Bi-objective approximation scheme for makespan and reliability optimization on uniform parallel machines

Emmanuel Jeannot[1], Erik Saule[2], and Denis Trystram[2]

[1] INRIA-Lorraine : emmanuel.jeannot@loria.fr
[2] INPG, LIG(MOAIS) : firstname.lastname@imag.fr

Abstract. We study the problem of scheduling independent tasks on a set of related processors which have a probability of failure governed by an exponential law. We are interested in the bi-objective analysis, namely simultaneous optimization of the makespan and the reliability. We show first that this problem can not be approximated by a single schedule. Then, we provide a $\langle \bar{2}, 1 \rangle$-approximation algorithm (*i.e.* for any fixed value of the makespan, the obtained solution is optimal on the reliability and no more than twice the given makespan). This solution is finally used to derive a $(2+\epsilon, 1)$-approximation of the Pareto set of the problem, for any $\epsilon > 0$.

1 Introduction

With the recent development of large parallel and distributed systems (computational grids, cluster of clusters, peer-to-peer networks, etc.), it is difficult to ensure that the resources are always available for a long period of time. Indeed, hardware failures, software faults, power breakdown or resources removal often occur when using a very large number of machines. Hence, in this context, taking into account new problems like reliability and fault-tolerance is a major issue. Several approaches have been proposed to tackle the problem of faults. One approach is based on duplication. The idea is that if one resource fails, the other resources can continue to correctly execute the application (thanks to redundancy). However, the main drawback of this approach is a possible waste of resources. Another solution consists in check-pointing the application from time to time and, in case of failure, to restart it from the last check-point [1].However, check-pointing an application is costly and may require to modify it. Furthermore, restarting an application slows it down. Therefore, in order to minimize the cost of the check-point/restart mechanism, it is necessary to provide a reliable execution that minimizes the probability of failure of this application. Scheduling an application correspond to determine which resources will execute the tasks and when they will start. Thus, the scheduling algorithm is responsible of minimizing the probability of failure of the application by choosing the set of resources that enables a fast and reliable execution.

In this paper, we consider an application modeled by a set of independent tasks to be scheduled on a set of heterogeneous machines that are characterized by their speed and their error rate. The goal of the scheduling algorithm is then to minimize the makespan of the application (its run-time) and to minimize the probability of failure of the execution. As no hypothesis is made on the relationship between the speed and the error rate of the resources, there is no correlation between minimizing the makespan and optimizing the reliability of a schedule(the most reliable schedule is not necessarily the fastest one). Moreover, it has been shown that minimizing the schedule length is NP-hard even in the case of two homogeneous machine and independent tasks [2]. Therefore, unless P=NP, it is not possible to find the set of optimal trade-off in polynomial time ; *i.e.* to answer the question "*find the shortest schedule that has a probability of success greater than a given value p*". Moreover, we show that approximating the problem within a constant ratio for both objectives at the same time is impossible. Therefore, in order to solve the problem, we first propose a $\langle \bar{2}, 1 \rangle$-approximation algorithm *i.e.* an algorithm which finds a schedule which length is at most twice as long as a guess value (given by the user), if such a schedule exists, and the reliability is optimal among the schedules that are shorter than this guess value. Based on this algorithm, we are able to construct a set of solutions that approximates the Pareto set (*i.e.* the set of Pareto optimal[3] solutions) by a factor of $2+\epsilon$ for the makespan and 1 for the reliability. For constructing this set, ϵ can be chosen arbitrarily small at the cost of a larger number of solutions.

The paper is organized as follows. In Section 2, we formally define the problem and briefly discuss the related works. Then, we show that the problem cannot be approximated within a constant ratio. We discuss the most reasonable way to approximate the Pareto set. Section 3 reports the main result which is the design and analysis of the $\langle \bar{2}, 1 \rangle$-approximation algorithm. In Section 4 we derive the Pareto set approximation algorithm.

2 Problem

Let T be a set of n tasks and Q be a set of m uniform processors as described in [3]. p_i denotes the processing requirement of task i. Processor j computes $1/\tau_j$ operations by time unit and has a constant failure rate of λ_j. $p_{ij} = p_i\tau_j$ denotes the running time of task i on processor j. In the remainer of the paper, i will be the task index and j will be the processor index.

A schedule is a function $\pi : T \rightarrow Q$ that maps a task to the processor that executes it. Let $T(j, \pi) = \{i \mid \pi(i) = j\}$ be the set of tasks mapped to processor j. The completion time of a processor j is $C_j(\pi) = \sum_{i \in T(j,\pi)} p_i\tau_j$. The makespan of a schedule $C_{max}(\pi) = max_j C_j(\pi)$ is the first time where all tasks are completed. The probability that a processor j executes all its tasks

[3] intuitively, a solution is said *Pareto optimal* if no improvement on every objective can be made

successfully is given by an exponential law: $p^j_{\text{succ}}(\pi) = e^{-\lambda_j C_j(\pi)}$. We assume that faults are independent, therefore, the probability that schedule π finished correctly is: $p_{\text{succ}} = \Pi_j p^j_{\text{succ}}(\pi) = e^{-\sum_j C_j(\pi)\lambda_j}$. The reliability index is defined by $rel(\pi) = \sum_j C_j(\pi)\lambda_j$. When no confusion is possible, π will be omitted.

We are interested in minimizing both C_{max} and rel simultaneously (*i.e.* minimizing the makespan and maximizing the probability of success of the whole schedule).

2.1 Related Works

We now discuss briefly how each single-objective problem has been studied in the literature.

Optimizing the makespan: Scheduling independent tasks on uniform processors in less than K units of time is a NP-complete problem because it contains PARTITION as a sub-problem which is NP-complete [2]. For the makespan optimization problem, a $(2 - \frac{1}{m+1})$-approximation algorithm has been proposed by [4]. It consists of classical list scheduling where the longest task of the list is iteratively mapped on the processor that will complete it the soonest. [5] proposes a PTAS based on the bin packing problem with variable bin sizes. However, the PTAS is only of theoretical interest because its runtime complexity is far too high.

Optimizing the reliability: Minimizing the objective function rel is equivalent to maximizing the probability of success of the schedule on a parallel system subject to failure. More precisely, if processor j can fail with a constant failure rate λ_j and if we assume that faults are statistically independent, the probability of success of a schedule is $p_{\text{succ}} = e^{-rel}$. It has been shown in [6] that a ρ-approximation on rel is a ρ-approximation on $1 - p_{\text{succ}}$. The minimal rel is obtained by scheduling all tasks on the processors j having the smallest $\lambda_j \tau_j$. Indeed, if a task t was scheduled on another processor j', migrating it to j will result in changing rel by the negative value $p_t \lambda_j \tau_j - p_t \lambda_{j'} \tau_{j'}$.

In [7, 8] several heuristics have been proposed to solve the bi-objective problem. However, none of the proposed heuristics have a guaranteed approximation ratio.

In [9], Shmoys and Tardos studied the problem of optimizing the makespan and the sum of costs of a schedule on unrelated machines. In their model, the cost is induced by scheduling a task on a processor and the cost function is given by a cost matrix. They proposed an algorithm that receives two parameters, namely, a target value M for the makespan and C for the cost and returns a schedule whose makespan lower than $2M$ with a cost better than C. This model can be directly used to solve our problem. However, their method is difficult to implement as it relies on Linear Programming and its complexity is high: $O(mn^2 \log n)$.

2.2 On the approximability

Proposition 1. *The bi-objective problem of minimizing C_{max} and rel cannot be approximated within a constant factor with a single solution.*

Proof. Consider the instance of the problem with two machines such that $\tau_1 = 1$, $\tau_2 = 1/k$ and $\lambda_1 = 1$, $\lambda_2 = k^2$ (for a fixed $k \in \mathbb{R}^{+*}$). Consider a single task t_1 with $p_1 = 1$. There are only two feasible schedules, namely, π_1 in which t_1 is scheduled on processor 1 and π_2 in which it is scheduled on processor 2. Remark that π_2 is optimal for C_{max} and that π_1 is optimal for rel.

$C_{max}(\pi_1) = 1$ and $C_{max}(\pi_2) = 1/k$. This leads to $C_{max}(\pi_1)/C_{max}(\pi_2) = k$. This ratio goes to infinity when k goes to infinity. Similarly, $rel(\pi_1) = 1$ and $rel(\pi_2) = \frac{k^2}{k} = k$ which leads to $rel(\pi_2)/rel(\pi_1) = k$. This ratio goes to infinity with k.

None of both feasible schedules can approximate both objectives within a constant factor.

2.3 Solving the Bi-Objective Problem

As we proved in the last section in Proposition 1, the bi-objective problem cannot be approximated with a single schedule. For such problems, several approaches can be used such as optimizing a linear (or convex) combination of objectives [10], or optimizing the objectives one after the other [11]. However, these methods usually do not provide all interesting solutions. We would like to obtain all the best compromise solutions and leave the final choice to a decision maker.

The notion of Pareto dominance [12] allows to formalize the best compromise solutions in multi-objective optimization. A solution is said to be **Pareto optimal** if no solution is as good as it is on all objective values and better on at least one. The **Pareto set** (denoted by Pc^*) of a problem is the set of all Pareto optimal solutions.

Unfortunately, on our problem deciding if a solution is Pareto optimal or not is an NP-complete problem (as the makespan decision problem is NP-complete[4]). Thus, computing the whole set is impossible in polynomial time unless P=NP. Like in standard single-objective optimization, we are interested in obtaining approximate solutions.

Pc is a (ρ_1, ρ_2)**-approximation of the Pareto set** Pc^* if each solution $\pi^* \in Pc^*$ is (ρ_1, ρ_2) approximated by a solution $\pi \in Pc : \forall \pi^* \in Pc^*, \exists \pi \in Pc, C_{max}(\pi) \leq \rho_1 C_{max}(\pi^*)$ and $rel(\pi) \leq \rho_2 rel(\pi^*)$. Figure 1 illustrates this concept. Crosses are solutions of the scheduling problem represented in the $(C_{max}; rel)$ space. The bold crosses are an approximated Pareto set. Each solution $(x; y)$ in this set (ρ_1, ρ_2)-dominates a quadrant delimited in bold whose origin is at $(x/\rho_1; y/\rho_2)$. All solutions are dominated by a solution of the approximated Pareto set as they are included into a (ρ_1, ρ_2)-dominated quadrant.

In [14], Papadimitriou and Yannakakis give a generic method to obtain an approximated Pareto set. The idea is to partition the solution space into rectangles of geometric increasing size of common ratio $(1+\epsilon)$ among all objectives. The set formed by taking one solution in each rectangle (if any) is a $(1+\epsilon)$-approximation

[4] The argument is straightforward in our context. The reader should be aware that the bi-objective decision problem could be NP-complete while both single-objective decision problems are polynomial [13].

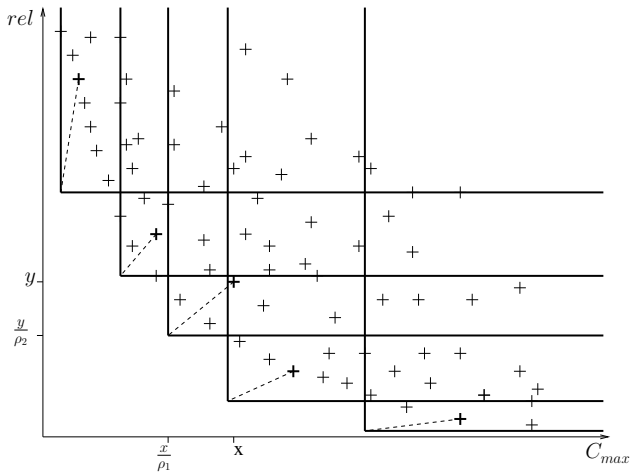

Fig. 1. Bold crosses are a (ρ_1, ρ_2)-approximation of the Pareto set.

of the Pareto set of the problem. We will use an adaptation of this method for designing a Pareto set approximation algorithm. A similar approach has been used in [15] for a single machine scheduling problem.

2.4 $\langle \bar{\rho_1}, \rho_2 \rangle$-Approximation Algorithm

Because it is impossible to get a solution approximating both objectives at the same time (Proposition 1), we are looking for the minimum reliability index among schedules whose makespan are greater than an arbitrary threshold.

Most existing algorithms that solve a bi-objective problem construct a ρ_2-approximation of the second objective constrained by a threshold on the first one. The threshold can be exceeded no more than a constant factor ρ_1. Such an algorithm is said to be a $\langle \bar{\rho_1}, \rho_2 \rangle$-approximation algorithm. More formally,

Definition 1. *Given ω a threshold value of the makespan, a $\langle \bar{\rho_1}, \rho_2 \rangle$-approximation algorithm delivers a solution whose $C_{max} \leq \rho_1 \omega$ and $rel \leq \rho_2 rel^{*,\omega^-}$ where rel^{*,ω^-} is the best possible value of rel in schedules whose makespan is less than ω.*

3 A Dual Approximation algorithm

In this section, we present a $\langle \bar{2}, 1 \rangle$-approximation algorithm called CMLT which has a better complexity and which is easier to implement than the general algorithm presented in [9].

Let ω be the guess value of the optimum makespan. Let $M(i) = \{j \mid p_{ij} \leq \omega\}$ be the set of processors able to execute task i in less than ω units of time. It

is obvious that if i is executed on $j \notin M(i)$ then, the makespan will be greater than ω.

The following proposition states that if task i has less operations than task i', then all machines able to schedule i' in less than ω time units can also schedule i in the same time. The proof is directly derived from the definition of M and is omitted.

Proposition 2. *$\forall i, i' \in T$ such that $p_i \leq p_{i'}$, $M(i') \subseteq M(i)$*

The ConstrainedMinLambdaTau algorithm (CMLT) is presented as follows: for each task i considered in non-increasing number of operations, schedule i on the processor j of $M(i)$ that minimizes $\lambda_j \tau_j$ with $C_j \leq \omega$ (or it returns no schedule if there is no processor j). Sorting tasks by non-increasing number of operations implies that more and more processors are used over time.

The principle of the algorithm is rather simple. However several properties should be checked to ensure that it is always possible to schedule all the tasks this way.

Lemma 1. *CMLT returns a schedule whose makespan is lower than 2ω or ensures that no schedule whose makespan is lower than ω exist.*

Proof. We need first to remark that if the algorithm returns a schedule, then its makespan is lower than 2ω (task i is executed on processor $j \in M(i)$ only when $C_j \leq \omega$). It remains to prove that if the algorithm does not return a schedule then there is no schedule whose a makespan lower than ω.

Suppose that task i cannot be scheduled on any processor of $M(i)$. Then all processors of $M(i)$ execute tasks during more than ω units of time, $\forall j \in M(i), C_j > \omega$.

Moreover, due to Proposition 2, each task $i' \leq i$ such that $p_{i'} > p_i$ could not have been scheduled on a processor not belonging to $M(i)$. Thus, in a schedule with a makespan lower than ω, all the tasks $i' \leq i$ must be scheduled on $M(i)$.

There is more operations in the set of tasks $\{i' \leq i\}$ than processors in $M(i)$ can execute in ω units of time.

Lemma 2. *CMLT generates a schedule such that $rel \leq rel^{*,\omega-}$*

Proof. We first need to construct a (non feasible) schedule π^* whose reliability is a lower bound of $rel^{*,\omega-}$. Then, we will show that $rel(CMLT) \leq rel(\pi^*)$.

From Theorem 2 of [6] it is known that the optimal reliability under the makespan constraint for unitary tasks and homogeneous processors is obtained by adding tasks to processors in the $\lambda\tau$ increasing order up to reaching the ω constraint. For our problem we can construct a schedule π^* where we apply a similar method. Task i is allocated to the processor of $M(i)$ that minimizes the $\lambda\tau$ product. But if i finishes after ω, the exceeding quantity is scheduled on the next processor belonging to $M(i)$ in $\lambda\tau$ order. Note that such a schedule exists because CMLT returns a solution. Of course this schedule is not always feasible as the same task can be required to be executed on more than one processor at

the same time. However, it is easy to adapt the proof of Theorem 2 of [6] and show that $rel(\pi^*) \leq rel^{*,\omega-}$.

The schedule generated by CMLT is similar to π^*. The only difference is that some operations are scheduled after ω. In π^*, these operations are scheduled on less reliable processors. Thus, the schedule generated by CMLT has a better reliability than π^*.

Finally, we have $rel(CMLT) \leq rel(\pi^*) \leq rel^{*,\omega-}$ which concludes the proof.

Algorithm 1: CMLT

```
begin
    sort tasks in non-increasing p_i order
    sort processors in non-decreasing τ_j order
    Let H be an empty heap
    j = 1
    for i = 1 to n do
        while P_j ∈ M(i) do
            Add P_j to H with key λ_j τ_j
            j = j + 1
        if H.empty() then
            Return no solution
        schedule i on j' = H.min()
        C_j' = C_j' + p_i τ_j'
        if C_j' > ω then
            Remove j' from H
end
```

Lemma 3. *The time complexity of CMLT is in* $O(n \log n + m \log m)$.

Proof. In fact, the algorithm should be implemented using a heap in the manner presented in Algorithm 1. The cost of sorting tasks is in $O(n \log n)$ and the cost of sorting processors is in $O(m \log m)$. Adding (and removing) a processor to (from) the heap costs $O(\log m)$ and such operations are done m times. Heap operations cost $O(m \log m)$. Scheduling the tasks and all complementary tests are done in constant time, and there are n tasks to schedule. Scheduling operations cost is in $O(n)$.

Theorem 1. *CMLT is a* $\langle \bar{2}, 1 \rangle$*-approximation algorithm of complexity* $O(n \log n + m \log m)$.

4 Pareto set approximation algorithm

Algorithm 2 described below constructs an approximation of the Pareto set of the problem by applying the $\langle \bar{2}, 1 \rangle$-approximation algorithm on a geometric sequence of makespan thresholds which requires a lower bound and an upper bound.

The lower bound $C_{max}^{min} = \frac{\sum_i p_i}{\sum_j \frac{1}{\tau_j}}$ is obtained by considering that a single processor is given the computational power of all the processors of the instance.

The upper bound $C_{max}^{max} = \sum_i p_i \max_j \tau_j$ is the makespan obtained by scheduling all tasks on the slowest processor. No solution can have a worse makespan without inserting idle times which are harmful for both objective functions. Note that C_{max}^{max} can be achieved by a Pareto optimal solution if the slowest processor is also the most reliable one.

Algorithm 2: Pareto set approximation algorithm

```
Data: ϵ a positive real number
Result: S a set of solutions
begin
    i = 0
    S = ∅
    while i ≤ ⌈log_{1+ϵ/2}(C_max^max / C_max^min)⌉ do
        ω_i = (1 + ϵ/2)^i C_max^min
        π_i = CMLT(ω_i)
        S = S ∪ π_i
        i = i + 1
    return S
end
```

Proposition 3. *Algorithm 2 is a* $(2+\epsilon, 1)$ *approximation algorithm of the Pareto set.*

Proof. Let π^* be a Pareto-optimal schedule. Then, there exists $k \in \mathbb{N}$ such that $(1 + \frac{\epsilon}{2})^k C_{max}^{min} \leq C_{max}(\pi^*) \leq (1 + \frac{\epsilon}{2})^{k+1} C_{max}^{min}$. We show that π_{k+1} is an $(2 + \epsilon, 1)$-approximation of π^*. This is illustrated in Figure 2.

- Reliability. $rel(\pi_{k+1}) \leq rel^*((1 + \frac{\epsilon}{2})^{k+1} C_{max}^{min})$ (by Theorem 1). π^* is Pareto optimal, hence $rel(\pi^*) = rel^*(C_{max}(\pi^*))$. But, $C_{max}(\pi^*) \leq (1+\frac{\epsilon}{2})^{k+1} C_{max}^{min}$. Since rel^* is a decreasing function, we have: $rel(\pi_{k+1}) \leq rel(\pi^*)$.
- Makespan. $C_{max}(\pi_{k+1}) \leq 2(1 + \frac{\epsilon}{2})^{k+1} C_{max}^{min} = (2 + \epsilon)(1 + \frac{\epsilon}{2})^k C_{max}^{min}$ (by Theorem 1) and $C_{max}(\pi^*) \geq (1 + \frac{\epsilon}{2})^k C_{max}^{min}$.
  Thus, $C_{max}(\pi_{k+1}) \leq (2 + \epsilon) C_{max}(\pi^*)$.

Remark that $CMLT(\omega_i)$ may not return a solution. However, this is not a problem. It means that no solution has a makespan lower than ω_i. $CMLT(\omega_i)$ approximates Pareto optimal solutions whose makespan is lower than ω_i. Hence, there is no forgotten solution.

The last points to answer are about the cardinality of the generated set and the complexity of the algorithm.

- Cardinality: The algorithm generates less than $\lceil \log_{1+\frac{\epsilon}{2}} \frac{C_{max}^{max}}{C_{max}^{min}} \rceil$
  $\leq \lceil \log_{1+\frac{\epsilon}{2}} max_i \tau_i \sum_j 1/\tau_j \rceil \leq \lceil \log_{1+\frac{\epsilon}{2}} m \frac{max_i \tau_i}{min_i \tau_i} \rceil$ solutions which is polynomial in $1/\epsilon$ and in the size of the instance.

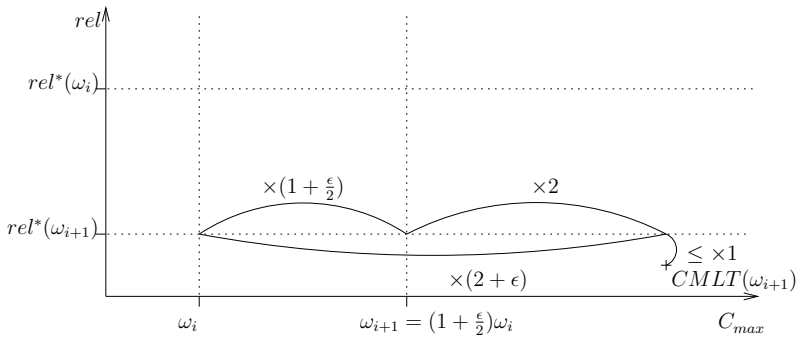

Fig. 2. $CMLT(\omega_{i+1})$ is a $(2+\epsilon, 1)$ approximation of Pareto optimal solutions whose makespan is between ω_i and ω_{i+1}. There is at most a factor of 1 in reliability between $CMLT(\omega_{i+1})$ and $rel^*(\omega_{i+1})$. The ratio in makespan between $CMLT(\omega_{i+1})$ and ω_{i+1} is less than 2 and $\omega_{i+1} = (1+\frac{\epsilon}{2})\omega_i$. Thus, $CMLT(\omega_{i+1})$ is a $(2+\epsilon, 1)$-approximation of $(\omega_i, rel^*(\omega_{i+1}))$

– Complexity: We can remark that CMLT sorts the tasks in an order which is independent of ω. Thus, this sorting can be done once for all.
Thus, the complexity of the pareto set approximation algorithm is $O(n \log n + \lceil \log_{1+\epsilon/2}(\frac{C^{max}_{max}}{C^{min}_{max}}) \rceil (n + m \log m))$.

In Section 2.1 we briefly recalled the work of Shmoys and Tardos which can be used in our context [9]. We can derive from it a $\langle 2, 1 \rangle$-approximation algorithm whose time-complexity is in $O(mn^2 \log n)$. This time-complexity is larger than the time-complexity of CMLT in $O(n \log n + m \log m)$. Moreover, in the perspective of approximating the Pareto set of the problem with the method previously presented; the algorithm derived from [9] will have a time-complexity of $\lceil \log_{1+\epsilon/2}(\frac{C^{max}_{max}}{C^{min}_{max}}) \rceil (mn^2 \log n)$. Unlike CMLT, their algorithm cannot be easily tuned to avoid a significant part of computations when the algorithm is called several time. Thus, CMLT is significantly better than the algorithm presented in [9] which has been established in a more general setting on unrelated processors.

5 Conclusion

As larger and larger infrastructures are available to execute distributed applications, reliability becomes a crucial issue. However, optimizing both the reliability and the length of the schedule is not always possible as they are often conflicting objectives. In this work, we have analyzed how to schedule independent tasks on uniform processors for optimizing both makespan and reliability. It has been proven that the problem cannot be approximated within a constant factor by a single solution. We designed the CMLT algorithm and proved that it is a $\langle 2, 1 \rangle$-approximation. Finally, we derived a $(2+\epsilon, 1)$-approximation of the Pareto set

of the problem. This bound is very good and will be hard to improve. Some previous work could have been used. However, it will have lead to a far worse complexity.

The natural continuation of this work is to address the problem of the reliability with precedence constraints. However, this problem is much more difficult. Firstly, no constant approximation algorithm for the makespan is known. Secondly, the reliability model is much more complex in presence of idle times.

References

1. Bouteiller, A., Herault, T., Krawezik, G., Lemarinier, P., Cappello, F.: Mpich-v: a multiprotocol fault tolerant mpi. International Journal of High Performance Computing and Applications (2005)
2. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, San Francisco (1979)
3. Graham, R., Lawler, E., Lenstra, J., Kan, A.R.: Optimization and approximation in deterministic sequencing and scheduling : a survey. ann. Discrete Math. **5** (1979) 287–326
4. Gonzalez, T., Ibarra, O., Sahni, S.: Bounds for LPT schedules on uniform processors. SIAM Journal of Computing **6** (1977) 155–166
5. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. SIAM Journal on Computing **17**(3) (1988) 539 – 551
6. Dongarra, J.J., Jeannot, E., Saule, E., Shi, Z.: Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: Proc. of SPAA. (2007) 280–288
7. Dogan, A., Ozgüner, F.: Matching and Scheduling Algorithms for Minimizing Execution Time and Failure Probability of Applications in Heterogeneous Computing. IEEE Trans. Parallel Distrib. Syst. **13**(3) (2002) 308–323
8. Dogan, A., Ozgüner, F.: Bi-objective Scheduling Algorithms for Execution Time-Reliability Trade-off in Heterogeneous Computing Systems. Comput. J. **48**(3) (2005) 300–314
9. Shmoys, D.B., Tardos, E.: Scheduling unrelated machines with costs. In: Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms. (1993) 448–454
10. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. In: Proc. of STACS. LNCS 3884 (2006) 621–633
11. Ho, K.: Dual criteria optimization problems for imprecise computation tasks. In Leung, J.Y.T., ed.: Handbook of Scheduling. (2004)
12. Voorneveld, M.: Characterization of pareto dominance. Operations Research Letters **31** (2003) 7–11
13. Agnetis, A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A.: Scheduling problems with two competing agents. **52**(2) (2004) 229–242
14. Papadimitriou, C., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proc. of FOCS. (2000) 86–92
15. Angel, E., Bampis, E., Gourvès, L.: Approximation results for a bicriteria job scheduling problem on a single machine without preemption. Information processing letters **94** (2005)