(01-018) - Determination of Search Domain of Resource Constrained Project Scheduling Problem

Bettemir, Onder Halis [1]

[1] Inonu University

Resource constrained project scheduling is an important problem of construction management. Procurement of excessive number of resources significantly increases the total project cost. Therefore, restrictions on the number of resources are defined and the optimum solution of resource constrained project scheduling problem provides the shortest project completion time. The aforementioned problem is NP-Hard since the search domain of the problem increases exponentially if the number of activities increases. In this study, the size of the search domain of the problem is examined by considering the number of activities. Contributions of the activities on the serial and parallel paths are examined. Equations representing the size of the search domain against the number of activities are derived for the aforementioned situations. The provided relationships represent the difficulty of the examined problem. Researchers can implement more robust optimization algorithms by considering the difficulty of the problem.

Keywords: Resource constrained project scheduling; search domain; optimization

Determinación del dominio de búsqueda del problema de programación de proyectos con recursos limitados

La programación de proyectos con recursos limitados es un problema importante de la gestión de la construcción. La contratación de un número excesivo de recursos aumenta significativamente el coste total del proyecto. Por lo tanto, se definen restricciones en el número de recursos y la solución óptima del problema de programación de proyectos con recursos limitados proporciona el menor tiempo de finalización del proyecto. Dicho problema es NP-Hard, ya que el dominio de búsqueda del problema aumenta exponencialmente si se incrementa el número de actividades. En este estudio, el tamaño del dominio de búsqueda del problema se examina considerando el número de actividades. Se examinan las contribuciones de las actividades en los caminos en serie y en paralelo. Se derivan ecuaciones que representan el tamaño del dominio de búsqueda frente al número de actividades para las situaciones mencionadas. Las relaciones proporcionadas representan la dificultad del problema examinado. Los investigadores pueden implementar algoritmos de optimización más robustos teniendo en cuenta la dificultad del problema.

Palabras clave: Programación de proyectos con recursos limitados; dominio de búsqueda; optimización

Correspondencia: Önder Halis Bettemir, onder.bettemir@inonu.edu.tr

1. Introduction

Resource-constrained project scheduling problem (RCPSP) aim to schedule all activities that constitute a project in a way that has the shortest project duration, taking into account their predecessor relationships, without violating the constraints on resources (Ulusoy and Ozdamar, 1994). In this way, the project is completed with the possible shortest duration with the limited available resources. RCPSP is very important for the construction industry as it ensures that the project is completed in a reasonable time without establishing a crowded construction site and provides significant savings in construction equipment investments.

Creating a resource-constrained construction schedule is related to the production process which can be the construction of the structure, or the scheduling of production groups. It is also of increasing importance for companies that work to order, which resort to capacity reduction in order to implement the lean business concept. In RCPSP, it is assumed that a project consisting of the set V={0,1,….,n,n+1} will be solved which consists of n activities. Activities up to 1,…,n in the set V are real activities, dummy activities located at the starting and ending of the project j=0, j=n+1, respectively are fictional activities. Activities are not allowed to be interrupted. In this type of problem, renewable resources are limited and all data are integers. The aim of RCPSP is to find the shortest construction schedule without violating the restrictions imposed by priority relationships and limited resource availability.

The shortest project duration without exceeding the allowable Tupper limit is the completion time of the activities j ϵ V repeated back and forth with early completion time E and late completion time G in the interval [E, G] using priority relations. Similarly, the interval [E, G] can be calculated from the early start time E and the late start time G, which are connected from below and above, respectively, to indicate the applicable start times first. RCPSP is a very important problem for the construction industry where project management is applied. Since each construction project is unique, a unique schedule will be created. The initial construction schedule is prepared with the assumption that an infinite number of resources are available. However, it is possible that the initial construction schedule may require more resources than the amount of available resources. In this case, it will not be possible to realize the existing construction schedule within the targeted time and delay of the construction schedule will be unavoidable. As a result of the delay in construction, the contractor may not be able to fulfil its commitment and may face delay penalties. Due to the mentioned financial burden, the optimum solution of RCPSP is very important for construction, manufacturing, service and industrial sectors. Due to the importance of the problem, there are many studies on the solution of the RCPSP in the literature.

Zhang et al. (2006) solved RCPSP with particle swarm optimization based algorithms. Particle swarm optimization patterns have been developed for resource-constrained project scheduling problems by considering two particle representations, priority-based and permutation-based representation. According to the solution results of the trial problems consisting of 480 projects with 30 activities, permutation-based PSO showed superior performance than priority-based PSO in solving the resource-constrained project scheduling problem. PSO-based method provided better schedules than meta-heuristics, genetic algorithm, and simulated annealing algorithms. Kanıt et al. (2005) discussed the impact of resource constraints on construction costs. If there are constraints on the resources that need to be used during the construction process, the impact of the constraints on the project duration and probable cost increase was investigated.

Kolish (1996) examined the serial and parallel resource-constrained project programming methods on RCPSP by conducting an in-depth computational study on 360 systematically generated examples. Merkle and Middendorf (2002) solved RCPSP with ant colony optimization (ACO) which obtains optimum solutions of 130 problems out of 396 sample

problems presented a better performance than genetic algorithm, simulated annealing, and taboo search. Hartmann (1998) solved RCPSP with genetic algorithm (GA) on 30 and 60 activity projects produced by ProGen. The permutation-based GA approach gave the best results in projects with 30 activities. Özdamar and Ulusoy (1995) classified RCPSP according to a specific purpose and constraints and discussed time-cost based objectives and renewable-non-renewable resource constraints. Christofides (1987) solved problems with 40 to 25 activities for 3 resource types, while Olaguíbel and Goerlich (1989) solved problems with 38, 51 and 103 activities for 6 resource types. Özdamar and Ulusoy (1995) obtained a solution with an average deviation of 2% from the optimum result with the heuristic pruning method using a comprehensive branching scheme.

Arauzo et al. (2009) solved RCPSP by dynamically allocating resources with a multi-factor system (MFS). The performance of priority rules for the resource-constrained multi-project scheduling problem (RCMPSP) with uncertain activity times is also discussed. It applied 17 hybrid rules, including minimum delay, maximum delay, and minimum delay (Wang et al., 2015). Zhu et al. (2019) solved multi-source RCPSP with discrete adversarial multi-version optimization algorithm. Lin et al. (2020) solved multi-source RCPSP with genetic programming hyper-heuristic method. Pellerin et al. (2020) examined the hybrid meta-heuristic methods applied for the RCPSP and listed the common features of the prominent methods. Birjandi and Mousavi (2019) solved the stochastic RCPSP using a heuristic rule. Laszczyk and Myszkowski (2019) solved the multi-source RCPSP with a multi-objective function evolutionary evolution algorithm. Chakrabortty et al. (2020) solved multi-mode RCPSP by modified variable neighbor search heuristic method. Tirkolaee et al. (2019) solved the multi-objective function multi-modal RCPSP with Pareto-based algorithms. Creemers (2019) examined stochastic RCPSP in a way that it can interrupt activities at any time and continue them after a period of time. Bettemir and Sönmez (2015) solved RCPSP by genetic algorithm simulating annealing meta-heuristic algorithm.

2. Methodology

In order to detect the entire search domain of RCPSP, the paths the network must be analysed. This process requires calculating how many different priority sequences can be assigned to all critical and non-critical activities in the network without violating successor-predecessor relationships. The first step in determining the search space is to determine how many paths are there in the network. After the determination of the paths of the network, the two paths are matched in order to determine the number of different priorities that can be assigned to create a single path. The resulting combined path is matched with a path that has not been examined before, and number of different priority arrangement that can be created by considering the resulting activity combinations is determined. Aforementioned procedure is similar to the computation of number of feasible schedules that can be obtained for the resource levelling problem (RLP). The number of feasible schedules is computed by considering the number of serial activities and their total float durations. The number of feasible schedule that can be produced by delaying noncritical activities for a serial path which has m activities with n total float durations is given in Eq. 1 (Bettemir and Erzurum, 2021).

$$FAPN = \frac{1}{m!}\prod_{i=1}^{m}(n+i) \tag{1}$$

The equation can be adopted for the computation of number of feasible activity priority listing for the resource constraint project scheduling problem. In Eq. 1, FAPN refers to the feasible activity priority number that can be produced by considering two parallel paths where activities are serially connected at each path. In a network more complex relationships between the

activities are expected. In such cases the network is divided into small parts to obtain pure serial activities and paths. The search domain of the corresponding paths is investigated by implementing Eq. 1 and the obtained permutations are added to the previously obtained permutations. This study aims to investigate the adoptability of the equation derived for the computation of the number of feasible schedule evaluations of resource levelling problem for the resource constraint scheduling problem. Number of available schedules are computed by adopting the Eq. 1 and manually by examining the all possible priority scenarios. The presented methodology is implemented on the presented case study problems.

3. Case Study Problems

3.1 Case study problem 1

First case study problem represented as 1(a) consists of four activity and two parallel paths each consisting of two serial activities. The number of permutation for activity priority list can be computed as $\frac{4!}{2!2!} = 6$. The four activities can be sorted i.e. prioritized in 4! different ways. However, activity A has to be in front of B and similarly activity C has to be in front of activity D. Therefore, these activities cannot produce unique sorting sequence and the aforementioned affect is compensated by dividing by the number of serial activities in the path. The feasible activity priority list of the first case study problem is given in Table 1.

Figure 1: Normal and modified network diagrams of the first case study problems

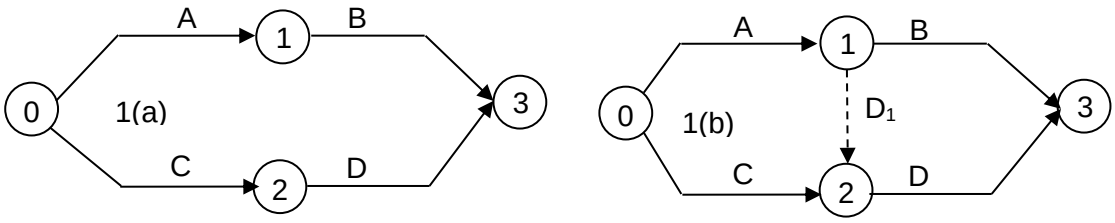


Table 1: Total list of combinations of feasible activity priorities

Shift C	Shift CD	Shift CDA
ABCD	ACDB	CDAB
ACBD	CADB	
CABD		

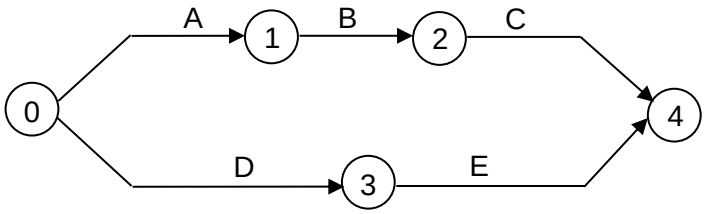
The simple first case study example is modified as shown in Figure 1(b) by adding a dummy activity between the first and second nodes. This prevents commencement of activity D without completion of activity A. The number of feasible priority combination lessens because of the modification.

Precedence relationships defined by D_1 is violated if activity D starts without the completion of activity A. 6th item of the activity priority list given as red highlighted in Table 1 violates the aforementioned logical relationship since it requires commencement of activity A after the finish of activity D. The search domain of the modified first case study problem contains 5 feasible activity priority alignments.

3.2 Case study problem 2

Second case study problem consists of five activities and two parallel paths where the first path contains three and the latter contains two serial activities. The number of permutation for activity priority list can be computed as $\frac{5!}{3!2!}=10$. Figure 2 illustrates the network diagram of the second case study problem.

Figure 2: Network diagram of the second case study problem



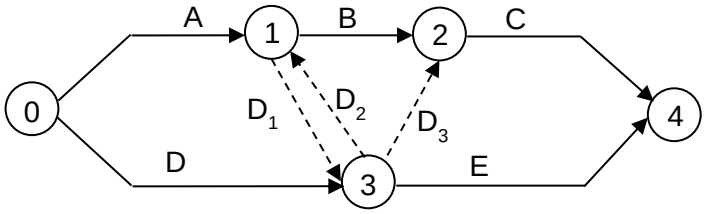
The activity priorities are formed by delaying the activities of path D-E on the activities of path A-B-C. The activities D and E are placed behind the path A-B-C and the first priority list is formed. In the first column of Table 2, activity D is shifted left by one activity at each row and four feasible priority alignments are obtained. In the second column activities D-E are shifted left by one activity and only activity D is shifted left by one and three unique feasible priority alignments are obtained. Third and fourth columns are obtained in a similar manner. Total number of feasible combination becomes 10 and it is compatible with the analytical formula. The feasible activity priority list of the second case study problem is given in Table 2.

Table 2: Combination list of feasible activity priorities of case study 2

Shift D	Shift DE-1	Shift DE-2	Shift DE-3
ABCDE	ABDEC	ADEBC	DEABC
ABDCE	ADBEC	DAEBC	
ADBCE	DABEC		
DABCE			

Activity precedence relationships are made more complex by adding dummy activities and in order to save space, three different modifications are shown in Figure 3 as one figure. Only one dummy activity is valid for each modification. Implemented dummy activity D_1 enforces to finish activity A before the commencement of activity E. The violation of the added precedence relationship can be done if activity A starts after the finish of activity E. In this case DEABC priority list, highlighted in turquoise, is removed from the list given in Table 2.

Figure 3: Network diagram of the modified second case study problem (only one dummy is applied at a time)

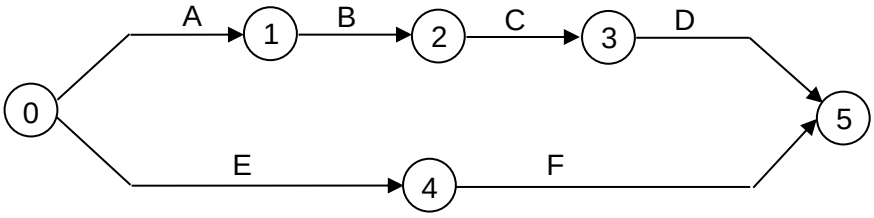


Attaching only dummy activity D_2 enforces to finish activity D before the start of activity B. Violation of the activity precedence relationship enforced by D_2 can be achieved when the priority of activity B is higher than activity D. The first path only one activity, activity C, starts after the activity B. The second path has two activities, activities D and E, which leads to formation of three activity priority arrangements. Priority list given in the first two rows of the first column and the first row of the second column of Table 2, highlighted in red, violates the aforementioned activity precedence relationship. Number of feasible activity priority arrangement of the modified project reduces to 7 when infeasible priority arrangements are eliminated. Final modification done on the second case study is the attaching dummy activity D_3. Violation of the activity precedence relationship enforced by D_3 can be achieved when the priority of activity C is higher than activity D. In this case there is not any activity starting after activity C and therefore the path D-E cannot be permutated. The only infeasible priority list is the one written in yellow colour in Table 2. Consequently, the number of feasible activity priority list becomes 9 for the third modification of the second case study problem.

3.3 Case study problem 3

Third case study problem consists of six activities and two parallel paths where the first path contains 4 and the latter contains two serial activities, respectively. The number of permutation for activity priority list can be computed as $\frac{6!}{4!2!}$=15. The feasible activity priority list of the second case study problem is given in Table 3.

Figure 4: Network diagram of the third case study problem

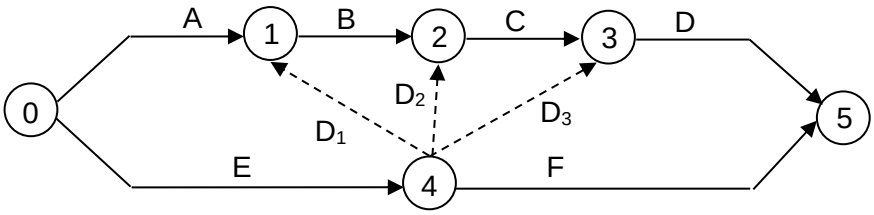


Activity priorities are formed by delaying the activities of path E-F on the activities of path A-B-C-D. The activities E and F are placed behind the path A-B-C-D and the first priority list given in the first row of the first column of Table 3 is formed. Activities, A-B-C-D, in front of the activity E provides four additional activity priority combination apart from the initial priority arrangement. The second column is obtained by left shifting the priority of activities E-F by one. In this case three additional priority arrangements are generated and the left shift of the path E-F is continued until it reaches to the in front of the priority list. Activities E-F are left shifted by one and the activity priorities shown in the third column are obtained by left shifting activity E by one at a time. The remaining two columns are obtained in a similar manner.

Table 3: Combination list of feasible activity priorities of case study 3

Shift E	Shift EF-1	Shift EF-2	Shift EF-3	Shift EF-4
ABCDEF	ABCEFD	ABEFCD	AEFBCD	EFABCD
ABCEDF	ABECFD	AEBFCD	EAFBCD	
ABECDF	AEBCFD	EABFCD		
AEBCDF	EABCFD			
EABCDF				

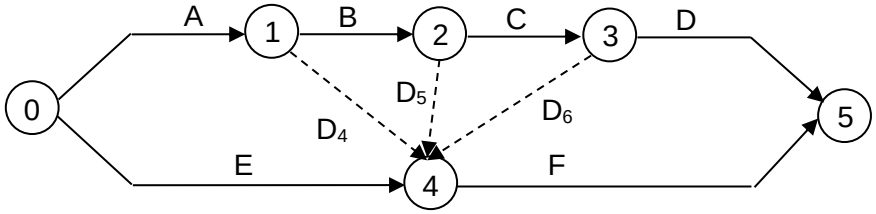
Figure 5: Network diagram of the modified third case study problem (only one dummy is applied at a time)



Effects of dummy activities are examined on the modified network. In order to save space, three different modifications are shown in the same figure. Only one dummy activity is valid for each modification among the modifications shown in Figure 5. First dummy activity D_1 enforces to finish activity E before the commencement of activity B. If activity B starts before the execution of the activity E the precedence relationship is violated. This case removes the top rows of the list given in Table 3 and only bottom two rows becomes feasible for each column except for the final column which has only one feasible activity priority list. The search domain becomes 9 for the first modification. Second dummy activity D_2 enforces to finish activity E before the commencement of activity C. The added precedence relationship can be violated if activity C starts before the execution of the activity E. Activity E can be on the left of activities A-B-C to produce feasible priority list. At each column activity E is left shift by one at each row therefore only the most bottom three rows become feasible for each column except for the last two columns which has only two and one feasible activity priority lists, respectively. The search domain becomes 12 for the second modification.

Third dummy activity D_3 enforces to finish activity E before the commencement of activity D. The added precedence relationship can be violated if activity D starts before the execution of the activity E. Activity E can be on the left of activities A-B-C-D to produce feasible priority list. At each column activity E is left shift by one at each row therefore only the most bottom four rows become feasible for each column. If the number of row is less than four, all of the items of the corresponding column become feasible. Only one item given in Table 3 become infeasible and the total search domain is computed as 14 for the third modification.

Figure 6: Network diagram of the remaining modifications of the third case study problem (only one dummy is applied at a time)



Dummy activity D_4 enforces to finish activity A before the commencement of activity F. The violation of the added precedence relationship can be done if activity F starts before the execution of the activity A. Among the activity priority lists presented in Table 3 the one given in the fifth column violates the rule. Total search domain becomes 14. Dummy activity D_5 enforces to finish activity B before the commencement of activity F. The violation of the added precedence relationship can be done if activity F starts before the execution of the activity B. Among the activity priority lists presented in Table 3 the last two rows violates the rule and the total search domain becomes 12. Dummy activity D_6 enforces to finish activity C before the commencement of activity F. The violation of the added precedence relationship can be done if activity F starts before the execution of the activity C. Among the activity priority lists presented in Table 3 the last three rows violates the rule and the total search domain becomes

9. Conjugate modifications D_1 to D_3 and D_4 to D_6 provide the same search domain even though the network precedence relations are not symmetric.

3.4 Case study problem 4

Fourth case study problem consists of six activities and two parallel paths where each path contains 3 serial activities as given in Figure 7. The number of permutation for activity priority list can be computed as $\frac{6!}{3!3!}$=20. The feasible activity priority list of the fourth case study problem is given in Table 4.

Figure 7: Network diagram of the fourth case study problem

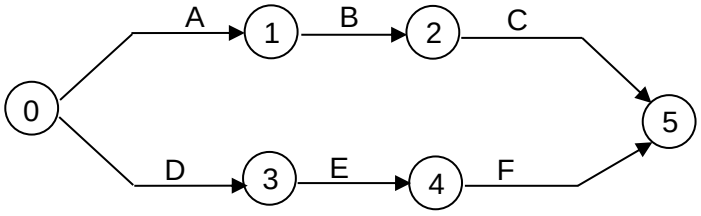


Table 4: Combination list of feasible activity priorities of case study 4

Shift D	Shift DE-1	Shift DE-2	Shift DE-3
ABCDEF	ABDECF	ADEBCF	DEABCF
ABDCEF	ADBECF	DAEBCF	
ADBCEF	DABECF		
DABCEF			

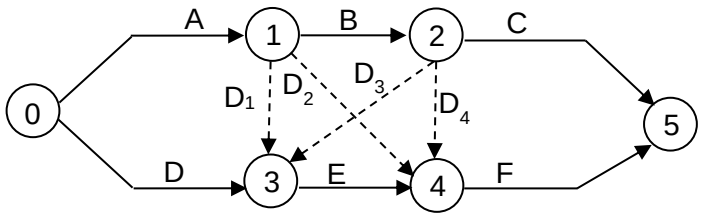
Shift DEF-2	Shift DE-1
ADEFBC	DEAFBC
DAEFBC	

Shift DEF-1	Shift DE-1	Shift DE-2
ABDEFC	ADEBFC	DEABFC
ADBEFC	DAEBFC	
DABEFC		

Shift DEF-3
DEFABC

Shortest path of the fourth case study problem consists of three activities. Therefore the activity priority lists are obtained in three steps as given in Table 4. Computation process starts with ABCDEF priority list and the first column of the first table is formed by left shifting the activity D. Second column of the same table starts by left shifting of the activities D-E and at the following rows only the activity D is left shifted until it reaches the front of the sequence. The left shifting of the activities D-E are continued until the activity pair reaches the most left of the sequence. This completes the first table given in Table 4 and activity set D-E-F is left shift by one activity which is given in the second table of Table 4. Left shifting process of activities D-E is carried on in a similar manner with the previous table and 5 additional activity priorities are formed. Activities D-E-F are left shift by one to form third table of the Table 4. This process continues until the activity set D-E-F reaches to the most left of the activity priority list. The search domain of the fourth case study problem is 20.

Figure 8: Network diagram of the modified fourth case study problem (only one dummy is applied at a time)



Activity precedence relationships shown in Figure 8 are applied one by one at a time. First dummy activity D_1 enforces to finish activity A before the commencement of activity E. This case removes the last item of the each table given in Table 4 and removes 4 activity priority arrangements and search domain reduces to 16. D_2 enforces activity A to finish activity F. This case removes only the last activity priority arrangement of the last table given in Table 4. Search domain of the second modification becomes 19. D_3 enforces to finish activity B before the commencement of activity E. This case removes the last two columns of the each table given in Table 4. Last table is an exception since it has only one column and the item at that column also violates the logical relationship. In the end the modification removes 10 activity priority arrangements. Search domain of the third modification becomes 10. D_4 enforces to finish activity B before the commencement of activity F. This case removes all of the items of the last two tables given in Table 4. The modification removes 4 activity priority arrangements and search domain of the fourth modification becomes 16.

3.5 Case study problem 5

Fifth case study problem consists of seven activities and two parallel paths where longer path contains 4 serial activities and the shorter path contains 3 serial activities as given in Figure 9. The number of permutation for activity priority list can be computed as $\frac{7!}{4!3!}$=35. The feasible activity priority list of the fifth case study problem is given in Table 5.

Figure 9: Network diagram of the fifth case study problem (only one dummy is applied at a time)

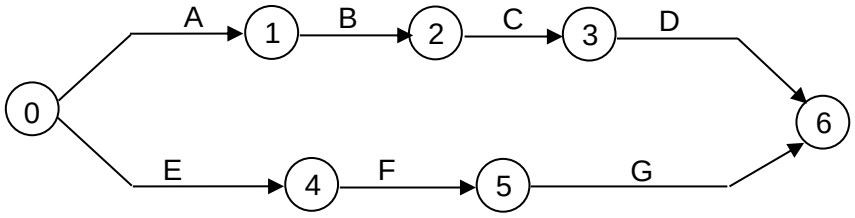


Table 5: Combination list of feasible activity priorities of case study 5

Shift E	Shift EF-1	Shift EF-2	Shift EF-3	Shift EF-4
ABCDEFG	ABCEFDG	ABEFCDG	AEFBCDG	EFABCDG
ABCEDFG	ABECFDG	AEBFCDG	EAFBCDG	
ABECDFG	AEBCFDG	EABFCDG		
AEBCDFG	EABCFDG			
EABCDFG				

Shift EFG-1	Shift EF-1	Shift EF-2	Shift EF-3

ABCEFGD	ABEFCGD	AEFBCGD	EFABCGD
ABECFGD	AEBFCGD	EAFBCGD	
AEBCFGD	EABFCGD		
EABCFGD			

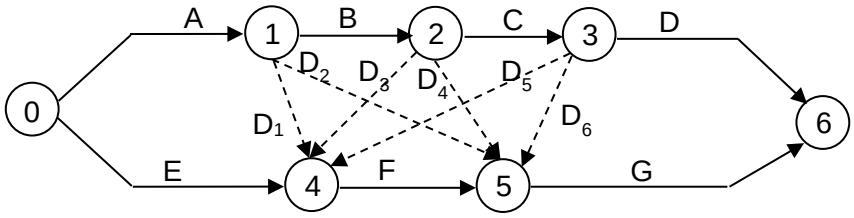
Shift EFG-2	Shift EF-1	Shift EF-2
ABEFGCD	AEFBGCD	EFABGCD
AEBFGCD	EAFBGCD	
EABFGCD		

Shift EFG-3	Shift EF-1
AEFGBCD	EFAGBCD
EAFGBCD	

Shift EFG-4
EFGABCD

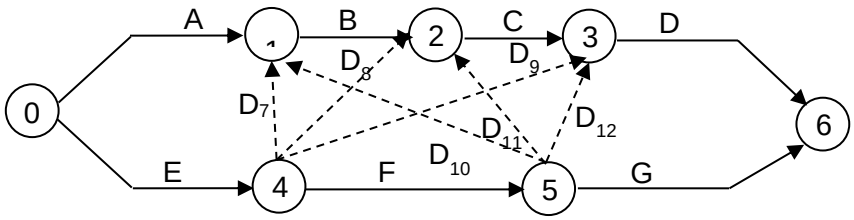
Shortest path of the fifth case study problem consists of three activities and therefore the activity priority lists are produced in three left shift stages as given in Table 5. Computation process starts with ABCDEFG priority list and the first column of the first table is formed by left shifting the activity E. Second column of the same table starts by left shifting of the activities E-F and at the following rows only the activity E is left shifted until it reaches the front of the sequence. The left shifting of the activities E-F are continued until the activity pair reaches the most left of the sequence. This completes the first table given in Table 5 and activity set E-F-G is left shift by one activity as given in the second table of Table 5. Left shifting process of activities E-F is carried on in a similar manner with the previous table until E-F reaches the front. Activities E-F-G are left shift by one to form third table of the Table 5. This process is continued until the activity set E-F-G reaches to the most left of the activity priority list. The search domain of the fifth case study problem is obtained as 35.

Figure 10: Network diagram of the modified fifth case study problem (only one dummy is applied at a time)



Dummy activity D_1 enforces activity A to finish before the commencement of activity F. This case removes the last item of the each table given in Table 5 and removes 5 activity priority arrangements. Search domain of the first modification becomes 30. D_2 enforces activity A to finish before the commencement of activity G and removes only the last activity priority arrangement of the last table given in Table 5, EFGABCD. Search domain of the second modification becomes 34. Third dummy activity D_3 enforces to finish activity B before the commencement of activity F. This case removes the last two columns of the each table given in Table 5. Third modification removes 13 activity priority arrangements and search domain becomes 22. D_4 enforces to finish activity B before the commencement of activity G. This case removes all of the items of the last two tables given in Table 5. The modification removes 4 activity priority arrangements and search domain becomes 31. D_5 enforces activity C to finish before the commencement of activity F. This case removes the last three columns of the each table given in Table 5. Fifth modification removes 22 activity priority arrangements and search domain becomes 13. D_6 enforces to finish activity C before activity G. This case removes all of the items of the last three tables given in Table 5. The modification removes 10 activity priority arrangements and search domain becomes 25.

Figure 11: Network diagram of the second modification of the fifth case study problem (only one dummy is applied at a time)

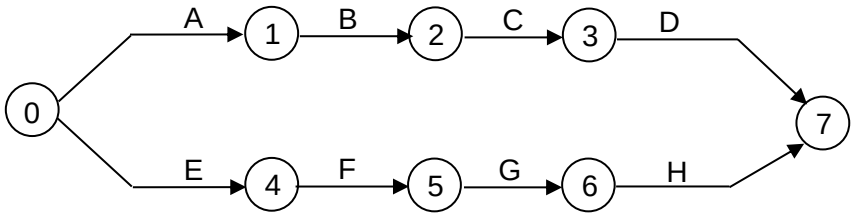


Dummy activity D_7 enforces activity E to finish before the commencement of activity B. This case removes the bottom two rows of the each table given in Table 5 and removes 25 activity priority arrangements. Search domain of the seventh modification becomes 10. D_8 enforces activity E to finish before the commencement of activity C. This case removes the first two rows of the first table and the first row of the second column of the first table given in Table 5. Moreover, first row of the first column of the second is removed and the search domain becomes 31. Ninth dummy activity D_9 enforces to finish activity E before the commencement of activity D. This case removes only ABCDEFG and search domain of the ninth modification becomes 34. Tenth dummy activity D_{10} enforces activity F to finish before the commencement of activity B. In this case only the last two columns of the tables given in Table 5 survives. Search domain of the tenth modification becomes 13. Eleventh dummy activity D_{11} enforces activity C to finish before the commencement of activity F. In this case last three columns of each table given in Table 5 survives and search domain of the eleventh modification becomes 22. Dummy activity D_{12} enforces activity F to finish before the commencement of activity D. In this case last four columns of the tables given in Table 5 survives. The modification removes 5 activity priority arrangements and search domain of the twelfth modification becomes 30.

3.6 Case study problem 6

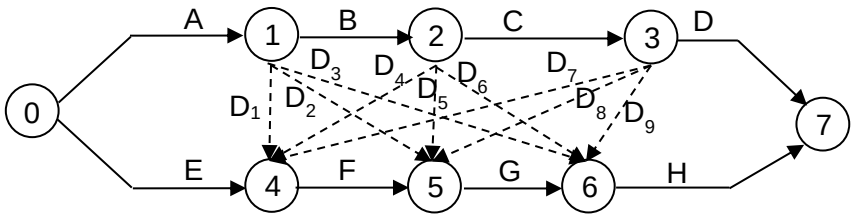
Sixth case study problem consists of eight activities and two parallel paths where each path contains 4 serial activities as given in Figure 12. The number of permutation for activity priority list can be computed as $\frac{8!}{4!4!}$=70. The feasible activity priority list of the second case study problem is given in Table 6.

Figure 12: Network diagram of the sixth case study problem



Search domain enumeration consists of five steps. At first step path E-F-G-H is at the farthest behind of the priority list and activity E is left shift at each row until it reaches to the front of the priority list. Then E-F is left shift by one and again activity E is left shifted by one until it reaches to the front of the priority list. This completes the 5x5 upper triangular matrix. Activities E-F-G are left shifted by one and the same procedure is implemented to obtain 4x4 upper triangular matrix. In order to save space the priority list, Table 6, is given in this drive link. In the Table 70 activity priority lists are provided. The given figure is compatible with the analytic computation.

Figure 13: Network diagram of the modified sixth case study problem (only one dummy is applied at a time)



D_1 enforces activity A to finish before the commencement of activity F. This case removes the last item of the each table given in Table 6 and removes 5 + 4 +3 +2 +1 = 15 activity priority arrangements. Search domain of the first modification becomes 55. D_2 enforces activity A to finish before the commencement of activity G. This case removes only the last item of the each computation step given in Table 6 and as there are five steps 5 activity priority arrangements are removed. Search domain becomes 65. D_3 enforces to finish activity A before the commencement of activity H. This case removes only the final priority list, EFGHABCD, and search domain of the third modification becomes 69. D_4 enforces to finish activity B before the commencement of activity F. This case removes all of the items of the last two tables. The modification removes 35 activity priority arrangements and search domain becomes 35. D_5 enforces to finish activity B before the commencement of activity G. This case removes the last two columns of the last two tables of each step given in Table 6. Fifth modification removes 4+4+4+4+1 = 17 activity priority arrangements and search domain of the fifth modification becomes 53. D_6 enforces to finish activity B before the commencement of activity H and removes the last five priority lists given in Table 6. The modification removes 5 activity priority arrangements and search domain becomes 65.

D_7 enforces to finish activity B before the commencement of activity F. This case removes all of the items of the last three tables given in Table 6. The modification removes 6 + 6 + 6 + 3 + 1 = 22 activity priority arrangements from the first step, 6 + 6 + 3 + 1 = 16, from the second step, 6 + 3 + 1 = 10 from the third step, 3 + 1 = 4 from the fourth step and 1 from the fifth step. 53 activity priority arrangements are removed and search domain of the seventh modification becomes 17. D_8 enforces to finish activity C before the commencement of activity G. This case removes the last three columns of each step given in Table 6. Eighth modification removes 6 + 3 + 1 = 10 activity priority arrangements for the first step, 10 for the second, 10 for the third, 4 for the fourth, and 1 for the fifth step, respectively. In total 34 activity priority arrangements are eliminated and search domain of the eighth modification becomes 36. D_9 enforces to finish activity C before the commencement of activity H. This case removes all of the priority lists produced at the third, fourth and fifth steps. The modification removes 15 activity priority arrangements and search domain of the sixth modification becomes 55.

4. Discussion of Results

In this study search domain of resource constrained project scheduling problem is examined when it is solved by activity priority based meta-heuristic search algorithms. Combinatorial activity priority solution procedure assigns priorities for each individual of the population and possible resource conflicts are resolved by considering the activity priorities where activity with the lower priority is delayed.

Search domain of the aforementioned solution process is examined on six case study problems, and detailed analyses are conducted on each problem by executing modifications on the case study problems. The case study problems revealed that the factorial expansion of the search domain theory is compatible with the complete enumeration of the feasible activity

priority arrangements. The search domain expands in factorial way according to the number of serial paths and the number of activities on each serial path which illustrates that resource constrained project scheduling problem is NP-Hard. The outcome of this study can be beneficial to test the performances of the implemented meta-heuristic algorithms based on the number of schedule evaluations and the size of the search domain.

Herroelen et al. (1998) mentioned the more complex the activity relationships, the easier the problem to solve. This is because as the modifications on the case study problems illustrated addition of dummy activities thus addition of more logical relationships between the activities decreases the number of feasible activity priority arrangements. Addition of dummy activities makes the network diagram more complex but restricts the shifting of activities and decreases the search domain of the problem. Consequently, the problem becomes easier to solve by meta-heuristic algorithms as the restricted regions may have local minima which are removed from the problem.

Proposed formulation to determine size of the search domain can provide solutions for the uncomplicated networks. In addition to this, the removed section of the problem is examined if a dummy activity is added to the problem. The examination was based on enumeration, however pattern of the size of the removed section is revealed according to the starting and finishing nodes of the included dummy activity. Every feasible combination of addition of dummy activities on the sample problems is examined. A robust formula cannot be derived at the end of the examination, however the pattern of the eliminated activity priority arrangements are detected. An explicit mathematical formula can be derived by considering the positions of the start and end nodes of the dummy activity and the total number of the activities of the network.

5. Conclusion

Resource constrained project scheduling problem is an important research area for the project management science. In this study, search domain of the problem is examined with respect to the activity priority based solution attempts. Search domain of the problem is detected for six case study problems by combinatorial methods and mathematical formula and both methods provided the same results. This proved that the proposed combinatorial computation scheme is a proper algorithm to estimate the search domain of the problem. Moreover, it is seen that the derived equation to compute the search domain of the resource levelling problem is also valid for the resource constraint project scheduling problem. On the other hand, it was not possible to derive a robust formula which covers all of the dummy activity combinations. However, an explicit formula can be derived as future study. The findings of the study reveal that the search domain enlarges factorial when the number of activity increases. Moreover, the search domain shrinks when complex logical relationships are defined between the activities.

References

Arauzo J.A., Galan J.M., Pajares J., Lopez-Paredes A., (2009) Efficient project portfolio management. An intelligent decision support system for engineering and consultancy firms. Dyna. 84(9), 761–772.

Bettemir Ö.H., Sonmez R., (2015) Hybrid genetic algorithm with simulated annealing for resource-constrained project scheduling. Journal of Management in Engineering. 31(5), 04014082.

Bettemir, Ö. H., Erzurum, T. (2021). Kaynak Dengeleme Probleminin Arama Uzayını Paralel Programlama ile Tarayarak Kesin Çözümü. Teknik Dergi, 32(3), 10767-10805.

Birjandi A., Mousavi S.M., (2019) Fuzzy resource-constrained project scheduling with multiple routes: A heuristic solution. Automation in Construction. 100, 84-102.

Brucker P, Knust S., Schoo A., Thiele O., (1998) A branch and bound algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research. 107 272-288.

Chakrabortty R.K., Abbasi A., Ryan M.J., (2020) Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. International Transactions in Operational Research. 27(1), 138-167.

Christofides N., Alvarez-Valdes R., Tamarit J.M., (1987) Project scheduling with resource constraints: a branch and bound approach. European Journal of Operational Research. 29, 262-273.

Creemers S., (2019) The preemptive stochastic resource-constrained project scheduling problem. European Journal of Operational Research. 277(1), 238-247.

Hartmann S., (1998) A competitive genetic algorithm for resource constrained project scheduling. Naval Research Logistics (NRL). 45(7) 733-750.

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. Computers & Operations Research, 25(4), 279-302.

Kanıt R., Baykan U.N., Erdal M., (2005) Kısıtlı Kaynak Koşullarının Yapı Maliyetine Etkisinin İncelenmesi. Politeknik Dergisi. 8(2), 209-22.

Kolisch R., (1996) Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. European Journal of Operational Research. 90(2), 320-333.

Laszczyk M., Myszkowski P.B., (2019) Improved selection in evolutionary multi–objective optimization of multi–skill resource–constrained project scheduling problem. Information Sciences. 481, 412-431.

Lin J., Zhu L., Gao K., (2020) A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. Expert Systems with Applications. 140 112915.

Merkle D., Middendorf M., Schmeck H., (2002) Ant colony optimization for resource-constrained project scheduling. IEEE transactions on evolutionary computation. 6(4), 333-346.

Olaguíbel R.A.V., Goerlich J.M.T., (1989) Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. Advances in project scheduling. 113-134.

Ozdamar L., Ulusoy G., (1995) A survey on the resource constrained project scheduling problem. IIE Transactions. 27, 574-586.

Pellerin R., Perrier N., Berthaut F., (2020) A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. European Journal of Operational Research. 280(2) 395-416.

Tirkolaee E.B., Goli A., Hematian M., Sangaiah A.K., Han T., (2019) Multi-objective multi-mode resource constrained project scheduling problem using Pareto-based algorithms. Computing. 101(6), 547-570.

Ulusoy G., Ozdamar L., (1994) A constrained-based perspective in resource constrained project scheduling. International Journal of Production Research. 32, 693- 705.

Wang X., Chen Q., Mao N., Chen X., Li Z., (2015) Proactive approach for stochastic RCMPSP based on multi-priority rule combinations. International Journal of Production Research. 53(4), 1098–1110.

Zhang H., Li H., Tam C.M., (2006) Particle swarm optimization for resource-constrained project scheduling. International Journal of Project Management. 24(1), 83-92.

Zhu L., Lin J., Wang Z.J., (2019) A discrete oppositional multi-verse optimization algorithm for multi-skill resource constrained project scheduling problem. Applied Soft Computing. 85, 105805.

Communication aligned with the Sustainable Development Goals