem (which is well-known to be an NP-complete problem). Therefore, we introduce the idea of planning horizons for parallel job scheduling, and provide a MILP formulation of the targeted scheduling problem. Due to the linear fashion of possible MILP objective functions, the proposed scheduling algorithm is flexible towards different optimization goals. We make use of data collected in a user-based study and workload traces from two real production systems, and demonstrate that our approach suffices to increase users' waiting time satisfaction. Additionally, we show that our MILP formulation outperforms the EASY scheduling technique with conservative backfilling, when neglecting the online character of job submissions.**

***Keywords*-Linear programming, Parallel job scheduling, user satisfaction**

I. Introduction

Parallel computing has become mainstream in scientific computing [1]–[3]. Typically, users submit computational jobs to a shared computing infrastructure, where a scheduler (mostly centralized) aggregates all submitted jobs, and takes decisions to where the jobs should be allocated (matching their requirements to the available computing resources), and when they start their computations (job execution). Parallel task scheduling is known to be an NP-complete problem, thus heuristics have been developed to tackle this problem. However, these heuristics still have to address several (concurrent) objectives, for example energy consumption, waiting times, fault tolerance, among others. In this paper, we propose a mixed-integer linear programming formulation (MILP) [4] to address the online parallel job scheduling problem. The advantage of using a MILP formulation approach is its capability of addressing a wide variety of objectives concurrently and solve them optimally. We first introduce a method to *discretize* an online parallel job scheduling problem, which allows us to explore the possibilities of MILPs. Then, we design our MILP formulation based on the analysis results obtained in the exploration step. Due to the flexibility of the linear optimization function, our formulation has the potential for a broad set of mentioned optimization goals, such as (1) maximizing utilization, (2) optimization of energy-consumption, and (3) user satisfaction. In this paper, we explore the possibilities of the MILP towards user satisfaction.

User satisfaction is the ultimate goal of every computing system, and therefore several works have focused on this aspect. For instance, Shmueli and Feitelson seek to increase user satisfaction by encouraging subsequent job submission [5]. Their analysis is based on the concept that users work in batches and sessions [6]. The term *batch* denotes a set of jobs, which are timely close submitted by a user, e.g., the inter-arrival time between job submissions lie within a certain threshold. The concept of *sessions* describes the subsequent submissions of batches. Once a batch, or certain jobs in the batch finish, the user might submit the next batch of jobs. Based on this model, they investigate whether a user is currently working within a session, and if so jobs are prioritized so that the users will receive the results faster, and consequently continue working in the current session. Contrary to this approach, this paper focus on a different aspect to increase user satisfaction. We use data collected in a user-based study [7]–[9] on waiting time satisfaction, and formulate optimization goals to:

1) Maximize the number of jobs that lie in an acceptable waiting time frame; and
2) Decrease the lateness of jobs according to an acceptable waiting time deadline.

The main contributions of this work include:

- The use of planning horizons in parallel job scheduling, which allows to discretely optimize schedules in online scheduling environments;
- A MILP formulation for the parallel job scheduling problem on parallel machines. The formulation is flexible towards linear optimization goals; and
- An evaluation of the performance of the MILP formulation with focus on increased waiting time satisfaction of users in parallel computing.

This paper is structured as follows. In the next section, we give an overview of the related work. In Section III, we introduce the scheduling approach, *discretizing* the online parallel job scheduling problem by interpreting it as a

consecutive optimization of independent sub-schedules. The flexible MILP solver is described in Section IV, where we also provide a complexity classification of the targeted scheduling problem. Section V contains an evaluation for two different user-based optimization goals. Section VI concludes this paper and highlights future works.

II. Related Work

Workload archives are widely used for research on distributed systems, to validate assumptions, to model computational activity, and to evaluate methods in simulation or in experimental conditions. Several works have used such archives (e.g., the Parallel Workloads Archive [10]) to develop and evaluate strategies to optimize multiple objectives in parallel computing including the categorization and prediction of workload behaviors [11]. However, most of these studies focus on the performance analysis of individual (or groups of) jobs. On the other hand, user behavior is seen as a key factor to understand job submission processes [12]–[14], evaluate newly designed scheduling strategies or policies [15], or increase user satisfaction [5].

A successful and efficient experiment execution mainly depends on how tasks are scheduled and executed. A common scheduling strategy is the EASY with conservative backfilling strategy [16]. EASY follows a first-come-first-serve (FCFS) approach, however it also allows jobs to execute earlier—in case they do not delay the execution of previously queued jobs. Due to the capability of EASY to handle online job scheduling and MILP is not suitable in a direct, static evaluation, we introduce discretized scheduling scenarios. Different backfilling strategies used in schedulers are evaluated in a survey conducted by Srinivasan et al. [17]. Schedulers used in practical applications provide a variety of parameter-based adjustments, e.g., the MAUI scheduler [18]. Grothklags and Streit [19] also use a MILP formulation to solve the parallel job scheduling problem, however their formulation is based on starting times of jobs, while we focus on job ordering. While the approach by Grothklags and Streit needs variables and linear constraints in the number of time-slots, the sizes of the variable set and the constraint set of our approach are independent of time. Furthermore, they use time-scaling to tackle the problem of large variable and constraint sets and calculate schedules on a scale of one minute rather than of one second. In contrast, our approach is immune towards this problem, since the order of jobs is not depending on actual time slots.

A few works have focused on the analysis of the user behavior in parallel computing. Feitelson [20] investigated correlations between system performance (in terms of response times) and the subsequent job submission behavior. Results of this work have driven, for example, the modeling of the user behavior [21], and the quality and modeling of runtime estimates [22]–[25]. However, these works derive their knowledge and assumptions of users, their behavior, and their needs from workload traces, while we use results from a survey [7], [8] to define objective functions concerning user satisfaction.

III. Planning Horizon

We introduce planning horizons to improve job scheduling and tackle the online aspect of job submission. A *planning horizon* is a time slot of constant, or dynamic length, in which a set of jobs is scheduled. Figure 1 shows the process of job scheduling using planning horizons. We use the MILP approach described in Section IV for scheduling the jobs in the queue $Q_i, i \geq 0$. The resulting schedule $S_i, i \geq 0$ is modified online (utilizing EASY backfilling) if and only if an incoming job fits into the existing schedule. Otherwise, the job is deferred to the next planning horizon, and therefore added to the queue $Q_{i+1}, i \geq 0$ that collects the jobs during execution of S_i.

Planning horizons are beneficial for the proposed method of applying MILP-techniques to parallel job scheduling. Due to the fact that solving MILPs is generally a runtime-intensive process, we do not reschedule queued jobs at every event of job arrivals, job completion (or preemption), or job starts, but only on certain, pre-defined moments in time. This is advantageous in parallel computing systems with a large number of jobs. The use of planning horizons reduces the complexity of dealing with the online arrival of jobs, and keep the system controllable by dividing the online scheduling problem into smaller subproblems. The length of a planning horizon might be adapted to specific application requirements.

Furthermore, if we re-arrange queued jobs and do not compute them in an order significantly influenced by the arrival time (in the basic case, in a first-come-first-serve order), we have to be aware of starvation, i.e., a job never starts processing because other jobs have higher priority. This situation cannot occur when considering planning horizons—every job is processed within its horizon, and jobs cannot run at any other moment outside its horizon.

IV. MILP for Parallel Job Scheduling on Parallel Machines

In this section, we first classify the theoretical complexity of the addressed scheduling problems. Then, we introduce an MILP-formulation as a flexible optimization framework for linear objectives.

A. Defining the complexity of the scheduling approach

We use the 3-field notation, which was introduced by Graham et al. [26]. Our analysis is based on the theoretical concept of parallel machines P_m and minimizing the makespan $C_{\max}$ (the turnaround time to complete the experiment execution, Eq. 1), which is known to be NP-hard [27]. In parallel job scheduling, additional constraints on the size of jobs m_j have to be met, as well as that jobs arrive online (Eqs. 2 and 3). However, these constraints do not weaken NP-hardness. The steps described in Section III for considering independent schedules then is, that we solve independent problems for some $t \in T$ (Eqs. 4 and 5). Therefore, the complexity of the addressed problem is NP-hard, since all objectives considered in this paper are stronger than $C_{\max}$. Furthermore, we introduce

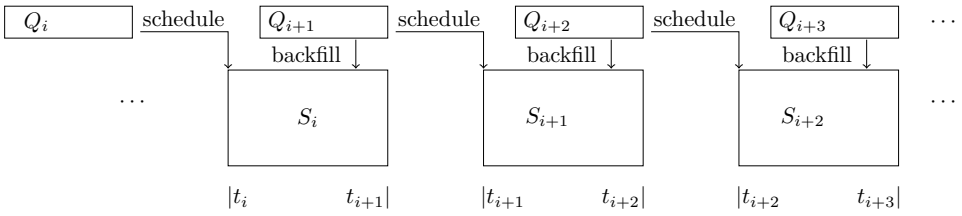

Fig. 1: Planning horizons divide the online job submissions into queues, which are used to backfill a current schedule until all jobs of the current schedule are processed. Meanwhile, new schedule is calculated from the queued jobs. $Q_i, i \geq 0$ is the queue of jobs, which are scheduled in schedule S_i at time t_i. Execution lasts from time t_i to $t_{i+1}, i \geq 0$. Jobs arriving between t_i to t_{i+1} are either allocated by the EASY backfilling strategy or, if they cannot be processed before t_{i+1}, queued in Q_{i+1}.

due-dates d_j, which we consider to reflect user satisfaction—if a due date is met, then it is satisfactory.

$$P_m \mid\mid C_{\max} \qquad \text{(NP-hard)} \tag{1}$$
$$\Big\downarrow \text{size}_j, \text{ online} \tag{2}$$
$$P_m \mid \text{size}_j, \text{online} \mid C_{\max} \tag{3}$$
$$\Big\downarrow \text{"offline":} \quad t \subset T \tag{4}$$
$$P_m \mid \text{size}_j \mid C_{\max} \tag{5}$$

Since the problem is NP-hard and we target optimal solutions, we choose a MILP (mixed-integer linear programming) for solving the targeted parallel job scheduling problem. We then assume that there is no additional abstraction layer, such as a meta-scheduler. Consequently, we neglect any kind of queuing and scheduling policies, which results in no user restrictions, and no sub-queues. We also assume that, once a job has been submitted, there is no further interaction between the users and the system. Although the approach allows in principle job removal from the queue, or the preemption of jobs at runtime, this is not considered in the evaluation (see Section V). Additionally, the proposed scheduling system does not give any guarantees. Hence, there are no service level agreements, or any guarantee that the job will complete within a deadline.

B. Mixed-integer linear programming formulation

The MILP requires a set of input variables described as follows:

$$p_j : \text{(requested) processing time of job } j, \tag{6}$$
$$m_j : \text{size of job } j, \tag{7}$$
$$w'_j : \text{previous waiting time of job } j, \tag{8}$$
$$a_i : \text{availability of resource i.} \tag{9}$$

Additionally, we define a set of jobs as J, and a set of resources as M.

The MILP requires two sets of binary decision variables, and one integer decision variable for the optimization process. The value ranges ensure that only a single job is executed per resource at the same time (i.e., there is no concurrency), and that there is a strict order between jobs allocated to a particular resource:

$$x_{ij} \in \{0,1\} : \text{indicator if job } j \text{ is processed on resource } i \tag{10}$$
$$y_{jj'} \in \{0,1\} : \text{indicator if job } j \text{ is processed before job } j' \tag{11}$$
$$t_j \geq 0 : \text{start time of job } j \tag{12}$$
$$Q \geq \max\{M\} + \sum_{i \in J} p_j + 1 : \text{arbitrarily large value} \tag{13}$$

The following set of constraints (Eqs. 14–19) represent the MILP formulation of the $P_m \mid \text{size}_j, d_j \mid \min \sum_j f(.)$ scheduling problem. The MILP is flexible since it is applicable to any linear optimization function $f(.)$ on the given variables:

$$\min \qquad \sum_{j \in J} f(.) \tag{14}$$
$$s.t. \qquad \sum_{i=1}^{M} x_{ij} = \text{m}_j \qquad \forall j \tag{15}$$
$$x_{ij} + x_{ij'} - 1 \leq y_{jj'} + y_{j'j} \forall i, j, j' \quad j > j' \tag{16}$$
$$y_{jj'} + y_{j'j} \leq 1 \qquad \forall j, j' \quad j \neq j' \tag{17}$$
$$t_{j'} + (1 - y_{jj'}) \cdot Q \geq t_j + p_j \qquad \forall j, j' \quad j \neq j' \tag{18}$$
$$t_j + (1 - x_{ij}) \cdot Q \geq \text{a}_i \qquad \forall j, i \tag{19}$$

Equation 15 denotes the size constraint of the jobs. This means that a job j must be executed on exactly $size_j$ resources in parallel. If two jobs run on the same resource, they must be executed consecutively (Eq. 16), and exclusively either j' before j or vice versa (Eq. 17). Non-overlapping (Eq. 18) adds a time constraint by guaranteeing that only a single job is executed per resource at an instant of time. Finally, the starting time of job j must respect the availability of the resource (Eq. 19).

V. Evaluation

In this section, we evaluate the practicability of the proposed MILP formulation. We first define two user-based optimization goals related to waiting times, i.e., the tardiness and the

number of late jobs, as well as the makespan, and compare them to results obtained with the EASY scheduling with conservative backfilling. EASY follows a FCFS approach with backfilling—jobs located farther in the queue may run before their predecessors if they do not delay the execution of any previous job. We derive *scenarios* from production platform workload traces, which are used as input for the MILP. In this evaluation, we ignore feedback effects such as think time and the online character of this scheduling problem, since they add significantly complexity and uncertainty to the user submission behavior. Since our approach uses planing horizon, this assumption does not weaken our evaluation. The schedulers face static scenarios, which are independent of online submission behavior.

Both schedulers, EASY and MILP, are based on runtime predictions. Whenever a runtime estimate is poor and a job completes significantly in advance to its estimated completion time, backfilling strategies are triggered to fill this gap. Therefore, this approach mitigates flaws in runtime estimates provided by users (which is typical in production systems).

A. Satisfaction

In this paper, we target the optimization of user-based waiting times satisfaction. Therefore, we use data gathered in a user-based survey conducted by means of the *Questionnaire on User Habits in Compute Clusters* (QUHCC). We collected data to determine to which extent users are whiling to accept waiting and response times for different job lengths [7], [9]. The questionnaire is composed of six questions on acceptable waiting times for different job sizes ranging from *interactive* and *short* jobs, which are defined as runs up to ten minutes; up to *large* jobs, which may run for several days. The surveyed data was collected from responses of 23 users of different clusters at TU Dortmund University, Germany. We assume that the results drawn from the collected data is to some extent representative for users of parallel computing infrastructures located at institutional facilities (e.g., campus clusters).

There can be several further aspects influencing user satisfaction, such as whether the system sizes suit users' needs, if they find that resources are fairly shared among researchers, among others. Nevertheless, waiting times are a measurable and negotiable component of parallel job processing, and therefore is the scope of this paper.

Figure 2 shows the median, .25-quantile, and .75-quantile values of acceptable response times for the provided answers. The *x*-axis represents job lengths of six job length categories of QUHCC in minutes. Acceptable response times are shown on the *y*-axis (also in minutes), where response time of a job j is defined as follows:

$$r_j = w_j + p_j, \tag{20}$$

where w_j is the waiting time, p_j the processing time. The increase on response time acceptance is of linear fashion (shown in a regression analysis). For all three datasets (median, .25-quantile, and .75-quantile), we perform a linear regression

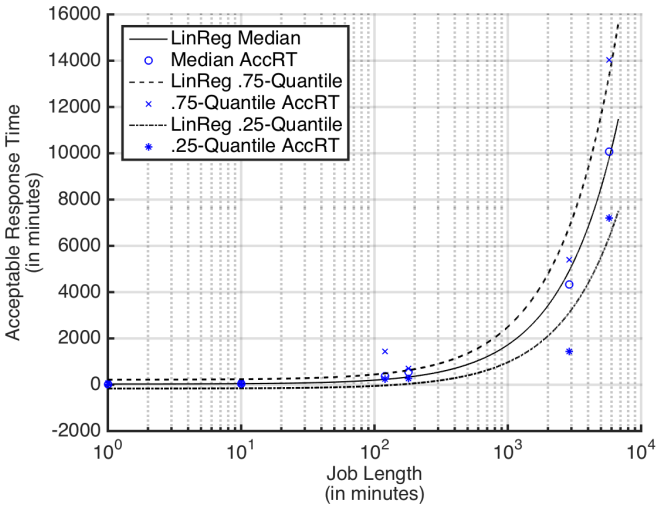

Fig. 2: Median, quantiles (.25 and .75), and linear regression of the acceptable response times for different job lengths (in terms of runtime). The increase on response time acceptance is linear.

TABLE I: Linear regression function parameters. (AccWT is the *acceptable waiting time*).

Data Set	c_1	c_2 (in min)	RMS (in min)
Median AccRT	1.70	29.51	286.00
.75-Quantile AccRT	2.28	215.75	746.11
.25-Quantile AccRT	1.14	-166.19	778.56

analysis minimizing least squares. The resulting functions are also plotted in Figure 2. The linear regression functions $r(p_j)$ are of the form:

$$r(p_j) := c_1 \cdot p_j + c_2. \tag{21}$$

The values of c_1 and $c2$ are listed in Table I, as well as the root mean square (RMS).

Based on the linear functions, we define an acceptable slowdown $S(p_j) \geq 1$, which describes a factor of how much greater the response time of a job j can be compared to its processing time p_j:

$$S(p_j) := \max \left\{ \frac{(c_1 \cdot p_j + c_2) - p_j}{p_j}, 1 \right\}. \tag{22}$$

Note that for the median and .25-quantile regression, c_2 is negative. When rating acceptability of waiting times according to the regression functions, one has to be aware of this relation, and therefore consider the maximum between the slowdown implied by the linear regression and one.

Optimization goals. In the following, we use two related optimization goals from the scheduling theory, i.e., $\sum_{j \in J} T_j$ (tardiness) and $\sum_{j \in J} U_j$ (unified lateness) [27], and apply

them to the targeted problem of increasing user satisfaction. *Tardiness* is the time span from a due date of a job until it is processed, while *unified lateness* describes whether a job can complete before its due date. Both objectives lead our simulation setup (Eqs. 1–5) to an NP-hard problem, since tardiness and unified lateness are more complex than $C_{\max}$ in parallel scheduling [27]. Both objective functions reasonably seek to exploit the findings on acceptable waiting times as possible minimization objectives for parallel job schedules. Thus, we derive the following two linear objective functions:

$$U_j(p_j) := \min\Big(1, \max\big(0, r_j - S(p_j)\cdot p_j\big)\Big) \quad (23)$$

$$T_j(p_j) := \max\Big(0, r_j - S(p_j)\cdot p_j\Big) \quad (24)$$

Since functions respect the waiting time of jobs, the idea is to distribute waiting times among all users.

B. Experimental Scenarios

Beside the objective functions, we also need realistic data to evaluate the performance and quality of the MILP scheduling approach in production environments. We use data from real workload traces, covering different system sizes and different temporal spaces (recorded in 1996 and 2014). Table II shows the main characteristics of the two traces used in this paper (KTH and MIRA), which include their names, a short handle, the system sizes (in terms of number of nodes), the number of individual users, the number of recorded jobs, and the duration covered in the trace. The KTH trace was obtained from the Parallel Workloads Archive (PWA) [10]. We arbitrarily chose the KTH trace from PWA, since it has been used in several studies in the past decades[1].

To evaluate the planning horizon approach (Section III), we derive scenarios from the workload traces. A scenario contains a queue status Q and a resource status M of trace l at an instant of time t (Eq. 25). At instant t, queue Q contains a set of jobs J, which have been submitted and have not started processing (Eq. 26). The resource status M describes the number of resources m_j that are allocated for a job j at t, and for how long the job will consume the resource c_j (Eq. 27).

$$\text{scenario}_{l,t} := \{Q, M\}, \quad (25)$$

$$Q := \{j \mid s_j \geq t \wedge w_j \leq s_j - t\}, \quad (26)$$

$$M := \{(m_j, c_j) \mid s_j \leq t \wedge c_j \geq t \wedge j \in J\} \quad (27)$$

C. Experimental Results and Discussion

First, we focus on minimizing the number of late jobs according to the unified lateness objective function defined in Eq. 23. We create 20 scenarios at arbitrary points in time t from the MIRA trace, and then we set the time limit to 10 minutes for the execution of the MILP. In order to decrease the complexity of the solution, and lead to more comprehensible results, the evaluation neglects any queueing, scheduling, or other policies, which were present during trace recording. The distribution of queue sizes in both KTH and MIRA are shown in Figure 3.

[1] http://www.cs.huji.ac.il/labs/parallel/workload/l_kth_sp2/index.html

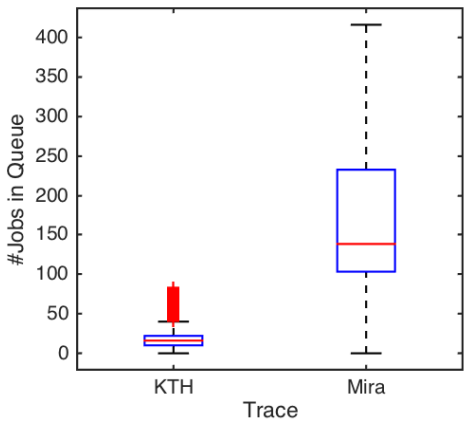

Fig. 3: Distributions of job queue sizes for the KTH and MIRA workload traces.

For the sake of simplicity, we do not consider jobs that will trespass their due date independently of any schedule, i.e., $t_j + w'_j > (S(p_j) - 1)\cdot p_j$. Thus, we filter these jobs in a preprocessing step (data preparation). We choose the median acceptance function and the .75-quantile function from Table I. We choose the median as an average value, which is robust against outliers. In situations of high queue saturation, this value cannot be satisfied. Therefore, we also choose the .75-quantile to represent less satisfaction but still concerning the answers provided in the survey. Additionally, we compare the results of both the EASY with conservative backfilling and our proposed MILP approach.

Figure 4 shows the scheduling results for the MIRA trace, as well as runtimes and the gap between best integer and relaxation solution of the MILP. Figure 4a shows the differences in the number of late jobs between the solution obtained from EASY and MILP when applying either the median or the .75-quantile function, respectively. Note that the obtained values are always positive (including zero), since the MILP always finds a solution, which is better (or at least not worse) than the EASY solution. This observation holds even when the runtime limit of ten minutes is reached (Figure 4b). Overall, the .75-quantile function allows the MILP to find more jobs, which can complete before their due date, whereas EASY cannot find suitable solutions due to its underlying FCFS strategy. Figure 4c shows that 80% of the scenarios can be processed optimally within the 10-min threshold time limit.

Figure 5 shows the results of optimizing the tardiness objective function defined in Eq. 24. Since we do not target the optimization of the unified lateness focusing only on the number of jobs that could be processed before their due date, but the sum of individual latenesses, we also report the $C_{\max}$-values of each schedule (Figure 5b). In this scenario, the runtime was limited to one hour since in practical application,

TABLE II: Characteristics of the real workload traces used in this paper.

Tracename	Handle	Year	System Size (#nodes)	#Users	#Jobs	Duration
KTH-SP2-1996-2.1-cln	KTH	1996	100	214	28,476	340 days
MIRA-2014	MIRA	2014	49,152	487	78,782	409 days

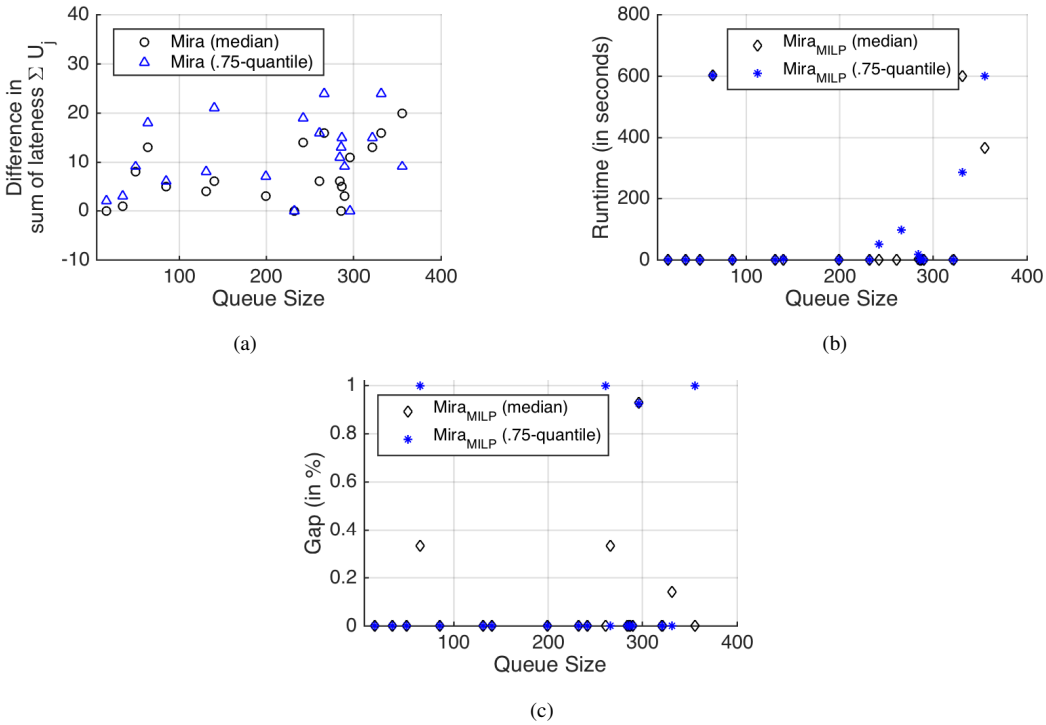

Fig. 4: Results of scheduling scenarios taken from the MIRA trace with preprocessing: (a) difference between results for EASY and MILP for $\sum U_j$ for median and .75-quantile satisfaction function, (b) runtimes, and (c) gap between best integer and relaxation solution.

we want to obtain a schedule within a realistic time frame. For both optimization functions, we observe that MILP outperforms EASY in most cases (Figure 5a). For three scenarios (queue sizes 16 (2x) and 30) the solution obtained by EASY is smaller than the solution obtained by MILP. For the median satisfaction function, the mean improvement is $\mu = 20.6\%$ ($\sigma = 20.3\%$), and for the .75-quantile function the mean improvement is $\mu = 32.2\%$ ($\sigma = 29.0\%$).

Although we reach the runtime limits for queue sizes of seven and nine already (Figure 5c), and subsequently also experience a gap $> 0\%$ (Figure 5d), for every scenario considered, MILP leads to better results than EASY. The MILP is able to find better schedules because it does not need to schedule jobs in FCFS order but can re-arrange jobs. Furthermore, the makespan $C_{\max}$ is smaller for all schedules up to queue sizes of about 16 jobs, although this is not the primary goal of the considered optimization functions. Since we experience a gap of more than about 20%, longer runtimes, or a deeper understanding and improvement of the considered scheduling approach will aid to improve the results.

Hence, we argue that we can distribute waiting times *better* among all jobs currently subject to schedule by introducing planning horizons and applying the MILP approach proposed in this paper under the assumptions discussed.

All experiments were conducted on five cores of an Intel Xeon CPU E5-2660 v3 @2,60Ghz with a limit of 64GB of RAM, running Windows Server 2012 and CPLEX Enterprise Server Version 12.6.1. The time limit was set to 10 and 120 minutes, respectively.

VI. Conclusion

In this paper, we have presented a way to explore MILP techniques for parallel job scheduling in parallel computing to increase user satisfaction by meeting job deadlines. Therefore, we have defined planning horizons and scenarios. The main results include:

1) The definition of planning horizons in the online parallel job scheduling problem, e.g., to exploit methods from discrete optimization;
2) An approach to optimize user satisfaction in parallel job scheduling; and
3) A MILP formulation capable of optimizing job queues of up to 350 jobs optimizing a $\sum U_j$ constraint (with preprocessing), and up to 30 jobs for a $\sum T_j$ constraint.

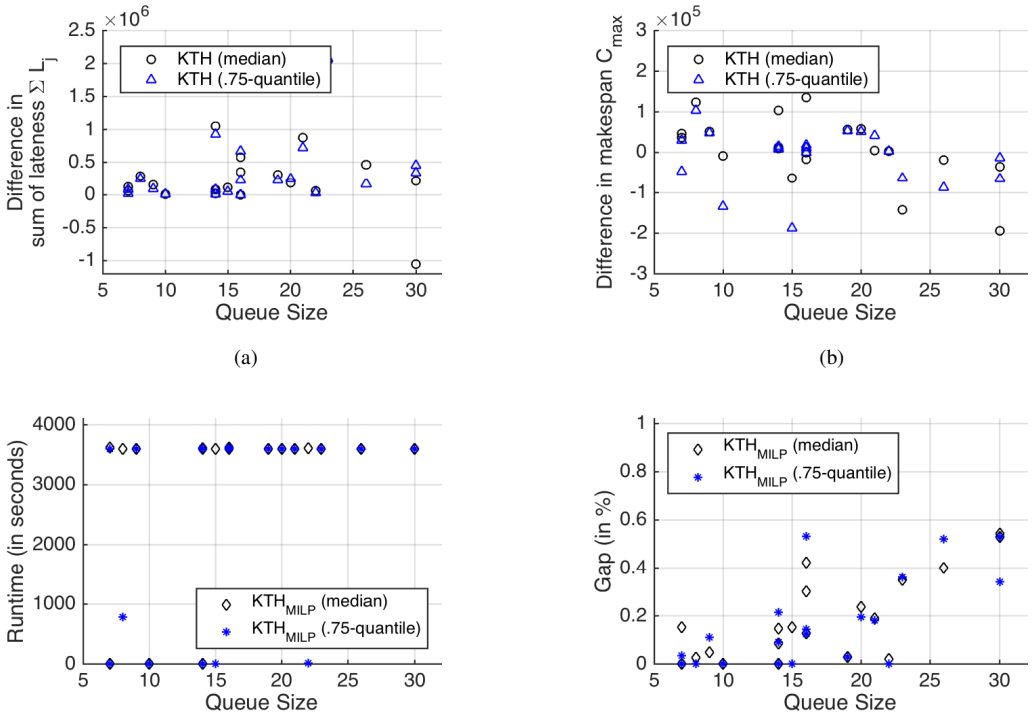

Fig. 5: Results of scheduling scenarios taken from the KTH trace: (a) Differences between results from EASY and MILP regarding the sum of latenesses $\sum L_j$, (b) differences between results from EASY and MILP regarding the makespan $\sum C_{\max}$, (c) runtimes, (d) gap between best integer and relaxation solution.

While we have demonstrated the practicability of a MILP-based scheduling technique for parallel job scheduling in the form of planning horizons, future work should address the problem of exploiting our findings. It is still necessary to apply the MILP to the consecutive scheduling of scenarios described in this work. Furthermore, future work should also consider the feedback effect between the users and the parallel computing environments. These studies might also include the influence of user-related inaccuracy in runtime estimation and further sources of uncertainty.

Acknowledgement

This work was partly funded by DOE under the contract number ER26110, "dV/dt – Accelerating the Rate of Progress Towards Extreme Scale Collaborative Science". We thank Bill Allcock from the Argonne National Laboratory (ANL) for making the traces of the Mira supercomputer available, and for the detailed information and insightful discussions.

References

[1] M. AbdelBaky, R. Tavakoli, M. F. Wheeler, M. Parashar, H. Kim, K. E. Jordan, V. Sachdeva, J. Sexton, H. Jamjoom, Z.-Y. Shae *et al.*, "Enabling high-performance computing as a service," *Computer*, no. 10, pp. 72–80, 2012.

[2] D. A. Reed and J. Dongarra, "Exascale computing and big data," *Communications of the ACM*, vol. 58, no. 7, pp. 56–68, 2015.

[3] A. Geist and D. A. Reed, "A survey of high-performance computing scaling challenges," *International Journal of High Performance Computing Applications*, p. 1094342015597083, 2015.

[4] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[5] E. Shmueli and D. G. Feitelson, "On simulation and design of parallel-systems schedulers: are we doing the right thing?" *Parallel and Distributed Systems, IEEE Transactions on*, vol. 20, no. 7, pp. 983–996, 2009.

[6] N. Zakay and D. G. Feitelson, "On identifying user session boundaries in parallel workload logs," in *Job Scheduling Strategies for Parallel Processing*. Springer, 2012, pp. 216–234.

[7] J. Renker, S. Schlagkamp, and G. Rinkenauer, "Questionnaire for user habits of compute clusters (QUHCC)," in *HCI International 2015-Posters' Extended Abstracts*. Springer, 2015, pp. 697–702.

[8] S. Schlagkamp, R. Ferreira da Silva, J. Renker, and G. Rinkenauer, "Analyzing users in parallel computing: A user-oriented study," in *14th International Conference on High Performance Computing & Simulation (HPCS)*, 2016.

[9] S. Schlagkamp and J. Renker, "Acceptance of waiting times in high performance computing," in *HCI International 2015-Posters' Extended Abstracts*. Springer, 2015, pp. 709–714.

[10] "Parallel workloads archive," http://www.cs.huji.ac.il/labs/parallel/workload.

[11] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[12] D. G. Feitelson, *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

[13] S. Schlagkamp, R. Ferreira da Silva, W. Allcock, E. Deelman, and U. Schwiegelshohn, "Consecutive job submission behavior at Mira supercomputer," in *ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, 2016.

[14] S. Schlagkamp, R. Ferreira da Silva, E. Deelman, and U. Schwiegelshohn, "Understanding user behavior: from HPC to HTC," in *International Conference on Computational Science (ICCS)*, 2016.

[15] U. Schwiegelshohn, "How to design a job scheduling algorithm," in

Job Scheduling Strategies for Parallel Processing. Springer, 2014, pp. 147–167.

[16] D. A. Lifka, "The anl/ibm sp scheduling system," in *Job Scheduling Strategies for Parallel Processing*. Springer, 1995, pp. 295–303.

[17] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan, "Characterization of backfilling strategies for parallel job scheduling," in *Parallel Processing Workshops, 2002. Proceedings. International Conference on*. IEEE, 2002, pp. 514–519.

[18] D. Jackson, Q. Snell, and M. Clement, "Core algorithms of the maui scheduler," in *Job Scheduling Strategies for Parallel Processing*. Springer, 2001, pp. 87–102.

[19] S. Grothklags and A. Streit, "On the comparison of cplex-computed job schedules with the self-tuning dynp job scheduler," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004.*, April 2004.

[20] D. G. Feitelson, "Looking at data," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*. IEEE, 2008, pp. 1–9.

[21] S. Schlagkamp, "Influence of dynamic think times on parallel job scheduler performances in generative simulations," in *JSSPP 2015 - 19th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2015)*, Hyderabad, India, May 2015.

[22] D. Tsafrir, Y. Etsion, and D. G. Feitelson, "Modeling user runtime estimates," in *JSSPP*, vol. 5. Springer, 2005, pp. 1–35.

[23] C. B. Lee and A. Snavely, "On the user–scheduler dialogue: studies of user-provided runtime estimates and utility functions," *IJHPCA*, vol. 20, no. 4, 2006.

[24] R. Ferreira da Silva, G. Juve, E. Deelman, T. Glatard, F. Desprez, D. Thain, B. Tovar, and M. Livny, "Toward fine-grained online task characteristics estimation in scientific workflows," in *8th Workshop on Workflows in Support of Large-Scale Science*, ser. WORKS '13, 2013, pp. 58–67.

[25] R. Ferreira da Silva, G. Juve, M. Rynge, E. Deelman, and M. Livny, "Online task resource consumption prediction for scientific workflows," *Parallel Processing Letters*, vol. 25, no. 3, 2015.

[26] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of discrete mathematics*, vol. 5, pp. 287–326, 1979.

[27] M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2008.