2009 International Conference on Intelligent Networking and Collaborative Systems

Scheduling Independent Tasks on Heterogeneous Distributed Environments by Differential Evolution

Pavel Krömer, Václav Snášel, Jan Platoš
Department of Computer Science,
VŠB - Technical University of Ostrava
17. listopadu 15, 708 33
Ostrava-Poruba, Czech Republic
Email: {pavel.kromer,
vaclav.snasel, jan.platos}@vsb.cz

Ajith Abraham
Norwegian Center of Excellence,
Center of Excellence for Quantifiable
Quality of Service,
Norwegian University of Science and
Technology, Trondheim, Norway
Email: ajith.abraham@ieee.org

Hesam Izakian
Islamic Azad University,
Ramsar Branch,
Ramsar, Iran
Email: hesam.izakian@gmail.com

***Abstract*—Scheduling is one of the core steps to efficiently exploit the capabilities of heterogeneous distributed computing systems and it is also an appealing NP-complete problem. There is a number of heuristic and meta-heuristic algorithms that were tailored to deal with scheduling of independent jobs. In this paper we investigate the efficiency of differential evolution on the scheduling problem.**

***Keywords*-scheduling; differential evolution**

I. Introduction

Grid computing and distributed computing, dealing with large scale and complex computing problems, is a hot topic in the computer science and research. Mixed-machine heterogeneous computing (HC) environments utilize a distributed suite of different machines, interconnected with computer network, to perform different computationally intensive applications that have diverse requirements [1], [2]. Miscellaneous resources should be orchestrated to perform a number of tasks in parallel or to solve complex tasks atomized to variety of independent subtasks [8]. Proper scheduling of the tasks on available resources is one of the main challenges of a mixed-machine HC environment.

To exploit the different capabilities of a suite of heterogeneous resources, a resource management system (RMS) allocates the resources to the tasks and the tasks are ordered for execution on the resources. At a time interval in HC environment a number of tasks are received by RMS. Task scheduling is mapping a set of tasks to a set of resources to efficiently exploit the capabilities of such.

It has been shown, that an optimal mapping of computational tasks to available machines in an HC suite is a NPcomplete problem [3] and as such, it is a subject to various heuristic and meta-heuristic algorithms. The heuristics applied to the task scheduling problem include min-min heuristic, max-min heuristic, longest job to fastest resource- shortest job to fastest resource heuristic, sufferage heuristic, work queue heuristic and others [2], [11], [10]. The meta-heuristics applied to the task scheduling problem include hybrid ant colony optimization [7], simulated annealing [9] and genetic algorithms [5], [4]. The meta-heuristic algorithms usually operate with a population of prospective problem solutions task schedules that are evolved (optimized) in order to obtain an improved schedule which is optimized according to some criteria.

In this paper is applied a powerful populational meta-heuristic algorithm the differential evolution to the task scheduling problem and its results are compared to selected existing algorithms. Moreover, to improve the efficiency of the general meta-heuristic solver, several widely used heuristic algorithms for scheduling in HC environments were used to improve the initial population for differential evolution.

II. Heuristic algorithms for mapping tasks in HC environment

There is a number of heuristic algorithms designed to schedule independent tasks in heterogeneous computing environments. Each algorithm exploits a heuristic based on certain intuition that helps to map tasks to machines so that selected objective is optimized. Unfortunately, different heuristics perform under various circumstances differently [11], [10].

From the optimization point of view, each heuristic represents a strategy, that finds a local optimum among all possible schedules. Number of papers compared known scheduling heuristics. A recent study [10] proposed new heuristic, called min-max, that outperformed some other popular heuristic algorithms. As for other algorithms, also the performance of min-max varies by the circumstances. Although it delivered the best result among the investigated algorithms in most cases, it was not the best in some experiments. Moreover, while min-max obtained best makespan, it did not delivered the best flowtime at the same time [10].

Efficient heuristic algorithms for scheduling in HC environments include [11], [10], [2]:

- Min-min heuristic that prioritizes tasks that can be completed earliest.
- Max-min heuristic that prioritizes tasks with the maximum earliest completion time. It aims to overlap long-running tasks with short-running ones.
- Sufferage heuristic that is based on the idea that better mappings can be generated by assigning a machine to

978-0-7695-3858-7/09 $26.00 © 2009 IEEE
DOI 10.1109/INCOS.2009.46

a task that would suffer most in terms of expected completion time if that particular machine is not assigned to it.

- Min-max heuristic that combines two metrics, the minimum execution time and the minimum completion time. It aims to assign the task to a machine that can handle it with lower execution time in comparison with other machines.

Despite the fact that the heuristic methods obtain good suboptimal results, its inconsistent behavior (different performance under different circumstances) encourages the research of global optimization methods for scheduling in HC environments. This paper deals with differential evolution for scheduling of independent tasks.

III. Differential evolution

Differential evolution (DE) is a reliable, versatile and easy to use stochastic evolutionary optimization algorithm [6]. DE is a population-based optimizer that evolves real encoded vectors representing the solutions to given problem. The real-valued nature of population vectors differentiates the DE notably from GAs that were designed to evolve solution encoded into binary or finite alphabets.

The DE starts with an initial population of N real-valued vectors. The vectors are initialized with real values either randomly or so, that they are evenly spread over the problem domain. The latter initialization usually leads to better results of the optimization process [6].

During the optimization, DE generates new vectors that are perturbations of existing population vectors. The algorithm perturbs vectors with the scaled difference of two randomly selected population vectors and adds the scaled random vector difference to a third randomly selected population vector to produce so called trial vector. The trial vector competes with a member of the current population with the same index. If the trial vector represents a better solution than the population vector, it takes its place in the population [6].

Differential evolution is parameterized by two parameters [6]. Scale factor $F \in (0, 1+)$ controls the rate at which the population evolves and the crossover probability $C \in [0, 1]$ determines the ratio of bits that are transferred to the trial vector from its opponent. The number of vectors in the population is also an important parameter of the population. The outline of DE is shown in Figure 1.

There are more variants of differential evolution. They differ mostly in the way new vectors are generated.

1 Initialize the population P consisting of M vectors
2 Evaluate an objective function ranking the vectors in the population
3 **while** *Termination criteria not satisfied* **do**
4 **for** $i \in \{1, \ldots, M\}$ **do**
5 Create trial vector $v_t^i = v_r^1 + F(v_r^2 - v_r^3)$, where $F \in [0, 1]$ is a parameter and v_r^1, v_r^2 and v_r^3 are three random vectors from the population P. This step is in DE called mutation.
6 Validate the range of coordinates of v_t^i. Optionally adjust coordinates of v_t^i so, that v_t^i is valid solution to given problem.
7 Perform uniform crossover. Select randomly one point (coordinate) l in v_t^i. With probability $1 - C$ let $v_t^i[m] = v^i[m]$ for each $m \in \{1, \ldots, N\}$ such that $m \neq l$
8 Evaluate the trial vector. If the trial vector v_t^i represent a better solution than population vector v^i, replace v^i in P by v_t^i
9 **end**
10 **end**

Fig. 1. A summary of Differential Evolution

IV. Differential evolution for scheduling optimization

An HC environment is composed of computing resources where these resources can be a single PC, a cluster of workstations or a supercomputer. Let $T = \{T_1, T_2, \ldots, T_n\}$ denote the set of tasks that is in a specific time interval submitted to RMS. Assume the tasks are independent of each other with no intertask data dependencies and preemption is not allowed (the tasks cannot change the resource they have been assigned to). Also assume at the time of receiving these tasks by RMS, m machines $M = \{M_1, M_2, \ldots, M_m\}$ are within the HC environment. For our purpose, scheduling is done on machine level and it is assumed that each machine uses First-Come, First-Served (FCFS) method for performing the received tasks. We assume that each machine in HC environment can estimate how much time is required to perform each task. In [2] Expected Time to Compute (ETC) matrix is used to estimate the required time for executing a task in a machine. An ETC matrix is a $n \times m$ matrix in which n is the number of tasks and m is the number of machines. One row of the ETC matrix contains the estimated execution time for a given task on each machine.

Similarly one column of the ETC matrix consists of the estimated execution time of a given machine for each task. Thus, for an arbitrary task T_j and an arbitrary machine M_i , $[ETC]_{j,i}$ is the estimated execution time of T_j on M_i. In ETC model we take the usual assumption that we know the computing capacity of each resource, an estimation or prediction of the computational needs of each job, and the load of prior work of each resource.

The objectives to optimize during the task mapping are makespan and flowtime. Optimum makespan (metatask execution time) and flowtime of a set of jobs can be defined as:

$$makespan = \min_{S \in Sched} \{ \max_{j \in Jobs} F_j \} \quad (1)$$

$$flowtime = \min_{S \in Sched} \{ \sum_{j \in Jobs} F_j \} \quad (2)$$

where $Sched$ is the set of all possible schedules, $Jobs$ stands for the set of all jobs and F_j represents the time in which job j finalizes. Assume that $[C]_{j,i}$ $(j = 1, 2, \ldots, n, i = 1, 2, \ldots, m)$ is the completion time for performing j-th task in i-th machine and W_i $(i = 1, 2, \ldots, m)$ is the previous workload of M_i, then $\sum (C_i + W_i)$ is the time required for M_i to complete the tasks included in it. According to the aforementioned definition, makespan and flowtime can be evaluated using Eq. (3) and Eq. (4) respectively.

$$makespan = \min_{i \in \{1,\ldots,m\}} \{ \sum C_i + W_i \} \quad (3)$$

$$flowtime = \sum_{i=1}^{m} C_i \quad (4)$$

Minimizing makespan aims to execute the whole metatask as fast as possible while minimizing flowtime aims to utilize the computing environment efficiently.

A. Schedule encoding

A schedule of n independent tasks executed on m machines can be naturally expressed as a string of n integers $S = (s_1, s_2, \ldots, s_n)$ that are subject to $s_i \in 1, \ldots, m$. The value at i-the position in S represents the machine on which is the i-the job scheduled in schedule S. Since the differential evolution uses for problem encoding real vectors, real coordinates must be used instead of discrete machine numbers. The real-encoded DE vector is translated to schedule representation by truncation of its elements.

B. Schedule evaluation

Assume schedule S from the set of all possible schedules $Sched$. For the purpose of differential evolution, we define a fitness function $fit(S) : Sched \rightarrow \mathbb{R}$ that evaluates each schedule:

$$fit(S) = \lambda \cdot makespan(S) + (1 - \lambda) \cdot \frac{flowtime(S)}{m} \quad (5)$$

The function $fit(S)$ is a sum of two objectives, the makespan of schedule S and flowtime of schedule S divided by number of machines m to keep both objectives in approximately the same magnitude. The influence of makespan and flowtime in $fit(S)$ is parameterized by the variable λ. The same schedule evaluation was used also in [4].

Flowtime and makespan are computed using a binary schedule matrix $B(S) : Sched \rightarrow \{0, 1\}^2$ which is constructed as follows: for a $n \times m$ ETC matrix that describes estimated execution times of n jobs on m machines, the $m \times n$ schedule matrix $B(S)$ has in i-th row and j-th column 1 iff the task j is scheduled for execution on machine i. Otherwise, $B(S)_{i,j}$ is equal to 0. Then $flowtime(S) : Sched \rightarrow \mathbb{R}$ and $makespan(S) : Sched \rightarrow \mathbb{R}$ can be defined with the help of matrix multiplication as:

TABLE I
A SUMMARY OF DE PARAMETERS.

Parameter	Value
Population size	20
Terminating generation	100000
Probability of crossover	$C = 0.9$
Scaling factor	$F = 0.1$
Makespan / flowtime ratio	$\lambda = 0.5$

$$makespan(S) = \sum [B(S) \cdot ETC]_{j,j} \quad (6)$$

$$flowtime(S) = \max_{j \in \{1,\ldots,m\}} \sum [B(S) \cdot ETC]_{j,j} \quad (7)$$

Less formally, makespan equals to the sum of all elements on the main diagonal of $B(S) \cdot ETC$ and flowtime equals to maximal value on the main diagonal on $B(S) \cdot ETC$.

V. EXPERIMENTS

We have implemented differential evolution for scheduling of independent tasks on heterogeneous independent environments. The differential evolution algorithm was implemented in its classic variant referred to as $DE/rand/1/bin$ [6]. To evaluate the performance of DE for minimizing makespan and flowtime, we have used the benchmark proposed in [2]. The simulation model is based on expected time to compute (ETC) matrix for 512 jobs and 16 machines. The instances of the benchmark are classified into 12 different types of ETC matrices according to the following properties [2]:

- *task heterogeneity* – represents the amount of variance among the execution times of tasks for a given machine
- *machine heterogeneity* – represents the variation among the execution times for a given task across all the machines
- *consistency* – an ETC matrix is said to be consistent whenever a machine M_j executes any task T_i faster than machine M_k; in this case, machine M_j executes all tasks faster than machine M_k
- *inconsistency* – machine M_j may be faster than machine M_k for some tasks and slower for others

The DE algorithm was used with parameters summarized in Table I. The parameters were set after brief initial tuning. The factor λ was set to 0.5 to have equal contribution of makespan and mean flowtime to the fitness value.

The experiments were conducted with two different settings for initial population. In the first case, whole initial population was generated randomly. In the second case, the initial population contained some vectors obtained by scheduling heuristics.

A. Experiments with random initial population

Table II and Table III show makespan and flowtime obtained by max-min heuristic, sufferage heuristic, min-min heuristic, and min-max heuristic. Table V and IV show makespan and flowtime of experimental schedule optimization by differential evolution with random initial population.

Each ETC matrix was named using the pattern $x - y - z$, where x describes task heterogeneity ($h$igh or $l$ow), y

TABLE II
MAKESPAN OBTAINED BY HEURISTIC ALGORITHMS.

ETC	max-min	sufferage	min-min	min-max
l-l-c	6753	5461	5468	5310
l-l-s	5947	3443	3599	3327
l-l-i	4998	2577	2734	2523
l-h-c	400222	333413	279651	273467
l-h-s	314048	163846	157307	146953
l-h-i	232419	121738	113944	102543
h-l-c	203684	170663	164490	164134
h-l-s	169782	105661	106322	103321
h-l-i	153992	77753	82936	77873
h-h-c	11637786	9228550	8145395	7878374
h-h-s	9097358	4922677	4701249	4368071
h-h-i	7016532	3366693	3573987	2989993

TABLE III
FLOWTIME OBTAINED BY HEURISTIC ALGORITHMS.

ETC	max-min	sufferage	min-min	min-max
l-l-c	108014	86643	80354	84717
l-l-s	95091	54075	51399	52935
l-l-i	79882	40235	39605	39679
l-h-c	6400684	5271246	3918515	4357089
l-h-s	5017831	2568300	2118116	2323396
l-h-i	3710963	1641220	1577886	1589574
h-l-c	3257403	2693264	2480404	2613333
h-l-s	2714227	1657537	1565877	1640408
h-l-i	2462485	1230495	1214038	1205625
h-h-c	185988129	145482572	115162284	125659590
h-h-s	145337260	76238739	63516912	69472441
h-h-i	112145666	47237165	45696141	46118709

describes machine heterogeneity ($h$igh or $l$ow) and z describes the type of consistency ($i$ncosnsistent, $c$onsistent or $s$emiconsistent).

As shown in the first four tables, DE with random initial population cannot compete with domain specific heuristics when optimizing makespan. It ranks fourth and its results are usually better than max-min heuristics, but worse than sufferage heuristics, min-min heuristics and min-max heuristics.

Table III and Table IV show the flowtime of optimized schedules. In this case, DE reached the best value for two of experimental matrices (l-h-c and h-h-c). Also in other cases, DE delivered quite competitive results. Obviously, used setting of scheduling DE suited better to the optimization of flowtime.

TABLE IV
FLOWTIME OBTAINED BY DE WITH RANDOM INITIAL POPULATION.

ETC	DE best	DE avg
l-l-c	85422	891272.4
l-l-s	53675	53964.4
l-l-i	43941	44846.2
l-h-c	3783520	3788428
l-h-s	2277816	2383501
l-h-i	1890529	1935355.4
h-l-c	2699241	2765402.2
h-l-s	1597594	1625219.6
h-l-i	1359241	1380342
h-h-c	100921177	104753227
h-h-s	67874790	70281581
h-h-i	57808847	58216428

TABLE V
MAKESPAN OBTAINED BY DE WITH RANDOM INITIAL POPULATION.

ETC	DE best	DE avg
l-l-c	7151	7303.2
l-l-s	4479	4582.2
l-l-i	3127	3203
l-h-c	451815	457741
l-h-s	212828	220334
l-h-i	141635	152186
h-l-c	212175	220142.2
h-l-s	141176	142405.2
h-l-i	99413	100307
h-h-c	13325802	13595908
h-h-s	6138124	6545734
h-h-i	4418167	454678

TABLE VI
FLOWTIME OBTAINED BY DE WITH UPGRADED INITIAL POPULATION.

ETC	DE best	DE avg
l-l-c	79580	80785.4
l-l-s	52729	52754.8
l-l-i	39674	39724.6
l-h-c	3829129	3983780.4
l-h-s	2280929	2288328.2
l-h-i	1586502	1589414.8
h-l-c	2468081	2496781.6
h-l-s	1573431	1580786.8
h-l-i	1204845	1206638.4
h-h-c	114841390	118413991.8
h-h-s	64502140	67964923.8
h-h-i	45446258	45954812.2

B. Experiments with optimized initial population

In the second set of experiments, the initial population of DE was upgraded with vectors obtained by scheduling heuristics. Max-min heuristic, sufferage heuristic, min-min heuristic, and min-max heuristic were used to obtain four vectors that were included in initial population of DE. Those vectors were superior to the rest of the initial population in terms of makespan and flowtime. The factor $lambda$ was set to 0.9 in order to preserve the suboptimal makespan from the initial population because initial experiment showed the tendency to improve flowtime at the expense of good initial makespan.

The results of second DE optimization are summarized in Table VI and VII respectively.

TABLE VII
MAKESPAN OBTAINED BY DE WITH UPGRADED INITIAL POPULATION.

ETC	DE best	DE avg
l-l-c	5250	5271
l-l-s	3326	3326.8
l-l-i	2498	2502.2
l-h-c	267773	270912.4
l-h-s	146125	146759.4
l-h-i	100904	101254.6
h-l-c	159770	161262.2
h-l-s	101824	102440.2
h-l-i	76096	76297.4
h-h-c	7775829	7856042.8
h-h-s	4368071	4372414.6
h-h-i	2922633	2953782.6

The best schedules obtained by DE with upgraded initial population were superior in terms of makespan in all cases. For all ETC matrices, except of h-h-s, outperformed also the average DE makespan scheduling heuristics.

The flowtime obtained by DE with upgraded initial population was not the best in all cases. However, differential evolution managed to optimize makespan and flowtime at once whereas the heuristic algorithms were not able to do that. Also, the value of $lambda$ used during the experiment prioritized makespan.

VI. Conclusions

This paper presents an algorithm for scheduling independent tasks on heterogeneous distributed environments based on differential evolution. The algorithm was implemented and experimental results suggest that it can deliver competitive results. With random initial population, the algorithm managed to optimize schedules for few ETC matrices so that the flowtime was best.

Much better results were obtained when we upgraded the initial population with candidate solutions obtained by the heuristic algorithms. In such case, the algorithm managed to exploit the different sub-optimal solutions provided at the beginning and converged to better schedules.

Presented algorithm has a number of parameters including C, F and λ. Fine tuning of DE parameters is subject of our future work. Moreover, the performance of the algorithm will be compared to other meta-heuristics for scheduling of independent tasks in heterogeneous computing environments.

Acknowledgment

This work was supported by the Czech Science Foundation under the grant no. $102/09/1494$.

References

[1] Ali, S., Braun, T., Siegel, H., Maciejewski, A.: Heterogeneous computing (2002). citeseer.ist.psu.edu/ali02heterogeneous.html

[2] Braun, T.D., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., Freund, R.F.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems (2001)

[3] Fernandez-Baca, D.: Allocating modules to processors in a distributed system. IEEE Trans. Softw. Eng. **15**(11), 1427–1436 (1989).

[4] Javier Carretero, Fatos Xhafa, Ajith Abraham: Genetic Algorithm Based Schedulers for Grid Computing Systems. In: International Journal of Innovative Computing, Information and Control, vol. 3 (2007)

[5] Page, A.J., Naughton, T.J.: Framework for task scheduling in heterogeneous distributed computing using genetic algorithms. Artificial Intelligence Review **24**, 137–146 (2004)

[6] Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution A Practical Approach to Global Optimization. Natural Computing Series. Springer-Verlag, Berlin, Germany (2005).

[7] Ritchie, G., Levine, J.: A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments. In: Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group (2004)

[8] Tracy, M.M., Braun, T.D., Siegel, H.J.: High-performance mixed-machine heterogeneous computing. In: 6th Euromicro Workshop on Parallel and Distributed Processing, pp. 3–9 (1998)

[9] YarKhan, A., Dongarra, J.: Experiments with scheduling using simulated annealing in a grid environment. In: GRID '02: Proceedings of the Third International Workshop on Grid Computing, pp. 232–242. Springer-Verlag, London, UK (2002)

[10] Izakian, H., Abraham, A., Snášel, V.: Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments. In: IWHGA '09: Proceedings of the IEEE International Workshop on HPC and Grid Applications, IEEE Press, USA, (2009)

[11] Munir, E.U., Jian-Zhong Li, Sheng-Fei Shi, Rasool, Q.: Performance Analysis of Task Scheduling Heuristics in Grid. In: ICMLC '07: Proceedings of the International Conference on Machine Learning and Cybernetics, Volume 6, Issue 19-22, pp. 3093 – 3098, (2007)