International Journal of Industrial Engineering, 29(2), 154-173, 2022

CLUSTER-BASED PRIORITY LIST GENERATION FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS

Ahmet Melik Öztürk[1] and Chulung Lee[2, *]

[1]Department of Industrial and Management Engineering
Korea University
Seoul, Korea

[2]School of Industrial and Management Engineering
Korea University
Seoul, Korea
[*]Corresponding author's e-mail: leecu@korea.ac.kr

Constructive Heuristics for the Resource Constraint Project Scheduling Problems (RCPSP) are preferred scheduling methods when the project network broadens. Then, to generate a good schedule from these heuristics, the priority list used in the algorithm becomes crucial. This paper proposes a Cluster-Based Priority List (CB-PL) method for generating lists to improve makespans of schedules obtained from constructive heuristics. The method creates more intellectual priority lists that generate lower makespans. The approach is built and fine-tuned upon the existing relative literature. The performance of the method is measured by comparing the makespan results. The experiment for the comparison uses serial and parallel scheduling schemes with seven priority rules. Then the experiment is tested through a set of benchmark data. Finally, schedules obtained through the CB-PL showed significant makespan reductions and increases in an overall number of better solutions.

Keywords: RCPSP; Clustering; Project network; Priority list; Heuristic

(Received on October 26, 2021; Accepted on March 29, 2022)

1. INTRODUCTION

The Resource-Constrained Project Scheduling Problem (RCPSP) is one of the primary keys to success in project management as it is the successor of the common project scheduling problem. The RCPSP aims to generate an applicable schedule for the projects with resource consumption. So, the scheduling becomes a deciding factor for the initial makespan planning for the projects. Therefore, the makespans obtained by these schedules are essential for the projects' on-time completion (Herroelen, 2005; Habibi *et al.*, 2018). This problem has mostly been seen as the team's problem of being ineffective in their scheduling. The basic version of the RCPSP is a single project including numerous activities to be scheduled as soon as possible. The problem contains precedence and successor relationship, resource consumption, and duration that it takes for the activities in the project. Also, the required resources for the project are assumed renewable.

The RCPSP is NP-hard (Blazewicz *et al.*, 1983). Due to this situation, the problem is hard to solve for an optimal schedule in a reasonable time, even with projects that have 30 activities (Coelho and Vanhoucke, 2018). The general solution for this immense time consumption of optimal solutions is to use heuristic algorithms (Demeulemeester and Herroelen, 2002; Roy and Sen, 2019). Heuristics used to create feasible schedules for RCPSP are called constructive heuristics. Generally, these heuristics are solved by adding activities to an empty schedule from pre-created priority lists (Demeulemeester and Herroelen, 2002; Ulusoy and Hazır, 2021). The priority lists used in these heuristics are lists based on rules that take project-related information into account.

Task clustering is a solid approach to different scheduling problems with efficient use (Lu *et al.*, 2015; Hajikano *et al.*, 2016; Santoso *et al.*, 2017; Roy *et al.*, 2021). The developed heuristics, used with clustered tasks, obtain lower makespans than existing heuristics of the same research area. Still, even though the approach's usefulness is proven, it has not been covered for the RCPSPs. Therefore, this paper presents a cluster-based priority list method (CB-PL) to improve makespans obtained out of constructive heuristics that use priority lists for basic RCPSP. As an exact replication from other research topics is not possible, the base of the method is developed onto the syntheses of existing approaches. Then, it is finalized with the tuning of the futures of RCPSP into this base. The base of the CP-PL approach used is covered in the multiprocessor scheduling problems (Boeres *et al.*, 2004; Jedari and Dehghan, 2009), which are the most relatable study RCPSPs. Because

both problem types, the scope of this paper, and multiprocessor scheduling problems use directed acyclic graphs. So, the main adaptation for the base is built upon these multiprocessor scheduling problems. From this base, the finalized method works by making some adjustments to the creation of the priority lists instead of making changes to the algorithms themselves. The CB-PL method enables us to obtain more sensible priority lists by using the clusters generated from the baseline schedules. Then, the new lists obtain lower makespans with the same constructive heuristics. In this perspective, the CB-PL method increases the performance of the most used and basic approaches in solving RCPSP.

Thus, there are two purposes to this paper. The first purpose is to develop a method for clustering activities that can be used with existing priority rules to create priority lists to increase their performance. So, the CB-PL method is adapted and developed by combining the methods from existing literature for activity clustering and properties of RCPSP. Then the second purpose is to present an evaluation method for the performance of the CB-PL method. The performance is evaluated by comparing makespans obtained with original priority lists and the lists generated by the CB-PL method. The remainder of this paper is organized as follows. Section 2 reviews the RCPSP, the applied activity clustering related studies for the scheduling and clustering related studies. Then the detailed description of the method and the application of an illustrative example are given in Section 3. In Section 4, the numerical experiment for evaluating the method is described in detail. Then Section 5 analyses the results of the numerical experiment. Finally, a conclusion for the method's performance and the summary of the paper are presented in Section 6.

2. LITERATURE REVIEW

2.1 RCPSP Related Literature

The RCPSP literature has a considerable number of studies on the type of the problem and solution procedures for them, which can be found by Brucker *et al.* (1999), Demeulemeester and Herroelen (2002), Kolisch and Hartmann (2006). The RCPSP is the extension of the project scheduling problem that is classified as $cpm|C_{max}$ by Herroelen *et al.* (1999), without any resource constraints and only a predecessor-constrained scheduling problem. In the problem representation, *cpm* outlines the strict finish-start precedence constraint relationship with zero-time lag, while C_{max} represents the makespan minimization of the project. This problem is especially used in completing Temporal Analysis on the project network. On the other hand, the basic RSPSP is classified as $\mathrm{m}, 1|cpm|C_{max}$ and this problem is proved to be NP-Hard (Blazewicz *et al.*, 1983). The problem classification adds *m* number of renewable resources to the problem. The "*1*" in the classification scheme depicts resources for this problem being renewable (Demeulemeester and Herroelen, 2002). Also another critical aspect of this problem is that the project network graph representing the problem by definition is acyclic. Then, by also assuming the graph's nodes are topologically numbered, the final graph is referred to as a directed acyclic graph (Demeulemeester and Herroelen, 2002).

The solution generation techniques for the RCPSP are studied under two subjects, exact procedures and heuristic procedures by Demeulemeester and Herroelen (2002). As the name implies, exact procedures generate optimal base schedules for the problem. The usual approaches for these procedures are usually the linear programming-based ones or the branch-and-bound method-based ones. However, exact procedures have one shortcoming. As this procedure achieves optimal makespan for the project network, delays in the implementation, especially for activities in the critical path, require the calculation of the schedule again. Also, the computation time needed for some project networks to achieve optimal solutions is large. Therefore, project managers prefer to use heuristic procedures. Heuristic procedures compared to exact ones give acceptable and feasible project schedules achievable in a reasonable short computational time. Heuristics procedures are examined under constructive heuristics and improvement heuristics. Out of these two, improvement heuristics are built upon an already existing schedule. They do not build schedules from scratch. However, the constructive ones build the schedule from scratch. Readers interested in improving heuristics can be referred to existing literature (Demeulemeester and Herroelen, 2002).

The constructive heuristics are based on scheduling schemes and priority rules. The scheduling schemes correspond to the way that a feasible schedule is constructed. At the same time, the priority rules decide the order in which the activities are scheduled in the scheduling schemes. The basic scheduling schemes are serial (Kelley, 1963) and parallel scheduling (Brooks and White, 1965) schemes. The mentioned schemes start creating the schedule from the dummy start node and build the schedule by adding the activities according to their priority rule. Especially the serial scheme does this by adding activities one by one depending on resource availability and precedence completion of the activities.

On the other hand, the parallel scheme adds all the available activities considering their constraints for the given decision point. Note that activity additions are always made in order of their priority order. If a constructive heuristic application uses one scheme and one priority rule as described so far, it is called the single-pass method. However, as the easiness of using the single-pass method became apparent, multi-pass methods, combinations of schemes, and priority rules for scheduling have been proposed (Li and Willis, 1992).

As a selection rule for the activities, the priority lists affect the makespan of the schedule, similar to scheduling schemes. By their usage in the algorithms, there are two priority rules, the static and the dynamic ones. The static priority rules are the ones that are calculated at the beginning of the scheduling, and then they are used for the activity selection throughout the algorithm without any change in the order of the activities. On the other hand, the dynamic ones require updates on the priority list each time an activity is scheduled. Other than this distinction, according to the priority list's attribute, the lists have been categorized under five major categories (Lawrence, 1985). These categories are as follows; Activity-based priority rules consider only the activity-related attributes, Network-based priority rules work with network-related attributes like successor and predecessor relationships except for the resource consumption, Critical path-based priority rules use attributes generated from the temporal analysis (critical path calculations) results of the network, Resource-based priority rules covers the rules that are generated based on recourse consumption of activities. Finally, Composite priority rules are generated by combining the previous priority values by a weighted sum. Lastly, if the activities are added to the priority list starting from the start node or end node, two types of lists can be created for the same rule: a Forward priority list and a backward priority list. It is essential to mention that depending on the rule type, these two types of lists might not be available if the list is based on a unique rule. Besides the type and classification of the priority lists, a crucial aspect of the lists is how they require the order of activities to be in sequential order. It means that for any priority rule in question, the activity added to the list requires all successors to be already ordered for forward activity additions and predecessors for the backward additions.

2.2 Clustering Method Application Related Literature

Clustering of tasks in scheduling is not a saturated subject as the variety of scheduling problems increases. Some of the latest studies that use task clustering for scheduling problems can be found for operation room scheduling for hospitals (Santoso *et al.*, 2017) and the automobile repair sector (Alex Joseph and Saini, 2018). It is important to note that the scheduling problem for both of these clustering applications is remotely related to the project scheduling problem. They cluster the tasks according to their properties and create a priority order accordingly. Later, using this priority, they create their schedules. In other words, they create categories for the available tasks in their areas. Even though these cases use the clustering method, their approaches are case-sensitive. Still, in this paper, these approaches are also used for developing the CB-PL method. The operating room scheduling application (Santoso *et al.*, 2017) does this by clustering surgical activities by their cluster. They use K-means clustering (MacQueen, 1967) with the Silhouette method (Rousseeuw, 1987). They can decide the clusters with a deciding factor for an optimal number of clusters. Then they order the priority of these clusters according to the activities' durations in the clusters. The priorities of clusters are decided by the Shortest Processing Time method as mentioned to give better results. Finally, they use the priority order they created in a genetic algorithm for scheduling. Their results show increased efficiency as patient waiting time and nurse overtime for the problem decrease.

On the other hand, the automobile repair example (Alex Joseph and Saini, 2018) uses a determined clustering approach. Initially, they defined a four-category that takes the criticality of the tasks into account. Then they define a priority order in these categories according to the further specific problem of the automobile that requires a repair. Finally, the scheduling approach used with this clustering method is shown that it does increase the on-time delivery of repaired automobiles. The intake from these papers can be summarized under the clustering method used with task attributes to improve the scheduling problems. Therefore, the method presented in this paper generalizes this improvement on project scheduling problems. However, these applications do not require a graph representation, which is the case for the RCPSP. Such difference differs the RCPSP from these types of problems. So, the base of the method cannot be referenced by Santoso *et al.* (2017) and Alex Joseph and Saini (2018). Yet, their approaches are required for the finalization of the CB-PL method.

The most similar subject to RCPSPs, which also uses attributes of the task for the clustering, is scheduling directed acyclic graph (DAG) tasks on multiprocessors. Even though both RCPSPs and multiprocessor task scheduling problems are based on similar graphs, task clustering is not utilized in the project scheduling as it has been in task scheduling on the multiprocessor. Clustering of tasks related studies for multiprocessor scheduling is one of the most extensive ones (Gerasoulis and Yang, 1992; Boeres *et al.*, 2004; Jedari and Dehghan, 2009). The earlier studies that present a framework for this subject can be found in Gerasoulis and Yang (1992). The clustering application aims to divide tasks into the available processors. As such, the tasks assigned to a cluster cannot be separated from the other. The way clustering is applied creates mini-DAGs out of the original graph, which is dealt with by their assigned processors. In which created mini-DAG does not have a task that breaks the predecessor and successor relationship inside the cluster. However, the successor relationship is protected between clusters. So, the clusters can be assigned to processors then later scheduled clusters are combined according to their successor relationships.

The more replicable study to project scheduling from a multiprocessor task scheduling subject is the study of Jedari and Dehghan (2009). They propose a new task scheduling algorithm for Heterogeneous Distributed Computing Systems (HeDCSs). The applied heuristics in these systems also show similarities with project scheduling problems heuristics. HeDCS heuristics also have list-based heuristic algorithms (Adam *et al.*, 1974), similar to the constructive heuristics that use priority

lists. The proposed Resource-Aware Clustering (RAC) for Directed Acyclic Graph (DAG) algorithm uses a clustering methodology applicable to the HeDCSs, systems with varied processors and resources. An example of resources, in this case, is the processing power of the processors. The algorithm first creates clusters of the tasks assigned to processors. An important aspect of their clustering method is to have the clustering application without considering resources so that the clustering is fair. Then the scheduling of clusters is then calculated with a list-based heuristic that uses Modified Bottom-Level to prioritize the tasks. The property used in the prioritizing is the tasks' longest distance to graph start or endpoints. Finally, their comparison results show improvement compared to other available list-based algorithms. More recent papers' like Hajikano *et al.* (2016) and Roy *et al.* (2021) also use task clustering for scheduling multiprocessors. The crucial differences between these two papers to Jedari and Dehghan's (2009) application are the method used in the list-based algorithm and the method heuristic. Also, although list-based heuristics are the main approaches in these papers, the clustering methodology is not covered explicitly. Therefore, this paper takes Jedari and Dehghan's (2009) approach as the base for replication to create a CB-PL method as it is more relatable than other task clustering applications.

2.3 Clustering Methods Related Literature

K-means algorithm (MacQueen, 1967) is the most commonly applied clustering algorithm available in the literature. The algorithm starts with k single random points, which then, by the addition of the other points it, generates the clusters. The amount of the clusters stays constant with the k amount as initially was used to declare the random points. The declaration of the amount, k, can be made intuitionally, or methods for finding the optimal number of clusters for the map can be used. A common method for finding such an optimal k is the Silhouette Method (Rousseeuw, 1987). However, the computational time increase is the negative side of using the Silhouette Method to have the optimal k amount for the k-means algorithm. For example, the operation scheduling problem study (Santoso *et al.*, 2017) uses this combination for their clustering method. The combination is useful for their case because they analyze the system and create the clusters one time for the variety of the tasks (surgical activities) in their systems. Therefore, they do not require the computation of clusters for any additional surgical activity that arrives in their system. They would only require it if a new type of surgical activity emerges in their system. However, such is not the case for the project scheduling problem. Any additional activities or changes in the project network would cause the network's graph, the mapping, to have changed. The change would require applying the clustering algorithm again as the new clusters differ from the previous ones.

The Affinity Propagation (AP) method (Frey and Dueck, 2007) overcomes this computational time need. At its core, the AP method aims to succeed in the shortcoming of the K-mean algorithm in other aspects. Especially the reliance of the K-means algorithm on its initial random point selection for finding satisfactory results. Compared to the K-mean algorithm, the AP method starts by considering all data points as possible clusters. Then, they create final clusters according to their proposed relationship of points by iterating until convergence of the graph is reached.

2.4 Conclusion

Firstly, the literature on the RCPSP problem was reviewed. Then, the possible clustering method studies from respective scheduling problem areas that are related or similar to the RCPSP were reviewed. However, none of the studies could be exactly replicated to the RCPSP for a clustering method application. Therefore, a new method utilizing the features of these studies and RCPSPs is required to take advantage of task clustering.

The clustering method applied by Jedari and Dehghan (2009) in their multiprocessors scheduling matches with the RCPSP according to the graph representation. Both are represented as directed acyclic graphs. However, their method is not directly replicable in the RCPSP because of its limitations. The first and foremost reason is that there cannot be a limit for the number of clusters, as it is limited by the number of processors in their case. Second, the resource availability in RCPSP's case is renewable throughout the schedule. If their method is to be applied, then the activities in the project need to be clustered according to their predecessor and successor relationship. Then the problem would have to schedule the activity clusters within themselves before combining them in the main schedule. Such an approach is not suggested in RCPSP. Because a schedule generated from such manner increases idle resources, as the scheduling would act like combining big puzzle pieces. Nevertheless, in this study, their method of creating the initial map was taken as the base for the CB-PL method's cluster generation step.

Following the map generation, the CB-PL method synthesized similar ideas from the operation room scheduling case by Santoso *et al.* (2017) and the automobile repair case by Alex Joseph and Saini (2018) for cluster and priority list generation. The method used the Santoso *et al.* (2017) way of generating the number of clusters, as for the RCPSPs' cluster amount cannot be sensibly decided like Jedari and Dehghan (2009). For this purpose, two general clustering methods from the literature were also reviewed. As the Affinity Propagation method (Frey and Dueck, 2007) performs better than the K-means algorithm (MacQueen, 1967), the clustering algorithm is decided to be AP for the CB-PL method. Finally, the Alex

Joseph and Saini (2018) method for activity prioritization was integrated for the priority list generation from generated clusters. Similar to their approach, the CB-PL method prioritizes clusters and then decides on the priority of activities by the original priority rules. The CB-PL method developed based on these integrations is covered in Section 3.

3. METHODOLOGY

This section covers cluster-based priority list methods' theoretical explanation, and an illustrative example of method application is given. In the theoretical explanations, the graph and the maps are defined and used for the clustering method for the RCPSP. Later, the obtained clusters are defined. Conclusively, the CB-PL method's way of creating modified priority lists is covered. Then finally, the method is applied for an illustrative example.

3.1 Cluster-Based Priority List Method

3.1.1 Project Network Graph Representation

Activity on Node (AoN) representation is used for the graph representation of the project networks of the RCPSP. Such that, let the graph be $G = (V, E)$ with V as the set of nodes (activities) and the E as the set of arcs (successor/predecessor relationship). By this, $|V| = n$ as it starts from 1 (Start dummy node) to n (End dummy node). It is also assumed that node numberings are topologically ordered such that node numbers are increasingly ordered according to their successor/predecessor relationship. By doing so, arcs always point from a smaller numbered node to a higher-numbered one. Note that notations of node and activity are used interchangeably as both would dictate the same. In this prospect, let i denote the number of the node that is in question such that $i = 1, \dots, n$. Then for the relationship of the nodes, let S_i denote the set of the immediate successors i and P_i denote the set of the immediate predecessor of node i. Then the S_1 , P_1, and S_n , P_n dummy nodes are defined accordingly. Also, let d_i denote the durations required for the activity i to be completed. Final notations for the problem would be for the project network itself would be the resources; therefore, let R_{ik} and R_avail_k represent the resource consumption of node i for the resource k and the availability of resource k. The k here represents the resource type as there can be multiple resources up to M such $M = |R\_avail|$, therefore $k = 1, \dots, M$.

3.1.2 Map Creation

The clustering method applied for the CB-PL differs from the multiprocessors scheduling example (Jedari and Dehghan, 2009). The main idea behind the CB-PL method is to find activity and resource consumption condensed areas in the project network. Let these areas be called the critical areas. In this aspect, the map created for the clustering algorithm requires such activity and resource-related information availability. As such, the CB-PL method's clustering method follows the idea of creating critical areas. Such that clusters do not need to be bounded by their relationships, as is the case for Jedari and Dehghan (2009). For example, nodes i and j that do not have either successor/predecessor relationship can be found in the same cluster. Therefore, a solid map representation that allows reasonable point representations for clustering is required. The representation is achieved by creating early and late start schedules from the critical path analysis mentioned in Demeulemeester and Herroelen (2002). These schedules are used to create points for the map. These schedules assume a resource-free state, or in other words, an infinite resource available state, for the generation of project schedules. Then the schedules are generated by using d_i and the predecessor and successor relationship of the activities. It is important to note that created clusters become fair clusters, similar to Jedari and Dehghan (2009), as map creation is based on these resource-free schedules. The resource-free schedules that can be created enable us to have two map types from early/late start schedules. Let the maps created from these schedules be named early and late maps. As the CB-PL method can be applied to both of these maps, each must be defined. However, it can be seen in Section 5 that late maps generate better results with the CB-PL application in the forward priority lists. For the early start schedule, let EST_i and EFT_i denote the earliest possible start and finish times of activity i, $1 \leq i \leq n$. Then for the late start schedule, let LST_i and LFT_i denote the latest possible start and finish times. These notations are used for the creation of points for the maps. Demeulemeester and Herroelen (2002) can be examined for the explanations of the creation of these schedules.

Considering the previously mentioned schedules, let us define the points of the map that is used for clustering generation. Note that dummy start and end nodes do not have any point reference as they do not affect the critical area. So, let P_{it} be the point in the map that represents activity $i, 2 \leq i \leq n-1$. Then the t represents the dimensions such that $1 \leq t \leq 1 + M$. Per this definition, each activity, except the dummy ones, is represented in a point system with the dimension amount of one plus the resource amount. The first dimension represents the midpoint of the activity's scheduled duration (mid-time). Depending on the early or late schedule type usage P_{i1} can be defined as $P_{i1} = EST_i + EFT_i / 2$ or as $P_{i1} = LST_i + LFT_i / 2$. The rest of the dimensions for the points are then calculated based on the resource consumption of the activity. However, as the project

networks can require more than one resource type, the varied consumption in multiple resource availability requires normalization. Otherwise, the clusters would diverge from the resource consumption condensed critical areas. As such resource-related dimensions are defined as $P_{it} = R_{ik}/ R\_avail_k,\ \{2 \leq t \leq M+1, 1 \leq k \leq M\}$ for each activity $i, 2 \leq i \leq n-1$.

3.1.3. Cluster Creation

Since the definition of the points P_{it} is completed, the Affinity Propagation (AP) method for the clustering of the activities can be applied easily. The variable that requires declaration for the AP method is the maximum number of iterations for the algorithm to stop if the convergence is not reached. The iteration amount can be decided intuitively depending on the number of activities in the network. For example, 500 iterations are used for the numerical experiment conducted in this paper. Now, let us define the results and attributes used in the priority list creation step from AP clustering. Let C_q be the clusters that contain the set of activity js that are clustered. Such that $|C| = u,\ u$ becomes the number of clusters obtained from the algorithm, such that $1 \leq q \leq u$. Then let C_center_{qt} be the center point of these clusters, $1 \leq q \leq u, 1 \leq t \leq 1+M$. After this definition, a priority is given to the clusters closer to the starting point of the initial schedule used for cluster generation. Therefore, let Co_q be ordered clusters containing the set of activities for cluster q such that clusters are ordered increasingly according to the initial dimension (mid-time dimension, C_center_{q1}) of C_center_{qt}. The ordering scheme allows us to use Co_q for forward and backward priority lists creations as the reverse of the Co_q prioritizes the clusters that are close to the end dummy node. Finally, the dummy start and end nodes are added as initial and final clusters for the easiness of the method applied to the Co_q, such that $0 \leq q \leq u+1$. Note that the clusters for the dummy nodes only include one node, which is themselves.

3.1.4 Cluster-Based Priority List Creation

The regular priority lists creation takes the selected priority rule and the successor/predecessor relationship into account. Also, whether the list type is forward or backward, activity relationships are considered according to the start or end dummy node. However, the CB-PL method gives a higher priority to the clusters according to their order. Therefore, the cluster-based priority list method uses defined Co_q with the selected priority rule to create the priority list. The activity clusters in Co_q are not definite to be schedulable in the cluster. So, the activities might not be scheduled in clusters according to the priority rule. Consequently, the combination of clusters by their priorities might not be schedulable either. Also, the highest prioritized cluster might not include all of the successors from the start dummy node for the forward priority list creation in CB-PL. The schedulability issue is also valid for the backward priority list as the predecessors from the initial cluster might have the same circumstances. As such, another scheme for creating the priority lists is required.

Let $CBPL$ be the final priority list according to the selected priority rule with the clustering method. Initially, $CBPL$ only includes the starting dummy node. Then let AAC be a list of available activities for the clusters in their order of priority list addition. Initialization of the AAC would take either Co_1 or Co_u depending on forward or backward list creation as the clusters from one of the dummy nodes would be already scheduled. Also, let PAA be the list of physically available activities. The set PAA is meant only to include activities in which all of the given activity's predecessors for the forward priority list (successors for the backward priority list) are listed in the list $CBPL$. Initialization of the PAA would include the successors of the start dummy node or predecessors of the end dummy node according to priority list type. Finally, let APC be the intersection of lists AAC and PAA, the set of activities available for addition to the list $CBPL$ according to the priority rule.

A priority list creation scheme would be as follows; At each activity selection point for the $CBPL$, decision point, according to the priority rule selection, the activity i from the newly calculated APC list would be added to $CBPL$ and deleted from the PAA and AAC lists. After the addition of i, each successor j of i is checked if their predecessors are in the $CBPL$ list. If so, the j is added to PAA as it would be physically possible to add activity j to $CBPL$. However, if the APC list is empty at the decision point, the activities from the cluster set in order are added to the AAC list. For example, the first time the APC list is empty, the activities from clusters Co_2 (for the forward list) or Co_{n-1} (for backward list) would be added to the AAC. However, if the APC list is empty and all of the clusters were added to the AAC list, then all of the activities are added to $CBPL$, and the priority list is created. The created list is called CB-PL and can be used in the constructive heuristics of the RCPSPs. An example pseudo-code for the forward priority list of the scheme can be seen in Algorithm 1. The summary of the notations that have been used throughout this section is presented in Table 1.

Algorithm 1. Pythonic priority list creation scheme pseudo-code for the forward priority list

```
CBPL = [1]
AAC = [j for j in Co_1]
PAA = [i for i in S_1]
clusters = [2 to u + 1]
for i = 1 to n:
        APC = AAC ∩ PAA
        while APC is Empty:
            if clusters is Empty:
                STOP (all activities are added to CBPL)
            end if
            AAC = AAC.extend([j for j in Co_clusters[0]])
            clusters.pop(0)
            APC = AAC ∩ PAA
        end while
        activity = Select according to priority rule from APC
        CBPL = CBPL.append(activity)
        PAA = PAA.remove(activity)
        AAC = AAC.remove(activity)
        Add successors of activity to PAA if their predecessors are in CBPL
end for
```

Table 1. Summary of the notations for the method

Notation	*Definition*
V	Set of nodes in the project network, $i \in V$
d_i	Duration required for node *i* to be completed
M	Amount of available renewable resource type, such that $k = 1, \ldots, M$
R_avail_k	Availability of the renewable resource *k*
R_{ik}	Consumed resource *k* for the node *i*
S_i	Set of immediate successors of node *i*
P_i	Set of immediate predecessors of node *i*
EST_i	Earliest start time of node *i* for the early schedule
EFT_i	Earliest finish time of node *i* for the early schedule
LST_i	Late start time of node *i* for late schedule
LFT_i	Late finish time of node *i* for late schedule
t	The dimensions of points in the map, such that $t = 1, \ldots, M + 1$
P_{it}	Map point representation of activity *i* with *t* dimension (Start and dummy nodes are not included)
C	Set of clusters that contain a group of activities, $q \in C$
C_center_{qt}	Center point of cluster q with *t* dimension
Co_q	Set of clusters that are increasingly ordered according to cluster centers (Start and dummy nodes are added as clusters)
$CBPL$	The priority list obtained by the method
AAC	List of activities that are available by clusters for priority list addition
PAA	List of activities that are physically available for priority list addition
APC	List of activities that are available both physically and cluster wise for priority list addition

3.2 Illustrative Example

In this part, the cluster-based priority list method is applied to the example from Demeulemeester *et al.* (1994) using the longest processing time (LPT) priority rule (Graham, 1969) with a forward priority list type. The example has seven activities with nine nodes, including the starting and ending dummy nodes. The activities only require one renewable resource type with an availability of 5. The graph of the project network example can be examined in Figure 1. In the project network graph, the d values represent the required duration for the activity, while the r values indicate the resource requirement.

The initial step in the CB-PL method is to create early and late schedules used to generate the maps. So, the schedules are generated according to the method presented in Demeulemeester and Herroelen (2002). Figure 2a represents the early schedule obtained, while Figure 2b represents the late schedule. Then, the early and late maps are generated accordingly. In the map generation part, the essential aspect is the creation of points from the schedules. The points' first dimension is based on the schedule start-finish midpoints, while the rest are based on resource consumption.

For example, for node 8, the resource dimension is calculated by normalizing the value. The value is obtained by dividing the resource requirement of the node by the resource availability for both early and late maps. So, it becomes 0.4 (2/5). If there had been more than one resource, each would have been processed similarly. Then the time (duration) dimension, the first dimension, for the early schedule is calculated by dividing the summation of its EST and EFT by 2. The time dimension then becomes 4.5 ([3+6]/2). The final point of node 8 in the map becomes (4.5, 0.4) for the early schedule. Later for the late schedule, the dimension is calculated using LST and LFT. The time dimension then becomes 5.5 ([4+7]/2) while the point in the map becomes (5.5, 0.4).

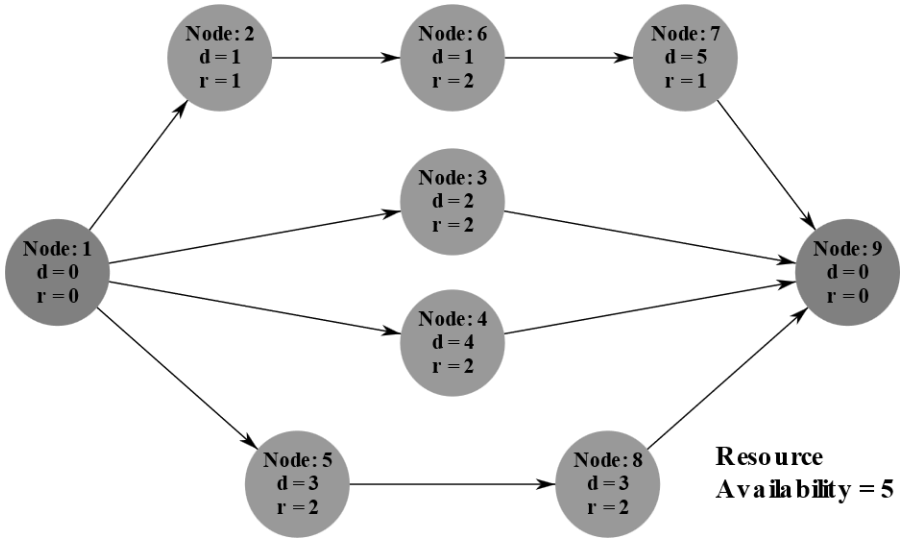

Figure 1. The project network example for the CB-PL method application

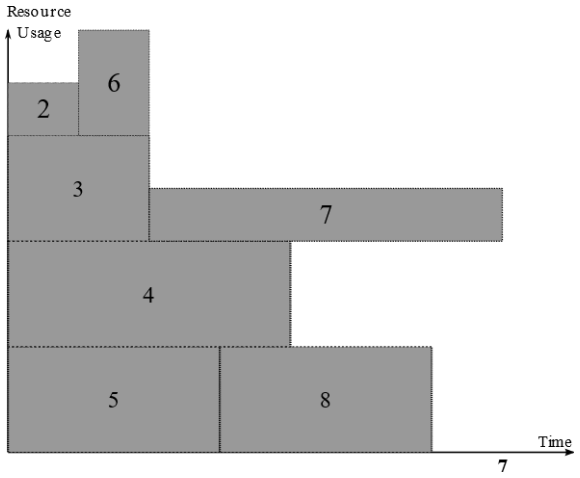

Figure 2a. Early Schedule

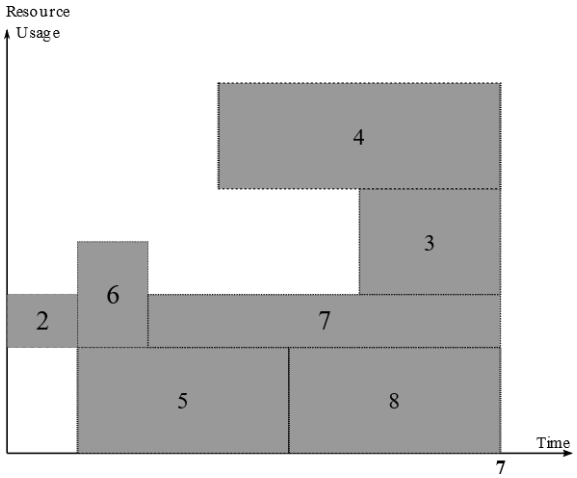

Figure 2b. Late Schedule

Figure 2. Early and late schedules of the example

After the generation of maps is completed, the clusters are generated using the Affinity Propagation method. Figure 3 shows the clustered nodes for the early map, while Figure 4 shows them for the late map. Note that generated clusters are

numbered increasingly according to their center point's time dimension. So, regardless of the clusters' order generated by the AP method, the clusters need to be ordered.

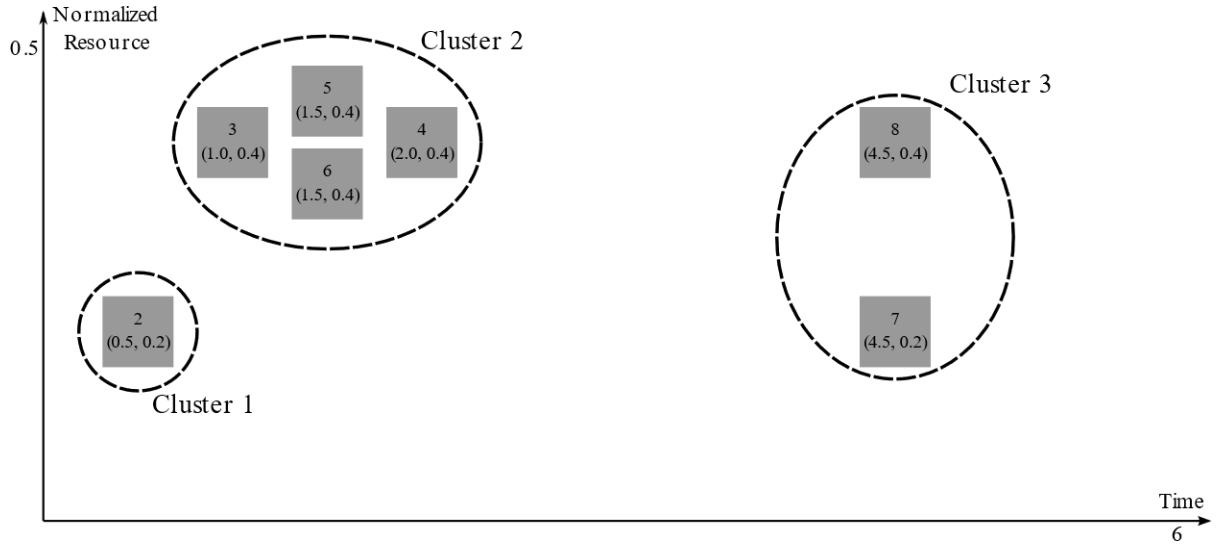

Figure 3. Early map with ordered clusters

Now that the clusters are generated, let us create the CB-PL for them with the LPT rule. Note that the original forward priority list (PL) using the LPT rule for the example is $PL = \{1, 4, 5, 8, 3, 2, 6, 7, 9\}$. For the early map-based CB-PL initialization, the CB-PL priority list is $CBPL = \{1\}$ and the physically available activities (PAA) are $PAA = \{2, 3, 4, 5\}$. On the other hand, the available activities for cluster priority (AAC) are $AAC = \{2\}$ as only the first cluster is added to the list. Then a node for addition to $CBPL$ is selected, which can only be chosen from nodes that are both available physically and with their cluster orders (APC). In this state, $APC = \{2\}$ as the only node available for both lists is node 2. As node 2 is the only node that can be ordered, the list is updated to $CBPL = \{1, 2\}$. Then the other lists are updated. The PAA is updated by adding newly physically available nodes and removing node 2. The AAC is updated by removing node 2. The lists become as follows $PAA = \{3, 4, 5, 6\}$ and $AAC = \{\}$.

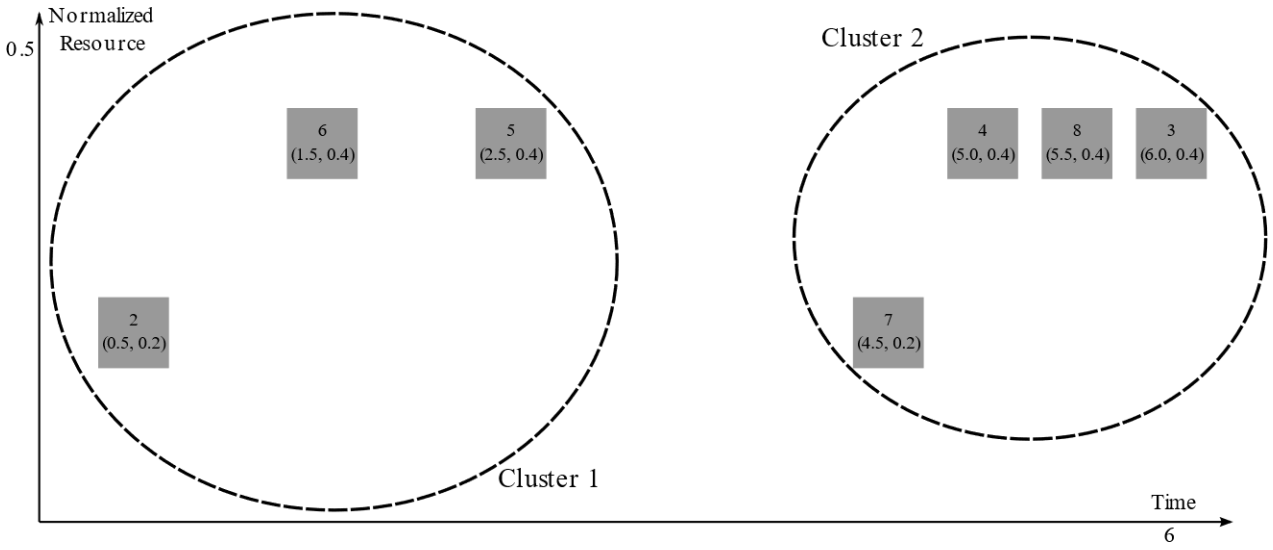

Figure 4. Late map with ordered clusters

At this stage, the intersection APC becomes empty. Therefore, the next cluster (Cluster 2) is added to the AAC, which becomes $AAC = \{3, 4, 5, 6\}$. Now, since both PAA and AAC are the same, all of the activities in $APC = \{3, 4, 5, 6\}$ can be

ordered. According to the LPT rule, the nodes are added to $CBPL$, which updates to $CBPL = \{1, 2, 4, 5, 3, 6\}$. Note that, if necessary, ties are broken by selecting the lower numbered node. Then finally, the last cluster (Cluster 3) is added and ordered. The priority list for early CB-PL becomes as follows $\{1, 2, 4, 5, 3, 6, 7, 8, 9\}$. A final note for the method application is that starting node and ending nodes are declared as single clusters for a smooth application named Cluster 0 for the start dummy node and Cluster 4 for the end dummy node in this case.

Similarly, the late map-based CB-PL can be constructed. The list then becomes as follows $\{1, 5, 2, 6, 7, 4, 8, 3, 9\}$. Then if constructed lists are used with the forward planning serial algorithm, the makespans of the project network example become 12 for the original priority lists, while it becomes 10 and 8 for the early map-based CB-PL and the late map-based CB-PL. The schedules of obtained makespans can be examined in Figures 5, 6, and 7.

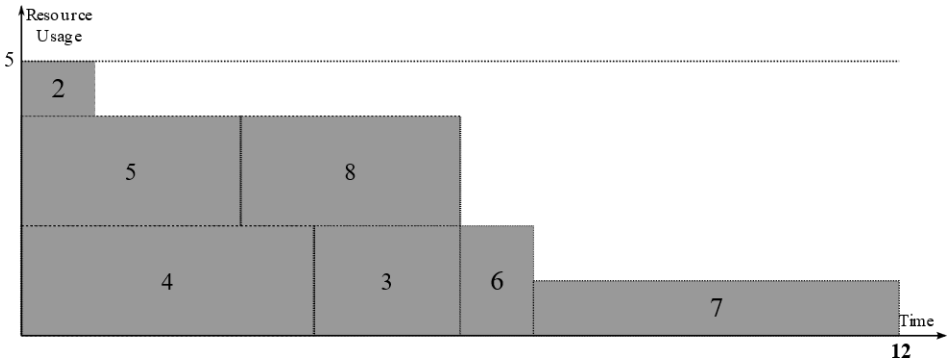

Figure 5. Normal priority list serial schedule

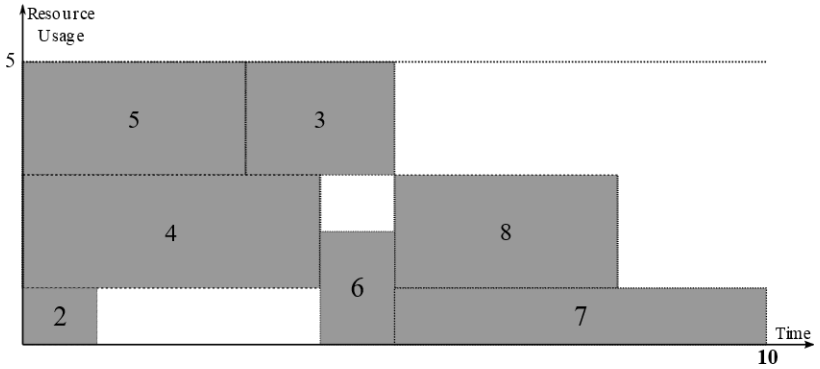

Figure 6. Early map-based CB-PL serial schedule.

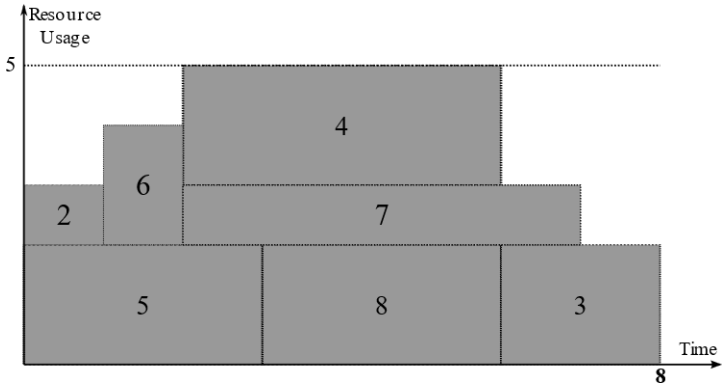

Figure 7. Late map-based CB-PL serial schedule

4. NUMERICAL EXPERIMENT

The Cluster-Based Priority List method that is presented in this paper is an improvement to the existing priority lists' literature. The method can be applied to numerous priority lists. Therefore, a comparative experiment of performance is required to highlight its effectiveness. The benchmark dataset used for this purpose is the RCPSP instances of the PSBLIB, a project scheduling problem library (Kolisch *et al.*, 1999). The library was first introduced by Kolisch and Sprecher (1997). Even though it does not have many activity instances, the initial library covers various types of problems in project scheduling. The Resource Constraint Project Scheduling, which focused on this paper, is one of them. Later, Kolisch *et al.* (1999) enhance this library. The initial library had two sets of instances with an activity amount of 30 and 60 activities for the RCPSP. The instance sets are called J30 and J60 sets. With the additions from Kolisch *et al.* (1999), sets with 90 and 120 activities are also introduced to the library, namely J90 and J120. All of the mentioned instances from the PSPLIB are widely used as benchmark instances in the literature (Merkle *et al.*, 2002; Möhring *et al.*, 2003; Gonçalves *et al.*, 2008). The conducted numerical experiment also uses the enchanted library's RCPSP project network instances from Kolisch *et al.* (1999). All of the sets of the library J30, J60, J90, each with 480 instances and J120 with 600 instances, are included in the experiment. Also, using these sets allows us to compare the CB-PL method on a varied number of activities for the project networks. Through this variety, the effectiveness of clustering for a different number of activities can be compared. The comparison is significant because the clustering methods' efficiency is dependent on the number of points in the given map.

The number of constructive heuristics limits the solution procedure set generating the makespans. Note that the constructive heuristics are performed through scheduling schemes and priority rules. Therefore, the set definition comprises some schemes with different priority rules for the experiment. The scheduling schemes used are two, base serial (Kelley, 1963) and parallel (Brooks and White, 1965) heuristics. The base ones are counted for the forward planning type, as the backward planning is categorized separately (Demeulemeester and Herroelen, 2002). Also, the ties for selecting activities are broken by choosing the lower numbered nodes while applying the heuristics.

Table 2. Literature review for priority rules used in the experiment

Priority Rule Category	Name	Abbreviation	Literature
Activity-Based	Shortest Processing Time	SPT	Olaguíbel and Goerlich (1989)
	Longest Processing Time	LPT	Graham (1969)
Critical Path-Based	Earliest Start Time	EST	Demeulemeester and Herroelen (2002)
	Earliest Finish Time	EFT	Demeulemeester and Herroelen (2002)
	Latest Start Time	LST	Kolisch (1995)
	Latest Finish Time	LFT	Davis and Patterson (1975)
	Minimum Slack	MSLK	Davis and Patterson (1975)

Then, the selection is made from the static priority rules for priority rules. Seven priority rules are used with the two of the scheduling schemes. Four of the rules out of the seven relate to their forward and backward list types. Only the forward priority lists generation is experimented with to prevent replication. The chosen priority rules with their categories from Lawrence (1985) and works of literature can be examined in Table 2. Combining the single schemes with the single priority rules defines the solution procedure set. Therefore, the set includes a total of fourteen procedures such that each scheme has seven procedures. By these definitions, each defined solution procedure becomes a single-pass method.

Eventually, the comparative analysis is made with this solution procedure set. First and foremost, the makespans with the original priority list generation way are calculated. Then the makespans from the early map and late map cluster prioritized cluster-based priority lists are calculated. Applying the CB-PL method for both maps is required as neither of the maps can be preferred without comparing the makespans obtained through these maps in any respected solution procedure. Also, the Affinity Propagation method has a stopping condition for cluster generation. The maps that do not converge with the given iteration amount of the AP method cannot be used for creating CB-PL. As they do not converge, the final total number of clusters is not found. Convergence not being reached means that there were activities that were not clustered meaningfully. These clusters would not have an affinity relationship in the system. Therefore, they would cause meaningless clusters, which are against the critical area creation of the CB-PL method. So, the makespans for the project network instances, either for the early or late map, which was not clustered, are matched with makespans of the original priority lists.

Finally, by comparing the makespans of the original priority lists and the CB-PLs, the performance of the CB-PL method is measured. The primary comparison method is the improvements on the makespan of the project networks. The results showed varied responses for maps and the applied priority rules. The makespans of the project network instances showed both improvement and depreciation compared to the makespans of the original priority lists. Still, notable makespans improvements are observed on average depending on the cluster map type and the priority rule applied. In the early map-

based CB-PLs, the average makespans change is dependent on the priority rule as both improvements and depreciation are available. However, the late map-based CB-PLs show significant improvement as there is no depreciation for the makespans on average for any priority rules. The detailed analysis of the numerical experiment results can be observed in the following section.

5. ANALYSIS

The analysis of the experiment is presented under five categories. Initially, the network instances that are not clustered by the Affinity Propagation clustering method are examined. Then the average makespan reductions from the original priority rules are presented for all set instances. After that, the clustering method's effect on the number of activities is examined for the preferred map type. Finally, the best-found makespan and computational time analysis are presented.

5.1 Clustering Analysis

First, let us analyze the effect of the Affinity Propagation clustering on the project networks for the clusters that do not converge. As the convergence has not been reached, the clusters are not generated. The number of early and late maps that do not get clustered is significantly less. Non-clustered ones only amount to 1.3% of network instances for the early map-based clusters and 1.8% for the late map-based clusters (Tables 3 and 4). The interesting finding is that the percentage of the project networks that do not get clustered increases per the number of activities in the networks.

Table 3. Convergence analysis for convergence not reached count per sets

	All Sets	J30	J60	J90	J120
Total Instances	2040	480	480	480	600
Early Map	27	0	1	4	22
Late Map	36	0	1	7	28
Both Maps	2	0	0	0	2

It can be seen in the increase in not clustered maps from J90 set to the J120 set (Table 4). The J120 set shows the highest percentage of not clustered networks for both maps. Also, it is the only set that shows a non-clustered network for both of the maps simultaneously with two networks out of 600. The project network instances in which clusters could not be generated are counted with instances that gave the same makespan as the original rule application.

Table 4. Convergence analysis for convergence not reached count percentage per sets

	All Sets	J30	J60	J90	J120
Early Map	1.3%	0.0%	0.2%	0.8%	3.7%
Late Map	1.8%	0.0%	0.2%	1.5%	4.7%
Both Maps	0.1%	0.0%	0.0%	0.0%	0.3%

5.2 Makespan Reduction Analysis

5.2.1 Definitions

Let us define the analytical values that are used in the makespan analysis. The values can be grouped under two distinct categories, average makespan reductions percentage and makespan change-direction count percentages. Also, the definitions of the categories are presented in combination with the dataset indicators. Therefore, the combinations are created with Total, J30, J60, J90, and J120 sets in which Total set values include all the instances from the other sets. For the average values, as the performance is measured based on the makespan reduction, the calculations are made so that a positive percentage would decrease the makespan. The values are as follows; Average%, the average of the percentage of makespans changes for the set, PositiveAvg%, the average of the percentage of only positive makespans changes for the set, NegativeAvg%, the average of the percentage of only negative makespans changes for the set, and finally, NonZeroAvg%, the average of the percentage of positive and negative makespans changes for the set. Then for the count values, NoChange%, the percentage of the number of makespans that did not change in the set, Positive%, the percentage of the number of makespans that improved in the set, and Negative%, the percentage of the number of makespans that deteriorated in the set.

5.2.2 General Analysis

The makespan reduction analysis for the CB-PL method of the whole dataset can be examined in Tables 5 and 6. Table 5 includes the forward planning serial algorithm (FP-SA), while Table 6 is for the forward planning parallel algorithm (FP-PA). The early map-based (EMB) cluster for the FP-SA shows an Average% makespan reduction change between -4% and %3 for the priority rules that have been used. On the other hand, late map-based (LMB) clusters show makespan reduction improvement between 0% and 10%. The FP-PA analysis also shows similar changes for the applied map types. Like FP-SA cluster map bases, there is a significant improvement increase between EMB and LMB cluster application of CB-PLs for the FP-PA.

An important result from these heuristic applications is that EST and EFT rules and LST and LFT rules show no improvement in makespan if the CB-PL is based on the early or late map, respectively. It is because the cluster priorities give a similar priority to the nodes as the EST and EFT priority rules. For example, if the CB-PL is constructed with the early map, the nodes with the early start and finish times are given priority with the clustering method. Then applying EST and EFT rules to these clusters is expected to generate the same results as the original EST and EFT priority rules. The cases are the same for late map-based CB-PL with LST and LFT rules. Therefore, using the CB-PL method with LST and LFT priority rules is not suggested for both forward serial and parallel heuristics with any map types. However, the EST and EFT priority rules can still be preferred with the method for the late map-based applications.

Table 5. Forward planning serial algorithm makespan reduction analysis

Cluster Map Type	**Earl Map Based**				**Late Map Based**			
Priority Rule Type	Average%	Positive Avg%	Negative Avg%	NonZero Avg %	Average%	Positive Avg%	Negative Avg%	NonZero Avg%
SPT	3%	7%	-5%	4%	10%	13%	-5%	13%
LPT	2%	7%	-5%	2%	7%	11%	-5%	10%
EST	0%	2%	-2%	0%	5%	7%	-3%	6%
EFT	0%	2%	-2%	0%	6%	9%	-4%	8%
LST	-4%	4%	-8%	-6%	0%	2%	-2%	0%
LFT	-4%	4%	-7%	-6%	0%	2%	-2%	0%
MSLK	0%	6%	-6%	0%	3%	6%	-4%	5%

Table 6. Forward planning parallel algorithm makespan reduction analysis

Cluster Map Type	**Earl Map Based**				**Late Map Based**			
Priority Rule Type	Average%	Positive Avg%	Negative Avg%	NonZero Avg%	Average%	Positive Avg%	Negative Avg%	NonZero Avg%
SPT	0%	5%	-4%	1%	5%	8%	-3%	7%
LPT	1%	5%	-4%	1%	5%	8%	-3%	7%
EST	0%	2%	-2%	0%	4%	7%	-3%	6%
EFT	0%	2%	-2%	0%	4%	7%	-3%	6%
LST	-3%	3%	-6%	-5%	0%	2%	-2%	0%
LFT	-3%	3%	-6%	-5%	0%	2%	-2%	0%
MSLK	-1%	4%	-5%	-1%	2%	5%	-3%	3%

Other good indicators for the performance of the CB-PL method are the PositiveAvg% and NegativeAvg% values for EMB and LMB clusters. For both algorithm cases, the Average% shows significant makespan improvement at the LMB cluster application. However, it would have been a bad case if the NegativeAvg% values decreased significantly compared to the increase in PositiveAvg%. Such is not the case for the conducted experiment. Except for the LST and LFT priority rules, other rules show an increase in PositiveAvg%. On the other hand, NegativeAvg% shows a varied response of small increases and decreases. However, NegativeAvg% still shows increases for most of the tested priority rules.

There is a clear makespan performance difference between EMB and LMB clusters. Therefore, another major explanatory point for the makespan reduction difference between EMB and LMB clusters is the change in the number of instances' makespan shift direction. Tables 7 and 8 show the analysis of this change. The difference can be explained by the decrease from the Negative%, which is way more significant and decreases from NoChange% sliding to the Positive%. Again, it is essential to note that the LST and LFT priority rules are an exception in the CB-PL method.

Table 7. Count analysis of forward planning serial algorithm makespan change direction

Cluster Map Type	Earl Map Based			Late Map Based		
Priority Rule Type	Negative%	NoChange%	Positive%	Negative%	NoChange%	Positive%
SPT	20%	28%	52%	2%	22%	75%
LPT	28%	26%	46%	3%	27%	70%
EST	16%	70%	14%	6%	26%	68%
EFT	9%	76%	15%	3%	23%	73%
LST	61%	31%	7%	9%	83%	8%
LFT	59%	33%	9%	8%	83%	10%
MSLK	36%	29%	35%	8%	40%	51%

Table 8. Count analysis of forward planning parallel algorithm makespan change direction

Cluster Map Type	Earl Map Based			Late Map Based		
Priority Rule Type	Negative%	NoChange%	Positive%	Negative%	NoChange%	Positive%
SPT	27%	41%	32%	5%	30%	65%
LPT	27%	35%	38%	5%	31%	64%
EST	13%	74%	13%	5%	32%	63%
EFT	9%	82%	9%	6%	30%	65%
LST	51%	40%	9%	6%	86%	8%
LFT	50%	41%	10%	6%	87%	7%
MSLK	36%	37%	27%	11%	49%	41%

A similar change can also be examined for count analysis of LST and LFT priority rules. Compared to the other rules, their shift is towards NoChange% than the Positive%. Therefore, their results from the average values are exempted for the values that are presented in the following. The values show an average increase of 35% in Positive% for late map-based FP-SA cases for the priority rules. The increase indicates the number of project network instances improving with late map-based CB-PLs. The FP-PA cases also show a 36% increase in this aspect. However, it is important to note that even with the increase, the FP-SA application's instances with positive makespan reduction change are on average higher than the FP-PA applications. On average, FP-SA has 68% Positive% while the FP-PA has 59% throughout the priority rules.

5.2.3 Activity Amount Effect Analysis

The experiment results also highlighted the effect of CB-PL per activity amount. The EMB results of different sets do not show much difference. The average makespan reduction percentage nearly stays the same per activity number. However, there is a gradual increase in the average makespan reduction percentage (Average%) and the number of positive change percentages in makespan (Positive%) in the data sets for the late map-based CB-PL. Therefore, the activity amount relationship with the late map-based CB-PL requires explanation.

Tables 9a and 9b show these for the FP-SA application, while Tables 10a and 10b do the same for the FP-PA application. One important aspect is that the gradual increase becomes more horizontal in the transaction from J60 to the J90 set. Such an observation holds for both the Average% and Positive% values of both applications. Then the performance indicators peak at the J120 set.

Table 9. Forward Planning Serial Algorithm analysis

Table 9a. Average makespan reduction (Average%)

Priority Rule Type	All Sets	J30	J60	J90	J120
SPT	10.0%	6.9%	9.3%	8.9%	14.0%
LPT	7.3%	4.8%	6.2%	6.4%	10.8%
EST	4.8%	2.5%	4.0%	4.6%	7.3%
EFT	6.5%	4.2%	5.6%	6.0%	9.4%
LST	0.0%	0.0%	0.0%	0.0%	0.0%
LFT	0.0%	0.1%	0.0%	0.0%	0.0%
MSLK	3.0%	1.9%	2.4%	2.7%	4.5%

Table 9b. Makespan positive change count % (Positive%)

Priority Rule Type	All Sets	J30	J60	J90	J120
SPT	75%	61%	71%	70%	95%
LPT	70%	55%	64%	63%	93%
EST	68%	49%	63%	66%	90%
EFT	73%	59%	68%	70%	93%
LST	8%	2%	5%	7%	17%
LFT	10%	3%	6%	7%	20%
MSLK	51%	34%	43%	42%	78%

Table 10. Forward Planning Parallel Algorithm analysis

Table 10a. Average makespan reduction (Average%)

Priority Rule Type	All Sets	J30	J60	J90	J120
SPT	5.1%	3.4%	3.8%	4.5%	7.9%
LPT	5.1%	3.4%	3.9%	4.3%	8.2%
EST	4.0%	2.3%	2.8%	3.8%	6.5%
EFT	4.3%	2.5%	3.3%	4.1%	6.7%
LST	0.1%	0.0%	0.0%	0.1%	0.1%
LFT	0.0%	0.0%	0.0%	0.0%	0.0%
MSLK	1.6%	1.0%	1.1%	1.2%	2.7%

Table 10b. Makespan positive change count % (Positive%)

Priority Rule Type	All Sets	J30	J60	J90	J120
SPT	65%	46%	56%	62%	91%
LPT	64%	45%	55%	59%	89%
EST	63%	39%	53%	63%	89%
EFT	65%	44%	57%	64%	88%
LST	8%	1%	4%	7%	19%
LFT	7%	2%	4%	5%	16%
MSLK	41%	21%	32%	35%	68%

Another aspect is the way Positive% values increase throughout the sets. The observed improvement in Positive% for set type is obtained differently than it was in cluster map type selection. The LMB cluster type selection allowed a significant decrease from Negative% instances to slide to Positive%. However, the major slide to Positive% comes from the NoChange%, while Negative% stays similar for all sets throughout the dataset. In this prospect, the J120 set can be examined. Table 11 shows the J120 set's count percentage of the project instance's makespan direction. The Negative% for the J120 set is under 5% for SPT, LPT, EST, and EFT priority rules in both heuristics. The NoChange% values for the same rules also nearly reach to limit. For the late map, 4.7% of project network instances in the dataset were not clustered because of the clustering method (Table 4). Therefore, the real NoChange% for these rules varies between 0%~2% for the FP-SA and 1%~4% for the FP-PA. However, the MSLK priority rule does not show similar remarkable performance compared to the other four rules for the J120 set.

Table 11. Count analysis of makespan change direction for J120 set

Heuristic Type	Forward Planning Serial Algorithm			Forward Planning Parallel Algorithm		
Priority Rule Type	Negative%	NoChange%	Positive%	Negative%	NoChange%	Positive%
SPT	0.3%	4.8%	94.8%	2.7%	6.3%	91.0%
LPT	1.0%	6.2%	92.8%	3.0%	7.8%	89.2%
EST	3.8%	6.5%	89.7%	2.7%	8.0%	89.3%
EFT	1.2%	6.2%	92.7%	4.8%	7.3%	87.8%
LST	19.2%	63.7%	17.2%	13.2%	68.0%	18.8%
LFT	18.0%	62.2%	19.8%	15.0%	69.0%	16.0%
MSLK	7.7%	14.0%	78.3%	14.8%	17.5%	67.7%

Nonetheless, it still shows nearly 4.5% and 2.7% makespan reduction, with more than 75% and 65% of instances having a positive makespan reduction for FP-SA and FP-PA, respectively. As such, the analyses made so far imply that the CB-PL method performs better as the number of activities in the project network increases.

5.3 Best Found Makespan Distribution Analysis

Best found makespans (BFM) for the dataset are analyzed under four groups. Group 1 only contains the original results of the algorithm with seven priority rules used in the experiment. Group 2 and 3 combine actual results with EMB and LMB results of the CB-PL method, respectively. Finally, Group 4 contains all three of the result types. The minimum makespan available in each group is taken as a base. Later, the count of priority rules matching with their minimum makespans is analyzed. The results of the analyses can be observed in Tables 12 and 13 for serial and parallel algorithms. For example, in group 3 of Table 12, the value for the LPT rule under the LMB application means 51.4% of the networks gave the same value as the BFM of the group.

Table 12. Best found makespan distribution for serial algorithm results

	Cluster Map Type	Priority Rule Type						
		SPT	LPT	EST	EFT	LST	LFT	MSLK
Group 1	Original	**20.1%**	**24.4%**	**23.5%**	**20.5%**	**79.2%**	**63.3%**	**36.8%**
Group 2	Original	**20.0%**	**23.8%**	**22.4%**	**19.9%**	**76.0%**	**61.0%**	**36.2%**
	EMB	19.3%	21.8%	22.2%	19.9%	32.3%	31.0%	28.7%
Group 3	Original	**20.0%**	**23.8%**	**22.1%**	**19.7%**	**64.2%**	**54.2%**	**35.7%**
	LMB	41.3%	51.4%	47.2%	45.4%	64.6%	55.0%	45.3%
Group 4	Original	**19.9%**	**23.6%**	**21.6%**	**19.6%**	**63.2%**	**53.4%**	**35.5%**
	EMB	19.2%	21.4%	21.6%	19.6%	30.0%	29.2%	27.9%
	LMB	40.9%	50.4%	46.3%	44.8%	63.5%	54.2%	44.7%

Table 13. Best found makespan distribution for parallel algorithm results

	Cluster Map Type	Priority Rule Type						
		SPT	LPT	EST	EFT	LST	LFT	MSLK
Group 1	Original	**26.3%**	**28.0%**	**27.4%**	**26.1%**	**68.0%**	**65.8%**	**43.3%**
Group 2	Original	**25.4%**	**27.3%**	**26.5%**	**25.3%**	**64.8%**	**62.9%**	**42.2%**
	EMB	25.5%	26.4%	26.3%	25.2%	37.4%	37.7%	35.6%
Group 3	Original	**25.2%**	**27.0%**	**26.4%**	**25.0%**	**55.5%**	**53.5%**	**40.0%**
	LMB	44.0%	45.5%	44.1%	42.0%	56.3%	53.7%	48.0%
Group 4	Original	**25.0%**	**26.7%**	**25.9%**	**24.7%**	**54.5%**	**52.5%**	**39.4%**
	EMB	25.0%	25.8%	25.7%	24.6%	35.2%	35.9%	34.0%
	LMB	43.2%	44.4%	43.3%	41.2%	55.2%	52.8%	47.0%

The values obtained in Tables 12 and 13 show similar changes to the average makespan reduction analysis. In Group 2, the EMB application of the CB-PL method does not increase the percentage of best results for the rules similar to makespan reduction analysis. However, in Group 3, the rules except for LST and LFT show a significant increase compared to the bench values in Group 1. The important outcome from tables 12 and 13 is that around 10~15% decrease of original results from LST and LFT values in Group 3 compared to Group 1. The decrease holds for both serial and parallel algorithm results. The decrease implies that the LMB application of the method generated lower makespans in the other rules than the LST and LFT. Therefore, this shows that the CB-PL method improved the results obtained from rules and lowered the makespans obtained compared to the original results of priority rules. 22~23% of the best-found makespans in the original results had decreased with the application of LMB for both serial and parallel algorithm approaches.

The downside of the CB-PL method in this experiment is that the method does not help other rules to overshadow the LST and LFT rules. However, the CB-PL method's contribution to decreasing overall makespans compared to the original application becomes notable with the emergence of machine learning applications for the priority rules selection in RCPSPs (Guo *et al.*, 2021).

5.4 Computational Time Analysis

The additional steps' effect on computational time is reported in Table 14 as the average percentage time increases for the serial algorithm. The parallel algorithm time complexity is the same as the serial algorithm, so it is skipped. The calculations are completed on a computer with an Intel Core i5-6600 CPU with a clock speed of 3.30GHz processor on a Windows OS.

Table 14. The average percentage of completion CPU time changes of the serial algorithm by dataset

Data Type	Cluster Map Type	Priority Rule Type						
		SPT	LPT	EST	EFT	LST	LFT	MSLK
All	EMB	18.3%	13.0%	14.5%	14.8%	15.6%	16.0%	16.3%
	LMB	17.9%	12.2%	13.3%	12.6%	12.0%	13.2%	13.8%
J30	EMB	24%	21%	22%	21%	30%	25%	26%
	LMB	28%	26%	26%	18%	21%	18%	17%
J60	EMB	21%	3%	14%	14%	7%	10%	9%
	LMB	15%	0%	12%	12%	3%	6%	7%
J90	EMB	14%	13%	10%	10%	9%	13%	14%
	LMB	14%	11%	7%	6%	8%	13%	14%
J120	EMB	16%	14%	12%	15%	16%	16%	16%
	LMB	16%	13%	10%	14%	15%	15%	16%

The CB-PL method, on average, increases the time needed for algorithms to generate results between 12~19%. Table 14 can also be examined for each data types' time increase analysis. Depending on the rule and the network properties, the method's time increase effect can be minimal, as seen in the J60 sets' LMB application. Therefore, the computational time effect of the CB-PL method is also observed as feasible as it does not increase the time requirement exponentially.

The analysis of the experiment shows that the CB-PL method application on the selected priority rules presents a significant makespan reduction on average for the late map-based CB-PL applications with a reasonable time increase. The only exception is the Latest Start Time (LST) and Latest Finish Time (LFT) priority rules, which show no deterioration but are still not significant, as the late map-based clustering uses a similar approach. However, the CB-PL method application increases the other rules' reliance, as they generate overall lower makespans than the original applications of LFT and LST rules. Further, the CB-PL method's performance increases as the number of activities in the project network increases.

6. SUMMARY AND CONCLUSION

This paper presents the cluster-based priority lists method for the Resource Constraint Project Scheduling Problems. The method uses a task clustering approach to project network activities to create a new type of priority list. The new list is built on the existing priority rules. The CB-PL method aims to improve the priority lists obtained from these rules. The improvements are observed by either decreasing the obtained result or by the way the method generates new lower bound values for the given RCPSP. To the best of our knowledge, there has been no research on activity cluster and priority list combination for the RCPSP, even though task clustering is an effective method in other scheduling problems. The task clustering for other scheduling problems was observed, but they were either case-specific or not identical.

Therefore, the CB-PL method was based on related study areas in which similar studies were covered. The base of the proposed method was integrated from the study of Jedari and Dehghan (2009) on multiprocessor scheduling problems. Their method was adjusted with a new proposition for applying to the RCPSPs. To that extent, two types of maps to generate CB-PLs were proposed. Namely, early map-based CB-PL and late map-based CB-PL. Then, the method's effectiveness was demonstrated by comparing original priority lists using seven different priority rules with two types of constructive heuristics. The numerical experiment results show that the CB-PL method outperforms original priority lists with the late map-based application of the method. Therefore, our results suggest that using the CB-PL method with a late map-based clustering approach can increase the performance of constructive heuristics applications in resource-constrained project scheduling problems.

The goal of the CB-PL method is to improve makespans obtained by the original lists as heuristics are preferred for projects with a large number of activities in RCPSP. The proposed method results show that, on top of its overall performance, the method performs exceptionally well with many activity instances compared to low ones. The performance improvement in large project instances is not just with the makespan improvement but also with the number of instances improved. So, the reliability of the method for improving the makespan becomes consistent. Such that, the addition of the CB-PL option before running a commercial program to solve an RCPSP would be practical. Another practical implication of the CB-PL method is how priority rule types are levelized at their heuristic performance. The levelized performance would result in decreasing changes to obtain worse results by less knowledgeable individuals responsible for the schedule creation. In other words, the selection of priority rule is relaxed on its significance. In favor of these implications, the experimental results in this paper show that the CB-PL method application in heuristics is effective compared to original priority list applications. Also, a recent study combines a machine learning approach to select priority rules in project networks for the RCPSP (Guo *et al.*, 2021). As machine learning approaches become more popular in recent years, the best-found makespan increase observed in the analysis

section gains more significance. Hence, we think that our method will provide more to improve managerial applications of the resource-constrained project scheduling problems.

ACKNOWLEDGEMENT

This work is a part of the first author's degree research, and the author was supported by the Republic of Turkey Ministry of National Education, Directorate General for Higher and Overseas Education. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A4A1029124). This research was supported by Brain Korea 21 FOUR.

REFERENCES

Adam, T. L., Chandy, K. M., and Dickson, J. R. (1974). A Comparison of List Schedules for Parallel Processing Systems. *Communications of the ACM*, 17(12): 685–690. https://doi.org/10.1145/361604.361619.

Alex Joseph, K. P. and Saini, G. (2018). Cluster-Based Scheduling Algorithm. *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 1327–1331. https://doi.org/10.1109/RTEICT42901.2018.9012401.

Blazewicz, J., Lenstra, J. K., and 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.

Boeres, C., Filho, J. V., and Rebello, V. E. F. (2004). A Cluster-Based Strategy for Scheduling Task on Heterogeneous Processors. *16th Symposium on Computer Architecture and High Performance Computing*, 214–221. https://doi.org/10.1109/SBAC-PAD.2004.1.

Brooks, G. H. and White, C. R. (1965). An Algorithm for Finding Optimal or Near Optimal Solutions to The Production Scheduling Problem. *The Journal of Industrial Engineering*, 16(1): 34–40.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., and Pesch, E. (1999). Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operational Research*, 112(1): 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5.

Coelho, J. and Vanhoucke, M. (2018). An Exact Composite Lower Bound Strategy for The Resource-Constrained Project Scheduling Problem. *Computers and Operations Research*, 93: 135–150. https://doi.org/10.1016/j.cor.2018.01.017.

Davis, E. W. and Patterson, J. H. (1975). A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science*, 21(8): 944–955. https://doi.org/10.1287/mnsc.21.8.944.

Demeulemeester, E., Herroelen, W., Simpson, W. P., Baroum, S., Patterson, J. H., and Yang, K.-K. (1994). On A Paper by Christofides *et al.* For Solving The Multiple-Resource Constrained, Single Project Scheduling Problem. *European Journal of Operational Research*, 76(1): 218–228. https://doi.org/10.1016/0377-2217(94)90018-3.

Demeulemeester, E. L. and Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. Springer US. https://doi.org/10.1007/b101924.

Frey, B. J. and Dueck, D. (2007). Clustering by Passing Messages Between Data Points. *Science*, 315(5814): 972–976. https://doi.org/10.1126/science.1136800.

Gerasoulis, A. and Yang, T. (1992). A Comparison of Clustering Heuristics for Scheduling Directed Acyclic Graphs On Multiprocessors. *Journal of Parallel and Distributed Computing*, 16(4): 276–291. https://doi.org/10.1016/0743-7315(92)90012-C.

Gonçalves, J. F., Mendes, J. J. M., and Resende, M. G. C. (2008). A Genetic Algorithm for The Resource Constrained Multi-Project Scheduling Problem. *European Journal of Operational Research*, 189(3): 1171–1190. https://doi.org/10.1016/j.ejor.2006.06.074.

Graham, R. L. (1969). Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, 17(2): 416–429. https://doi.org/10.1137/0117039

Guo, W., Vanhoucke, M., Coelho, J., and Luo, J. (2021). Automatic Detection of The Best Performing Priority Rule for The Resource-Constrained Project Scheduling Problem. *Expert Systems with Applications*, 167: 114116. https://doi.org/10.1016/j.eswa.2020.114116.

Habibi, F., Barzinpour, F., and Sadjadi, S. (2018). Resource-Constrained Project Scheduling Problem: Review of Past and Recent Developments. *Journal of Project Management*, 3(2): 55–88.

Hajikano, K., Kanemitsu, H., Kim, M. W., and Kim, H.-D. (2016). A Task Scheduling Method after Clustering for Data Intensive Jobs in Heterogeneous Distributed Systems. *Journal of Computing Science and Engineering*, 10(1): 9–20. https://doi.org/10.5626/JCSE.2016.10.1.9.

Herroelen, W. (2005). Project Scheduling—Theory and Practice. *Production and Operations Management*, 14(4): 413–432. https://doi.org/10.1111/j.1937-5956.2005.tb00230.x.

Herroelen, W., Demeulemeester, E., and De Reyck, B. (1999). A Classification Scheme for Project Scheduling. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications* (pp. 1–26). Springer US. https://doi.org/10.1007/978-1-4615-5533-9_1.

Jedari, B. and Dehghan, M. (2009). Efficient DAG Scheduling with Resource-Aware Clustering for Heterogeneous Systems. In R. Lee, G. Hu, and H. Miao (Eds.), *Computer & Information Science 2009* (pp. 249–261). Springer. https://doi.org/10.1007/978-3-642-01209-9_23.

Kelley, J. E. J. (1963). The Critical-Path Method: Resource Planning and Scheduling. *Industrial Scheduling*, 347–365.

Kolisch, R. (1995). *Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes*. Physica-Verlag Heidelberg. https://doi.org/10.1007/978-3-642-50296-5

Kolisch, R. and Hartmann, S. (2006). Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. *European Journal of Operational Research*, 174(1): 23–37. https://doi.org/10.1016/j.ejor.2005.01.065.

Kolisch, R., Schwindt, C., and Sprecher, A. (1999). Benchmark Instances for Project Scheduling Problems. In J. Węglarz (Ed.), *Project Scheduling: Recent Models, Algorithms and Applications* (pp. 197–212). Springer US. https://doi.org/10.1007/978-1-4615-5533-9_9.

Kolisch, R., and Sprecher, A. (1997). PSPLIB - A Project Scheduling Problem Library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1): 205–216. https://doi.org/10.1016/S0377-2217(96)00170-1.

Lawrence, S. (1985). Resource Constrained Project Scheduling – A Computational Comparison of Heuristic Scheduling Techniques, Technical Report. *Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh*.

Li, K. Y. and Willis, R. J. (1992). An Iterative Scheduling Technique for Resource-Constrained Project Scheduling. *European Journal of Operational Research*, 56(3): 370–379. https://doi.org/10.1016/0377-2217(92)90320-9.

Lu, H., Cao, J., Lv, S., Wang, X., and Liu, J. (2015). A Comparative Study of DAG Clustering. *2015 International Conference on Information Society (i-Society)*, 84–89. https://doi.org/10.1109/i-Society.2015.7366864.

MacQueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 281–297.

Merkle, D., Middendorf, M., and Schmeck, H. (2002). Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE Transactions on Evolutionary Computation*, 6(4): 333–346. https://doi.org/10.1109/TEVC.2002.802450.

Möhring, R. H., Schulz, A. S., Stork, F., and Uetz, M. (2003). Solving Project Scheduling Problems by Minimum Cut Computations. *Management Science*, 49(3): 330–350. https://doi.org/10.1287/mnsc.49.3.330.12737.

Olaguíbel, R. A.-V. and Goerlich, J. M. T. (1989). Chapter 5 - Heuristic Algorithms for Resource-Constrained Project Scheduling: A Review and An Empirical Analysis. In R. Słowiński and J. Węglarz (Eds.), *Advances in Project Scheduling* (pp. 113–134). Elsevier. https://doi.org/10.1016/B978-0-444-87358-3.50009-2.

Rousseeuw, P. J. (1987). Silhouettes: A Graphical Aid to The Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20: 53–65. https://doi.org/10.1016/0377-0427(87)90125-7.

Roy, B., and Sen, A. K. (2019). Meta-heuristic Techniques to Solve Resource-Constrained Project Scheduling Problem. In S. Bhattacharyya, A. E., Hassanien, D., Gupta, A. Khanna, and I. Pan (Eds.), *International Conference on Innovative Computing and Communications* (pp. 93–99). Springer. https://doi.org/10.1007/978-981-13-2354-6_11.

Roy, S. K., Devaraj, R., Sarkar, A., and Senapati, D. (2021). *SLAQA*: Quality-level Aware Scheduling of Task Graphs on Heterogeneous Distributed Systems. *ACM Transactions on Embedded Computing Systems*, 20(5): 45:1-45:31. https://doi.org/10.1145/3462776.

Santoso, L. W., Sinawan, A. A., Wijaya, A. R., Sudiarso, A., Masruroh, N. A., and Herliansyah, M. K. (2017). Operating Room Scheduling Using Hybrid Clustering Priority Rule and Genetic Algorithm. *AIP Conference Proceedings*, 1902(1): 020032. https://doi.org/10.1063/1.5010649.

Ulusoy, G. and Hazır, Ö. (2021). Resource Constrained Project Scheduling. In G. Ulusoy and Ö. Hazır (Eds.), *An Introduction to Project Modeling and Planning* (pp. 199–250). Springer International Publishing. https://doi.org/10.1007/978-3-030-61423-2_7.