Dynamic Resource Management Heuristics for Minimizing Makespan while Maintaining an Acceptable Level of Robustness in an Uncertain Environment

Ashish M. Mehta*, Jay Smith†, H. J. Siegel*‡, Anthony A. Maciejewski*, Arun Jayaseelan*, and Bin Ye*

*Electrical and Computer Engineering Department
‡Computer Science Department
Colorado State University, Fort Collins, CO 80523–1373
Email: {ammehta, hj, aam, arunj}@engr.colostate.edu, binye@simla.colostate.edu
†IBM 6300 Diagonal Highway Boulder, CO 80301
Email: bigfun@us.ibm.com

Abstract

Heterogeneous parallel and distributed computing systems may operate in an environment where certain system performance features degrade due to unpredictable circumstances. Robustness can be defined as the degree to which a system can function correctly in the presence of parameter values different from those assumed. This paper presents a mathematical model for quantifying robustness in a dynamic environment where task execution times estimates are known to contain errors. This research proposes, evaluates, and compares ten different dynamic heuristics for their ability to maintain or maximize the proposed dynamic robustness metric in an uncertain environment. In addition, the makespan results of the proposed heuristics are compared to a lower bound.

1. Introduction

Heterogeneous parallel and distributed computing is the coordinated use of various compute resources of different capabilities to optimize certain system performance features. An important research problem (i.e., resource management) is how to determine a resource allocation and scheduling of tasks to machines (i.e., a mapping) that optimizes a system performance feature while maintaining an acceptable level of quality of service. This research focuses on a dynamic mapping environment where task arrival times are not known *a priori*. A mapping environment is considered dynamic when tasks are mapped as they arrive, e.g., in an on-line fashion [20]. The general problem of optimally mapping tasks to machines (resource management) in heterogeneous parallel and distributed computing environments has been shown in general to be NP-complete (e.g., [10, 14, 16]). Thus, the development of heuristic techniques to find a near-optimal solution for the mapping problem is an active area of research (e.g., [1,4,5,13,15,18,20,21,26]).

Dynamic mapping heuristics can be grouped into two categories: immediate mode and batch mode [20]. In immediate mode, when a task arrives (i.e., a mapping event) it is immediately mapped to some machine in the suite for execution. In batch mode, tasks are accumulated until a specified condition is satisfied (e.g., a certain number of tasks have accumulated, or some amount of time has elapsed); whereupon the entire batch of accumulated tasks and the previously enqueued but not executing tasks are considered for mapping. A pseudo-batch mode can be defined where the batch of tasks considered for mapping is determined upon the arrival of a new task (i.e., a mapping event) and consisting of all tasks in the parallel and distributed system that have not yet begun execution on some machine. Both immediate mode and pseudo-batch mode heuristics were considered for this research.

Heterogeneous parallel and distributed systems may operate in an environment where certain system performance features degrade due to unpredictable circumstances and inaccuracies in estimated system parameters. Robustness is defined as the degree to which a system can function correctly in the presence of parameters different from those assumed [2]. For a given set of tasks, the makespan is defined as the completion time for the entire set of tasks. For this research, makespan is required to be robust against errors in the estimated execution time of each task. For a given

*This research was supported by the DARPA Information Exploitation Office under contract No. NBCHC030137, by the Colorado State University Center for Robustness in Computer Systems (funded by the Colorado Commission on Higher Education Technology Advancement Group through the Colorado Institute of Technology), and by the Colorado State University George T. Abell Endowment. Approved for public release, distribution unlimited.

application domain, the estimated time to compute (ETC) each task i on each machine j is assumed known, denoted ETC(i, j). However, these estimates may deviate from the actual computation times; e.g., the actual times may depend on characteristics of the input data to be processed. The tasks considered in this research are taken from a frequently executed predefined set, such as exists in a lab or business environment. This research focuses on determining a dynamic mapping for a set of tasks that minimizes the predicted makespan (using the provided ETC values) while still being able to tolerate a quantifiable amount of variation in the ETC values of the mapped tasks. Hence, the goal is to obtain a mapping that has the minimum makespan and can still guarantee a certain level of robustness at each mapping event.

One of the areas where this work is directly applicable is the development of resource allocations in enterprise systems that support transactional workloads sensitive to response time constraints, e.g., time sensitive business processes [22]. Often, the service provider in these types of systems is contractually bound through a service level agreement to deliver on promised performance. The dynamic robustness metric can be used to measure a resource allocation's ability to deliver on a performance agreement.

The contributions of this paper include:

- a model for quantifying dynamic (as opposed to static) robustness in this environment,
- heuristics for solving the above resource management problem,
- simulation results for the proposed heuristics, and
- an lower bound on the total makespan for the resource management problem.

The remainder of the paper is organized as follows. Section 2 formally states the problem statement for this research. Section 3 briefly discusses the heuristics studied in this research including the definition of a lower bound on the total makespan of the mapping problem. Section 4 outlines the simulation setup. The simulation results are presented and discussed in Section 5. The related work is considered in Section 6 and Section 7 concludes the paper.

2. Problem Statement

In this study, T independent tasks (i.e., there is no inter-task communication) arrive at a mapper dynamically, where the arrival times of the individual tasks are not known in advance. For example - these independent tasks may have been generated by different users. Arriving tasks are each mapped to one machine in the set of M machines that comprise the heterogeneous computing system. Each machine is assumed to execute a single task at a time (i.e., no multitasking). In this environment, the robustness of a resource allocation must be determined at every mapping event—recall that a mapping event occurs when a new task arrives to the system. Let $T(t)$ be the set of tasks either currently executing or pending execution on any machine at time t, i.e., $T(t)$ does not include tasks that have already completed execution. Let $F_j(t)$ be the predicted finishing time of machine j for a given resource allocation μ based on the given ETC values. Let $MQ_j(t)$ denote the subset of $T(t)$ previously mapped to machine j's queue and let $scet_j(t)$ denote the starting time of the currently executing task on machine j. Mathematically, given some machine j

$$F_j(t) = scet_j(t) + \sum_{\forall i \in MQ_j(t)} ETC(i, j). \quad (1)$$

Let $\beta(t)$ denote the maximum of the finishing times $F_j(t)$ for all machines at time t—i.e., the predicted makespan at time t. Mathematically,

$$\beta(t) = \max_{\forall j \in [1,M]} \{F_j(t)\}. \quad (2)$$

The robustness metric for this work has been derived using the procedure defined in [2]. In our current study, given uncertainties in the ETC values, a resource allocation is considered robust if, at a mapping event, the *actual* makespan is no more than τ seconds greater than the *predicted* makespan. Thus, given a resource allocation μ, the robustness radius $r_\mu(F_j(t))$ of machine j can be quantitatively defined as the maximum collective error in the estimated task computation times that can occur where the actual makespan will be within τ time units of the predicted makespan. Mathematically, building on a result in [2],

$$r_\mu(F_j(t)) = \frac{\tau + \beta(t) - F_j(t)}{\sqrt{|MQ_j(t)|}}. \quad (3)$$

The robustness metric $\rho_\mu(t)$ for a given mapping μ is simply the minimum of the robustness radii over all machines [2]. Mathematically,

$$\rho_\mu(t) = \min_{\forall j \in [1,M]} \{r_\mu(F_j(t))\}. \quad (4)$$

With the robustness metric defined in this way, $\rho_\mu(t)$ corresponds to the collective deviation from assumed circumstances (relevant ETC values) that the resource allocation can tolerate and still ensure that system performance will be acceptable (the actual makespan is within τ of the predicted).

To define the dynamic robustness metric as a constraint let α be the minimum acceptable robustness of a resource allocation at any mapping event; i.e., the constraint requires that the robustness metric at each mapping event be at least α. Thus, the goal of the heuristics in this research is to dynamically map incoming tasks to machines such that the

total makespan is minimized, while maintaining a robustness of at least α i.e., $\rho_\mu(t) \geq \alpha$ for all mapping events. The larger α is, the more robust the resource allocation is.

3. Heuristics

3.1. Overview

Five immediate mode and five pseudo-batch mode heuristics were studied for this research. For the task under consideration, a feasible machine is defined to be a machine that will satisfy the robustness constraint if the considered task is assigned to it. This subset of machines is referred to as the feasible set of machines.

3.2. Immediate Mode Heuristics

The following is a brief description of the immediate mode heuristics. Recall that in the immediate mode of heuristics, only the new incoming task is considered while making a mapping decision. Thus, the behavior of the heuristic is highly influenced by the order in which the tasks arrive.

3.2.1. Feasible Robustness Minimum Execution Time (FRMET). FRMET is based on the MET concept in [20, 27]. For each incoming task, FRMET first identifies the feasible set of machines. From the feasible set of machines the incoming task is assigned to its minimum *execution* time machine.

3.2.2. Feasible Robustness Minimum Completion Time (FRMCT). FRMCT is based on the MCT concept in [8, 20, 27]. For each incoming task, FRMCT first identifies the feasible set of machines for the incoming task. From the feasible set of machines the incoming task is assigned to its minimum *completion* time machine.

3.2.3. Feasible Robustness K-Percent Best (FRKPB). FRKPB is based on the KPB concept in [17, 20]. It tries to combine the aspects of both MET and MCT. FRKPB first finds the feasible set of machines for the newly arrived task. From this set, FRKPB identifies the *k*-percent feasible machines that have the smallest *execution* time for the task. The task is then assigned to the machine in the set with the minimum *completion* time for the task. For a given α the value of *k* was varied between 0 and 100, in steps of 12.5, for sample training data to determine the value that provided the minimum makespan. A value of $k = 50$ was found to give the best results.

3.2.4. Feasible Robustness Switching (FRSW). FRSW is based on the SW concept in [17, 20]. As applied in this research, FRSW combines aspects of both the FRMET and the FRMCT heuristics. A *load* balance ratio (LBR) is defined to be the ratio of the minimum number of tasks enqueued on any machine to the maximum number of tasks enqueued on any machine. The LBR can have any value in the interval[1,0]. Two threshold points T_{high} and T_{low} are chosen for the ratio LBR such that $T_{high} > T_{low}$. FRSW then switches between FRMET and FRMCT based on the value of the load balance ratio. The heuristic starts by mapping tasks using FRMCT. When the ratio raises above the set point T_{high} FRSW switches to the FRMET heuristic. When the ratio falls below T_{low} FRSW switches to the FRMCT heuristic. The values for the switching set points were determined experimentally using sample training data.

3.2.5. Maximum Robustness (MaxRobust). MaxRobust has been implemented for comparison only, trying to greedily maximize robustness without considering makespan. MaxRobust calculates the robustness radius of each machine for the newly arrived task, assigning the task to the machine with the maximum robustness radius.

3.3. Pseudo-Batch Heuristics

The pseudo-batch mode heuristics implement two sub-heuristics, one to map the task as it arrives, and a second to remap pending tasks. For the pseudo-batch mode heuristics, the initial mapping is performed by the previously described FRMCT heuristic (except for the MRMR heuristic). The remapping heuristics each operate on a set of mappable tasks; a mappable task is defined as any task pending execution that is not next in line to begin execution. To avoid idle machines in the system caused by waiting for a heuristic to complete a mapping event, a task that is next in line to begin execution on any machine will not be considered for mapping at a mapping event. While it is still possible that a machine may become idle, it is highly unlikely for the assumptions in this research (the average execution time of a task is 100 seconds while the average execution time of a mapping event is less than 0.6 seconds). The following is a brief description of the pseudo-batch mode re-mapping heuristics.

3.3.1. Feasible Robustness Minimum Completion Time-Minimum Completion Time (FMCTMCT). FMCTMCT uses a variant of Min-Min heuristic defined in [16]. For each mappable task, FMCTMCT finds the feasible set of machines, then from this set determines the machine that provides the minimum completion time for the task. From these task/machine pairs, the pair that gives the overall minimum completion time is selected and

that task is mapped onto that machine. This procedure is repeated until all of the mappable tasks have been remapped.

3.3.2. Feasible Robustness Maximum Robustness-Minimum Completion Time (FMRMCT). FMRMCT builds on concept of the Max-Min heuristic [16]. For each mappable task, FMRMCT first identifies the feasible set of machines, then from this set determines the machine that provides the minimum completion time. From these task/machine pairs, the pair that provides the maximum robustness radius is selected and the task is assigned to that machine. This procedure is repeated until all of the mappable tasks have been remapped.

3.3.3. Feasible Minimum Completion Time-Maximum Robustness (FMCTMR). For each mappable task, FMCTMR first identifies the feasible set of machines, then from this set determines the machine with the maximum robustness radius. From these task/machine pairs, the pair that provides the minimum completion time is selected and the task is mapped to that machine. This procedure is repeated until all of the mappable tasks have been remapped.

3.3.4. Maximum Weighted Sum-Maximum Weighted Sum (MWMW). MWMW builds on a concept in [24]. It combines the Lagrangian heuristic technique [9, 19] for deriving an objective function with the concept of Min-Min heuristic [16] here to simultaneously minimize makespan and maximize robustness. For each mappable task, the feasible set of machines is identified and the machine in this set that gives the maximum value of the objective function (defined below) is determined. From this collection of task/machine pairs, the pair that provides the maximum value of the objective function is selected and the corresponding assignment is made. This procedure is repeated until all of the mappable tasks have been remapped.

When considering assigning a task *i* to machine *j*, let $F'_j(t) = F_j(t) + \sum ETC(i,j)$ for all tasks currently in the machine queue and the task currently under consideration. Let $\beta'(t)$ be maximum of the finishing times $F'_j(t)$ at time t for all machines. Let $r'_\mu(F'_j(t))$ be the robustness radius for machine *j*. Let *maxrob(t)* be the maximum of the robustness radii at time *t*. Given η , an experimentally determined constant using training data, the objective function for MWMW is defined as

$$s(j,t) = \eta\left(1 - \frac{F'_j(t)}{\beta'(t)}\right) + (1-\eta)\left(\frac{r'_\mu(F'_j(t))}{maxrob(t)}\right) \quad (5)$$

3.3.5. Maximum Robustness-Maximum Robustness (MRMR). MRMR is provided here for comparison only as it optimizes robustness without considering makespan. As a task arrives it is initially mapped using the MaxRobust heuristic. Task remapping is performed by a variant of the Max-Max heuristic [16]. For each mappable task, the machine that provides the maximum robustness radius is determined. From these task/machine pairs, the pair that provides the maximum overall robustness radius is selected and the task is mapped to that machine. This procedure is then repeated until all of the mappable tasks have been remapped.

3.4. Lower Bound (lb)

A lower bound on makespan for the described system can be found by identifying the task whose arrival time plus minimum execution time on any machine is the greatest. More formally, given the entire set of tasks *S* where each task i has an arrival time of $arv(i)$, the lower bound is given by

$$\text{lb} = \max_{\forall i \in S}\left((arv(i) + \min_{\forall j \in [1,M]} ETC(i,j)\right). \quad (6)$$

This is a lower bound on makespan. Thus, no heuristic can achieve a smaller makespan. However, it is possible that this lower bound is not achievable even by an optimal mapping.

4. Simulation Setup

The simulated environment consists of *T* = 1024 independent tasks and *M* = 8 machines. This number of tasks and machines was chosen to present a significant mapping challenge for each heuristic and to prevent an exhaustive search for an optimal solution (however, our techniques can be applied to different numbers of tasks and machines). As stated earlier, each task arrives dynamically and arrival times are not known *a priori*. For this study, 100 different ETC matrices were generated, 50 with high task heterogeneity and high machine heterogeneity (HIHI) and 50 with low task heterogeneity and low machine heterogeneity (LOLO) ([8]). The LOLO ETC matrices model an environment where different tasks have similar execution times on a machine and also the machines have similar capabilities, e.g., a cluster of workstations employed to support transactional data processing. In contrast, the HIHI ETC matrices model an environment where the computational requirements of tasks vary greatly and there is a set of machines with diverse capabilities, e.g., a computational grid comprising of SMPs, workstations, and supercomputers.

All of the ETC matrices generated were inconsistent (i.e., machine A being faster than machine B for task 1 does not imply machine A is faster than machine B for task 2) [8].

All ETC matrices were generated using the gamma distribution method presented in [3]. The arrival time of each task was generated according to a Poisson distribution with a mean task inter-arrival rate of eight seconds.

In the gamma distribution method of [3], a mean task execution time and coefficient of variation (COV) are used to generate the ETC matrices. In the high-high case, the mean task execution time was set to 100 seconds and a COV of 0.9 was used for both the task and the machine heterogeneity. The low-low heterogeneity case uses a mean task execution time of 100 seconds and a COV of 0.3 for task heterogeneity and a COV of 0.3 for machine heterogeneity.

The value of τ chosen for this study was 120 seconds. The performance of each heuristic was studied across all 100 different trials (ETC matrices).

5. Results

In Figures 1 through 4, the average makespan results (with 95% confidence interval bars) are plotted, along with a lower bound on makespan. Figures 1 and 2 present the makespan results for the immediate mode heuristics for HIHI and LOLO heterogeneity, respectively. While, Figures 3 and 4 present the makespan results for the pseudo-batch mode heuristics for HIHI and LOLO heterogeneity, respectively. Each of the heuristics was simulated using multiple values for the robustness constraint α. For each α the performance of the heuristics was observed for 50 HIHI and 50 LOLO heterogeneity trials. In Figures 1 and 2, the number of failed trials (out of 50) is indicated above the makespan results for each heuristic, i.e., the number of trials for which the heuristic was unable to successfully find a mapping for every task given the robustness constraint α.

The average execution time of each heuristic over all mapping events (on a typical unloaded 3GHz Intel Pentium 4 desktop machine) in all 100 trials are shown in Table 1 and Table 2 for immediate and pseudo-batch mode, respectively. For the immediate mode heuristics, this is the average time for a heuristic to map an incoming task. For the pseudo-batch mode heuristics, this is the average time for a heuristic to map an entire batch of tasks.

For the immediate mode heuristics, FRMET resulted in the lowest makespan for HIHI, and FRMET and FRSW performed the best for LOLO. The immediate mode FRMET heuristic for both HIHI and LOLO heterogeneity performed better than anticipated given prior studies including a minimum execution time (MET) heuristic in other environment (that do no involve robustness and had different arrival rates and ETC matrices).

It has been shown, in general, that the minimum execution time heuristic is not a good choice for minimizing makespan for both the static and dynamic environments [8,20], because it ignores machine loads and machine available times when making a mapping decision. The establishment of a feasible set of machines by the FRMET heuristic indirectly balances the incoming task load across all of the machines (i.e., incorporates some sense of machine loads). Also, because of the highly inconsistent nature of the data sets coupled with the high mean execution time (100 seconds), FRMET is able to maintain a lower makespan compared to FRMCT.

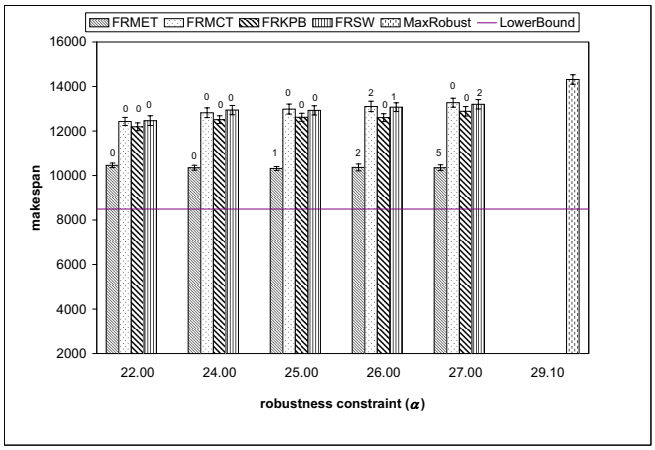

Figure 1. Simulation results of makespan for different values of robustness constraint (α) for immediate mode heuristics for HIHI heterogeneity.

Table 3 shows the maximum and average number of mapping events (out of a possible 1024) over successful trials (out of 50) for which the MET machine was not feasible. That is, the table values were calculated based on only the subset of the 50 trials for which FRMET could determine a mapping that met the constraint. For each of these trials, there were 1024 mapping events. Thus, even though the vast majority of tasks are mapped to their MET machine, it is important to prevent those rare cases where doing so would make the mapping infeasible.

The FRKPB heuristic performed better than FRMCT (in terms of makespan) for LOLO heterogeneity and comparable to FRMCT for HIHI heterogeneity. FRKPB selects the *k*-percent feasible machines that have the smallest *execution* time for the task and then assigns the task to the machine in the set with the minimum *completion* time for the task. Thus, rather then trying to map the task to its best completion time machine, it tries to avoid putting the current task onto the machine which might be more suitable for some yet to arrive task. This foresight about task heterogeneity is missing in FRMCT, which might assign the task to a poorly matched machine for an immediate marginal improvement in completion time. This might possibly deprive some subsequently arriving better matched tasks of that machine, and

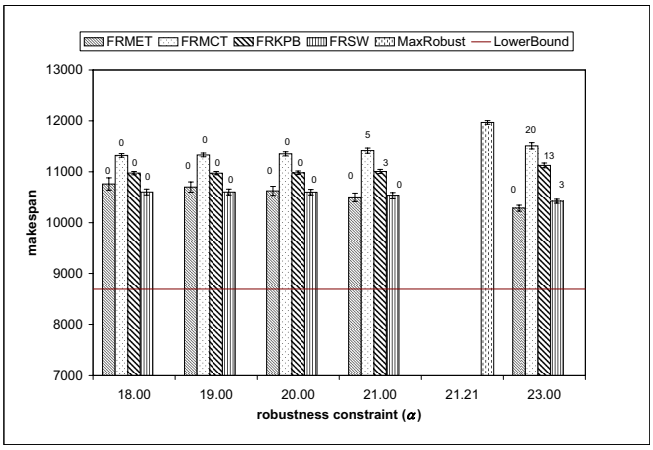

Figure 2. Simulation results of makespan for different values of robustness constraint (α) for immediate mode heuristics for LOLO heterogeneity.

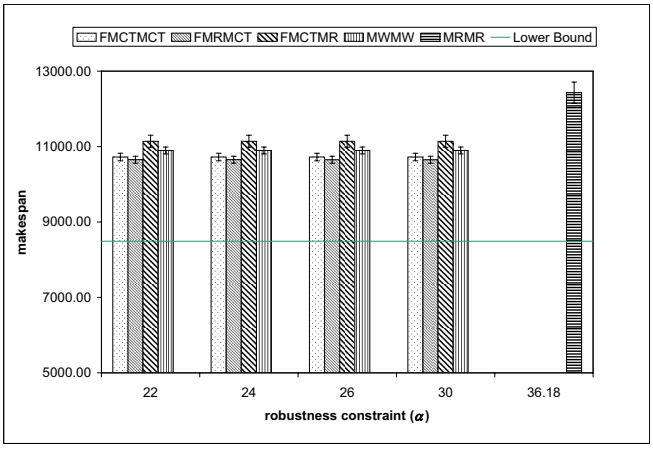

Figure 3. Simulation results of makespan for different values of robustness constraint (α) for pseudo-batch mode heuristics for HIHI heterogeneity.

Table 1. Average execution times, in seconds, of a mapping event for the proposed immediate mode heuristics.

heuristic	average execution time (sec.)
FRMET	0.001
FRMCT	0.0019
FRKPB	0.0019
FRSW	0.0015
MaxRobust	0.0059

Table 2. Average execution times, in seconds, of a mapping event for the proposed pseudo-batch mode heuristics.

heuristic	average execution time (sec.)
FMCTMCT	0.023
FMRMCT	0.028
FMCTMR	0.028
MWMW	0.0211
MRMR	0.0563

eventually leading to a larger makespan than FRKPB.

An interesting observation was that the FRMCT heuristic was able to mantain a robustness constraint of $\alpha = 27$ for the 50 trials used in this study, but only for 48 trials for $\alpha = 26$ (for HIHI heterogeneity). This could be attributed to the volatile nature of the greedy heuristics. The looser robustness constraint ($\alpha = 26$) allowed for a paring of task to machine that was disallowed for a tighter robustness constraint ($\alpha = 27$). That is, the early greedy selection proved to be a poor decision because it ultimately led to a mapping failure.

The MWMW heuristic used a value of $\eta = 0.7$ for HIHI and $\eta = 0.6$ for LOLO. Among the pseudo-batch mode heuristics, for the HIHI heterogeneity trials, FMRMCT performed the best on average, while FMCTMCT gave comparable results. For the LOLO heterogeneity trials, all of the heuristics performed comparably. As can be seen from figures 3 and 4, the MRMR heuristic was able to maintain a high level of robustness. But consequently, it even had the worst makespan among the heuristics studied.

6. Related Work

The research presented in this paper was designed using the four step FePIA procedure described in [2]. A number of papers in the literature have studied robustness in distributed systems (e.g., [6, 11, 23, 25]).

The research in [6] considers rescheduling of operations with release dates using multiple resources when disruptions prevent the use of a preplanned schedule. The overall strategy is to follow a preplanned schedule until a disruption occurs. After a disruption, part of the schedule is reconstructed to match up with the pre-planned schedule at some future time. Our work considers a slightly different environment where task arrivals are not known in advance. Consequently, in our work it was not possible to generate a

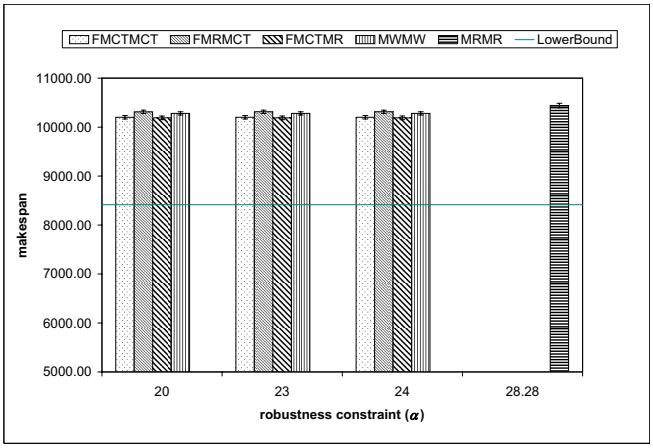

Figure 4. Simulation results of makespan for different values of robustness constraint (α) for pseudo-batch mode heuristics for LOLO heterogeneity.

preplanned schedule.

The research in [11] considers a single machine scheduling environment where processing times of individual jobs are uncertain. Given the probabilistic information about processing times for each job, the authors in [11] determine a normal distribution that approximates the flow time associated with a given schedule. The risk value for a schedule is calculated by using the approximate distribution of flow time (i.e., the sum of the completion times of all jobs). The robustness of a schedule is then given by one minus the risk of achieving sub-standard flow time performance. In our work, no such stochastic specification of the uncertainties is assumed.

The study in [23] defines a robust schedule in terms of identifying a Partial Order Schedule (POS). A POS is defined as a set of solutions for the scheduling problem that can be compactly represented within a temporal graph. However, the study considers the Resource Constrained Project Scheduling Problem with minimum and maximum time lags, (RCPSP/max), as a reference, which is a different problem domain from the environment considered here.

In [25], the robustness is derived using the same FePIA procedure used here. However the environment considered is static (off-line), as opposed to the dynamic (on-line) environment in this research. The robustness metric and heuristics employed in a dynamic environment are substantially different from those employed in [25].

7. Conclusions

This research established a method for quantifying the robustness of a resource allocation in a dynamic environment. Ten different heuristics were designed, developed, and simulated for the presented parallel and distributed environment. FRMET performed the best among the immediate mode heuristics, while FMRMCT performed the best (on average) among the pseudo-batch mode heuristics. A theoretical lower bound on the makespan of the resource allocation was also developed. The immediate mode heuristics described here can be used when the individual guarantee for the submitted jobs is to be maintained (as there is no reordering of the submitted jobs), while the pseudo-batch heuristics can be used when the overall system performance is of importance.

Table 3. Maximum and average number of mapping events (over successful trials) for which the MET machine was not feasible for HIHI and LOLO heterogeneity.

HIHI		
Robustness constraint(α)	maximum	average
22.00	41	14
24.00	54	22
25.00	73	30
26.00	79	36
27.00	88	42

LOLO		
Robustness constraint(α)	maximum	average
18.00	5	0
19.00	10	1
20.00	14	3
21.00	26	6
21.21	26	6
23.00	56	17

References

[1] S. Ali, J.-K. Kim, Y. Yu, S. B. Gundala, S. Gertphol, H. J. Siegel, A. A. Maciejewski, and V. Prasanna, "Utilization-based techniques for statically mapping heterogeneous applications onto the HiPer-D heterogeneous computing system," *Parallel and Distributed Computing Practices*, Special Issue on Parallel Numerical Algorithms on Faster Computers, Vol. 5, No. 4, Dec. 2002.

[2] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 7, July 2004, pp. 630-641.

[3] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Representing task and machine heterogeneities for hetero-

geneous computing systems," *Tamkang Journal of Science and Engineering*, Special 50^{th} Anniversary Issue, Vol. 3, No. 3, Nov. 2000, pp. 195-207 (invited).

[4] H. Barada, S. M. Sait, and N. Baig, "Task matching and scheduling in heterogeneous systems using simulated evolution," 10th IEEE Heterogeneous Computing Workshop (HCW 2001), 15^{th} *International Parallel and Distributed Processing Symposium (IPDPS 2001)*, Apr. 2001.

[5] I. Banicescu and V. Velusamy, "Performance of scheduling scientific applications with adaptive weighted factoring," 10^{th} *IEEE Heterogeneous Computing Workshop (HCW 2001)*, 15^{th} International Parallel and Distributed Processing Symposium (IPDPS 2001), Apr. 2001.

[6] J. Bean, J. Birge, J. Mittenthal, C. Noon, "Matchup scheduling with multiple resources, release dates and disruptions," *Journal of the Operations Research Society of America*, Vol. 39, No. 3, June. 1991, pp. 470-483.

[7] W. F. Boyer and G.S. Hura, "Dynamic scheduling in distributed heterogeneous systems with dependent tasks and imprecise execution time estimates," 16^{th} *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, Nov. 2004.

[8] T. D. Braun, H. J. Siegel, N. Beck, L. Boloni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and Bin Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, Vol. 61, No. 6, June 2001, pp. 810-837.

[9] R. Castain, W. W. Saylor, and H. J. Siegel, "Application of lagrangian receding horizon techniques to resource management in ad-hoc grid environments," 13^{th} *Heterogeneous Computing Workshop (HCW 2004)*, in the proceedings of the 18^{th} International Parallel and Distributed Processing Symposium (IPDPS 2004), Apr. 2004.

[10] E. G. Coffman, Jr. ed., *Computer and Job-Shop Scheduling Theory*, John Wiley & Sons, New York, NY, 1976.

[11] R. L. Daniels and J. E. Carrilo, "β-Robust scheduling for single-machine systems with uncertain processing times," *IIE Transactions*, Vol. 29, No. 11, Nov. 1997, pp. 977-985.

[12] J. Dorn, R. M. Kerr, and G. Thalhammer, "Reactive scheduling: Improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning," *International Journal on Human-Computer Studies*, Vol. 42, No. 6, June 1995, pp. 687-704.

[13] M. M. Eshaghian, ed., *Heterogeneous Computing*, Norwood, MA, Artech House, 1996.

[14] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transaction on Software Engineering*, Vol. SE-15, No. 11, Nov. 1989, pp. 1427-1436.

[15] I. Foster and C. Kesselman, eds., *The Grid: Blueprint for a New Computing Infrastructure*, San Fransisco, CA, Morgan Kaufmann, 1999.

[16] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," *Journal of the ACM*, Vol. 24, No. 2, Apr. 1977, pp. 280-289.

[17] J. -K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli, "Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines," 12^{th} *IEEE Heterogeneous Computing Workshop (HCW 2003)*, 17^{th} International Parallel and Distributed Processing Symposium (IPDPS 2003), Apr. 2003.

[18] V. J. Leon, S. D. Wu, and R. H. Storer, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, Vol. 26, No. 5, Sep. 1994, pp. 32-43.

[19] P. Luh, X. Zhao, Y. Wang, and L. Thakur, "Lagrangian relaxation neural networks for job shop scheduling," *IEEE Transactions on Robotics and Automation*, Vol. 16, No. 1, Feb. 2000, pp. 78-88.

[20] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, Vol. 59, No. 2, Nov. 1999, pp. 107-121.

[21] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, New York, NY, Springer-Verlag, 2000.

[22] V. K. Naik, S. Sivasubramanian, D. Bantz, and S. Krishnan,"Harmony: A desktop grid for delivering enterprise computations," 4^{th} *International Workshop on Grid Computing (GRID 03)*, Nov. 2003.

[23] N. Policella, *Scheduling with uncertainty, A proactive approach using partial order schedules*, PhD thesis, Dipartimento di Informatica e Sistemistica "Antonio Ruberti" Universit'a degli Studi di Roma "La Sapienza," 2005.

[24] S. Shivle, H. J. Siegel, A. A. Maciejewski, P. Sugavanam, T. Banka, R. Castain, K. Chindam, S. Dussinger, P. Pichumani, P. Satyasekaran, W. Saylor, D. Sendek, J. Sousa, J. Sridharan, and J. Velazco, "Static allocation of resources to communicating subtasks in a heterogeneous ad hoc grid environment," *Journal of Parallel and Distributed Computing*, Special Issue on Algorithms for Wireless and Ad-hoc Networks, Vol. 66, No. 4, pp. 600-611, Apr. 2006.

[25] P. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, and A. Pippin, "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints," *Journal of Parallel and Distributed Computing*, accepted, to appear.

[26] M.-Y. Wu, W. Shu, and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems," 9^{th} *IEEE Heterogeneous Computing Workshop (HCW 2000)*, May 2000, pp. 375-385.

[27] V. Yarmolenko, J. Duato, D. K. Panda, P. Sadayappan, "Characterization and enhancement of dynamic mapping heuristics for heterogeneous systems," *International Conference on Parallel Processing Workshops (ICPPW 00)*, Aug. 2000, pp. 437-444.