Crimson Publishers
Wings to the Research

Research Article

Software Project Scheduling: A Systematic Literature Review

Natasha Nigar*

Department of Computer Science, University of Engineering and technology (RCET), Pakistan

***Corresponding author:** Natasha Nigar, Department of Computer Science, University of Engineering and technology (RCET), Pakistan

Submission: May 30, 2022

Published: August 10, 2022

Volume 2 - Issue 3

How to cite this article: Natasha Nigar. A Short History of Cataract Surgical Training. COJ Rob Artificial Intel. 2(3). COJRA. 000537. 2022.

Abstract

Context: During project management, software project scheduling is performed to generate a schedule for system. It is one of the most important steps taken before project development.

Objective: This study aims to identify and analyze common trends, gaps and future studies in line with formulated research questions.

Method: According to the guidelines proposed in evidence-based software engineering literature, a state of art literature review has been conducted.

Result: We selected a total of 41 primary studies from the search process. This selection comprises of 14 journal articles, 25 conference papers, 1 book chapter, and 1 article.

Conclusion: Project scheduling has been discussed by researchers significantly. There are number of limitations in existing scheduling proposed techniques which include: lack of scalability, inclusion of less dynamic events, inefficiency in the presence of more dynamics, a low number of project scenarios handling. The existing techniques have been applied in complex and real settings in a limited means too.

Keywords: Dynamic software project scheduling; Search-based optimization; Mathematical model

Introduction

Due to exponential increasing trends in software market, software projects' success heavily depends on efficient project plans and reduced development costs. This refers to a Scheduling Problem (SP) where decisions are made about, who does what during project life cycle [1]. Software Project Management (SPM) is a complex task due to unpredictable dimensions. Software project scheduling is a main domain of SPM and it involves the management of large teams in a dynamic environment with uncertain parameters [2]. There exists many software project management supporting tools such as MS Project [3] but they fail in a dynamic environment when dealing with uncertainties. Search-Based Software Engineering (SBSE) is an emerging field to cure such complex software engineering problems. Search-based optimization techniques have been applied to wide variety of software engineering problems [1] and, this research focuses on the software project scheduling.

SBSE search algorithms can be categorized into three main groups as listed below:

A. Exact optimization methods: Branch and Bound algorithm, and Integer Linear Programming are examples of such methods and they guarantee to find an optimal solution [4].

B. Heuristic algorithms: These algorithms find a 'good' or near-optimal solution such as greedy algorithms.

C. Metaheuristics: is the most preferred approach in the SBSE field [5] that continue the search beyond the first encountered local optimum. Examples are Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO).

Some hybrid approaches are also used that combine techniques from these three main groups. The study aims to chronologically review and select the published literature and present an overview of existing software project scheduling problems and techniques used to solve those problems.

Existing published literature review has been classified as follows:

A. Traditional Literature Review (TLR): TLR attempts to establish current research trends. A critical summary is written by examining the body of published literature.

B. Systematic Literature Review (SLR): SLR uses a structured methodology towards clarifying the precise set of formulated research questions and identifies gaps, contradictions, and inconsistencies in the literature.

To the authors' best knowledge, no SLR exists that focuses on software project scheduling taxonomies, techniques, targeted software models for scheduling, data sets used and limitations/benefits of existing scheduling techniques. The essence of this SLR is to present the available evidence regarding (1) the existing scheduling taxonomies (2) techniques (3) software models used (4) types of data sets used (5) their limitations/benefits. Therefore, this SLR will provide insight for both researchers and project managers in the academia and industries to create more efficient project plans. This article is structured as follows. Research method used in this study is described in section 2. Section 3 presents the results and discussions. Research finding are described in section 4. This paper concludes in section 6.

Review Process

We have adopted the approach proposed by [6] in performing this SLR (Figure 1). Refereeing to Figure 1, the review process consists of six phases. In first phase, a set of research questions were formulated. Search strategies were designed in phase 2 in accordance with formulated research question. In this phase, search terms were identified and choice of literature resources was made. Data is extracted from literature resources in phase 3. Phase 4 concentrates on refinement of extracted data by scrutinizing titles. A quality assessment criterion is applied in fifth phase to further evaluate data. In last phase, studies for analysis are selected and subsequent actions are performed.

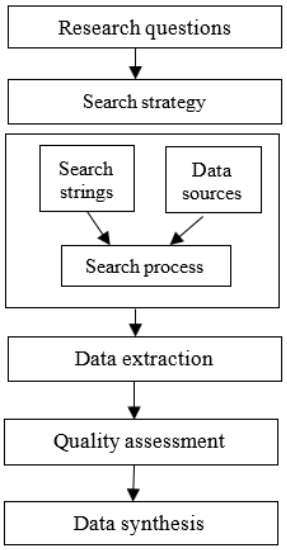

Figure 1: Phases of review process.

Research questions

The basic purpose of this study is to summarize the scheduling scenarios and techniques and identify area of further research. Five Research Questions (RQs) were formulated as presented below:

A. RQ1: What are taxonomies of scheduling?

B. RQ2: What are software models that have been targeted for scheduling?

C. RQ3: What are existing optimization techniques used for scheduling problem?

D. RQ4: What kind of dataset have used in existing proposed techniques?

E. RQ5: What are limitations/benefits of existing scheduling techniques?

PICOC [6] is described below, based on above research questions:

Population (P): Scheduling for software projects development

methods (medium to large scale).

Intervention (I): Scheduling Methods/techniques

Comparison (C): No comparison intervention.

Outcomes (O): Accurate scheduler for software projects under dynamic environment.

Context (C): Software project scheduling

Search strategy

The search strategy consists of identification of search terms, data sources and search process as explained below:

Search strings: According to the guidelines in [6], search terms are built as presented below:

a) Major terms are derived from research questions.

b) Synonyms are identified relevant to major terms.

c) Keywords are identified in relevant research papers or books.

d) Boolean OR operator is used to incorporate alternative spellings and synonyms.

e) Boolean AND operator is used to link the major terms.

The resulting search terms are written as follows:

1. Software project scheduling OR (problems/OR techniques/)

2. (Dynamic/OR Optimization) AND Software project scheduling

Data sources: We conducted search on five electronic database resources and used title, abstract and index terms for journals papers, conference proceedings, and workshops, symposiums, and books chapters to primarily extract data. These database resources include: IEEE Xplore, ACM Digital Library, ScienceDirect, Springer, and Google Scholar.

Search process: The search process employed in this research consists of two stages as depicted in Figure 2.

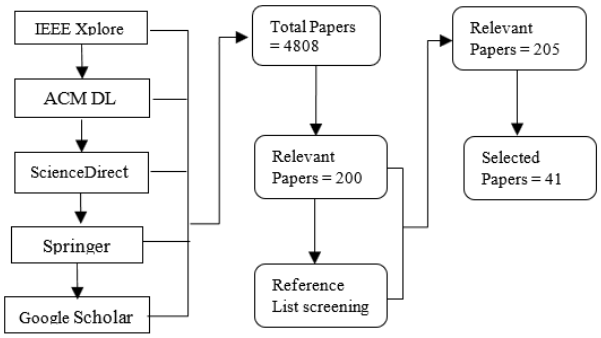

Figure 2: Search and selection process.

Stage 1: Papers were collected by performing as extensive search on five electronic database sources.

Stage 2: In this stage, the screening of reference list of all relevant papers (collected in stage 1) is performed to identify any additional relevant papers and combine them with ones in stage 1.

Data extraction

In search process stage 1, 4808 papers were collected from five electronic databases. Thereafter, titles of these studies were used to eliminate duplicate and irrelevant studies and we got 200 relevant studies consequently. Next, the reference list of each of these 200 papers was screened to detect any relevant study that might have been missed out during initial search process. This effort led to the identification of 5 additional relevant papers and total of selected relevant papers was 205. Table 1 describes inclusion and exclusion criteria.

Table 1: Inclusion/Exclusion Criteria.

Inclusion Criteria	Exclusion Criteria
a. Papers that are published in English language.	a. Papers that are not published in English language.
b. Papers that are most relevant to Software project scheduling	b. The most complete, recent and improved version of a paper is included. The rest are excluded.
c. Papers that are published from 2001-2017.	c. Papers those are not available in full text and have missing information such as publication year, volume was excluded.
d. Papers that have potential to answer at least one research questions.	d. Papers that don't answer research questions.

Quality assessment

Table 2: Quality assessment criteria.

Sr. No.	Quality assessment criteria
1	Are research aims clearly defined?
2	Is the proposed technique clearly described?
3	Are the findings based on multiple project data sets or case study?
4	Do the researchers discuss any problems with the validity of their results?

The quality assessment (QA) checklist for selected studies was customized based on guidelines mentioned in [6]. In Table 2, we formulated the quality assessment questions (Table 2) to evaluate the credibility, completeness and relevance of the selected studies. Each question has only three optional answers: "Yes", "partly" or "No". These three answers have been assigned scores as follows: "Yes" = 1, "Partly" = 0.5 and "No" = 0. Each study could obtain 0-5 points. We used the 2.5 (50%) point as the cut off point for including a study. If a study score less than 2.5 it would be eliminated from our final list of primary studies. Finally, we applied quality assessment criteria and 41 studies were selected. Table 2 quality assessment criteria.

Data synthesis

In this phase, selected studies are summarized to address the research questions precisely. The aim is to synchronize data to enhance the clarity. To synthesize data 41 selected studies were further processed based on criteria defined in Table 3 to access detailed contents of each study. The data extracted in this phase consists of both quantitative and qualitative data. Data related to RQ1 and RQ2, result was presented in a tabular and text form to address taxonomies of scheduling and targeted software models respectively. RQ3 and RQ4 was organized using visualization tools such as pie chart to present the multi-objective techniques used for scheduling problem and types of data sets used respectively. In RQ5, limitations/benefits of existing scheduling technique are described in a tabular form.

Table 3: Data synthesis criteria.

Criteria	Description
Type of study	Journal, conference paper, book chapter.
Study focus	Problems, scope, objectives.
Research methodology	Case study, survey, experiment
Constraints	Limitations and areas for future work.
Application domain	Application domain. For example, academic or industry.

Results and Discussions

This section describes the detailed description of the finding of this review in line with formulated research questions. We selected 41 studies. Out of which 14 were journal papers, 25 conference papers, 1 book chapter, and 1 was article.

Taxonomies of scheduling (RQ1)

The applications of scheduling have wide areas. We divide the taxonomies of scheduling into three different categories:

5.1.1.Project scheduling: It refers to the arranging activities in a sequence and allocation of resources to these activities. The software project scheduling problem is a variant of project scheduling problem [1]. We can define SPSP as the allocating software engineering (employees) to software tasks (activities) in such a way that all tasks are covered to develop a software project. It is a NP-hard problem. The main objective of project scheduling is to minimize the project cost, duration and maximize the quality of project. The most traditional popular methods used in project scheduling are CPM [7] and PERT [8]. Many SBSE approaches have been proposed for it as well. Which will be discussed further in detail.

Machine scheduling: Machine scheduling deals with jobs and machine as resources. It can be further divided into 'Single machine scheduling' and 'Parallel machine scheduling':

A. Single machine scheduling

Single Machine Scheduling Problem (SMSP) constitutes the foundation of scheduling theory. SMSP is the simplest form of scheduling. All other problems arise from it and their role is vital in both theory and practical application. In SMSP, multiple jobs sequencing is done on a single machine. It can be elaborated by running of multiple processes on a single CPU. In a single machine environment makespan (total time) is independent to the schedule. The primary rules for solving SMSP are Shortest-Processing-Time (SPT), Earliest Due Date (EDD), Minimum Slack Time (MST) and Weighted Shortest Processing Time (WSPT). Several techniques like Dynamic Programming (DP), branch & bound approach have been adopted to solve this problem.

B. Parallel machine scheduling

Parallel Machine Scheduling Problem (PMSP) is a generalization of SMSP. If we extend SMSP, the first area is PMSP. Assigning processes on a multi-process computer is an example of PMSP. Performance measures for PMSP are makespan, mean flow time, weighted mean flow time, number of tardy jobs and maximum lateness. Makespan performance measure is meaningful and objective for PMSP. According to the types of machines, Brucker et al. [9] has categorized PMSP into three classes.

a) Identical machines: In this class, the specification of all the machines is same. There is no difference among machines regarding the processing of jobs. All machines process the jobs in the same way.

b) Uniform machines: Each machine have different speed to process a job. In this class, each job requires different processing requirement.

c) Unrelated machines: It is generalized from uniform machines. The processing time of each job is different on different machines.

In PMSP, jobs may have precedence constraints or may be independent of each other.

Resource-constrained project scheduling problem: Resource-Constrained Project Scheduling Problem (RCPSP) [10] finds an optimal schedule that minimizes the project duration and meets the precedence and resource requirements. RCPSP have several kind of resources and each activity requires different quantities of resources. RCPSP is considered as general scheduling problem and open-shop, job-shop, and flow-shop scheduling problems are considered its special cases.

A. Flow-shop scheduling

Flow-shop is a special case of job-shop scheduling problem in which the flow control enables an appropriate sequencing for each job and for processing on a set of machines or with other resources in compliance with given processing orders. There are 'm' machines and 'n' jobs and each machine is bound not to perform more than one operation simultaneously. Performance measures are flowtime, makespan, and tardiness. Flow-shop scheduling problem

can be solved by either exact method such as branch and bound or heuristics algorithms such as genetic algorithm. A special type of flow shop scheduling problem is called permutation flow shop scheduling.

B. Job-shop scheduling

The Job-Shop Scheduling Problem (JSP) is a generalization of flow-shop scheduling problem. In JSP, there are 'n' jobs and 'm' machines, and each job is made of sequence of 'o' operations. Each operation has attached processing time and precedence constraints of jobs are defined between each job operation. The objective is to find the optimal and feasible schedule. The main difference between job-shop and flow-shop scheduling problem is that flow-shop follows a unidirectional sequence while workflow is not unidirectional. Therefore, in the route of jobs machine number consideration is necessary in job-shop scheduling.

C. Open-shop scheduling

This problem is a special case of flow-shop scheduling problem. In open-shop there are no precedence constraints in between the operations of jobs. If there are 'n' jobs and 'm' machines and each job is made of sequence of 'o' operations. Operation 'o' must be processed on machine 'm'. The main purpose is to find the job sequences and machine sequences. Job sequences means that same job order of operations and machine sequences means that on same machine, the order of operations to be performed.

Software models for scheduling (RQ2)

In this research question, we have identified how many studies clearly defined which software process model they have used in their software project scheduling problem. By doing detailed SLR, we have come to know that out of 41 studies; only five studies clearly defined that which software model they have used. Four studies [11-14] use waterfall model and one study [15] use Agile methodology. For rest of studies, we get perception that they are working in waterfall model but still not sure.

Existing optimization techniques (RQ3)

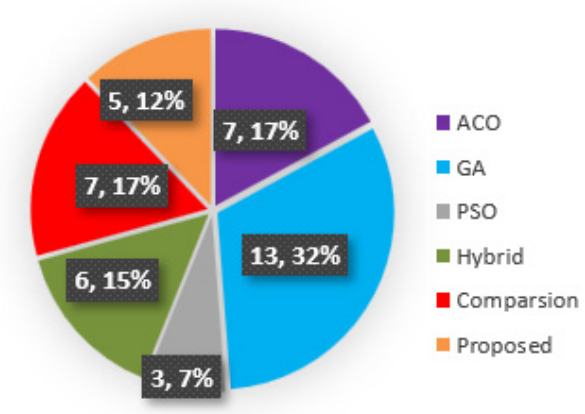

Figure 3: Existing optimization techniques.

In SLR related to software project scheduling, out of 41 studies that were identified, The 13(32%) studies [1,11,13,16-25] used genetic algorithm as optimization technique. Ant colony optimization was used by 7(17%) studies [26-32] to solve the SPS problem. Particle swarm optimization technique was used by only 3(7%) studies [12,33,34]. 5(12%) studies [35-39] used their proposed algorithm to solve this NP-hard problem. Hybrid algorithms were used by 6(15%) studies [14,15,40-43,]. In addition, 7(17%) studies [44-50] did comparison between different metaheuristics algorithms. Hence, we can conclude that genetic algorithm has been used widely to solve the software project scheduling problem. After genetic algorithm, researchers have used ACO to deal with this problem. Following pie chart denotes percentage of each algorithm used (Figure 3).

Data sets used (RQ4)

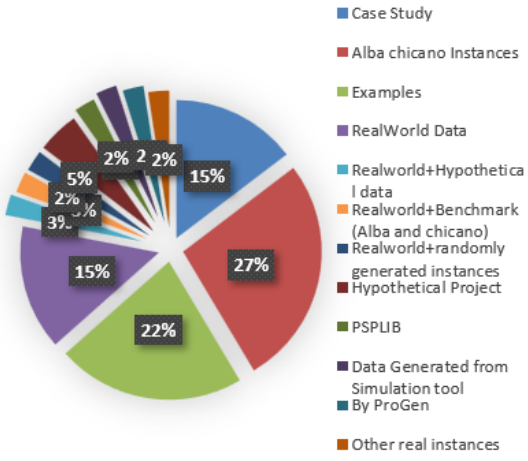

Figure 4: Data used.

In this research question, we have identified that what kind of data sets have been used by researchers to evaluate their proposed techniques. By doing detailed SLR, we found that out of 41 studies, only 6 (15%) studies have used real-world data [14,15,19,36,39,42], 6 (15%) studies considered case studies [12,17, 18,25,29,43] to evaluate their approach. 11(27%) studies [1, 24,30,31,35,41,44-48] used Alba et al. [1] (The employee allocation model of SPSP was first designed by Alba et al. [1]). 9 (22%) studies proved their approach by using examples [13,16,20,32-34,38,40]. One study used combination of real-world and hypothetical data [50]. One study used real-world and benchmark data (Alba et al. [1], [36]. Combination of real world and randomly generated instances was used by one study [27]. Two studies evaluated their approach by using hypothetical project [22,25]. One study used data from PSPLIB [33]. One study used data generated from simulation tool [37]. Data generated by ProGen was used by one study [21]. Other real instances were considered by one study [50]. Most researchers (27% studies) used Alba et al [1] instances to evaluate their proposed approach and only few studies used real-world

data. Hence, we can conclude that more real-world data should be used to evaluate an approach. Following pie chart denotes the percentage of data used (Figure 4).

Limitations and benefits of existing scheduling techniques (RQ5)

We identified following limitations and benefits.

Limitations:

a) Most of the studies deal with two objectives to be optimized usually time and cost.

b) Lack of real world data set usage while evaluating their techniques. Most of the studies have used case studies or some benchmark problems.

c) Most of the studies deal with static SPS considering that no disruption occurs.

d) Lack of real-world and dynamic scenarios inclusion is ignored in most of the studies.

e) Researchers don't clearly define which software model they are being used while dealing with SPS problem.

f) In most of the studies, tasks' efforts are known in advance.

g) While proposing their technique, most of the studies do not do comparison with other already existing techniques.

h) Several studies have used the same problem formulation as defined by Alba et al. [1].

i) The changing objectives problem has not been addressed by any study by considering them as extra dynamic events.

j) Most of the studies evaluate their proposed techniques on small-scale projects.

k) Very few studies have deal with Agile as software model.

l) Developers' skills and expertise, communication overhead models, are not considered.

5.5.1. Benefits:

a) Many studies have included 'robustness' as objective in their proposed technique. Provide robust solution in the presence of uncertainty.

b) Some studies also allowed variation of human factors.

c) Instance generator have been used by researchers to analyze different project scenarios.

d) Some models have also been proposed to consider 'quality' as important objective.

e) Many studies did comparison of different metaheuristic techniques (using same problem formulation of Alba et al. [1]) to identify metaheuristics works best for SPSP.

f) Some studies have dealt with SPSP for global software development.

Research Findings

Software project scheduling under dynamic and uncertain environment is a big challenge for software engineering community. From SLR, the following finding were discovered.

A. Genetic algorithm is most widely used to solve the software project scheduling problem.

B. Most of the studies lack of inclusion of dynamic scenarios.

C. Most studies do not evaluate their approach on real world data.

D. Most studies have used Alba et al. [1] instances to evaluate their approach.

E. Most of the studies use same problem formulation as defined by Alba et al. [1].

Conclusion

This work aims to examine and identify the status quo for dynamic software project scheduling [51,52]. The research method utilized was systematic literature review. In this method, some research questions are formulated and this study revolves around to identify answers of these questions. The essence of this study was to examine the software models that have been targeted for scheduling, scheduling techniques, their taxonomies and their limitations/benefits and ultimately to identify areas for future research.

References

1. Alba E, Chicano JF (2007) Software project management with gas. Inf Sci 177(11): 2380–2401.
2. Parunak Van Dyke H (1991) Characterizing the manufacturing scheduling problem. Journal of manufacturing systems 10(3): 241-259.
3. (2015) Project planning tools - popularity ranking. Project Management Zone.
4. Rothlauf F (2011) Optimization methods. In: Design of modern heuristics. Natural Computing Series. Springer, Berlin, Heidelberg, Germany.
5. Harman M, Jones BF (2001) Search-based software engineering. Information and software Technology 43(14):833-839.
6. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. EBSE 2007-001.
7. Golenko-Ginsburg Dand, Ganik A (1997) Stochastic network project scheduling with non-consumable limited resources. Int l J Production Econ 48(1): 29-37.
8. Wiest JD, Levy FK (1977) A Management Guide to PERT/CPM: with GERT/PDM/CPM and Other Networks. Prentice-Hall , Englewood Cliffs, NJ, USA, p. 178-196.
9. Bruker P, Schli R (1990) Job-shop scheduling with multi-purpose machines. Computing 45: 369-375.
10. Hartmann S (1998) A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics (NRL) 45(7): 733-750.
11. Seo D, Shin D, Bae DH (2015) Quality based software project staffing and scheduling with cost bound. 2015 Asia-Pacific Software Engineering Conference (APSEC) IEEE, New Delhi, India, pp. 269-276.

12. Gonsalves T, Ito A, Kawabata R, Itoh K (2008) Swarm intelligence in the optimization of software development project schedule. 2008 32nd Annual IEEE International Computer Software and Applications Conference IEEE, Turku, Finland, pp. 587-592.

13. Shan X, Jiang G, Huang T (2010) The optimization research on the human resource allocation planning in software projects. 2010 International Conference on Management and Service Science, IEEE, Wuhan, China, pp. 1-4.

14. Antoniol G, Di Penta M, Harman M (2004) A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. 10th International Symposium on Software Metrics, 2004. Proceedings, IEEE, Chicago, IL, USA, pp. 172-183.

15. Britto R, Neto PS, Rabelo R, Ayala W, Soares T (2012) A hybrid approach to solve the agile team allocation problem. 2012 IEEE Congress on Evolutionary Computation, IEEE, Brisbane, QLD, Australia, pp. 1-8.

16. Hanchate DB (2010) Analysis, mathematical modeling and algorithm for software project scheduling using BCGA. 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, IEEE, Xiamen, China, 3: 1-7.

17. Ge Y (2009) Software project rescheduling with genetic algorithms. 2009 International Conference on Artificial Intelligence and Computational Intelligence, IEEE, Shanghai, China, pp. 439-443.

18. Ge Y, Chang C (2006) Capability-based project scheduling with genetic algorithms. 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), IEEE, Sydney, NSW, Australia, pp. 161-161.

19. Kroll J, Friboim S, Hemmati H (2017) An empirical study of search-based task scheduling in global software development. 2017 IEEE/ ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), IEEE Press, Buenos Aires, Argentina, pp. 183-192.

20. Chang CK, Christensen MJ, Zhang T (2001) Genetic algorithms for project management. Ann Softw Eng 11: 107–139.

21. Yannibelli V, Amandi A (2011) A knowledge-based evolutionary assistant to software development project scheduling. Expert Systems with Applications 38(7): 8403-8413.

22. Chang CK, Jiang H, Di Y, Zhu D, Ge Y (2008) Time-line based model for software project scheduling with genetic algorithms. Inf Softw Technol 50: 1142-1154.

23. Duggan J, Byrne J, Lyons GJ (2004) A task allocation optimizer for software construction. IEEE software 21(3):76-82.

24. Stylianou C, Andreou AS (2011) Intelligent software project scheduling and team staffing with genetic algorithms. In Artificial Intelligence Applications and Innovations, Springer, Berlin, Heidelberg, Germany, pp. 169-178.

25. Xiao J, Wang Q, Li M, Yang Q, Xie L, et al. (2009) Value-based multiple software projects scheduling with genetic algorithm. Proc Int l Conf Software Process 5543: 50-62.

26. Suri B, Jajoria P (2013) Using ant colony optimization in software development project scheduling. 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, Mysore, India, pp. 2101-2106.

27. Chen WN, Zhang (2013) Ant colony optimization for software project scheduling and staffing with an event-based scheduler. IEEE Trans Softw Eng 39(1): 1-17.

28. Han W, Zhang X, Jiang H, Li W (2014) An Ant Colony Optimization Algorithm for Software Project Management. 2014 7th International Conference on Control and Automation, IEEE, Hainan, China, pp. 19-23.

29. Zhang W, Yang Y, Xiao J, Liu X, Babar MA (2015) Ant colony algorithm based scheduling for handling software project delay. In Proceedings of the Int'l Conference on Software and System Process, pp. 52-56.

30. Crawford B, Soto R, Johnson F, Monfroy E, Paredes F (2014) A max–min ant system algorithm to solve the software project scheduling problem. Expert Systems with Applications 41(15): 6634-6645.

31. Xiao J, AO XT, Tang Y (2013) Solving software project scheduling problems with ant colony optimization. Computers & Operations Research 40(1): 33-46.

32. Crawford B, Soto R, Johnson F, Monfroy E (2013) Ants can schedule software projects. In International Conference on Human-Computer Interaction, Springer, Berlin, Heidelberg, Germany, pp. 635-639.

33. Jia YH, Chen WN, Hu XM (2014) A PSO approach for software project planning. In Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, pp. 7-8.

34. Gonsalves T, Itoh K (2010) Multi-objective optimization for software development projects. In Proceedings of the international multi conference of engineers and computer scientists 2010, Hong Kong, China.

35. Minku LL, Sudholt D, Yao X (2014) Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis. IEEE Trans Softw Eng 40(1): 83-102.

36. Shen X, Minku LL, Bhasoon R, Yao X (2016) Dynamic software project scheduling through a proactive-rescheduling method. IEEE Trans Softw Eng 42(7): 658-686.

37. Rodríguez D, Ruiz M, Riquelme JC, Harrison R (2011) Multiobjective simulation optimisation in software project management. In Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM, pp. 1883-1890.

38. Hanne T, Nickel S (2005) A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. European Journal of Operational Research 167(3): 663-678.

39. Ren J, Harman M, Penta MD (2011) Cooperative co-evolutionary optimization of software project staff assignments and job scheduling. In Proceedings of the Third International Conference on Search Based Software Engineering. SSBSE'11. Berlin, Heidelberg, Springer-Verlag, Germany, pp. 127-141.

40. Stylianou C, Gerasimou S, Andreou AS (2012) A novel prototype tool for intelligent software project scheduling and staffing enhanced with personality factors. 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, IEEE, Athens, Greece, 1: 277-284.

41. Xiao J, Gao ML, Huang MM (2015) Empirical study of multi-objective ant colony optimization to software project scheduling problems. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, pp. 759-766.

42. Gueorguiev S, Harman M, Antoniol G (2009) Software project planning for robustness and completion time in the presence of uncertainty using multi-objective search based software engineering. Proc 11th Annu Geneic Evol Compu Conf pp. 1673-1680.

43. Barreto A, Barros MD, Werner CM (2008) Staffing a software project: A constraint satisfaction and optimization-based approach. Computers & Operations Research 35(10): 3073-3089.

44. Luna F, González-Álvarez DL, Chicano F, Vega-Rodríguez MA (2011) On the scalability of multi-objective metaheuristics for the software scheduling problem. 2011 11th International Conference on Intelligent Systems Design and Applications, IEEE, Cordoba, Spain, pp. 1110-1115.

45. Chicano F, Luna F, Nebro AJ, Alba E (2011) Using multi-objective metaheuristics to solve the software project scheduling problem. In Proceedings of the 13th annual conference on Genetic and evolutionary

computation, ACM, pp. 1915-1922.

46. Al Khatib SM, Noppen J (2017) Benchmarking and comparison of software project human resource allocation optimization approaches. ACM SIGSOFT Software Engineering Notes 41(6): 1-6.

47. Luna F, González-Álvarez D, Chicano F, Vega-Rodríguez MA (2014) The software project scheduling problem: A scalability analysis of multi-objective metaheuristics. Appl Soft Comput 15: 136-148.

48. Luna F, Chicano F, Alba E (2012) Robust solutions for the software project scheduling problem: A preliminary analysis. International Journal of Metaheuristics 2(1): 56-79.

49. Crawford B, Soto R, Johnson F, Misra S, Paredes F (2014) The use of metaheuristics to software project scheduling problem. In International Conference on Computational Science and Its Applications, Springer, Cham, pp. 215-226.

50. Chicano F, Cervantes A, Luna F, Recio G (2012) A novel multiobjective formulation of the robust software project scheduling problem. Applications of Evolutionary Computation, Springer, NY, USA, pp. 497-507.

51. Nuevo E, Piattini M, Pino FJ (2011) Scrum-based methodology for distributed software development. 2011 IEEE Sixth International Conference on Global Software Engineering, IEEE, Helsinki, Finland, pp. 66-74.

52. Juran JM (1988) Juran's Quality Control Handbook, McGraw-Hill, New York, USA.