ISSN (Print) 2307-6968, ISSN (Online) 2663-2209
Вчені записки Університету «КРОК» №4 (76), 2024

УДК: 005.8:519.8
DOI: 10.31732/2663-2209-2024-76-256-265

ПЛАНУВАННЯ ПРОЕКТІВ У РОЗПОДІЛЕНОМУ СЕРЕДОВИЩІ З ВРАХУВАННЯМ РОБОЧИХ ГРАФІКІВ ПРАЦІВНИКІВ

Олександр Боголій
Аспірант, ВНЗ «Університет економіки та права «КРОК», м. Київ, Україна, email: boholiiom@krok.edu.ua, ORCID: https://orcid.org/0000-0003-0253-667X

PROJECT SCHEDULING IN A DISTRIBUTED ENVIRONMENT CONSIDERING EMPLOYEES' WORKING HOURS

Oleksandr Bogolii
PhD Student, Higher Education Institution the "KROK" University, Kyiv, Ukraine, email: boholiiom@krok.edu.ua ORCID: https://orcid.org/0000-0003-0253-667X

Анотація: *Останніми роками розподілена розробка програмного забезпечення набула значної популярності, дозволяючи компаніям підвищувати продуктивність завдяки використанню глобальних ресурсів, водночас знижуючи витрати на виробництво та скорочуючи час виходу на ринок. Проте, така організаційна модель ставить перед управлінцями низку викликів. Одним із таких викликів є складність планування завдань у віддаленому, розподіленому середовищі. Окрім традиційних факторів, які враховуються в умовах спільної роботи на одному місці, менеджери повинні зважати на різноманітність робочих годин та часових поясів, в яких працюють географічно розподілені члени команди. Незважаючи на те, що за останні десятиліття було розроблено багато методів планування, обмежена кількість досліджень присвячена плануванню з урахуванням робочих годин співробітників. Метою цього дослідження є розробка нового підходу до планування, який враховує календарні обмеження співробітників та надати цінні поради для керівників проектів, особливо тих, що працюють у віддалених, розподілених середовищах. Запропонована методологія включає розробку нового алгоритмічного підходу для створення оптимального плану проекту, який враховує робочий час працівників. Порівняльний аналіз із класичним двофазним методом планування для розподілених команд показав потенціал скорочення загальної тривалості проекту на 6% і продемонстрував особливу ефективність у проектах, що характеризуються високою складністю графа завдань. Крім того, експерименти показали, що планування з урахуванням робочого часу є ще ефективнішим, коли різниця в часових поясах між підкомандами становить приблизно 8 годин, що відповідає типовому робочому дню працівника. В подальшому запропонований підхід можна додатково покращити, враховуючи додаткові фактори та обмеження в процесі розподілу ресурсів, зокрема необхідність синхронізації між інженерами, які працюють у різних часових поясах.*

Ключові слова: *планування проєктів; робочий графік; гнучка розробка програмного забезпечення; розподілене середовище; часовий пояс.*

Формул: 0, рис.: 9, табл.: 1, бібл.: 20

Abstract: *In recent years, distributed software development has gained significant popularity, enabling companies to enhance productivity by leveraging global resources while simultaneously reducing production costs and time-to-market. However, this organizational model presents management with distinct challenges. One such challenge lies in the complexity of scheduling tasks in a remote, distributed environment. In addition to the traditional factors considered in co-located settings, managers must now account for the diverse working hours and time zones of geographically dispersed team members. Although numerous scheduling techniques have been developed in recent decades, limited research has focused on scheduling in relation to employees' working hours. This research aims to develop a novel scheduling approach that incorporates employee calendar constraints and provides valuable insights for project managers, particularly those operating in remote, distributed environments. The proposed methodology encompasses the development of a new algorithmic approach to produce an optimal project schedule that accounts for employee working hours. Comparative analysis against classical two-phase calendarization method and co-located setups showed the potential to reduce overall project duration by 6% and demonstrates particular efficiency in projects characterized by high task graph complexity. In addition, experiments showed that scheduling with consideration of working hours is even more effective when the time zone difference between subteams is approximately 8 hours, aligning with the typical employee workday. In the future, the proposed technique can be further refined by considering additional factors and constraints in the resource allocation process, specifically the need for synchronization between engineers working in different time zones.*

Keywords*: project scheduling; calendarization; agile software development; distributed environment; time zone.*

Formulas: 0, fig.: 9, tabl.: 1, bibl.: 20

1.Introduction

In a remote, distributed setup, members of one team often work in different time zones. In such a setup, it is important to adopt task scheduling approaches that take into account different working hours. It is especially important when tasks are interconnected, and team members can wait much longer while dependent tasks are completed by their colleagues in other time zones. Figure 1 shows an example of a project that consists of three interconnected tasks. Tasks are presented in circles, with numbers inside depicting the numero of the task at the top and the time needed to complete the task at the bottom. Arrows show dependencies between tasks.

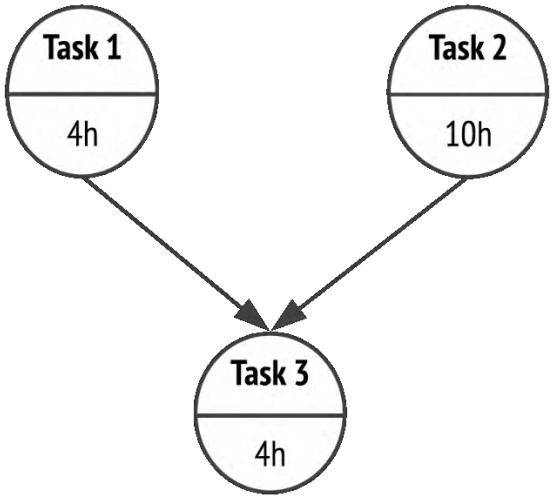

Figure 1. Project with 3 interconnected tasks
Source: Figure created by authors

Let's assume we have 2 employees that work on a project with the same working schedule, 8 hours per day. If we use the Earliest Finish Time (EFT) scheduling algorithm, the project could be finished by 14 o'clock on the second day. Figure 2 shows the schedule of the project in this setup in the form of a timing diagram, which is used to illustrate the allocation of the parallel project tasks among the team members and the execution order of the tasks.

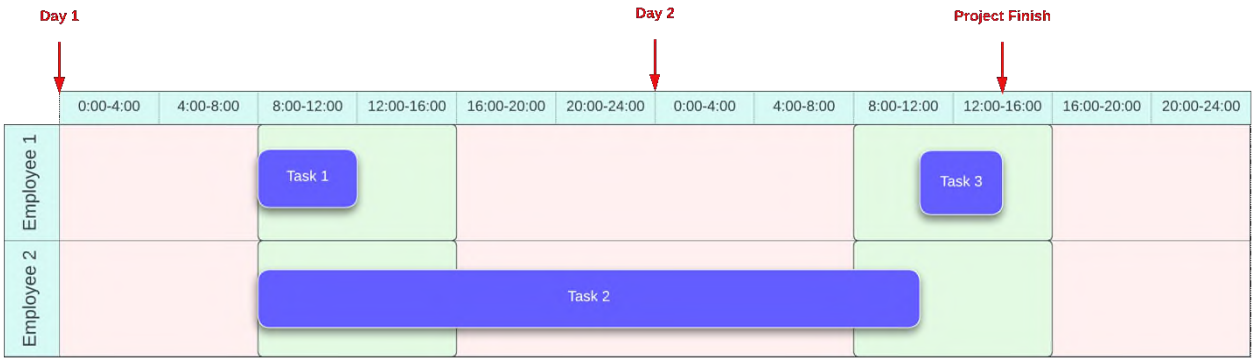

Figure 2. Project schedule when employees have the same working hours
Source: Figure created by authors

If we take 2 employees that work in different time zones, for example, first in Eastern Europe (GMT+1) and second in California (GMT-7). Using the same task assignments as above but not adapting to the time difference may result in a slightly longer project finish time. In Figure 3, we see that the first employee cannot start working on the third task, as it depends on task 2, which is assigned to his colleague. The planned project finish is at 12 o'clock on the third day.

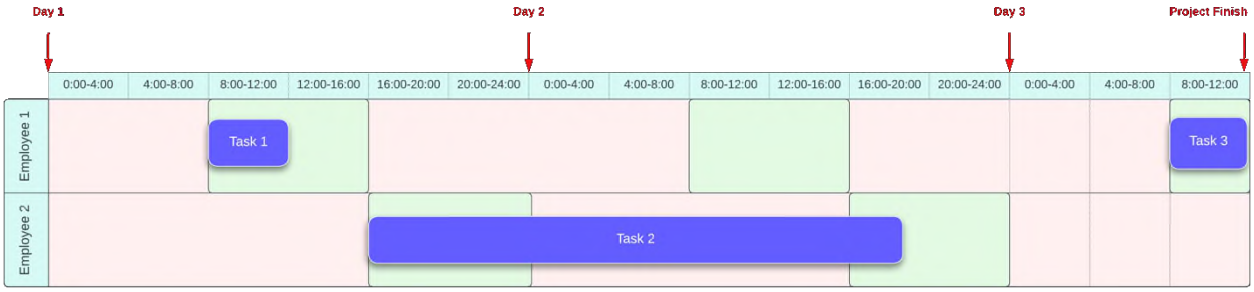

Figure 3. Project schedule when employees have different working hours
Source: Figure created by authors

If we adopt our scheduling algorithm to take into consideration working hours, we can improve project timing. EFF will assign the second task to the first employee, as with his schedule, he will finish it faster than if it is assigned to the second employee. With such a schedule, the project will be finished at 16 o'clock on the second day (Figure 4).

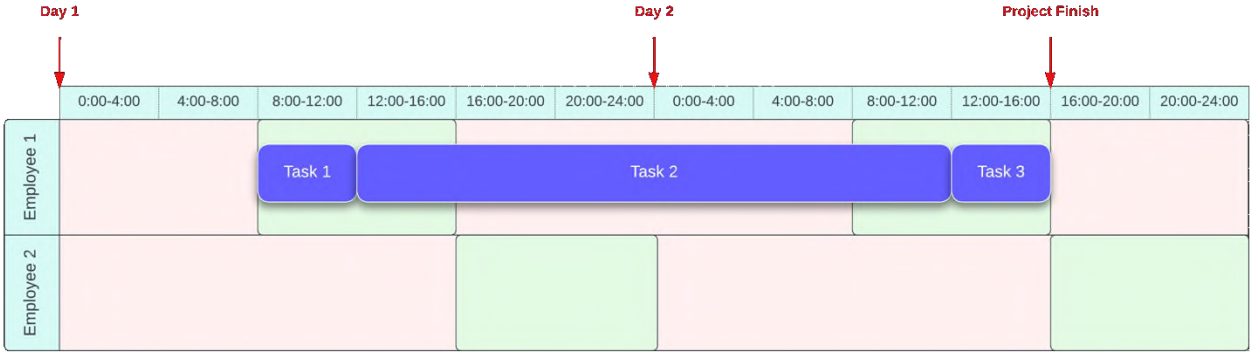

Figure 4. Improved project schedule when employees have different working hours
Source: Figure created by authors

The remainder of the paper is organized as follows: Section 2 reviews related literature on project scheduling; Section 3 introduces the proposed algorithm for task scheduling, which accounts for employees' working hours; Section 4 details the experimental setup used to evaluate the algorithm; and Section 5 presents a discussion of the evaluation results. Finally, Section 6 offers the paper's conclusion.

2.Literature Review

Project Managers often use different scheduling techniques and tools to improve planning and organization, optimize resource allocation, reduce risk and uncertainty, and increase accountability (Fox & Spence, 1998; Pollack-Johnson & Liberatore, 1998).

The Critical Path Method (CPM) and the Program Evaluation and Review Technique (PERT) are the two most popular approaches for project scheduling.

CPM (Moder, 1988) is a deterministic technique that utilizes a task graph where each task is allocated a deterministic duration. CPM computes the longest path within this graph, known as the "critical path." The "critical path" length is the earliest project completion time (Khodakarami et al., 2007).

PERT (Malcolm et al., 1959) is another network technique. It uses a statistical approach to calculate the probability of projects and tasks being completed on time. PERT requires three different task duration estimates: pessimistic, optimistic, and most likely. Then the "critical path" and the start and finish dates are calculated. PERT is beneficial when there are significant variations in optimistic and pessimistic estimates and great uncertainty regarding project outcomes.

Both CPM and PERT assume that the resources required by project activities exist in unlimited quantities. In reality, practitioners often face high contention for scarce resources, which frequently causes missed deadlines and commitments to stakeholders. To prevent this, a feasible plan must be implemented, which requires the reflection of a limited number of

resources. This is a standard problem in project management and is often referenced in literature as a Resource-Constrained Project Scheduling Problem (RCPSP) and shown to be an NP-hard problem (Blazewicz et al., 1983). Many RCPSP techniques have been developed that seek to achieve the earliest project completion, considering dependencies between tasks and resource constraints. RCPSP strategies often apply some heuristics from real life (Goldratt, 1997), make use of constraint programming (Kreter et al., 2017; Vanhoucke & Coelho, 2016), genetic algorithms (Zhang et al., 2008), and neural networks (Golab et al., 2023).

Critical Chain Project Management (CCPM), proposed by Goldratt (1997), is one of the most well-known resource-constrained planning strategies. The core concept of CCPM is the identification and management of the project's critical chain, which is the sequence of tasks that determines the project's overall duration. CCPM has proven to be effective in resolving resource contentions as well as tackling problems concerning human resource behavior.

In most cases, researchers work on new RCPSP solutions, taking into consideration some assumptions from real life, like the type of available resources (renewable, non-renewable, double-constrained), project activity characteristics (preemptive, varying in time, multi-mode, etc.), objection function type (time-based, economic, resource-based, and others), and availability of information (deterministic, non-deterministic) (Habibi et al., 2018).

However, despite numerous RCPSP techniques being proposed in recent decades, very little research has been done to address scheduling concerning employees' working hours. Zhan (1992) presented a method for time planning for a project with regard to the working and non-working days of employees (calendarization problem). This method combines two phases. In the first phase, the earliest start times of activities for a project are determined without considering the calendar. In the second phase, start times are mapped to the dates on employee’s calendars.

Another algorithm for project scheduling with calendar constraints was proposed by Franck et al. (2001). The proposed algorithm considers minimum and maximum time lags between activities and time intervals during which some resources, such as manpower, are not available and examines different priority rules for the selection of the next eligible activity during scheduling. This method schedules activities with regard to the working-time calendar common to all resources, while in practice, different resources generally have different calendars.

Project scheduling with different calendars is especially vital for the software development industry, where engineers, even in the same team, often work from different locations and in different time zones. Having a way to efficiently allocate project tasks among team members, taking into consideration the specifics of modern software development, would be beneficial.

3.Methodology

This research paper seeks to determine whether adapting widely used scheduling techniques to align with employees' working hours can enhance overall project completion time.

In this section, we define an algorithm for project scheduling for distributed software development teams. The proposed algorithm uses the following assumptions, typical for the majority of agile software development teams:

-Each project task can be assigned to exactly one engineer.

-Each engineer can work at the same time on only one task (no multitasking).

-Each engineer has its own working time calendar.

-Engineers can work on any task from the project.

At first, we define the project task queue, together with the corresponding dependencies set D_i for each task i. In addition, we define working time calendars C_k for each team member k from the team. Figure 5 illustrates the proposed algorithm.

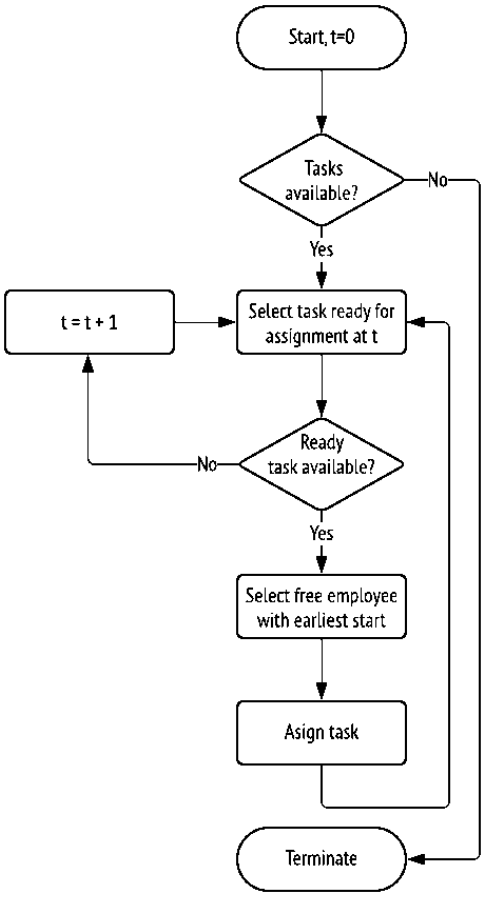

Figure 5. Calendarization algorithm
Source: Figure created by authors

We start our algorithm with the first time point $t = 0$, and proceed with the next steps:

1.Select the highest priority available task for assignment. If all tasks are already assigned, terminate algorithm execution.

2.If no tasks are available at the moment t, meaning dependencies are not yet resolved, we repeat from step 1 for $t = t + 1$.

3.If there is a task, that is ready to be executed, we are iterating among free team members, and for each team member k, we are calculating the earliest start time for tasks ES_{ki} according to their working time calendar. We assign the task to the team member k who has the earliest ES_{ki}

4.Repeat assignments from step 1.

We conducted several experiments to evaluate the proposed algorithm against the standard two-phase calendarization technique, where scheduling is performed first without considering the calendar, and then start times are mapped to the dates on employees' calendars. In addition, we evaluated the project duration for the local team, where all employees are working in the same time zone.

Each experiment consisted of 100 projects. We randomly generated a task set for each project using the model proposed by Watts & Strogatz (1998) to generate relationships between tasks. This model can capture both randomness and clustering, which are common features in real projects.

In our experiments, we used different project and team configurations to show the impact of different factors, such as the varying number of task dependencies and time differences between sites. We have considered the following five different scheduling parameters: number of tasks, number of dependencies between tasks, team size, size 1 time zone, and site 2 time zone.

To save costs, many companies from North America and Western countries

outsource software development to an overseas engineering team. According to an analysis performed by Divakova (2023), the most comfortable locations for software development outsourcing are Eastern Europe (Ukraine, Poland, the Czech Republic, and Romania) and Asia (India, China, the Philippines, and Vietnam). In conducted experiments, we simulate setup when the engineering team is extended with outsourced engineers at another time zone:

1.The engineering team in Western Europe outsources to Asia (5 hours difference).

2.The engineering team in California outsources to Eastern Europe (8 hours difference).

3.The engineering team in California outsources to Asia (13 hours difference).

Multiple engineering practitioners agree in the opinion that the optimal Agile team size is around 5-7 members (Levison, 2020; Cohn, 2024). In our experiments, we assumed that the team consisted of six members. The average task duration can vary significantly depending on various factors such as the complexity of tasks, team experience, team velocity, and the nature of the project. In Agile methodologies like Scrum, estimation is often performed in story points, which represent a relative effort to accomplish a task. Big stories are usually broken down into smaller, manageable tasks that can be done in 1 to 3 days (Fuqua, 2015). For our experiments, we assigned each task a random story point value from the Fibonacci sequence (Cohn, 2022) with a maximum of 8 and then converted it to absolute time by multiplying it by team velocity, which is chosen based on maximum task duration. We used a total of 15 tasks per project, which is the typical number of tasks in Agile Sprint (Fuqua, 2015).

For experiments 1–3, we change the value of dependencies between tasks while keeping the other parameters constant. Having more dependencies between tasks makes it harder to parallelize the work, increasing the importance of coordination between employees. Thus, having an efficient scheduling approach that makes use of working hours is supposed to be beneficial.

In experiment 4, we study the impact of project task duration. We increased the maximum task duration from 8 to 24 hours with an 8-hour step. We want to verify if the proposed approach can take advantage of shorter task durations by having a portion of tasks assigned and finished before other employees' time zone shifts, resulting in a more optimal project schedule.

Table **1** depicts the experimental setup for this study.

Table 1. Experimental setup

№	Number of Tasks	Number of Dependencies	Time zone 1	Time zone 2	Time Difference (hours)	Team Size	Max Task Duration (hours)
1	15	15-60 (step 15)	Western Europe (GMT+01:00)	Asia (GMT+06:00)	5	6	24
2	15	15-60 (step 15)	California (GMT-07:00)	Eastern Europe (GMT+01:00)	8	6	24

№	Number of Tasks	Number of Dependencies	Time zone 1	Time zone 2	Time Difference (hours)	Team Size	Max Task Duration (hours)
3	15	15-60 (step 15)	California (GMT-07:00)	Asia (GMT+06:00)	13	6	24
4	15	30	California (GMT-07:00)	Eastern Europe (GMT+01:00)	8	6	8-24

Source: Table created by authors

4. Results and Discussion

In the first experiment, we tested a setup where a team from Western Europe outsources to Asia. The time difference is 5 hours. The experiment results presented in Figure 6 show that the proposed resource task allocation strategy performs significantly better than two-phase calendarization and prove that distributed work from different time zones could lead to faster project delivery than work from the same location (time zone). It is even more beneficial to use the proposed approach as the number of task dependencies increases.

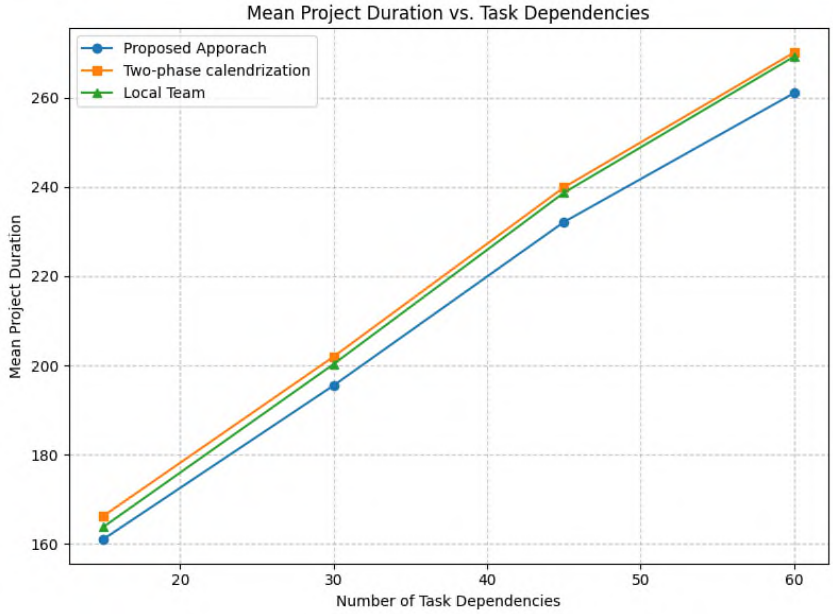

Figure 6. Results of the 1st Experiment

Source: Figure created by authors

Figure 7 and Figure 8 show results for experiments 2 and 3, respectively. It is clear from the charts that the proposed approach is beneficial in these experiments as well. Another interesting observation, derived from experiments 2 and 3, is that without efficient task allocation, the distribution of development may not always lead to a reduction in project duration. We can also notice that the maximum gain from distributed development is achieved when the time zone difference between subteams is 8 hours (second experiment), which correlates to the working day hours of employees, resulting in a more optimal concurrent task schedule.

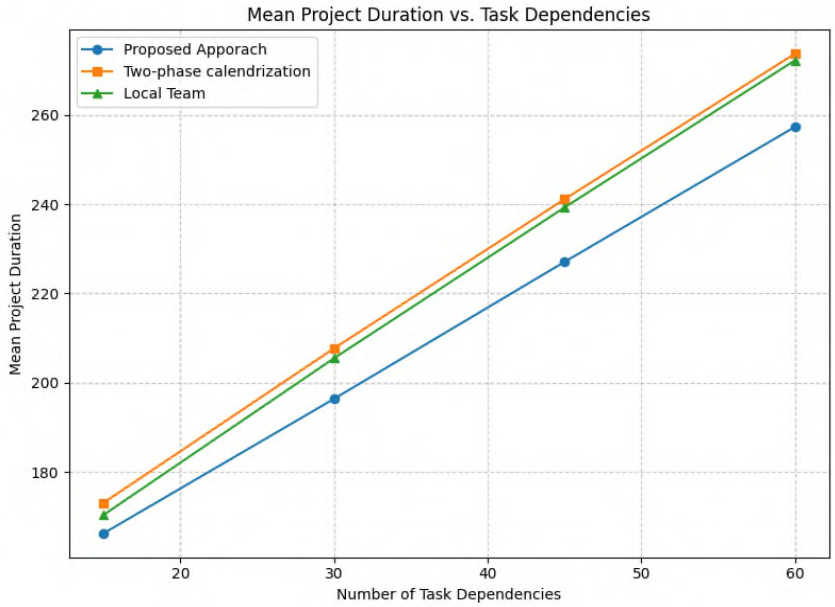

Figure 7. Results of the 2nd Experiment
Source: Figure created by authors

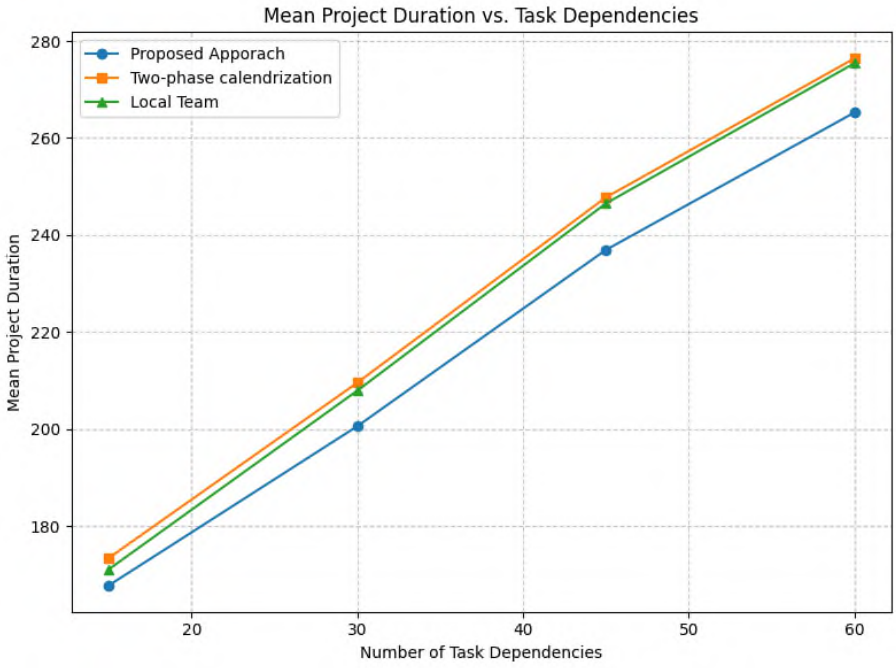

Figure 8. Results of the 3rd Experiment
Source: Figure created by authors

Figure 9 presents the results of the 4th experiment, which confirms the impact of task durations on possible project duration. Project makespan tends to increase between the proposed and two-phase strategy when we increase the maximum project task duration.

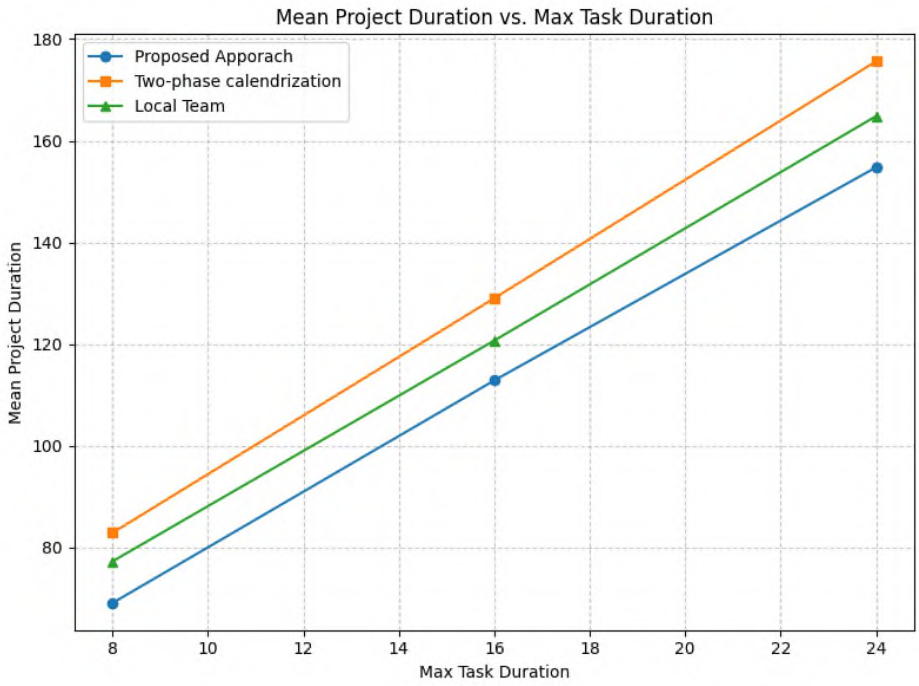

Figure 9. Results of the 4th Experiment
Source: Figure created by authors

On average, the scheduling technique presented in this paper resulted in approximately 6% faster project completion time in comparison to the two-phase calendarization approach.

Conclusion

In this paper, we study the resource-constrained project scheduling problem to minimize the software development project's makespan with regard to employee calendar constraints. Unlike the previous approach, where scheduling was first performed without considering the calendar and then start times were mapped to the dates of employees' calendars, we proposed a method to perform project scheduling that takes into account employees' working hours before task assignment.

Different parameters were tested to evaluate the effectiveness of the proposed strategy compared to its counterpart. To be closer to the real world, we conducted experiments for popular distributed team configurations, each containing the onsite part in one time zone and the outsource part in another. Overall, there were three experiments, with varying time differences between subteams ranging from 5 to 13 hours. In each experiment, the value of task dependencies increased while keeping the other factors constant.

After conducting extensive experiments, it was proven that the proposed technique was up to 6% more efficient than the classical two-phase approach. The efficiency increases with the increase in project task graph complexity.

It is also observed that the use of the proposed approach is even more efficient when the time zone difference between subteams is 8 hours in comparison to a shorter 5 or 13 hours, which correlates to the employee's working day duration.

In the future, the proposed technique can be further improved by considering additional factors and constraints in the resource allocation process, specifically the need for synchronization between engineers working in different time zones.

References:

1.Blazewicz, J., Lenstra, J. K., & Kan, A. H. G. R. (1983). Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics, 5(1), 11–24. https://doi.org/10.1016/0166-218X(83)90012-4

2.Golab, A., Gooya, E. S., Falou, A. A., & Cabon, M. (2023). A convolutional neural network for the

resource-constrained project scheduling problem (RCPSP): A new approach. Decision Science Letters, 12(2), 225–238. https://doi.org/10.5267/j.dsl.2023.2.002

3.Goldratt, E. M. (1997). Critical chain. Great Barrington: The North River Press Publishing Corporation.

4.Khodakarami, V., Fenton, N., & Neil, M. (2007). Project Scheduling: Improved Approach to Incorporate Uncertainty Using Bayesian Networks. Project Management Journal, 38(2), 39–49. https://doi.org/10.1177/875697280703800205

5.Franck, B., Neumann, K., & Schwindt, C. (2001). Project scheduling with calendars. OR-Spektrum, 23(3), 325–334. https://doi.org/10.1007/PL00013355

6.Habibi, F., Barzinpour, F., & Sadjadi, S. J. (2018). Resource-constrained project scheduling problem: Review of past and recent developments. Journal of Project Management, 55–88. https://doi.org/10.5267/j.jpm.2018.1.005

7.Kreter, S., Schutt, A., & Stuckey, P. J. (2017). Using constraint programming for solving RCPSP/max-cal. Constraints, 22(3), 432–462. https://doi.org/10.1007/s10601-016-9266-6

8.Fox, T. L., & Spence, J. W. (1998). Tools of the Trade: A Survey of Project Management Tools. Project Management Journal, 29(3), 20–27. https://doi.org/10.1177/875697289802900305

9.Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a Technique for Research and Development Program Evaluation. Operations Research, 7(5), 646–669. https://doi.org/10.1287/opre.7.5.646

10.Moder, J. (1988). Network techniques in project management. Project Management Handbook, 2nd edition (pp. 324–373). New York: Van Nostrand Reinhold.

11.Pollack-Johnson, B., & Liberatore, M. J. (1998). Project Management Software Usage Patterns and Suggested Research Directions for Future Developments. Project Management Journal, 29(2), 19–28. https://doi.org/10.1177/875697289802900205

12.Vanhoucke, M., & Coelho, J. (2016). An approach using SAT solvers for the RCPSP with logical constraints. European Journal of Operational Research, 249(2), 577–591. https://doi.org/10.1016/j.ejor.2015.08.044

13.Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. Nature, 393(6684), 440–442. https://doi.org/10.1038/30918

14.Zhan, J. (1992). Calendarization of time planning in MPM networks. ZOR, Zeitschrift Für Operations Research Methods and Models of Operations Research, 36(5), 423–438. https://doi.org/10.1007/BF01415759

15.Zhang, H., Xu, H., & Peng, W. (2008). A Genetic Algorithm for Solving RCPSP. 2008 International Symposium on Computer Science and Computational Technology, 246–249. https://doi.org/10.1109/ISCSCT.2008.255

16.Cohn, M. (2024, January 2). The Ideal Size for Your Agile Team. https://www.mountaingoatsoftware.com/blog/the-just-right-size-for-agile-teams

17.Levison, M. (2020, March 13). Scrum Team Size - How Big? How Small? Agile Pain Relief. https://agilepainrelief.com/blog/scrum-team-size.html

18.Fuqua, A. (2015, May 29). Agile Story Points: How Many Stories Per Sprint? LeadingAgile. https://www.leadingagile.com/2015/05/agile-story-points-how-many-user-stories-per-sprint-rules-of-thumb/

19.Divakova, G. (2023, October 16). Offshore Developers — Rates in 2024: Best Countries and Best Platforms. YouTeam. https://youteam.io/blog/offshore-developers-rates-in-2020-best-countries-and-best-platforms-to-hire-a-remote-development-team/

20.Cohn, M. (2022, September 10). Agile Estimation: Why The Fibonacci Sequence Works. Mountain Goat Software. https://www.mountaingoatsoftware.com/blog/why-the-fibonacci-sequence-works-well-for-estimating