Globus An International Journal of Management & IT
A Refereed Research Journal
Vol 8 / No 2 / Jan-Jun 2017 ISSN: 0975-721X

PERFORMANCE STUDY OF TASK AND RESOURCE SECHEDULING IN SOFTWARE PROJECT

*I.K. Gulam Mohiddin

Abstract

Software projects scheduling plays a significant role in software project management. Software project management is the process of scheduling and leading the software projects in which the software projects are designed, executed and managed. A software project has the capability for testing and maintaining the software product during the specific period of time. Software project management is designed to address every software projects that is essential to manage the difficult processes of software projects. The main aim of task scheduling is to schedule the tasks on processors and also decrease the make span of schedule efficiently. The task preemption is exactly represented to provide the resource efficiently with reducing the software time by using software project management. Due to the requirement for scheduling many activities and improper task preemption cause severe threats is the major issue. The study helps to prioritizing the multiple projects' activities which resulting in minimum scheduling time. This in turn to decreases the thread stacks and therefore improves the optimization of memory with lesser cost.

Keywords: Project Scheduling, Software Project Management, Task Scheduling, and Software Project.

Introduction

Software project scheduling is one of the most significant scheduling parts that are mainly focused on software project management group. With the huge opportunity in internet, large-scale administered software projects in manufacturing, production and others are becoming more extensive. The project scheduling is to illustrate the process of constructing and detecting the scheduling technique for software development projects. In order to build the difficult software systems, the several engineering tasks are required to take place with one another to complete the project during particular interval of time. Both the software engineering as well as the software management is very necessary for efficient software project.

Software project management is defined as the procedure of organizing, staffing, observing, managing and leading the software project. Software project manager is efficiently leads to software development team for scheduling the several project activities with minimum cost. Scheduling of software projects involves the resource allocation to ascertain the start and completion periods of the comprehensive activities. The software project manager's work is essential to guarantee the software project with its resource constraints and delivers software in time. Thus the process of software project management is a technique of providing every activity that is relevant to its project and parts. Also, the management of software project is necessary to require, since the professional software engineering is mainly focused on organizational budget and schedule constraints.

When the task preemption is accurately described, the resources scheduling is organized efficiently in which decreasing the software time and cost. Projects scheduling is the process of scheduling the task with shared stack and transition between the preemptive and non-preemptive threads. An inappropriate task preemption approach creates severe threats that are essential to minimize the threats by using multiple project managements. Project management is very useful for all variety of software projects but it is extensively employed to manage the difficult processes of software developments projects. During the project management, the resources scheduling has the ability to execute the task of software project efficiently and effectiveness.

*Lecturer, Department of Computer Science, Eritrean Institute of Technology, Asmara, Eritrea

This paper is organized as follows: Section II discusses task and resource scheduling in software project. Section III describes the existing task and resource scheduling in software project technique, Section IV identifies the possible comparison between them, Section V explains the limitations as well as the related work and Section VI concludes the paper, research work is given as to optimizing various projects activities which resulting in minimum scheduling time and cost.

Literature Review

Multi Agent Optimization Algorithm (MAOA) in [1] deployed an integrated key behavior of agent and population-based method of swarm intelligence to determine Resource Constrained Project Scheduling Problem (RCPSP). Though, RCPSP method is resulted in finding better solutions with acceptable running time, more reasonable search operators remained unaddressed. RCPSP technique in [2] describes combinatorial NP-difficult issues for implementing a number of precedence based tasks is subjected to limited uncertain resources. The design of Max-Min Ant System algorithm using Hyper-Cube structure in [8] designed to reduce the software project cost and duration. A technique with event-based scheduler and ant colony optimization algorithm (EBS-ACO) in [7] allows the resource conflict and task preemption method for optimizing resources usage.

Evolutionary Algorithms (EAs) in [6] design the functions considered relative significant among the cost and completion time depends on their weights. However, less emphasis is made on runtime analysis. RCSP technique in [11] describes the resource consumption in construction activities. But, the RCSP method has very large size and complexity. Particle Swarm Optimization (PSO) based hyper-heuristic algorithm in [7] to solve RCPSP presented that worked an upper-level algorithm and controlled several low-level heuristics run on solution space. An improved differential evolution (IDE) algorithm intended in [15] designed to address the software project scheduling problem (SPSP) technique. Multi Threaded Local Search (MTLS) in [4] intended a master thread that controlled several worker threads, running in a parallel manner, where the information exchange is done only when the worker thread reached local optimum. Despite local optima, state transition between threads did not evolved over time.

To remove the resource contention, Dynamic Task Aware Scheduling (DTAS) in [3] designed to run complementary tasks such as compute-bound or memory-bound during run time. RCPSP technique in [10] deal with realistic like energy constraints energy limitation, constraint demand and power utilization. Nvidia GPU processor description in [13] describes the intermittent errors that is determined accurately and contains the limited impact of well defined architecture tile. Though, it enhances the shared memory and register usage lead to overhead and also higher false positive rate. A novel time planning procedure in [17] designed to discover the feasible start times of activities and the longest paths among start times of activities. However, the resource optimization rate is high.

Multi-Mode Resource-Constrained Multi-Project Scheduling Problem in [18] determined to detect a feasible schedule when reducing the total project delay (TPD) and total makespan (TMS). Though, the scheduling time is high. A Time Division Multiplexing (TDM) global scheduler and preemptive Fixed Priority (FP) local schedulers running multiple applications on a single platform designed in [9]. A two-step procedure developed in [14] where the TCT is employed with Microsoft excel software to achieve the project deadline considering unlimited resources. But, computational time gets maximized. Cooperative Coevolutionary multi-objective algorithm (CCMOA) in [12] designed to provide the high-quality results and efficiently solve the optimization issues. A software project scheduling/rescheduling method in [16] planned to support dynamic staffing and rescheduling by using hybrid approach based on Genetic Algorithm (GA) and Hill Climbing (HC). Though, the performance of project scheduling is not effective.

In this paper, in order to overcome the above mentioned limitation, a task and resource scheduling in software project is designed. That aims to guarantee the scheduling multiple project activities of several software projects with minimal resource cost. Hence, the scheduling technique is essential to enhance the resource optimization rate for achieving the high performance by using software project.

Task and Resource Scheduling in Software Project

Project scheduling is one of the most essential techniques for preventing the delays during the software project. Software project is the process of software development that is efficiently required for testing and maintenance according to the planned software product is attained in a certain time interval. Software project managers are responsible for planning and scheduling of software project development in an efficient manner. The process of resource scheduling is more flexible to enables the project managers that are essential to describe the staffing requirements and resource managers to fulfill the overall

necessities. Task management has the ability to control the task through its entire life cycle that contains planning, testing, tracking and reporting. In addition, the task scheduler is assists to achieve the goals that is allowed to monitor and also executes the scheduling tasks automatically.

The performance of task and resource scheduling in software project is compared against with the three existing methods including Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm, Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm and Dynamic Task-Aware Scheduling (DTAS) technique.

A Particle Swarm Optimization Based Hyper-Heuristic Algorithm for the Classic Resource Constrained Project Scheduling Problem

Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm is essential to deal with more familiar and challenging technique to solve the Resource Constrained Project Scheduling Problem (RCPSP). The hyper-heuristic is designed to control the upper-level algorithm that limits the various low-level heuristics in which they work to solution space. The process of solution representation is mainly depends upon the random keys. Also, the active schedules are very useful to create serial scheduling generation systems that are changed through low-level heuristics method.

The hyper-heuristic is the main part of a flexible multi-level process which does not need to detect how the low-level heuristics run to solution space. But, it only requires the information about the function and its value. It is the main attributes of hyper-heuristic structure that make very easier to improve the problem-independent hyper heuristic algorithms and also share the low-level heuristics for project scheduling problems (PSP).

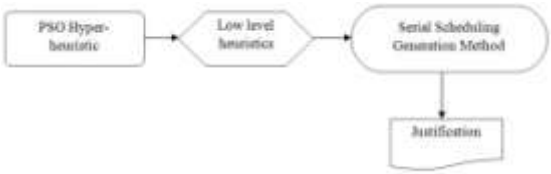

Figure 1: Block Diagram of Particle Swarm Optimization Based Hyper-Heuristic Algorithm

Figure 1 describes the diagram of PSO Hyper-heuristic algorithm. Initially, the PSO-HH approach is essential to change the sequence in which the low-level heuristics are efficiently employed to solution space. After applying the low-level and addressing the vector of priorities, the serial scheduling generation method is utilized to build the possible schedule and also compute the effective makespan. Finally, the process of justification is employed to implement the local search and more feasibly to increase the resulted makespan. In order to evaluate the functionality of hyper-heuristic during the selected algorithmic parameters settings that demonstrates good results involved in the effectiveness of PSO Hyper-heuristic algorithm. In addition, the flexibility of hyper-heuristics approach and the high quality solutions are described to increase the algorithm of hyper-heuristics for project scheduling problems.

Ant Colony Optimization for Software Project Scheduling and Staffing With an Event-Based Scheduler

A new technique is employed for developing the software project planning problem to be solved. The major attributes of the method are classified into two types such as Event-Based Scheduler (EBS) and Ant Colony Optimization (ACO) algorithm. Initially, the approach is essential to establish the event-based scheduler. Next, the process of ACO is utilized to determine more complex software planning issues. Therefore, the EBS-ACO technique is designed to improve the flexible and effective model for software project planning. Both the EBS and ACO method is performed by using task list and an employee allocation matrix. Consequently, both the problem of task scheduling and employee allocation is taken into consideration. In starting time of project, the time when resources released from ending tasks and time when employees join or leave the project is considered as events by using EBS method. The fundamental idea of EBS technique is to modify the distribution of employees at events and also maintain the distribution cannot modify at nonevents.

The EBS-ACO technique is essential to implement the modeling of resource conflict as well as task preemption and protect the flexibility involved during the distribution of human resource. The EBS-ACO strategy is essential to describe the process of ACO algorithm. Initially, the approach initializes the parameter of ACO method. During all iteration, ants set out to construct the plans for problem. While the plan is used for evaluating the problem that is mainly consists of task list and also designed employee allocation matrix. Thus the process of solution construction in the EBS- ACO algorithm is classified into two steps namely construction of task list and construction of employee allocation matrix. After that, the pheromone values are updated by using the global and local updating rules. Finally, a local mutation procedure is evaluated as the local search that maximizes the performance of EBS- ACO algorithm. Furthermore, the EBS- ACO algorithm

is more effective to handle the better plans with very lesser costs and highly stable workload task is compared to other technique.

Kernel Mechanisms with Dynamic Task-Aware Scheduling to Reduce Resource Contention in NUMA Multi-Core Systems

Many systems with multi-core processors, Non-Uniform Memory Access (NUMA) architecture increase the system scalability by separating the processors and memory into multiple nodes. When processors take effort to access the shared resources simultaneously, a resource contention leads to minimize the performance of system. In order to prevent the resource contention in NUMA multicore systems, processor cores that share resources is essential to operate the complementary tasks. Dynamic Task-Aware Scheduling (DTAS) technique is designed to decrease the resource contention during NUMA multi-core systems. It is automatically detect the resource usage for running tasks at run time and dynamically recognizes the task's categorization like either compute-bound or memory-bound task. Also, the processors are classified into compute-bound processors or memory-bound processors.

Figure 2 explains the flow diagrams of task and processor classification. The task and processor are classified into two types including compute-bound and memory-bound. Initially, the compute-bound processor is responsible for executing compute-bound tasks and thus set to higher processor frequency to efficiently guarantee the enhanced performance. Next, the memory-bound processor is responsible for executing memory-bound tasks and thus set to lower processor frequency to reduce the utilization of power. When the task is compute-bound, it is dispatched to compute-bound processor or else, it is dispatched to memory-bound processor. Based on this classification of tasks and processors, the DTAS dispatches a compute-bound task to run on compute-bound processor and dispatches a memory-bound task to run on a memory-bound processor. If the utilization of power gets reduced, processors are required to run at proper frequency for saving the power while maintaining the excellent performance. Finally, a results show that the DTAS mechanism is performed to decreases the utilization of power by reducing resource contention between the processor cores.

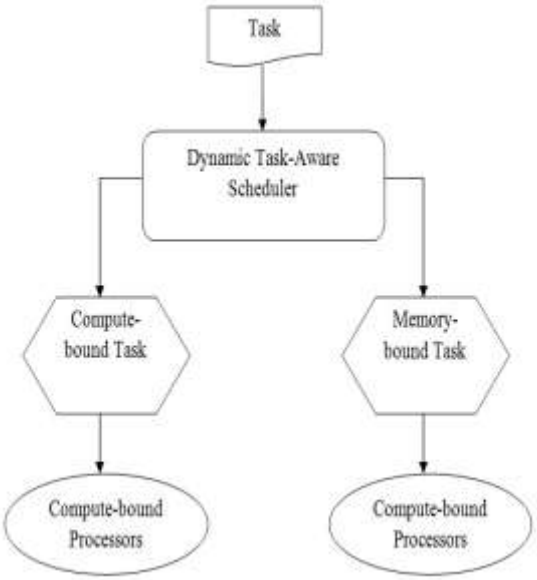

Figure 2: Flow Diagrams of Task and Processor Classification

The experimental evaluation using task and resource scheduling in software project is conducted on various factors including resource scheduling efficiency, memory optimization rate and scheduling time.

Comparison of Task and Resource Scheduling in Software Project Using Different Techniques and Suggestions

In order to compare the task and resource scheduling in software project method using different techniques, number of project activities is taken to perform this experiment. Various parameters are used for task and resource scheduling in software project techniques.

Resource Scheduling Efficiency (RSE)

The resource scheduling efficiency is defined as the ratio of exactly scheduling the multiple project activities to the total number of project activities. Resource scheduling efficiency is measured in terms of percentage (%) and mathematically formulated as below,

$$\text{RSE}\ (\%) = \frac{\text{Number of exactly scheduled project activities}}{\text{Total number of project activities}} \times 100$$

When the resource scheduling efficiency is higher, the method is said to be more efficient.

Table 1 Tabulation of Resource Scheduling Efficiency

Number of Project Activities	Resource Scheduling Efficiency (%)		
	PSO-HH	EBS-ACO	DTAS
10	60	72	55
20	65	74	57
30	68	77	59
40	70	79	62
50	74	82	65
60	77	85	68
70	79	87	71

Table 1 describes the resource scheduling efficiency versus different number of project activities in the range of 10 to 70. The resource scheduling efficiency comparison takes place on existing Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm, Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm and Dynamic Task-Aware Scheduling (DTAS) technique.

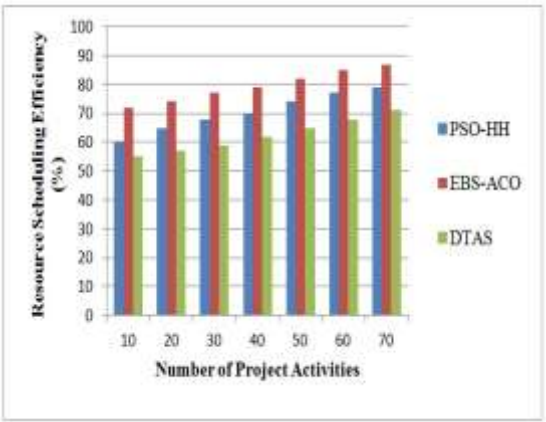

Figure 3: Measurement of Resource Scheduling Efficiency

Figure 3 measures the resource scheduling efficiency of existing techniques. Resource scheduling efficiency of Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm is comparatively higher than that of Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) and Dynamic Task-Aware Scheduling (DTAS) methods. Research in Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm has 22% higher resource scheduling efficiency than Dynamic Task-Aware Scheduling (DTAS) technique and 11% higher resource scheduling efficiency than Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm.

Memory Optimization Rate

The memory optimization rate is measured as the difference between the total number of memory used to the unused memory in project activities. Memory optimization rate is measured in terms of percentage (%) and mathematically formulated as below,

$$Memory\ Optimization\ Rate = Total\ number\ of\ memory - Unused\ memory$$

When the memory optimization rate is higher, the method is said to be more efficient.

Table 2 Tabulation of Memory Optimization Rate

Number of Project Activities	Memory Optimization Rate (%)		
	PSO-HH	EBS-ACO	DTAS
10	48	41	57
20	50	44	59
30	53	46	62
40	54	49	65
50	56	51	68
60	59	54	70
70	62	56	72

Table 2 describes the memory optimization rate versus different number of project activities in the range of 10 to 70. The memory optimization rate comparison takes place on existing Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm, Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm and Dynamic Task-Aware Scheduling (DTAS) technique.

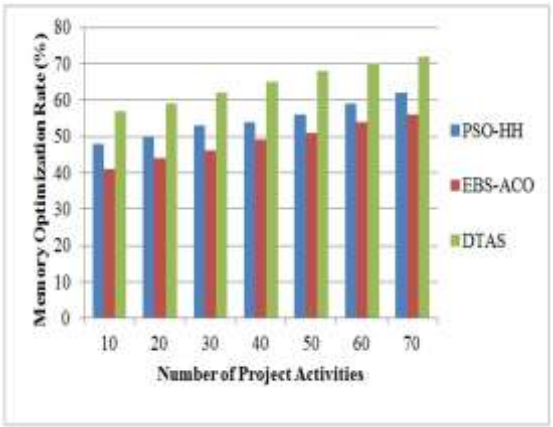

Figure 4: Measurement of Memory Optimization Rate

Figure 4 measures the memory optimization rate of existing techniques. Memory optimization rate of Dynamic Task-Aware Scheduling (DTAS) method is comparatively higher than that of Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) and Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm. Research in Dynamic Task-Aware Scheduling (DTAS) technique consumes 16% improved memory optimization than Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm and 25% improved memory optimization rate than Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm.

Scheduling Time

Scheduling time is defined as the difference between ending time and starting time required for scheduling the software projects. Scheduling time is measured in terms of milliseconds (ms) and mathematically formulated as below,

$$\begin{aligned} &Scheduling\ Time\ (ms) \\ &= Ending\ time \\ &- Starting\ time\ for\ scheduling \\ &\qquad software\ projects \end{aligned}$$

When the scheduling time is lower, the method is said to be more efficient.

Table 3 Tabulation of Scheduling Time

Number of Project Activities	Scheduling Time (ms)		
	PSO-HH	EBS-ACO	DTAS
10	34	50	42
20	37	52	45
30	39	55	48
40	42	58	50
50	45	61	53
60	48	65	58
70	53	70	62

Table 3 describes the scheduling time rate versus different number of project activities in the range of 10 to 70. The scheduling time comparison takes place on existing Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm, Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm and Dynamic Task-Aware Scheduling (DTAS) technique.

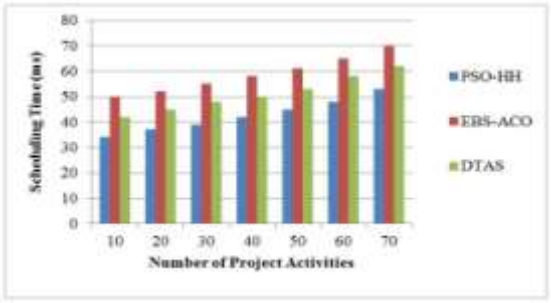

Figure 5: Measurement of Scheduling Time

Figure 5 measures the scheduling time of existing techniques. Scheduling time of Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm is comparatively lesser than that of Dynamic Task-Aware Scheduling (DTAS) technique and Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm. Research in Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) algorithm contains 39% lesser scheduling time than Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) algorithm and 20% lesser scheduling time than Dynamic Task-Aware Scheduling (DTAS) approach.

Discussion and Limitation of Task and Resource Scheduling in Software Project Using Different Techniques

In Dynamic Task-Aware Scheduling (DTAS) approach, a user does not contains the sufficient knowledge about the multi-core processor topology. Scheduling of multiple tasks is not ensured by using DTAS technique. The problem of choosing most appropriate method for migrating and reducing the superfluous task and page migration are not investigated. Also, the cost of migrate the task with its allocated memory is greater and performance of the DTAS system is decreased.

In Event-Based Scheduler and Ant Colony Optimization (EBS-ACO) approach, the uncertainty treatment is involved in software project planning model that cause very complicated and challenging issues. The employee experience and training model creates more extensive problem is not considered by using EBS-ACO algorithm. During the EBS technique, the comprehensive model complex event remained unaddressed. Particle Swarm Optimization based Hyper-Heuristic (PSO-HH) explains the enhanced scheduling time is need if the size of problem gets enhanced, it means that they difficult to solve larger scale issues.

Future Direction

The future direction of task and resource scheduling in software project can be employed to scheduling the several projects' activities which resulting in smallest cost. In addition, to increases the resource optimization while reducing the threads stacks during multiple software project.

Conclusion

The comparison of different techniques for task and resource scheduling in software project technique is carried out. Scheduling of multiple tasks is not guarantee and it does not contain the sufficient information about the multi-core processor system by using DTAS technique. When the problem size gets improved, the PSO-HH approach is described to require maximum scheduling time in which the larger scale problems are very hard to determine. Then the cost of migrate the task is improved and performance of the DTAS system gets minimized. During the proper planning through EBS approach, the prioritization of projects is remains unaddressed. Also, the improper task preemption model is designed to cause severe threats during the multiple software project scheduling. Finally, from the result, the research work can decrease the thread stacks and scheduling the multiple projects' activities which resulting in minimum cost. That helps to improve the resource scheduling efficiency, memory optimization rate and reducing the scheduling time involved in the analysis of software projects with better efficient.

References

1. Xiao-long Zheng and Ling Wang (2015), "A multi-agent optimization algorithm for resource constrained project scheduling problem", ELSEVIER: Expert Systems with Applications, Volume 42, Issue 15, Pages 6039-6049.
2. Margarita Knyazeva, Alexander Bozhenyuk, and Igor Rozenberg (2015), "Resource-constrained project scheduling approach under fuzzy conditions", ELSEVIER: Procedia Computer Science, Volume 28, Pages 56-64.
3. Mei-Ling Chiang, Chieh-Jui Yang and Shu-Wei Tu (2016), "Kernel mechanisms with dynamic task-aware scheduling to reduce resource contention in NUMA multi-core systems", ELSEVIER: The Journal of Systems and Software, Volume 121, Pages 72-87.
4. Martin Josef Geiger (2016), "A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling", ELSEVIER: European Journal of Operational Research.
5. Leandro L. Minku, Dirk Sudholt, and Xin Yao (2014), "Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis", IEEE Transaction on Software Engineering, Volume 40, Issue 1, Pages 83-102.
6. Wei-Neng Chen and Jun Zhang (2013), "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler", IEEE Transaction on Software Engineering, Volume 39, Issue 1, Pages 1-17.
7. Georgios Koulinas, Lazaros Kotsikas and Konstantinos Anagnostopoulos (2014), "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem", ELSEVIER: Information Sciences, Volume 277, Pages: 680-693.
8. Broderick Crawford, Franklin Johnson, Ricardo Soto, Eric Monfroy and Fernando Paredes (2014), "A Max-Min Ant System algorithm to solve the Software Projects Scheduling Problem", Expert Systems with Applications, Volume: 41, Issue: 15, Pages: 6634-6645.
9. Laura Carnevali, Alessandro Pinzuti and Enrico Vicario (2013), "Compositional Verification for Hierarchical Scheduling of Real-Time Systems", IEEE Transaction on Software Engineering, Vol 39, Issue 5, Pages: 638-657.
10. Hironori Okubo, Toshiyuki Miyamoto, Satoshi Yoshida, Kazuyuki Mori, Shoichi Kitamura and Yoshio Izui (2015), "Project Scheduling under Partially Renewable Resources and Resource Consumption during Setup Operations" ,Computer and Industrial Engineering, Volume 83, Pages 91-99.
11. Sofia Kaiafa and Athanasios P. Chassiakos (2015), "A genetic algorithm for optimal resource-driven project scheduling", ELSEVIER: Procedia Engineering, Volume 123, Pages 260-267.
12. Jian Xiong, Roel Leus, Zhenyu Yang, Hussein A. Abbass (2015), "Evolutionary Multi-Objective Resource Allocation and Scheduling in the Chinese Navigation Satellite System Project", European Journal of Operational Research, Volume 251, Pages 662-675.
13. David Defour and Eric Petit (2016), "A software scheduling solution to avoid corrupted units on GPUs", ELSEVIER: Journal of Parallel and Distributed Computing, Volume 90-91, Pages 1-8.
14. Sanjay Tiwari, Sparsh Johari (2015), "Project Scheduling by Integration of Time Cost Trade-off and Constrained Resource Scheduling", Journal of Institution of Engineers (India): Series A, Volume 96, Issue 1, Pages 37-46.
15. C. Biju, T. Aruldoss Albert Victoire and Kumaresan Mohanasundaram (2015), "An Improved Differential Evolution Solution for Software Project Scheduling Problem", Hindawi Publishing Corporation, the Scientific World Journal, Article ID 232193, Pages 1-9.

16. Yujia Ge, Bin Xu (2015), "Dynamic Staffing and Rescheduling in Software Project Management: A Hybrid Approach", PLoS ONE, Volume 11, Issue 6, Pages 1-28.
17. Stefan Kreter, Julia Rieck, Jürgen Zimmermann (2016), "Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars", Elsevier: European Journal of Operational Research, Volume 251, Issue 2, Pages 387-403.
18. Tony Wauters, Joris Kinable, Pieter Smet, Wim Vancroonenburg, Greet Vanden Berghe, Jannes Verstichel (2016), "The Multi-Mode Resource-Constrained Multi-Project Scheduling Problem", Journal of Scheduling, Springer, Volume 19, Issue 3, Pages 271-283.